

1.1 Introduction to HCI

Compiled by Shipra De, Summer 2017

Humans

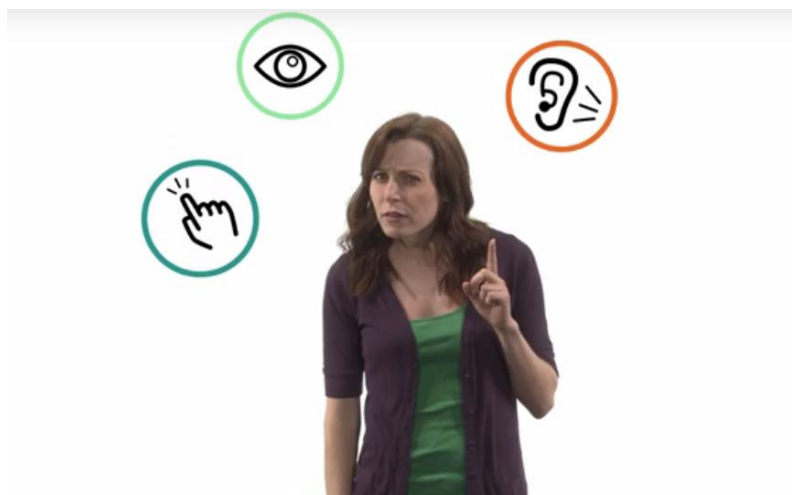


[MUSIC] This is Morgan.

>> Hello.

>> Morgan is a human.

>> Last time I checked.



>> As a human, Morgan has various ways of perceiving the world around her, like seeing, hearing, and feeling.

>> Is anyone else seeing these?



>> There are a few more as well like smelling and tasting, but we won't deal with those as much.

>> Thank goodness.



>> But Morgan has more than senses. She also has memories, experiences, skills, knowledge.

>> Thanks.



>> In human computer interaction, we have to take into consideration every element of the human, from the way they perceive and interact with the world, to their long history of using computers and technology.

Computers



This is a computer. Or at least, this is probably what you think of when you think of the term computer. But this is also a computer. And so is this. And so is this. And so is this. And so is this.

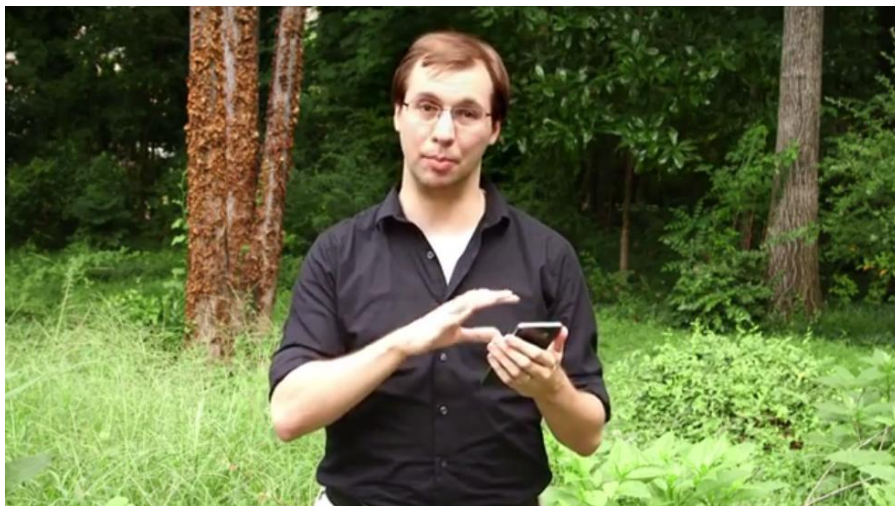
>> Hey!

>> This is Amanda, my video producer.

>> Go on, I'm rolling.

>> Right, and so is this. And so is this, and this, and this, and this, and even this. And so is this. And so is this. And so is this. [SOUND] And so is this. And so is this.

>> Hey David?

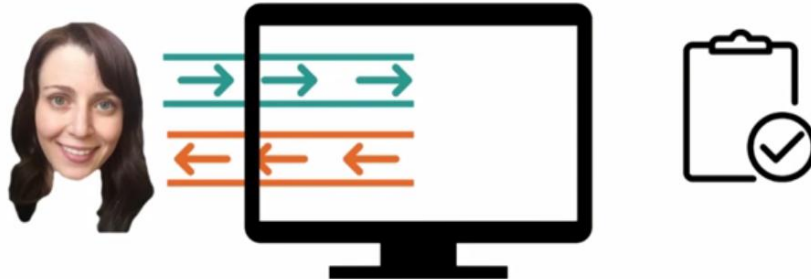


>> One second, trying to get to Squirtle. There we go. With mobile devices and augmented reality, HCI is quite literally everything. Pokemon Go was released a few days before I recorded this and augmented reality games like this turn effectively the entire world into an instance of HCI. Even out here in the middle of nowhere, I'm still doing something with computers.

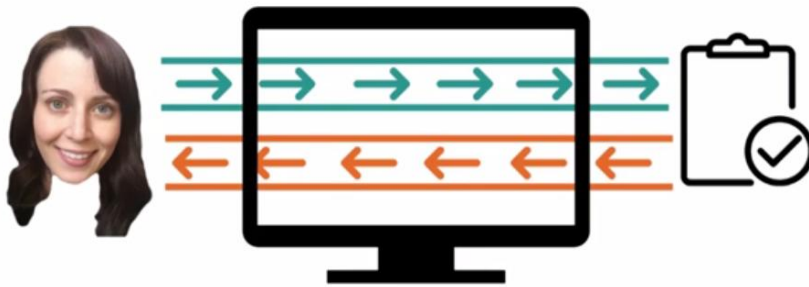
Interaction



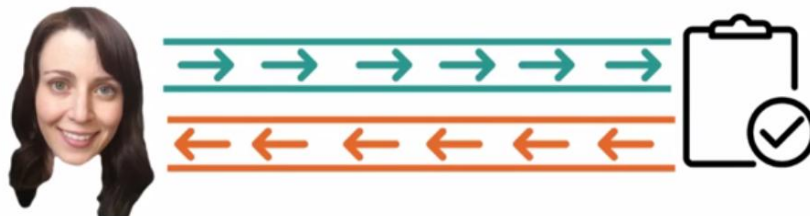
We have humans and we have computers and we're interested in the interaction between them. That interaction can take different forms though. The most obvious seems to be the human interacts with the computer and the computer interacts with the human in response. They go back and forth interacting, and that's a valid view. But it perhaps misses the more interesting part of HCI.



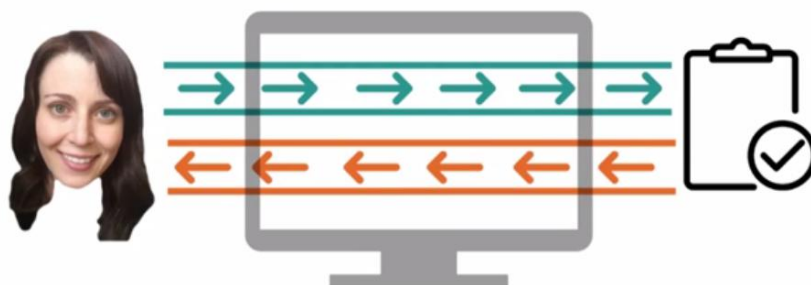
We can also think of the human interacting with the task, through the computer.



The interaction is really between the human and the task and the computer in the middle just mediates that interaction. Or to put this differently, the human and the computer together, interact with the task.



Ideally in this case, we're interested in making the interface as invisible as possible, so the user can spend as little time focusing on the interface and instead focus on the tasks that they're trying to accomplish.



Realistically, our interfaces are likely to stay somewhat visible. But our goal is to let the user spend as much time as possible thinking about the task, instead of thinking about our interface. We can all probably remember times when we've interacted with a piece of software and we felt like we spent all our time thinking about how to work the software. As opposed to accomplishing what we were using

the software to do in the first place and that's frustrating. So our goal as designers, is to help the human feel like they're interacting directly with that task. While our interface, kind of vanishes, in the middle of that interaction.

Quiz: Reflections: Interacting & Interfaces



We'll talk extensively about the idea of disappearing interfaces and designing with tasks in mind. But in all likelihood, you've used computers enough to already have some experience in this area. So take a moment and reflect on some of the tasks you do each day involving computers. Try to think of an example where you spend most of your time thinking about the task and an example where you spend most of your time thinking about the tool.



Video games actually give us some great examples of interfaces becoming invisible. A good video game is really characterized by the player feeling like they're actually inside the game world, as opposed to controlling it by pressing some buttons on a controller. We can do that through some intuitive controller design like pressing forward moves forward, and pressing backward moves backwards. But a lot of times we'll rely on the user to also learn how to control the game over time. But as they learn, it becomes invisible between them and their interaction. A classic example of a place where interaction is more visible, is the idea of having more than one remote control that controls what feels like the same system. So I have these two controllers that control my TV and my cable box together. And for me it

feels like this is just one task, watching TV. But technologically, it's actually different tasks. So I have to think about am I using the number pad on this controller or this controller, depending on what I'm trying to do at a given time. So I spend a lot of time thinking about the interface and not as much thinking about the task.

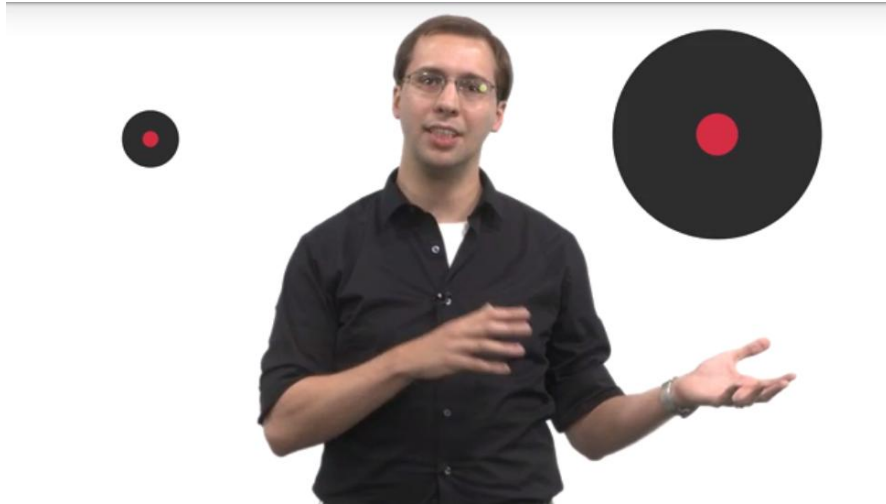
The HCI Space



One of the most exciting parts of HCI is it's incredible ubiquity. Computers are all around us and we interact with them everyday. It's exciting to think about designing the types of tools and interfaces we spend so much time dealing with, but there's a danger here too. Because we're all humans interacting with computers, we think we're experts at human-computer interaction. But that's not the case.



We might be experts at interacting with computers, but that doesn't make us experts at designing interactions between other humans and computers. We're like professional athletes or world-class scientists. Just because we're experts doesn't mean we know how to help other people also become experts.

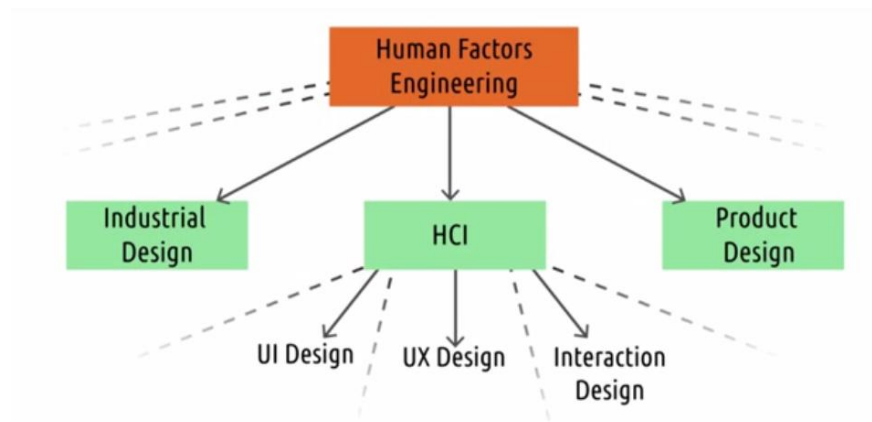


In my experience, many people look at HCI like this. The red dot represents what they know and the black circle represents what they think there's to know. They know there's probably some things they don't know yet, but they're already pretty at it, and it wouldn't be too hard to become an expert. After studying HCI for a bit though, they look more like this. You can see that they've increased what they know but their perception of what there is to know has grown even more. That's the journey we'll be taking together. You'll learn to do work in HCI, but perhaps more importantly, you'll learn how complex and large the field of HCI is. Your knowledge will increase, but yet you might exit the class less confident in your HCI ability than when you started. You're taking the first step into a larger world.

HCI in the Big Picture



Now, what we've described so far is a huge field, far too big to cover in any one class. In fact there are lots of places where you can get an entire masters degree or PhD in human computer interaction. Here are some of the schools that offer programs like that. And these are just the school that offer actual degree programs in HCI, not computer science degrees with specializations in HCI, which would be almost any computer science program. So let's look more closely at what we're interested in for the purpose of the next several weeks. To do that, let's look at where HCI sits in a broader hierarchy of fields.



We can think of HCI as a subset of a broader field of human factors engineering. Human factors engineering is interested in a lot of the same ideas that we're interested in. But they aren't just interested in computers. Then there are also sub disciplines within HCI. This is just one way to represent this. Some people, for example, would put UI design under UX design, or put UX design on the same level as HCI, but this is the way I choose to present it. Generally, these use many of the same principles that we use in HCI, but they might apply them to a more narrow domain, or they might have their own principles and methods that they use in addition to what we talk about in HCI in general. So to get a feel for what we're talking about when we discuss HCI, let's compare it to these other different fields.

HCI vs Human Factors



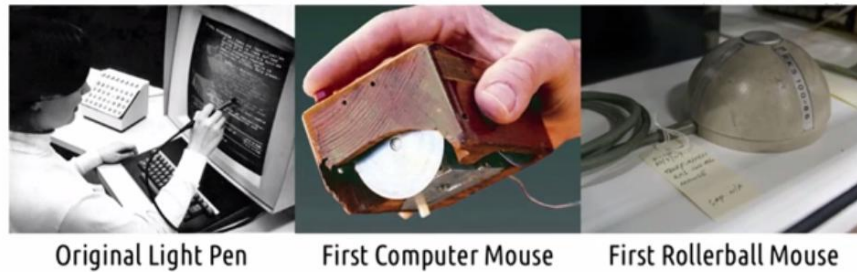
First let's start by comparing HCI to the broader field of human factors. Human factors is interested in designing interactions between people and products, systems or devices. That should sound familiar. We're interested in designing the interactions between people and computers. But computers are themselves products or systems. But human factors is interested in non-computing parts of this, as well.



Let's take an example. I drive a pretty new electric car, which means there are tons of computers all over it. From a HCI perspective, I might be interested in visualizing the data on the dashboard, or helping the driver control the radio. Human factors is interested not only in how I interact with the computerized parts of the car but the non-computerized parts as well. It's interested in things like the height of the steering wheel, the size of the mirrors, or the positioning of the chair. It's interested in designing the entire environment, not just the electronic elements. But that means it's interested in a lot of the same human characteristics that we care about, like how people perceive the world, and their own expectations about it. So many of the principles we'll discuss in this class, come from human

factors engineering, applied more narrowly to computerized systems. But the exciting is that as computers become more and more ubiquitous, the number of application areas for HCI is growing. 20 years ago, a car might not have been an application of HCI. Ten years ago a wrist watch would have been more about industrial design than HCI. And without only the past couple years, things like showerheads and refrigerators have started to become truly computerized devices. As computers integrate themselves into more and more of our devices, the gap between human computer interaction and human factors engineering is shrinking. As computers become more and more ubiquitous there is coming a time when pretty much every single thing on your car, will actually be run through a computer. Don't believe me? Check out the inside of Teslas model S. When you look at the console of a Tesla, almost everything you see is giant computer. Cars have become computers on wheels. Watches have become computers on wristbands. Car keys have become computers on keychains. Computers are everywhere. And so, HCI is everywhere.

HCI vs User Interface Design

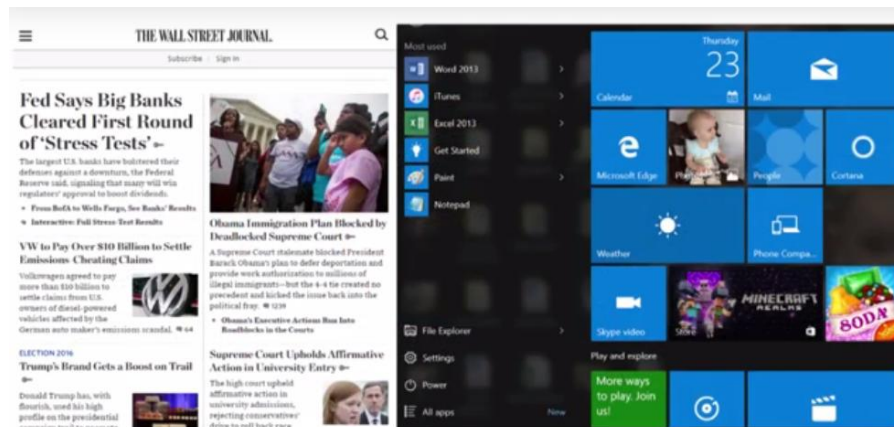


Original Light Pen

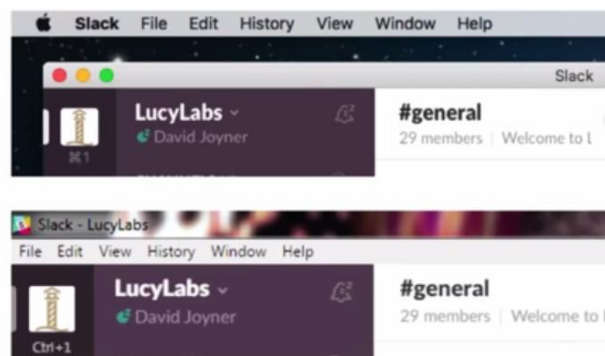
First Computer Mouse

First Rollerball Mouse

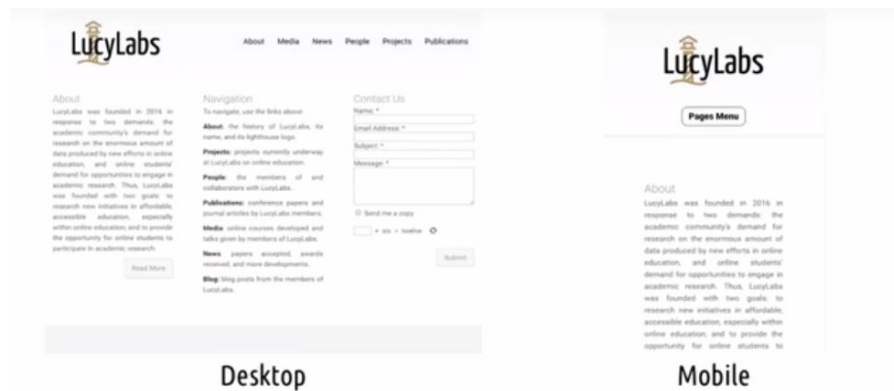
For many years, human-computer interaction was largely about user interface design. The earliest innovations in HCI were the creation of things like the light pen, the first computer mouse, which allow for flexible interaction with things on screen. But the focus was squarely on the screen.



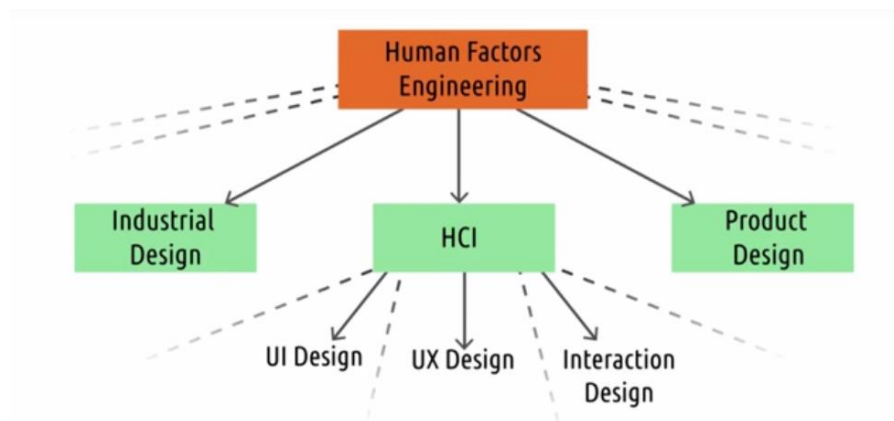
And so, we developed many principles about how to design things nicely for a screen. We borrowed from the magazine and print industries and identify the value of grids in displaying content and guiding the users eyes around our interfaces.



We created laws that govern how difficult it is for users to select what they want on screen. We examined for example whether it's easier to select a menu on a Mac, where the menus are always at the top of the screen. Or on a PC, where they're grouped with the individual window.

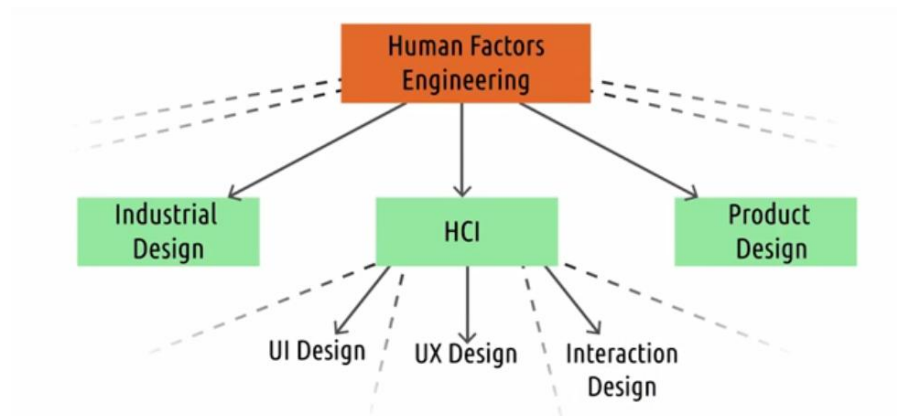


We developed techniques for helping interfaces adapt to different screen sizes and we developed methods for rapidly prototyping user interfaces using pen and paper or wire frames.

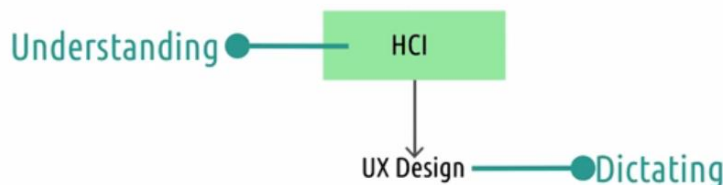


Through this rich history, UI design really became its own well defined field. In fact, many of the concepts we'll cover in HCI were originally developed in the context of UI design. But in HCI, we're interested in things that go beyond the user's interaction with a single screen. Technically, you can cover that in UI design as well, but traditionally most of the UI design classes I see focus on on-screen interaction. In HCI, we'll talk about the more general methods that apply to any interface.

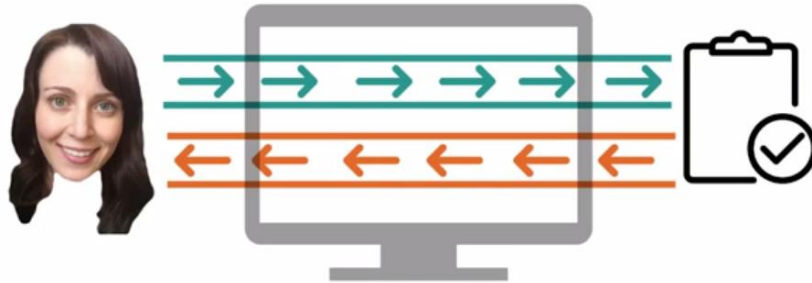
HCI vs User Experience Design



The relationship between HCI and user experience design is a little bit closer. In fact, if you ask a dozen people working in the field, you'll likely get a dozen different answers about the difference.

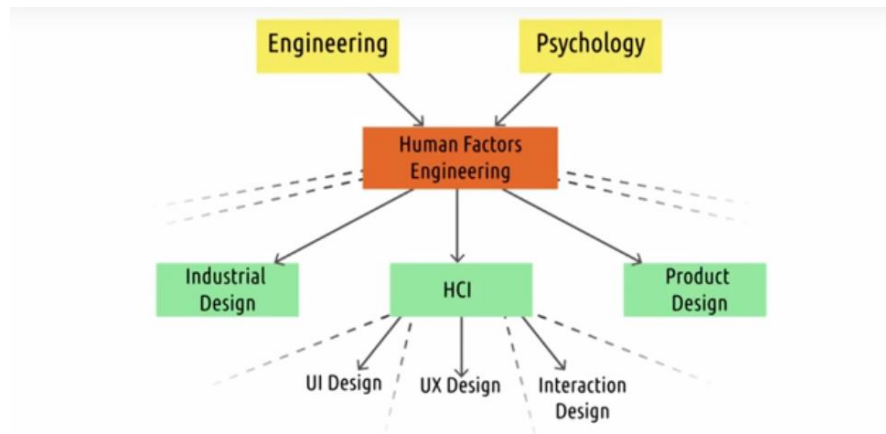


For the purposes of our conversations though, we'll think about the difference like this. HCI is largely about understanding the interactions between humans and computers. User experience design is about dictating the interactions between users and computers. In order to design user experiences very well, you need to understand the user, you need to understand their interactions with interfaces. And that's why I personally consider user experience design to be a subfield of the broader area of HCI. In our conversations, we'll use the principles and methods from HCI to inform how we design user experiences. But it's important to note that this relationship is deeply symbiotic. We might use that understanding to inform how we design the user experiences. But then we evaluate those designs and based on their success or failures, we'll use that to inform our increasing knowledge of human computer interaction itself. If our understanding leads us to create good designs, that provides evidence that our understanding is correct. If we create a design with some understanding and that design doesn't work, then maybe our understanding was flawed and now our understanding of human computer interaction as a whole will increase. This is similar to something called design based research, which we'll talk about later, using the results of our designs to conduct research.



You might also notice that this is very related to our concept of feedback cycles. Just as a user uses an interface to participate in some task and then evaluates the result of their interaction, so also we design interfaces and evaluate their success. You'll find that feedback cycles are really all over this course and all over the world in general.

HCI vs Psychology



The research side of HCI connects to the relationship between HCI and psychology. And if we zoom out even further on this hierarchy of disciplines, we might say that human factors engineering itself is in many ways the merger of engineering and psychology. As well as other fields of design and cognitive science. In HCI, the engineering side takes the form of software engineering, but this connection to psychology remains, and in fact, it's symbiotic. We use our understanding of psychology, of human perception, of cognition to inform the way we design interfaces. We then use our evaluations of those interfaces to reflect on our understanding of psychology itself. In fact, at Georgia Tech, the Human Computer Interaction class is cross listed as a Computer Science and Psychology class.

▼ CHI '92

May 3 - 7, 1992

A 'Pile' Metaphor for Supporting Casual Organization of Information

Richard Mander, Gitta Salomon and Yin Yin Wong

Human Interface Group, Advanced Technology
Apple Computer, Inc.
20525 Mariani Ave., MS 76-3H
Cupertino, California 95014
(408)996-1010

ABSTRACT

A user study was conducted to investigate how people deal with the flow of information in their workspaces. Subjects reported that, in an attempt to quickly and informally manage their information, they created piles of documents. Piles were seen as complementary to the folder filing system, which was used for more formal archiving. A new desktop interface element – the pile – was developed and

We were also interested in how people work with assistants when dealing with information.

By examining individuals' information management schemes, we were able to extract and extrapolate a number of interesting interface ideas for a graphical interface. Our intent was not simply to emulate physical world functionality – several investigators have argued against this proce-

So let's take an example of this. In 1992, psychologists working at Apple wanted to study how people organized the rapid flow of information in their work spaces.

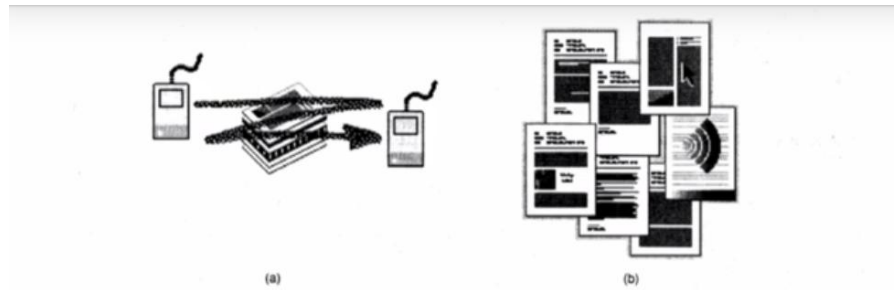


Figure 4. *Browsing by spreading out a pile.* Gesturing sideways with the mouse pointer, or with a finger in the case of a touch screen, causes the pile contents to spread out. Individual items can now be directly manipulated.



They observed that people tended to form piles of related material, kind of like a less formal filing system, and so they then designed a computer interface, that would mimic that ability.

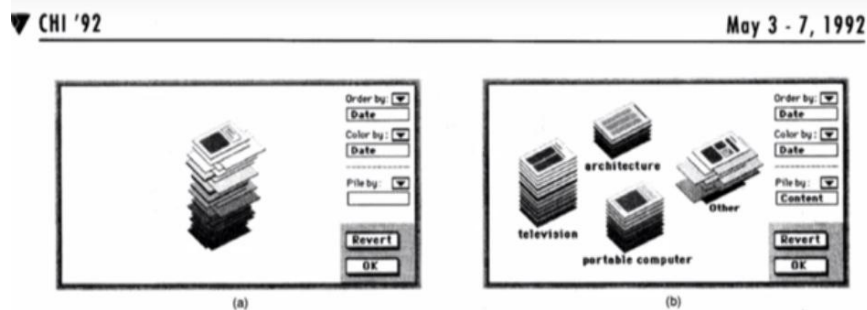


Figure 6. *Visualizing a pile's contents.* The pile shown is within a 'visualizing environment' that allows the user to select and visualize several criteria. Criteria can be mapped to the pile's order, the color of the items within the pile, or the way a pile is broken into sub-piles. In (a) the pile is both ordered and colored by date. In (b) the user chose to 'pile by' content. Therefore, the system separated the original pile into four content-based piles. Three are labeled with specific terms suggested by the system (e.g. 'architecture'), appear neat and are now scripted to maintain similar content. The remaining disheveled pile, 'other,' contains items which did not fit into any of the other three piles.

By virtue of using miniatures of the actual documents, we offered edge browsing capabilities. In addition, we explored gestural inputs as a way to invoke other browsing methods. For example, a horizontal gesture would spread out a pile so that miniatures of each item's first page were

Method

Five men and five women in nontechnical positions at Apple Computer were individually tested in approximately one hour sessions. The subjects were asked to think aloud [5] while working through 5 tasks, and the sessions were

Finally, they used the results of that development to reflect on how people were managing their work spaces in the first place. So in the end, they had a better understanding of the thought processes of their users as well as an interface that actually helped users. So in the end, their design of an interface with an HCI informed their understanding of psychology more generally. We came away with a better understanding of the way humans think about their work spaces because of our experience designing something that was supposed to help them think about their work spaces.

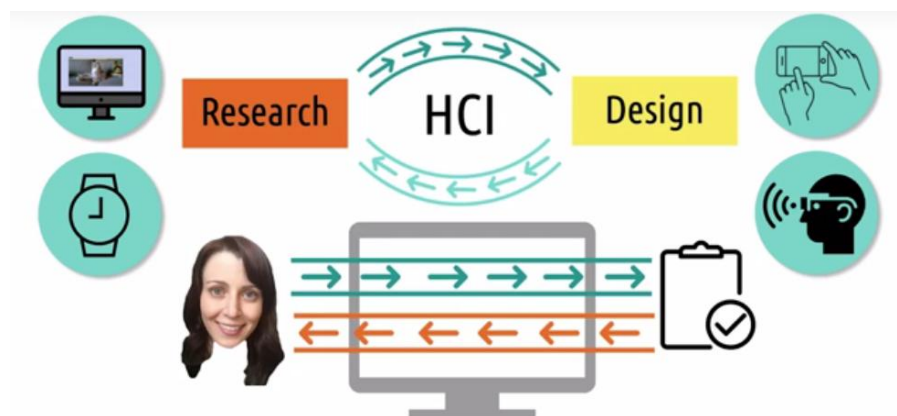
HCI: Research and Design



Now that we've talked at length about what HCI isn't, let's talk a little bit about what HCI actually is. On the one hand, HCI is about research. Many of the methods we'll discuss are about researching the user, understanding their needs, and evaluating their interactions with designs that we've prototyped for them. But on the other hand, HCI is about design. After all, design is that prototyping phase, even though we're prototyping with research in mind. HCI is about designing interactions to help humans interact with computers, oftentimes using some known principals for good interaction. Things like designing with distributed cognition in mind, or making sure the user develops good mental models of the way the interface works, or making sure they design with universal design in mind. We'll talk about all these topics later in our conversations. You don't need to worry about understanding any of these right now. What is important to understand right now, is that these aren't two isolated sides.



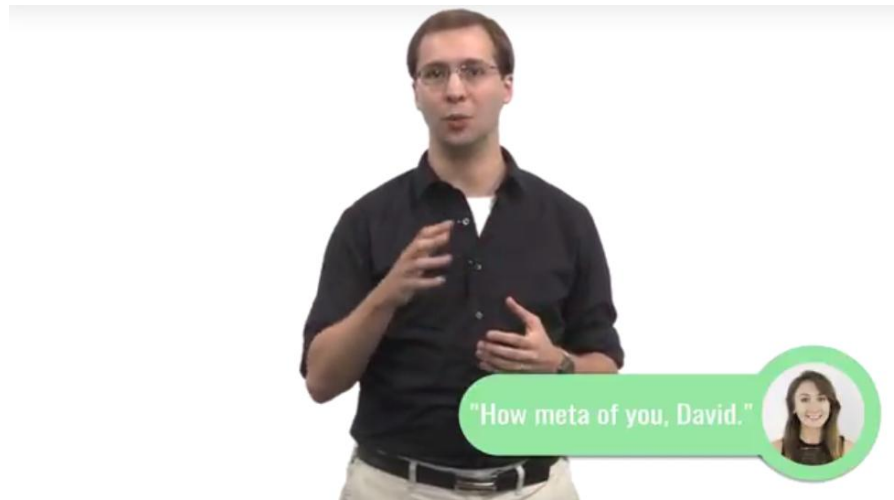
The results of our user research inform the designs we construct, and the results of our designs inform our ongoing research.



Again, you might notice this is very similar to the feedback cycles we're designing for our users. They use what they know to participate in the task, and then use the feedback from that participation to

inform what they know. We use what we know to design good interfaces, and then use the results of those interfaces to inform our ongoing research. This is the heart of what HCI is, for the purpose of our conversations. Using research to inform our designs and using the results of those designs to inform our ongoing research. And this cycle appears anywhere that humans are using computers to participate in tasks. That could be sitting on a desk using a screen. That could be using a smartphone or a smartwatch. That could be participating with some kind of touch or gesture base system, or could be some interesting in merging technologies like virtual and augmented reality. In HCI were interested in the general principles and methods for designing and researching all of these things.

Welcome to HCI!



So now you know what we're going to cover in our exploration of HCI. We're going to talk about some of the fundamental design principles that HCI researchers have discovered over the years. We're going to talk about performing user research, whether it be for new interfaces, or exploring human cognition. We're going to talk about the relationship between these two, how our research informs what we design, and how what we design helps us conduct research. And we're going to talk about how these principles work in a lot of domains, from technologies like augmented reality, to disciplines like healthcare. I hope you're excited. I know I am. I like to think of this not just as a course about human-computer interaction, but also an example of human-computer interaction. Humans using computers in new and engaging ways to teach about computer interaction. We hope this course exemplifies the principles as well as teaches them.

1.2 Introduction to CS6750

Compiled by Shipra De, Summer 2017

Introduction to CS6750



[MUSIC] Now that you understand a little bit about what human-computer interaction is, let's talk about what this class is going to be like. In this lesson, I am going to take you through a high level overview of this class. What material we'll cover, how it fits together, and what you should expect to know by the end of the course. I'll also talk a little bit about the assessments we'll use in the class, but be aware, these assessments are only applicable to students taking this class through Georgia Tech. If you're watching this course on your own or taking it to complement other courses you're taking, those assessments won't apply to you, but you'll get to hear a little bit about what students in the actual course do. If you are a student in the course, you should know the assessments do tend to change a bit semester to semester. I'm going to try and stay as general as possible to capture future changes, but make sure to pay attention to the specific materials you're provided for your semester.

Learning Goals



In education, a learning goal is something we want you to understand at the end of the course. It's the knowledge contained within your head that you might not have had when we got started. In this class we have three major learning goals.



First, we want you to understand some of the common principles in human computer interaction. These are the tried and true rules on how to design good interactions between humans and computers.

Goal #2: to understand the design life cycle.

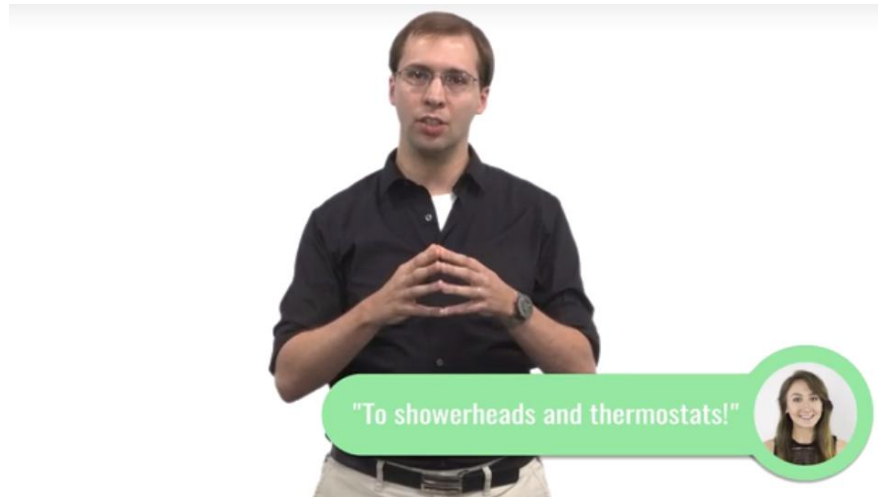


Second, we want you to understand design life cycle. That's how interfaces go from conception to prototypes to evaluation. And we especially want you to understand the roll of iteration in this process.

Goal #3: to understand current applications of HCI.

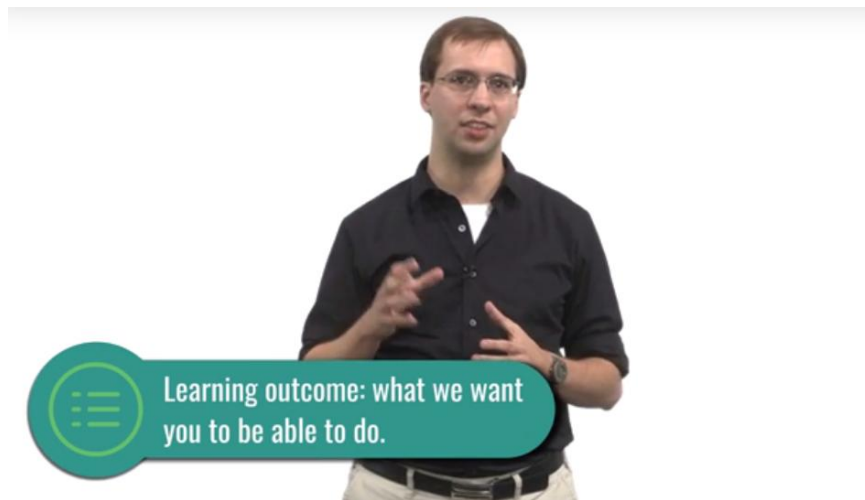


Third, we want you to understand the expanse of the human computer interaction field and the current applications available for HCI. HCI is really everywhere, from domains like healthcare, to technologies like virtual reality, to emerging techniques like sonification.



We want you to understand the broad range of application areas for HCI in the modern world.

Learning Outcomes: To Design



While learning goal is something we want you to know at the end of the course, a learning outcome is something we want you to be able to do. This course really has one learning outcomes but there are some nuances to it.



The learning outcome for this course is to be able to design effective interactions between humans and computers. The first part of this learning outcome is to design. But what is design? Well for us design is going to take two forms.



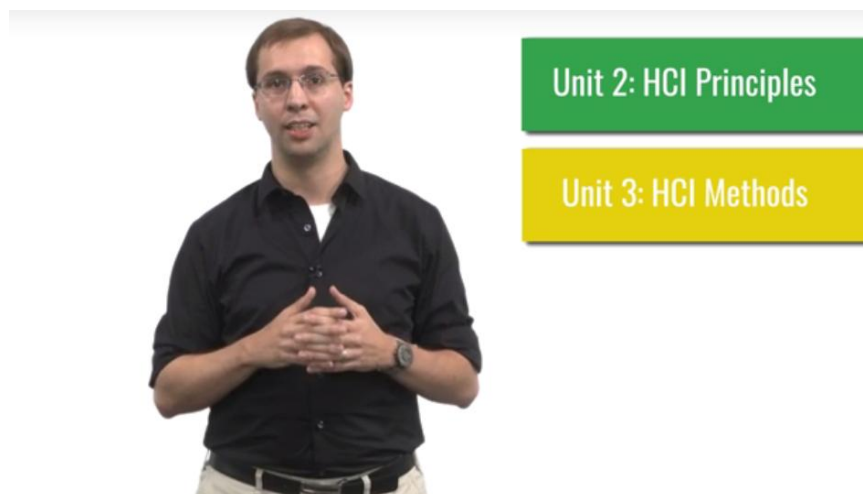
First, design is an activity where you're applying known principles to a new problem. For example we'll talk a lot about the importance of getting users to write kind of feedback at the right time. That's a plan of principle of feedback to some new design problem we encounter.



But design is a second form as well, design is also a process where you gather information, use it to develop design alternatives, evaluate them with users and revise them accordingly. When designing interface for some tasks, I would ask some potential users how they perform some task right now. I develop multiple different ideas for how we can help them. I give those to the users to evaluate, and I will use the experiences to try to improve the interface on that time.



So let's take an example of this. Imagine I was designing a new thermostat. On the one hand, designing a new thermostat means applying known HCR principles, like feedback and error tolerance to some new design. On the other hand, designing a new thermostat means creating different ideas, giving them to users, getting their feedback and then revising those designs. Both these sides of design are very important. You don't want to ignore decades of experience when designing new interfaces, but simply applying known principles to a new problem doesn't guarantee you have a good design. Designing is about both these things. And in fact, these two things are a vast majority of material that we'll cover in this course.

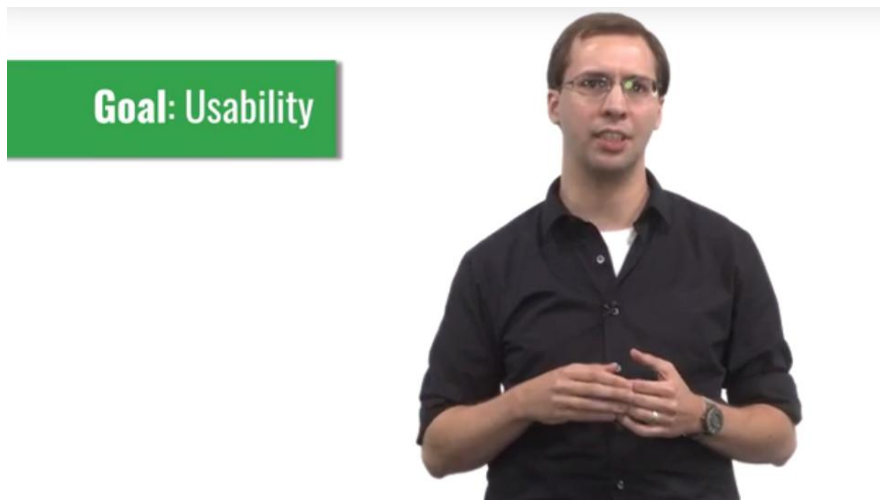


We'll cover the principles uncovered by a human factors engineering and human computer interaction research. And we'll cover the methods used in the HCI. We're gathering user requirements, developing designs, and evaluating new interfaces.

Learning Outcomes: Effectiveness



The first part of this learning outcome to design needed some definition, but the second part seems pretty straightforward, right? Not exactly. Effectiveness is defined in terms of our goal.



The most obvious goal here might be usability and for a lot of that's exactly what we're interested in. If I'm designing a new thermostat, I want the user to be able to create the outcome they want as easily as possible.

Goal: Research



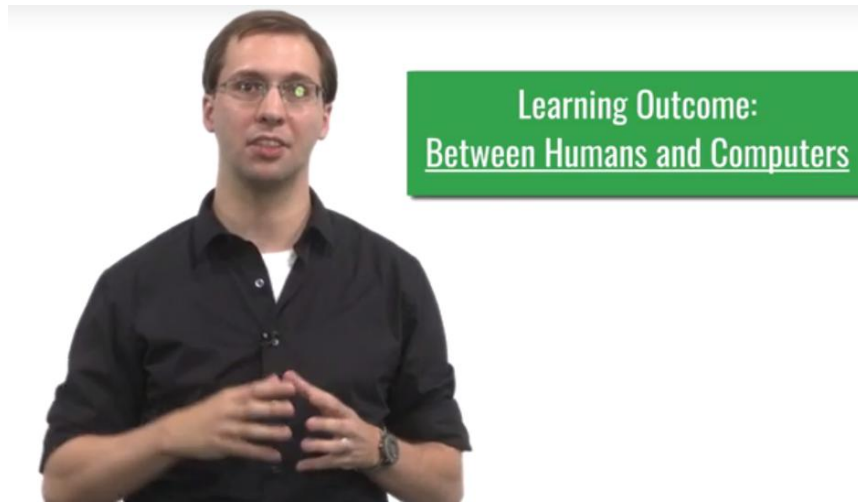
But maybe usability isn't my goal, maybe it's research. Maybe I'm interested in investigating what makes people think that the thermostat is working correctly. In that case, I might deliberately create some thermostats that are harder to read, just to see how that changes people's perceptions of the system.

Goal: Change



Or it could be that my goal isn't to make the certain activity easier but rather to change that activity. Maybe I'm interested in reducing a home's carbon footprint. In that case, my goal is to get people to use less electricity. I might design the interface of the thermostat specifically to encourage people to use less. Maybe I'd show them a comparison to their neighbor's usage, or allow them to set energy usage goals. Or I could make the thermostat physically harder to turn up and down. So effectiveness is very much determined by the goal that I have in mind. We'll generally assume that our goal is usability, unless we state otherwise. But we'll definitely talk about some of those other goals as well.

Learning Outcomes: Between Humans and Computers



The final part of our desired learning outcome is between humans and computers. We want to design effective interactions between humans and computers. Well, what is important to note here, is where we're placing the emphasis. Note that we didn't say designing effective interfaces, because that puts the entire focus on the interface. We're deeply interested in the human's role in this interaction.



So rather than designing interfaces, designing programs, designing tools, we're designing interactions. We're designing tasks. We're designing how people accomplish their goals, not just the interface that they use to accomplish their goals.



Take our thermostat for example. When we started this process, our goal shouldn't be to design a thermostat. Our goal should be to design the way in which a person controls the temperature in their home. That subtle shift in emphasis is powerful. If you set out to design a better thermostat, you might design a wall-mounted device that's easier to read or easier to use.



But if you set to design a better way for people to control the temperature in their home, you might end up with Nest. A device that learns from the user and starts to control the temperature automatically.

Learning Strategies: Video Material



Learning strategies are how we plan to actually impart that knowledge to you. This is how we attempt to help you achieve the learning goals and learning outcomes. Within these videos, we'll use a number of different strategies to try to help you understand the core principles and methodologies of HCI.



We'll use learning by example. Every lesson and, in fact, this course, as a whole. Is organized around a collection of running examples that will come up over and over again.



We use learning by doing. Throughout the course we'll ask you to engage in designing interactions to solve different problems in different contexts. These aren't required, since there's really no way we can verify if you've done them, but we really hope you'll take a few minutes and think about these. We'll also use learning by reflection a lot.



We'll ask you to reflect on times when you've encountered these things in your own everyday life. These strategies are useful because they connect to your own personal experiences but once again, there's a danger here.

“You are not your user!”



One of the recurrent points in HCI is that when you are designing interactions, you are not your own user. Focusing too much on your own experiences can give you a false sense of expertise. So I'll use some strategies to help take you out of that comfort zone and confront how little you might understand these tasks with which you thought you were so familiar.

Learning Strategies: Georgia Tech

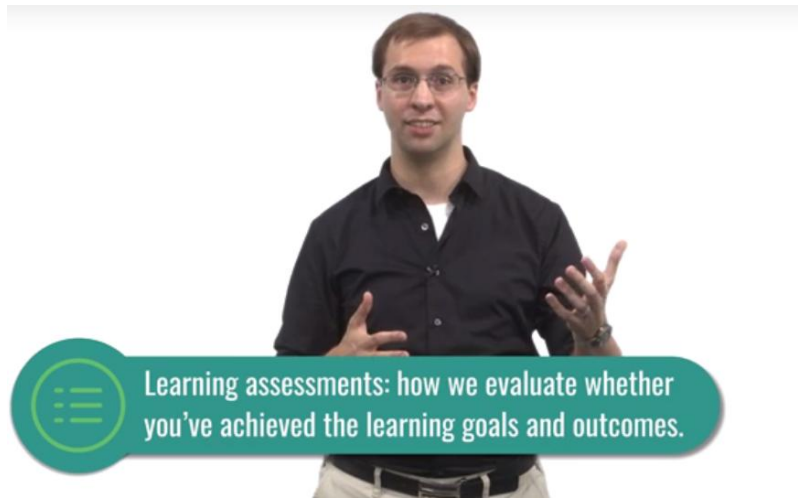


Within the full course at Georgia Tech, there are a number of other strategies in which you'll engage as well. First, we're really passionate about leveraging the student community in this class to improve the experience for everyone. Taking this class with you are people with experience in a variety of industries, many of whom have significant experience in HCI. So some strategies we'll use include peer learning, collaborative learning, learning by teaching, and communities of practice. You'll learn both from each other and with each other. You'll play the role of student, teacher, and partner, and you will learn from each perspective.



In addition, the entire course is built around the idea of project-based learning. Early in the semester, you'll form a team and start looking at a problem we've selected, or maybe one in which you're already interested. This project will then become the domain through which you explore the principles and methods of human-computer interaction. Who knows? By the end of the semester, you might even generate something with the potential to go forward as a real-world product, or as a research project.

Learning Assessments



Learning goals are what we want you to understand. Learning outcomes or what we want you to be able to do. Learning assessments then, are how we evaluate whether you can do what we want you to be able to do and understand what we want you to understand. The learning outcome to this class is to be able to design effective interactions between humans and computers.



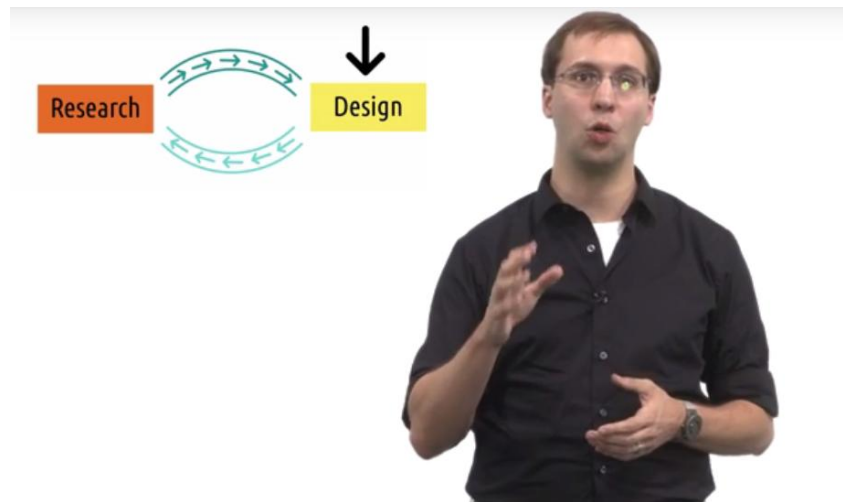
So the primary assessments in this class are to, say it with me, design effective interactions between humans and computers. You'll start with some relatively small scale tasks, recommending improvements to existing interfaces or undertaking some small design challenges. But as the semester goes on, you'll scope up towards a bigger challenge. You'll initially investigate that challenge individually and then you'll merge into teams to prototype and evaluate a full solution to the challenge you chose.

**"CS6750 is a journey, not a destination."
-Ralph Waldo Emerson (kind of)**



At the end, you'll be evaluated not just on the final design you generate but on the process by which it was generated.

Course Structure



We'll close by talking about the overall structure of the content you'll be consuming. The course's lessons are designed to be as independent as possible, so you should be able to skip around if you want, but there's a certain logic to our planned presentation order. We discussed earlier the model HCI, how design informs research and research then informs design, so we'll start by discussing some of the core design principles of HCI.



Then we'll discuss the research methodologies for uncovering new user information, the iterative design lifecycle. We'll close by giving you the opportunity to peek at what's going on in the HCI community at large.

5 Tips Doing Well in CS6750



Here are five quick tips for doing well in this course. [SOUND] Number one. Look over the assignments early. Some of our assignments you can sit down and do them in an hour. But others require some advance coordination to talk to users, develop prototypes or test with real people. So go ahead and at least read all the assignment descriptions. Number 2. Start the assignments early. That's not just typical teacher talk saying, you can't get this done at the last minute. You probably can, but you're using interfaces like these in your everyday life. By starting early you're likely to get inspiration just in your day to day routine, and that's going to make writing the assignment significantly easier than trying to sit down and come up with something on the spot. Number three, participate, interact with your classmates, post on the forums, read others' posts. The knowledge and experience you gain there is just as valuable as anything you'll get listening to me in these videos. Number four, select an application area to explore. Next lesson you'll hear about several of the interesting areas of HCR research and development going on right now. Developing in many of these areas is outside the scope of this class, but I encourage you to pick an area in which you're interested and mentally revisit it throughout the course. Number five, leave behind what you know, or at least try. HCI is a huge area and yet many people believe that because they're already good at using computers, they'd be good at designing user experiences. But HCI above all else is about gaining a grounded understanding of the user's needs not assuming we already understand them. So while it's great to apply the course's principles to your every day life, be cautious about designing too narrowly based only on your own experiences.

Conclusion



In this lesson, I've tried to give you some expectations of what this course will be like. We've gone over the course's goals, outcomes, learning strategies and assessments.



We've covered the course's learning outcome in great detail, to design effective interactions between humans and computers. Now I focus mostly on the video material, because the assignments, projects and exams are separate from these videos, and are likely to change pretty significantly semester to semester.



The video material here will cover three general areas, principles, methods and applications. To really get into the applications, it's useful to understand the principles and methods. But at the same time, it's useful to keep the applications in mind, while learning about the principles and methods. So next we're going to briefly preview some of the application areas for you to keep in mind during the rest of our conversations. Then after we cover principles and methods, we'll invite you to revisit these application areas, and leave room to explore whatever you find most interesting.

1.3 Exploring HCI

Compiled by Shipra De, Summer 2017

Introduction to Exploring HCI

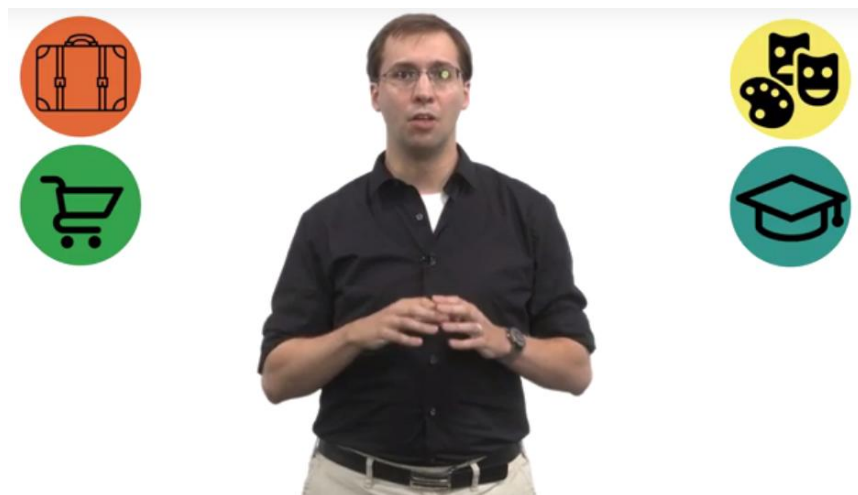


[MUSIC] [SOUND] Computers are finding their way into more and more devices, and as a result HCI is becoming more and more ubiquitous. It used to be that you wouldn't really need to think too much about HCI when designing a car or designing a refrigerator, but more and more computing is pervading everything. At the same time, new technological developments are opening up new areas for exploration. We're seeing a lot of really fascinating progress in areas like virtual reality, augmented reality, wearable devices. As we study HCI, we're going to talk a lot about things you've already used like computers and phones. But we want you to keep in mind some of these more cutting edge application areas as well. After all, if you're really interested in going into HCI professionally, you'll be designing for these new application areas. So we're going to quickly preview some of these. We'll divide them into three areas, **technologies**, **domains** and **ideas**. Technologies are emerging technological capabilities that let us create new and interesting user interactions. Domains are pre-existing areas that could be significantly disrupted by computer interfaces like healthcare and education. Ideas span both of these. They are the theories about the way people interact with interfaces and the world around them. Now, our delineation of this is kind of artificial. There's a lot of overlap. New technologies like augmented reality are what allow emerging ideas like contact sensitive computing to really have the power that they do. But for organization, we'll group our application areas into these three categories. When one of these areas catches your eye, take a little while and delve into it a little bit deeper. Then keep that topic area in mind as you go through the rest of the HCI material. We'll revisit your chosen area throughout the course, and ask you to reflect on the application of the course's principals and methods to your application area.

Technology: Virtual Reality

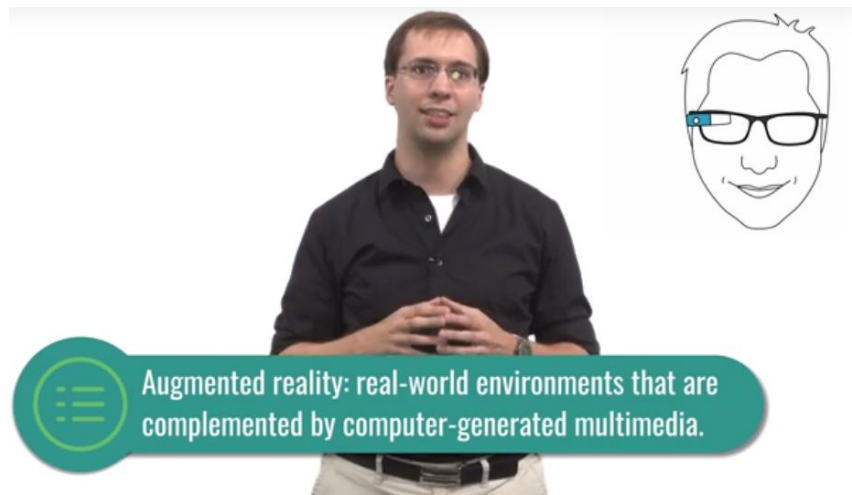


The year that I'm recording this is what many have described as the year that virtual reality finally hits the mainstream. By the time you watch this, you'll probably be able to assess whether or not that was true, so come back in time and let me know. Virtual reality is an entire new classification of interaction and visualization and we're definitely still at the beginning of figuring out what we can do with these new tools. You could be one of the ones who figures out the best way to solve motion sickness or how to get proper feedback on gestural interactions.

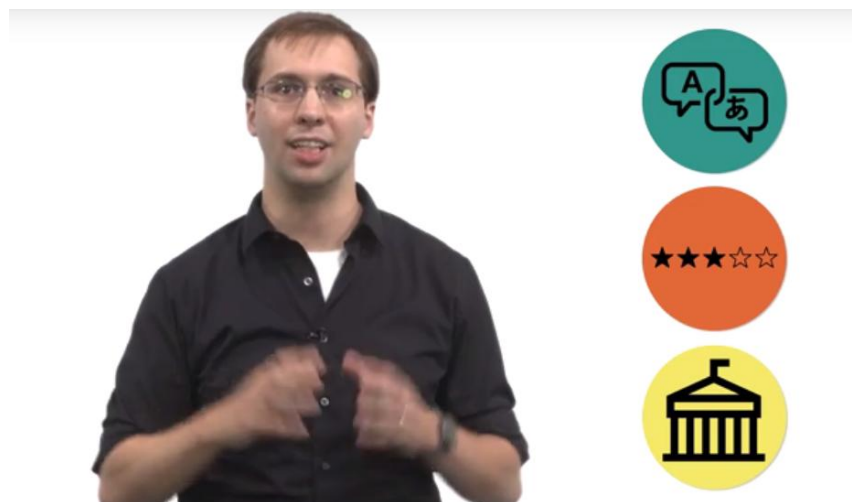


A lot of the press around virtual reality has been around video games, but that's definitely not the only application. Tourism, commerce, art, education, virtual reality has applications to dozens of spaces. For example, there is a lab in Michigan that's using virtual reality to treat phobias. They're creating a safe space where people can very authentically and realistically confront their fears. The possible applications of virtual reality are really staggering. So I'd encourage you to check them out as you go through this class.

Technology: Augmented Reality



Virtual reality generally works by replacing the real world's visual, auditory, and sometimes even all factory or kinesthetic stimuli with it's own input. Augmented reality on the other hand, compliments what you see and hear in the real world. So for example, imagine a headset like a Google Glass that automatically overlays directions right on your visual field. If you were driving, it would highlight the route to take, instead of just popping up some visual reminder. The input it provides complements stimuli coming from the real world, and instead of just replacing them. And that creates some enormous challenges, but also some really incredible opportunities as well.



Imagine the devices that can integrate directly into our everyday lives, enhancing our reality. Imagine systems that could, for example, automatically translate text or speech in a foreign language, or could show your reviews for restaurants as you walk down the street. Imagine a system that students could use while touring national parks or museums, that would automatically point out interesting information, custom tailored to that student's own interests. The applications of augmented reality could be truly stunning, but it relies on cameras to take input from the world, and that actually raises

some interesting societal problems. There are questions about what putting cameras everywhere would mean. So keep those in mind when we get to interfaces and politics, in unit two.

Technology: UbiComp and Wearables



Ubiquitous Computing [SOUND] refers to trend towards embedding computing power in more and more everyday objects. You might also hear it referred to as pervasive computing, and it's deeply related to the emerging idea of an Internet of Things. A few years ago, you wouldn't have found computers in refrigerators and wristwatches, but as microprocessors became cheaper and as the world became increasingly interconnected, computers are becoming more and more ubiquitous. Modern HCI means thinking about whether someone might use a computer while they're driving a car or going on a run. It means figuring out how to build smart devices that offloads some of the cognitive load from the user, like refrigerators that track their own contents and deliver advice to the users right at the right time.



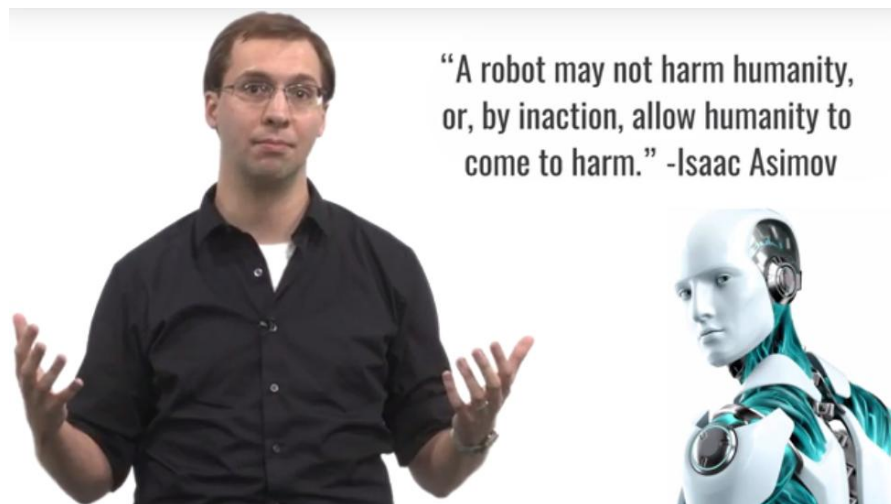
This push for increasing pervasiveness has also led to [SOUND] the rise of wearable technologies. Exercise monitors are probably the most common examples of this, but smart watches, Google Glass, augmented reality headsets, and even things like advanced hearing aids and robotic prosthetic limbs, are all examples of wearable technology. This push carries us into areas usually reserved for human

factors engineering and industrial design, which exemplifies the increasing role of HCI in the design of new products.

Technology: Robotics



A lot of the current focus on robotics is on their physical construction and abilities or on the artificial intelligence that underlies their physical forms. But as robotics becomes more and more mainstream, we're going to see the emergence of a new subfield of human-computer interaction, human-robot interaction. The field actually already exists. The first conference on human robot interaction took place in 2006 in Salt Lake City, and several similar conferences have been created since then. Now as robots enter the mainstream, we're going to have to answer some interesting questions about how we interact with them.



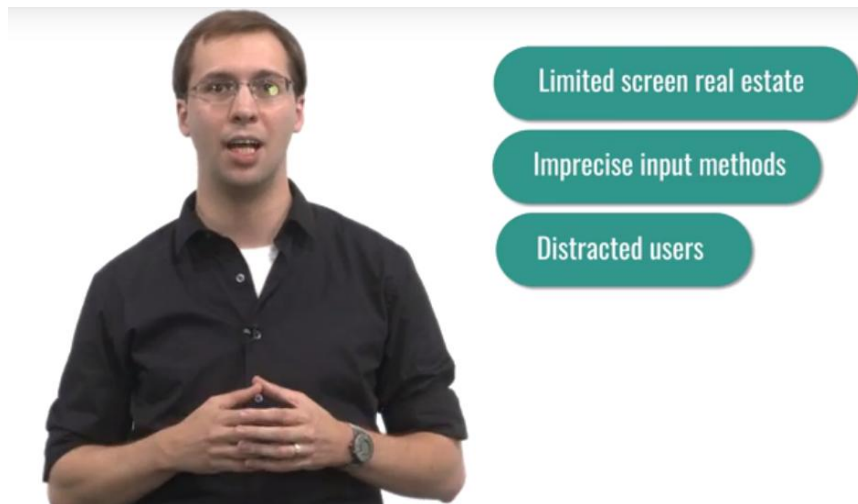
For example, how do we ensure that robots don't harm humans through faulty reasoning. How do we integrate robots into our social lives, or do we even need to? As robots are capable of more and more, how do we deal with the loss of demand for human work? Now these questions all lie at the intersection of HCI, artificial intelligence and philosophy in general. But there are some more concrete questions we can answer as well. How do we pragmatically equip robots with the ability to naturally

interact with humans based on things like voice and touch? How do we provide tasks that subtle feedback to humans interacting with robots to confirm their input is being received and properly understood? How do we support humans in teaching things to robots, instead of just programming them? Or alternatively, can we create robots that can teach things to humans? We already see robotics advances applied to things like healthcare and disability services. And I'm really excited to see where you take it next.

Technology: Mobile



One of the biggest changes to computing over the past several years has been the incredible growth of mobile as a computing platform. We really live in a mobile first world and that introduces some significant design challenges.



Screen real estate is now far more limited, the input methods are less precise and the user is distracted. But mobile computing also presents some really big opportunities for HCI. Thanks in large part to mobile we're no longer interested just in a person sitting in front of a computer. With mobile phones, most people have a computer with them at all times anyway. We can use that to support experiences from navigation to star gazing. Mobile computing is deeply related to fields like context aware computing, ubiquitous computing and augmented reality, as it possesses the hardware necessary to compliment those efforts. But even on its own, mobile computing presents some fascinating challenges to address.

Social networking
Personal organization
Games



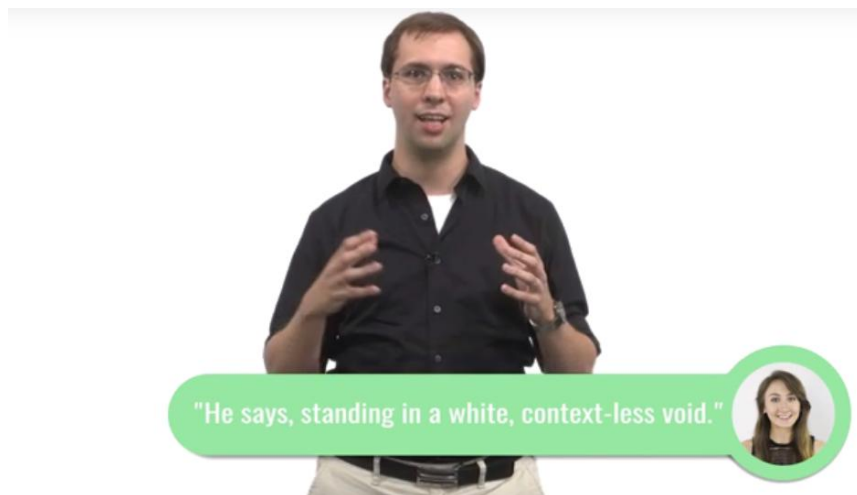
Writing essays
Programming
Video editing

For me, the big one is that we haven't yet reached a point where we can use mobile phones for all the tasks we do on computers. Smart phones are great for social networking, personal organization, games, and lots of other things. But we haven't yet reached a point where the majority of people would sit down to write an essay, or do some programming on smart phones. Why haven't we? What do we need to do to make smart phones into true replacements for traditional desktop and laptop computers?

Idea: Context-Sensitive Comp



What time is it? >> You can go ahead and go to lunch. >> Did that exchange make any sense? I asked Amanda for the time and she replied by saying I can go ahead and go get lunch. The text seems completely non-sensical and yet hearing that, you may have filled in the context that makes this conversation logical. You might think that I asked a while ago what time we were breaking for lunch, or maybe I mentioned that I forgot to eat breakfast. Amanda would have that context and she could use it to understand why I'm probably asking for the time. Context is a fundamental part of the way humans interact with other humans. Some lessons we'll talk about even suggest that we are completely incapable of interacting without context.



If context is such a pervasive part of the way humans communicate, then to build good interfaces between humans and computers, we must equip computers with some understanding of context.



That's where context-sensitive computing comes in. Context-sensitive computing attempts to give computer interfaces the contextual knowledge that humans have in their everyday lives. For example, I use my mobile phone differently depending on whether I'm sitting on the couch at home, or using it in my car, or walking around on the sidewalk. Imagine I didn't have to deliberately inform my phone of what mode I was in though. Imagine if it just detected that I was in my car and automatically brought up Google Maps and Audible for me. Services have started to emerge to provide this, but there's an enormous amount of research to be done on context sensitive computing. Especially as it relates to things like wearables, augmented reality, and ubiquitous computing.

Idea: Gesture-Based Interaction



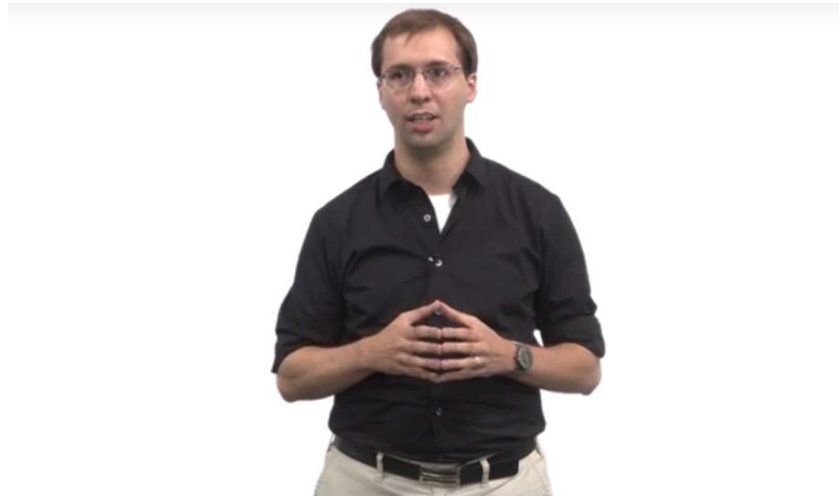
As this course goes on, you'll find that I'm on camera more often than you're accustomed to seeing in a Udacity course. Around half this course takes place with me on camera. There are a couple of reasons for that. The big one is that this is Human Computer Interaction. So it makes sense to put strong emphasis on the Human. But another big one is that when I'm on camera, I can express myself through gestures instead of just word and voice intonations. I can for example make a fist and really drive home and emphasize a point. I can explain that a topic applies to a very narrow portion of the field or a very wide portion of the field. We communicate naturally with gestures every day. In fact, we even have an entire language built out of gestures. So wouldn't it be great if our computers could interpret our gestures as well?



That's the emerging field of Gesture-Based Interaction. You've seen this with things like the Microsoft Connect which has far reaching applications from healthcare to gaming. We've started to see some applications of gesture based interaction on the go as well with wrist bands that react to certain hand motions. Gesture based interaction has enormous potential. The fingers have some of the finest muscle

movements, meaning that a system based on finger movements could support an incredible number of interactions. We might see a day when it's possible to type invisibly in the air in front of you based on system's recognition of the movement in the muscles of your wrist. That might finally allow mobile devices to replace traditional computers altogether.

Idea: Pen- and Touch-Based Interaction



I always find it interesting how certain technologies seem to come around full circle. For centuries we only interacted directly with the things that we built and then computers came along. And suddenly we needed interfaces between us and our tasks. Now, computers are trying to actively capture natural ways we've always interacted. Almost every computer I encounter now days has a touch screen. That's a powerful technique for creating simple user interfaces because it shortens the distance between the user and the tasks they're trying to accomplish. Think about someone using a mouse for the first time. He might need to look back and forth from the screen to the mouse, to see how interacting down here, change things he sees up here. With a touch based interface, he interacts the same way he uses things in the real world around him. A challenge can sometimes be a lack of precision, but to make up for that we've also create pen based interaction. Just like a person can use a pen on paper, they can also use a pen on a touch screen. And in fact, you might be quite familiar with that, because most Udacity courses use exactly that technology. They record someone writing on a screen. That gives us the precision necessary to interact very delicately and specifically with our task. And as a result tablet based interaction methods have been used in fields like art and music. Most comics you find on the internet are actually drawn exactly like this, combining the precision of human fingers with the power of computation.

Idea: Information Visualization



One of the biggest trends of the information age is the incredible availability of data. Scientists and researchers use data science and machine learning to look at lots of data and draw conclusions. But often times those conclusions are only useful if we can turnaround and communicate them to ordinary people. That's where information visualization comes in. Now at first glance you might not think of data visualization as an example of HCI. After all, I could draw a data visualization on a napkin and print in a newspaper and there's no computer involved anywhere in that process. But computers give us a powerful way to re-represent data in complex, animated, and interactive ways. We'll put links to some excellent examples in the notes. Now what's particularly notable about data visualization in HCI is the degree with which it fits perfectly with our methodologies for designing good interfaces. One goal of a good interface is to match the user's **mental model** to the reality of the task at hand. In the same way, the goal of information visualization is to match the reader's mental model of the phenomenon to the reality of it. So the same principles we discussed for designing good representations apply directly to designing good visualizations. After all, a visualization is just a representation of data.

Idea: CSCW



CSCW stands for Computer-Supported Cooperated Work. The field is just what the name says. How do we use computers to support people working together. You're watching this course online. So odds are that you've experienced this closely. Maybe you've worked on a group project with a geographically distributed group. Maybe you've had a job working remotely. Distributed teams are one example of CSCW in action but there are many others. The community often breaks things down into two dimensions.



Time and place. We can think of design as whether or not we're designing for the users in the same time and place or users at different times in different places. This course is an example of designing for different time and different place. You're watching this long after I recorded this, likely from far away from our studio. Work place chat utilities like slack and hipchat would be examples of same time, different place. They allow people to communicate instantly across space, mimicking the real-time

office experience. Now imagine a kiosk at a museum that asks visitors to enter their location to create a map of where everyone comes from. Now that would be different time, same place. Everyone uses the interface in the same place, but across time. And even when we're in the same time and place, computers can still support cooperation. In fact, right now, Amanda's running our camera, Ben's running the teleprompter and I'm standing up here talking at you. These different computers are supporting us in cooperating to create this course. So we can often think of CSCW as mediating cooperation across traditional geographic or temporal borders. But it can also help us with collocated simultaneous cooperation.

Idea: Social Computing



Social computing is the portion of HCI that's interested in how computers affect the way we interact and socialize. One thing that falls under this umbrella is the idea of recreating social norms within computational systems. So for example, when you chat online, you might often use emojis or emoticons. Those are virtual recreations of some of the tacit interaction we have with each other on a day-to-day basis. So, for example, these all take on different meanings depending on the emotion provided. Social computing is interested in a lot more than just emojis, of course.

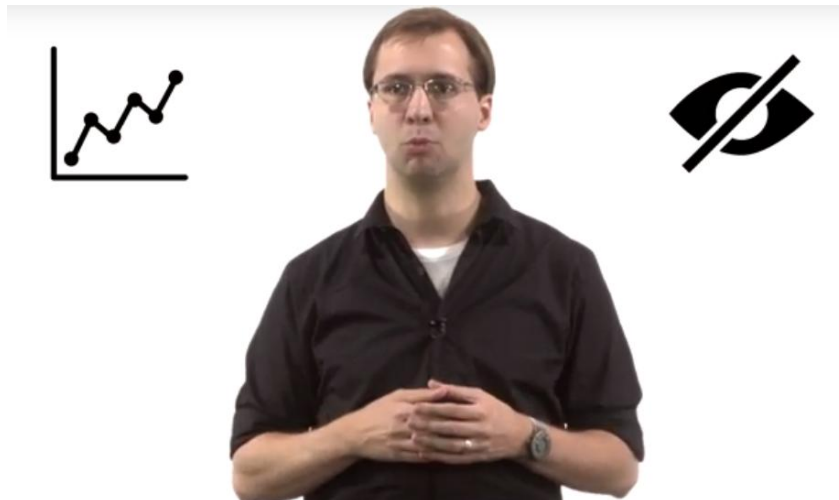


From online gaming and Wikipedia, to social media, to dating websites, social computing is really interested in all areas where computing intersects with our social lives.

Domain: Special Needs



One of the most exciting application areas for HCI is in helping people with special needs. Computing can help us compensate for disabilities, injuries, aging. Think of a robotic prosthetic, for example. Of course, part of that is engineering, part of it is neuroscience. But it's also important to understand how the person intends to use such a limb in the tasks they need to perform. That's HCI intersecting with robotics.



Or take another example from some work done here at Georgia Tech by Bruce Walker, how do you communicate data to a blind person? We've talked about informational visualization, but if it's a visualization, it's leaving out a significant portion of the population. So Dr. Walker's sonification lab works on communicating data using sound. A lot of the emerging areas of HCI technology could have extraordinary significance to people with special needs. Imagine virtual reality for people suffering from some form of paralysis. Or imagine using artificial intelligence with context-aware computing to

create an autonomous wheelchair. These are projects that would only target a small portion of the population, but the impact of that portion would be absolutely indescribable.

Domain: Education



Hi, and welcome to educational technology. My name is David Joyner and I'm thrilled to bring you this course.



As you might guess, education is one of my favorite application areas of HCI. In fact, as I'm recording this, I've been teaching educational technology at Georgia Tech for about a year, and a huge portion of designing educational technology is really just straightforward HCI. But education puts some unique twists on the HCI process. Most fascinatingly, education is an area where you might not always want to make things as easy as possible. You might use HCI to introduce some desirable difficulties, some learning experiences for students. But it's important to ensure that the cognitive loads students experience during a learning task is based on the material itself. Not based on trying to figure out our interfaces. The worst thing you can do in HCI for education is raise the student's cognitive load because they're too busy thinking about your interface instead of the subject matter itself. Lots of very noble

efforts in designing technology for education have failed due to poor HCI. So if you're interested in going into educational technology, you'll find a lot of valuable lessons in Human Computer Interaction.

Domain: Healthcare



A lot of current efforts in healthcare are about processing the massive quantities of data that are recorded everyday. But in order to make that data useful, it has to connect to real people at some point. Maybe it's equipping doctors with tools to more easily visually evaluate and compare different diagnoses. Maybe it's giving patients the tools necessary to monitor their own health and treatment options. Maybe that's information visualization so patients can understand how certain decisions affect their well-being. Maybe it's context aware computing that can detect when patients are about to do something they probably shouldn't do. There are also numerous applications of HCI to personal health like Fitbit for exercise monitoring or MyFitnessPal for tracking your diet. Those interfaces succeed if they're easily usable for users. Ideally, they'd be almost invisible. But perhaps the most fascinating upcoming intersection of HCI and health care is in virtual reality. Virtual reality exercise programs are already pretty common to make living an active lifestyle more fun, but what about virtual reality for therapy? That's actually already happening. We can use virtual reality to help people confront fears and anxieties in a safe, but highly authentic place. Healthcare in general is concerned with the health of humans. And computers are pretty commonly used in modern healthcare. So the applications of human computer interaction to healthcare are really huge.

Domain Security

Classes on network security are often most concerned with the algorithms and encryption methods that must be safeguarded to ensure secure communications. But the most secure communication strategies in the world are weakened if people just refuse to use them. And historically, we've found people have very little patience for instances where security measures get in the way of them doing their tasks. For security to be useful it has to be usable. If it isn't usable, people just won't use it. XEI can increase the usability of security in a number of ways. For one, it can make those actions simply easier to perform. CAPTCHAs are forms that are meant to ensure users are humans. And they used to involve recognizing letters in complex images, but now they're often as simple as a check-box. The computer recognizes human-like mouse movements and uses that to evaluate whether the user is a human. That makes it much less frustrating to participate in that security activity. But HCI can also make security more usable by visualizing and communicating the need. Many people get frustrated when systems require passwords that meet certain standards or complexity, but that's because it seems arbitrary. If the system instead expresses to the user the rationale behind the requirement, the requirement can be much less frustrating. I've even seen a password form that treats password selection like a game where you're ranked against others for how difficult your password would be to guess. That's a way to incentivize strong password selection making security more usable.

Domain: Games

Video games are one of the purest examples of HCI. They're actually a great place to study HCI, because so many of the topics we discuss are so salient. For example, we discussed the need for logical mapping between actions and effects. A good game exemplifies that. The actions that the user takes with the controller should feel like they're actually interacting within the game world. We discussed the power of feedback cycles. Video games are near constant feedback cycles as the user performs actions, evaluates the results and adjust accordingly. In fact, if you read through video game reviews you'll find that many of the criticisms are actually criticisms of bad HCI. The controls are tough to use, it's hard to figure out what happened. The penalty for failure is too low or too high. All of these are examples of poor interface design. In gaming though there's such a tight connection between the task and the interface. Their frustrations with a task can help us quickly identify problems with the interface.

Reflections: Exploring HCI

Throughout our conversations we are going to explore some of the fundamental principles and methods of HCI. Depending on the curriculum surrounding this material, you will complete assignments, projects, exams and other assessments in some of these design areas. However we'd also like you to apply what you learn to an area of your choice. So pick an area, either one we've mentioned here, or one you'd like to know about separately, and keep it in mind as we go through the course. Our hope is that by the end of the course you'll be able to apply what you learn here to the area in which you're interested in working.

Conclusion to Exploring HCI

In this lesson, our goal has been to give you an overview of the exciting expanse of ongoing HCI research and development. We encourage you to select a topic you find interesting, read about it a little bit more, and think about it as you go through the course. Then in unit four we'll provide some additional readings and materials on many of these topics for you to peruse. And in fact you can feel free to jump ahead to there now as well. But before we get too far into what we want to design, we first must cover the fundamental principles and methods of HCI.

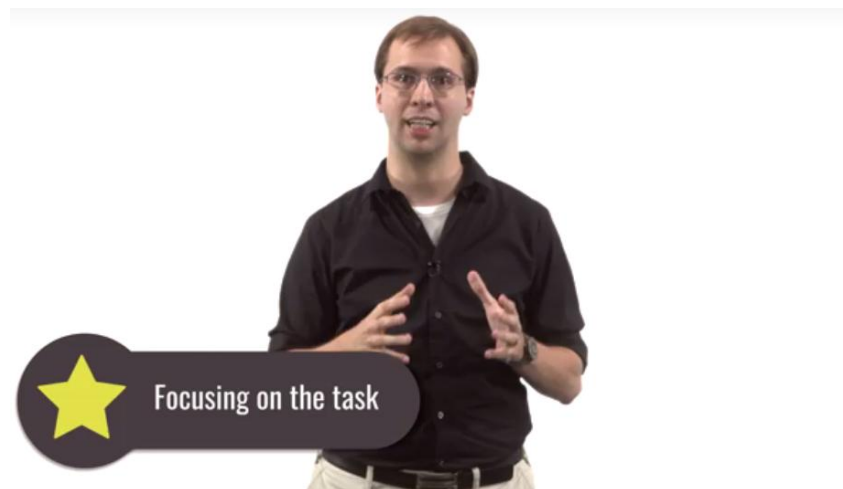
2.1 Introduction to Principles

Compiled by Shipra De, Summer 2017

Introduction to Design Principles



[MUSIC] For this portion of our conversation about human computer interaction, we're going to talk about some established principals that we'd uncovered after decades of designing user interfaces. We want to understand the fundamental building blocks of HCI, and separately we'll talk about how to build on those foundations to do new research and new development. To get started, though, let's first define some of the overarching ideas of design principles.



In this lesson, we're going to talk about the way we focus on users and tasks in HCI, not on tools and interfaces on their own.

(c) 2017 by David Joyner, all rights reserved.



We're going to talk about the role of the interface and how it mediates between user and the task.

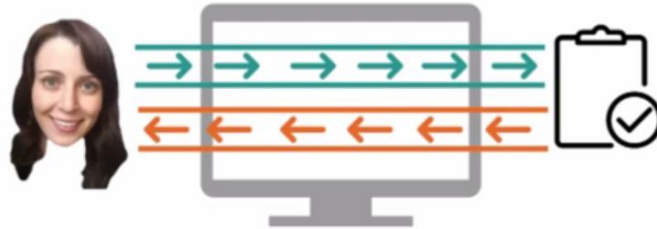


We're going to discuss different views on the user's role in the system.



And we're going to talk about user experience more generally and how it exists at several different levels. Along the way, we'll tackle some design challenges, reflect on our own experiences, and try to apply what we learn to the broader field of HCI.

Interfaces: Between Users and Tasks



At the heart of Human Computer Interaction is the idea that users use interfaces to accomplish some task. In general, that interface wouldn't actually have to be technological. This cycle exists for things like using pencils to write things or using a steering wheel to drive a car. But in HCI, we're going to focus on interfaces that are in some way computational or computerized. What's most important here though is our focus on the interaction between the user and the task through the interface, not just the interaction between the user and the interface itself. We're designing interfaces, sure, but to design a good interface, we need to understand both the user's goals and the tasks they're trying to accomplish.



Understanding the task is really important. One of the mistakes many novice designers make, is jumping too quickly to the interface, without understanding the task. For example, think about designing a new thermostat. If you focus on the interface, the thermostat itself, you're going to focus on things like the placement of the buttons or the layout of the screen, on whether or not the user can actually read what's there, and things like that. And those are all important questions. But the task is controlling the temperature in an area. When you think about the task rather than just the interface, you think of things like Nest, which is a device that tries to learn from its user and act autonomously. That's more than just an interface for controlling whether the heat or the air conditioning is on. That's an interface for controlling the temperature in your house. By focusing on the task instead of just the

interface, we can come up with more revolutionary designs like the Nest rather than just iterative improvements to the same thermostats we've been using for years.

Quiz: Identifying a Task



Let's try identifying a task real quick. We're going to watch a short clip of Morgan. Watch what she does, and try to identify what task she is performing. [MUSIC] What was the task in that clip?

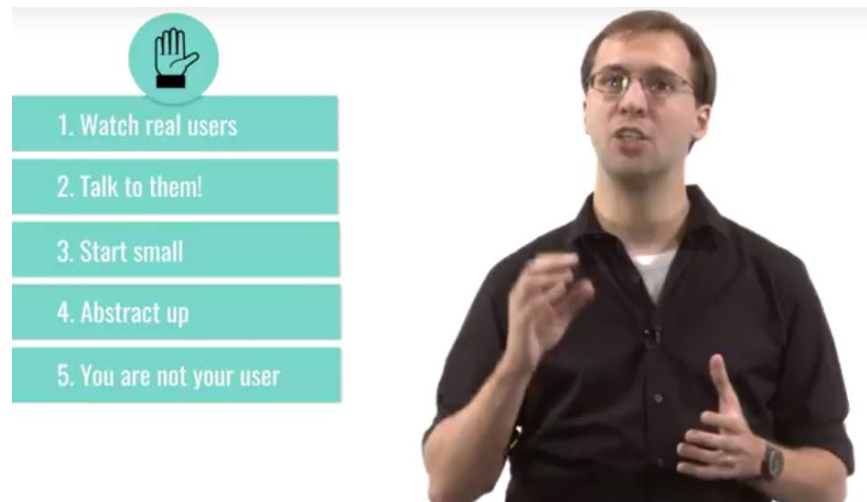


What was the task that Morgan was performing?

Making a purchase

If you said she's swiping her credit card, you're thinking a little too narrowly. Swiping her credit card is just how she accomplishes her task. We're interested in something more like she's completing a purchase. She's purchasing an item. She's exchanging goods. Those all put more emphasis on the actual task she's accomplishing and let us think more generally about how we can make that interface even better.

5 Tips: Identifying a Task



Here are five quick tips for identifying a user task. One: Watch real users. Instead just speculating or brainstorming, get out there and watch real users performing in the area in which you're interested. Two: Talk to them! You don't have to just watch them. Recruit some participants to come perform the task and talk their way through it. Find out what they're thinking, what their goals are, what their motives are. Three: Start small. Start by looking at the individual little interactions. It's tempting to come in believing you already understand the task, but if you do, you'll interpret everything you see in terms of what you already believe. Instead, start by looking at the smallest operators the user performs. Four: Abstract up. Working from those smaller observations, then try to abstract up to an understanding of the task they're trying to complete. Keep asking why they're performing these actions until you get beyond the scope of your design. For example: what is Morgan doing? Swiping a credit card. Why? To make a purchase. Why? To acquire some goods. Why? To repair her car. Somewhere in that sequence is likely the task for which we want to design. Five: You are not your user. Even if you yourself perform the task for which you're designing, you're not designing for you: you're designing for everyone that performs the task. So, leave behind your own previous experiences and preconceived notions about it. These five quick tips come up a lot in the methods unit of HCI. HCI research methods are largely about understanding users, their motivations, and their tasks. So, we'll talk much more about this later, but it's good to keep in mind now.

Usefulness and Usability



The ultimate goal of design in HCI is to create interfaces that are both useful and usable. Useful means that the interface allows the user to achieve some task. But usefulness is a pretty low bar. For example, a map is useful for finding your way from one place to another, but it isn't the most usable thing in the world. You have to keep track of where you are. You have to plot your own route. And you have to do all of this while driving the car. So before GPS navigation, people would often manually write down the turns before they actually started driving somewhere they hadn't been before. So our big concern is usability. That's where we get things like navigation apps. Notice how we have to focus on understanding the task when we're performing design. If we set out to design a better map, we probably wouldn't have ended up with a navigation app.

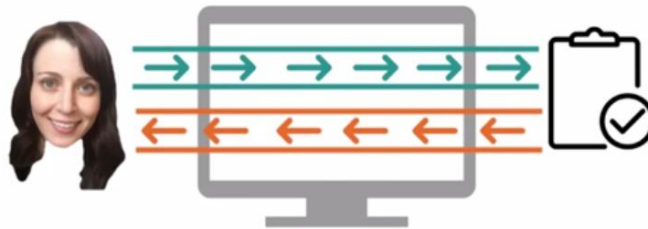


It was through understanding the task of navigation itself that we realized we could offload a lot of the cognitive load of navigation onto the interface, closing the loop between the user and the task of navigation.

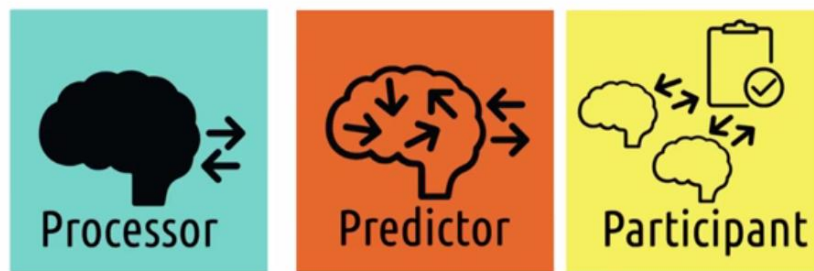
Exploring HCI: HCI Principles

Throughout this unit I've repeatedly asked you to revisit the area of HCI that you chose to keep in mind throughout our conversations. Now take a second and try to pull all those things together. You've thought about how your chosen area applies to each of the models in the human's role, how it applies to the various different design guidelines, and how it interacts with society and culture as a whole. How does moving it through those different levels change the kinds of designs you have in mind? Are you building it from low level interactions to high level effects? Are you starting at the top with a desired outcome and working your way down to the individual operations? There are no right or wrong answers here. The important thing is reflecting on your own reasoning process.

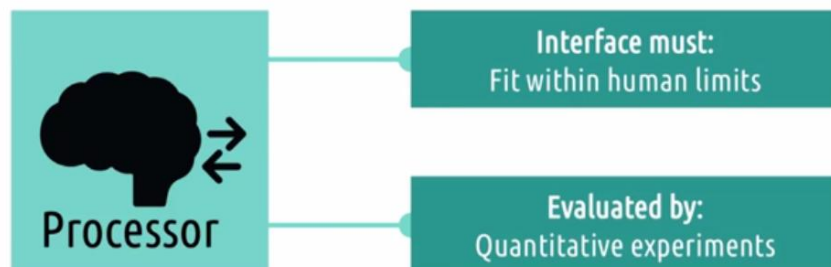
Views of the User: Processor



In looking at human-computer interaction, it's important that we understand the role that we expect the human to play in this overall system.



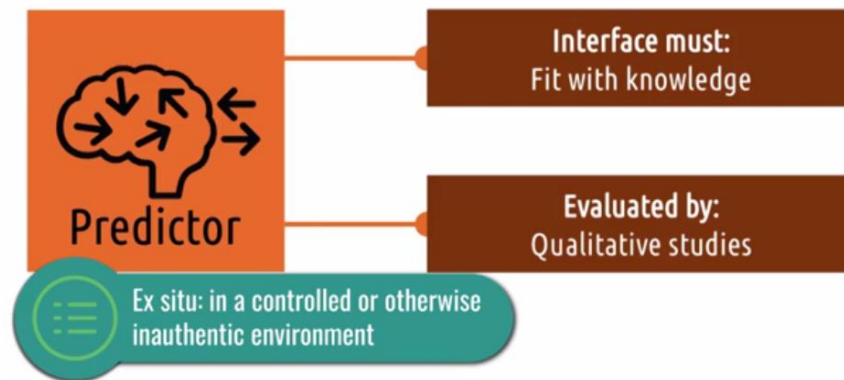
Let's talk about three different possible types of roles the human can play, processor, predictor, and participant.



First, we might think of the human as being nothing more than a sensory processor. They take input in and they spit output out. They're kind of like another computer in the system, just one that we can't

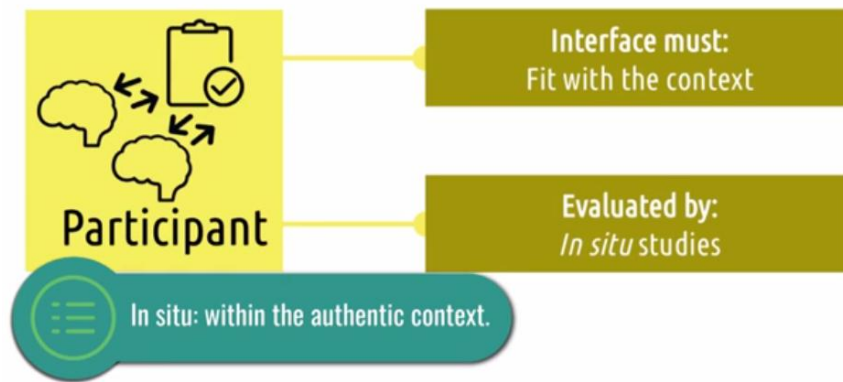
see the inside of. If we are designing with this role in mind then our main concern is that the interface fit within known human limits. These are things like what humans can sense, what they can store in memory, and what they can physically do in the world. In this case, usability means that the interface is physically usable. User can see all the colors, touch all the buttons, and so on. With this model, we evaluate our interfaces with quantitative experiments. That means we take numeric measurements on how quickly the user can complete some task or how quickly they might react to some incoming stimulus. Now, as you might have guessed, the processor view is not the one we'll generally take when we talk about good design. Instead, we'll probably divide our time pretty equally between the other two perspectives.

Views of the User: Predictor



A second way of viewing the human is to view them as a predictor. Here, we care deeply about the human's knowledge, experience, expectations, and their thought process. That's why we call them the predictor. We want them to be able to predict what will happen in the world as a result of some action they take. So we want them to be able to map input to output. And that means getting inside their head. Understanding what they're thinking, what they're seeing, what they're feeling when they're interacting with some task. If we're taking this perspective, then the interface must fit with what humans know. It must fit with their knowledge. It must help the user learn what they don't already know and efficiently leverage what they do already know. And toward that end, we evaluate these kind of interfaces with qualitative studies. These are often ex situ studies. We might perform task analyses to see where users are spending their time. Or perform cognitive walk-throughs to understand the user's thought process throughout some task. We can see pretty clearly that this view gives us some advantages over viewing the user simply as a sensory processor, just as another computer in the system. However, here we're still focusing on one user and one task. And sometimes that's useful. But many times we want to look even more broadly than that. That's when we take the third participant peel.

Views of the User: Participant



A third view on the user is to look at the user as a participant in some environment. That means we're not just interested in what's going on inside their head. We're also interested in what's going on around them at the same time, like what other tasks or interfaces they're using, or what other people they're interacting with. We want to understand for example, what's competing for their attention? What are their available cognitive resources? What's the importance of the task relative to everything else that's going on? So if we take this view, then our interface must fit with the context. It's not enough that the user is able to physically use the system and knows how to use the system. They must be able to actually interact with the system in the context where they need it. And because context is so important here, we evaluate it with in situ studies. We can't simply look at the user and the interface in a vacuum. We have to actually view and evaluate them in the real world using the interface in whatever context is most relevant. If we're evaluating a new GPS application, for example, we need to actually go out and look at it in the context of real drivers driving on real roads. The information we get from them using the app in our lab setting isn't as useful as understanding how they're going to actually use it out in the real world. These are in situ studies, which are studies of the interface and the user within the real complete context of the task.

Good Design, Bad Design



[MUSIC] Good design, a GPS system that warns you 20 seconds before you need to make a turn.

>> In 1,000 feet, turn left.



>> Bad design, a GPS system that warns you two seconds before you need to make a turn.

>> Turn left now, hurry. [SOUND]



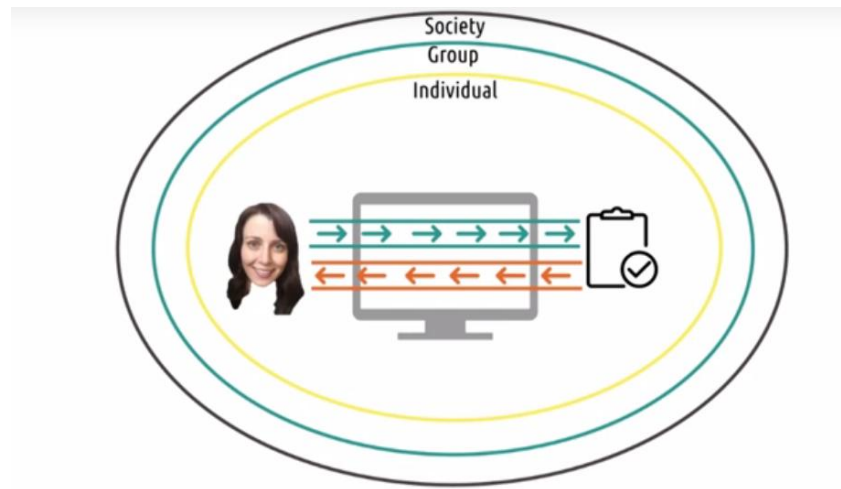
It sounds funny, but which view you take on the user can have a huge impact on the success of the interface. If you view the user just as a sensory processor, you might think that we only need to alert them a second before the upcoming turn because, after all, human reaction time is less than a second. If you view the user as a predictor, you understand they need time to slow the car down and actually make the turn. So they might need a few more seconds to execute the action of turning before being alerted they need to turn. And if you view the user as a participant, you'll understand this is happening while they're going 50 miles down the road with a screaming toddler in the backseat, trying to merge with the driver on a cell phone and the other one eating a cheeseburger. So it would probably be a good idea to give them a few or more reminders before the turn and plenty of time to get in the right position.

Quiz: Reflections: Views of the User

Let's take a moment to reflect on when you've encountered these different views of the user in your own history of interacting with computers. Try to think of a time when a program, an app or a device clearly treated you as each of these types of users for better or for worse.

For me, we have a system at Udacity we use to record hours for those of us that work on some contract projects. It asks us to enter the number of hours of the day we spend on each of a number of different types of work. The problem is that, that assumes something closely resembling the processor model. A computer can easily track how long different processes take. But for me, checking the amount of time spent on different tasks can be basically impossible. Checking my e-mails involves switching between five different tasks a minute. How am I suppose to track that? The system doesn't take into consideration a realistic view of my role in the system. Something more similar to the predictor view would be, well, the classroom you're viewing this in. Surrounding this video are a visual organization of the lesson's content, a meter measuring your progress through the video, representations of the video's transcript. These are all meant to equip you with the knowledge to predict what's coming next. This classroom takes a predictor view of the user. It offloads some of the cognitive load onto the interface allowing you to focus on the material. For the third view I personally would consider my alarm clock an example. I use an alarm clock app called Sleep. It monitors my sleep cycles, rings at the optimal time and tracks my sleep patterns to make recommendations. It understand its role as part of a broader system needed to help me sleep. It goes far beyond just interaction between me and an interface. It integrates into the entire system.

User Experience, Sans Design



By my definition, user experience design is attempting to create systems that dictate how the user will experience them. Preferably that the user will experience them positively. User experience in general, though, is a phenomenon that emerges out of the interactions between humans and tasks via interfaces. We might attempt to design that experience. But whether we design it or not, there is a user experience. It's kind of like the weather, there's never no weather, there's never no user experience. It might be a bad experience if we don't design it very well. But there's always some user experience going on and it emerges as a result of the human's interactions with the task via the interface. But user experience also goes beyond this simple interaction. It touches on the emotional, personal, and more experiential elements of the relationship. We can build this by expanding our understanding of the scope of the user experience. For just a particular individual, this is based on things like the individual's age, sex, or race, personal experiences, gender, expectations for the interface, and more. It goes beyond just designing an interface to help with a task. It touches on whether the individual feels like the interface was designed for them. It examines whether they're frustrated by the interface or joyous about it. Those are all parts of this user experience. We can take this further and talk about user experience at a group level. We can start to think about how interfaces lead to different user experiences among social or work groups. For example, I've known that school reunions seem to be much less important to people who've graduated within the past 15 years. And I hypothesize it's because Facebook and email have played such significant roles in keeping people in touch. It's fundamentally changed the social to group user experience. Those effects can then scope all the way up to the societal level. Sometimes these are unintended. For example, I doubt that the creators of Twitter, foresaw when they created their tool, how it would play a significant role in big societal changes like the Arab spring or, sometimes these might be intentional. For example, it was a significant change when Facebook added new relationship statuses to its profiles to reflect things like civil unions. That simultaneously reflected something that was already changing at the societal level. But it also participated in that change and helped normalize those kinds of relationships. And that then relates back to the individual by making sure the interface is designed such that each individual feels

like it's actually designed with them in mind. The options are there for them to feel like they're properly represented within the system. These are all components of the general user experience that we need to think about as we design interfaces.

Quiz: Design Challenge: Morgan on the Street



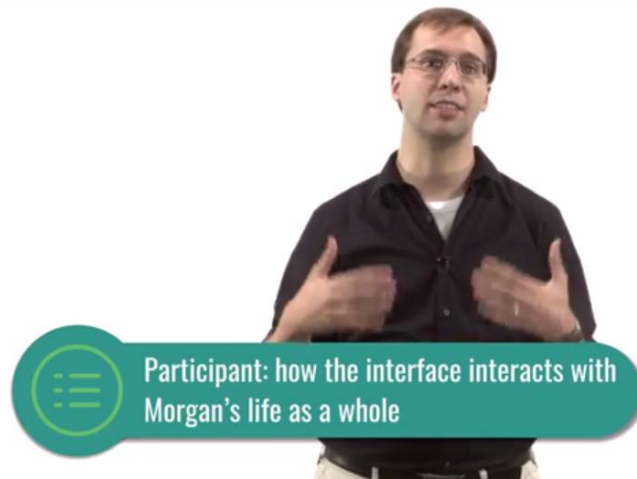
So keeping in mind everything we've talked about, let's design something for Morgan. Morgan walks to work, she likes to listen to audio books, mostly nonfiction. But she doesn't just want to listen, she wants to be able to take notes and leave bookmarks. And do everything else you do when you're reading. What would designing for her look like, from the perspectives of viewing her as a processor, a predictor, and a participant? How much this different designs affect user experience as an individual, in her local group of friends, and the society as a whole if the design caught on.



As a processor, we might simply look at what information is communicated to Morgan, when, and how.



As a predictor, we might look instead at how the interface meshes with Morgan's needs with regard to this task, how easy it is to access, how easy the commands are to perform, and so on.



As a participant, we might look at the broader interactions between this interface and Morgan's other tasks and social activities. You might look at how increased access to books changes her life in other ways. But really, this challenge is too big to address this quickly. So instead, let's return to this challenge throughout our conversations, and use it as a running dialogue to explore HCI principles and methods.

Conclusion to Intro to Design



In this lesson we've covered some of the basic things you need to understand before we start talking about design principles.



We've covered the idea that interface is mediated between users and tasks. And the best interfaces are those that let the user spend as much time thinking about the task as possible.



We covered the idea of usability and how we have to keep in mind the efficiency and user satisfaction of the interface.



We covered three views of the user and how those different views affect how we define usability and evaluation.

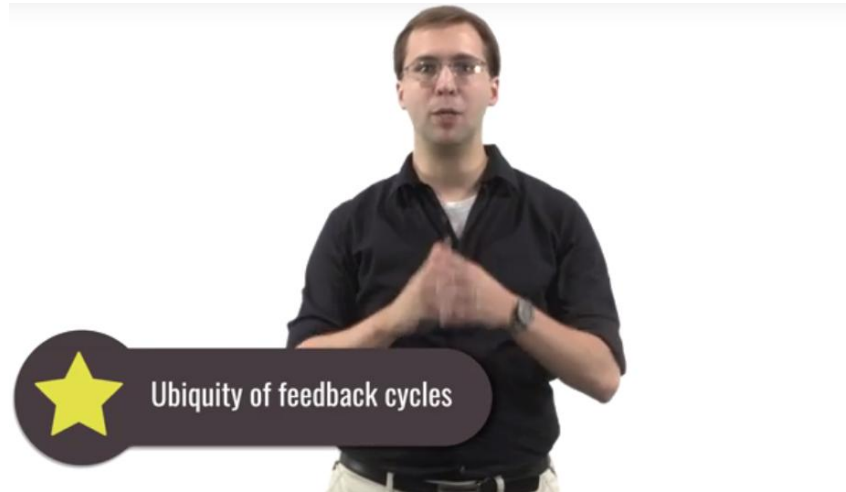


We covered how the user experience is not just at the user level, but also at group and even societal levels.

2.2 Feedback Cycles

Compiled by Shipra De, Summer 2017

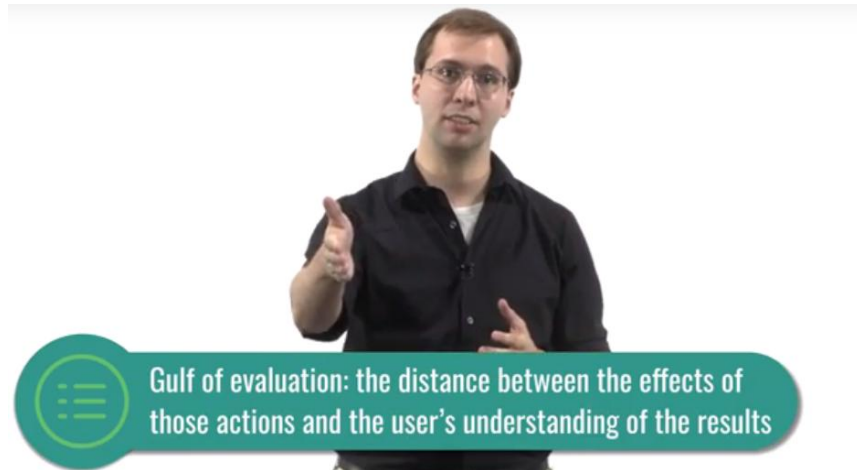
Introduction to Feedback Cycles



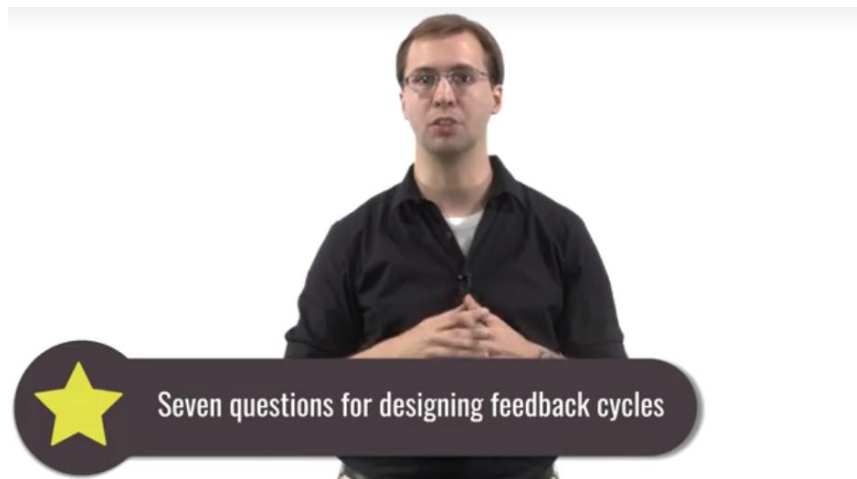
[NOISE] Feedback cycles are the way in which people interact with the world, and then get feedback on the results of those interactions. We'll talk about the ubiquity of those feedback cycles.



Then we'll talk about the gulf of execution, which is the distance between a user's goals and the execution of the actions required to realize those goals.



Then we'll talk about the Gulf of evaluation, which is the distance between the effects of those actions and the user's understanding of those results.



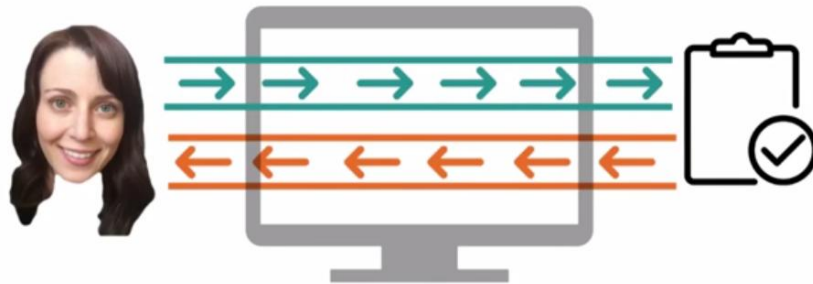
We'll discuss seven questions we should ask ourselves when designing feedback cycles for users.



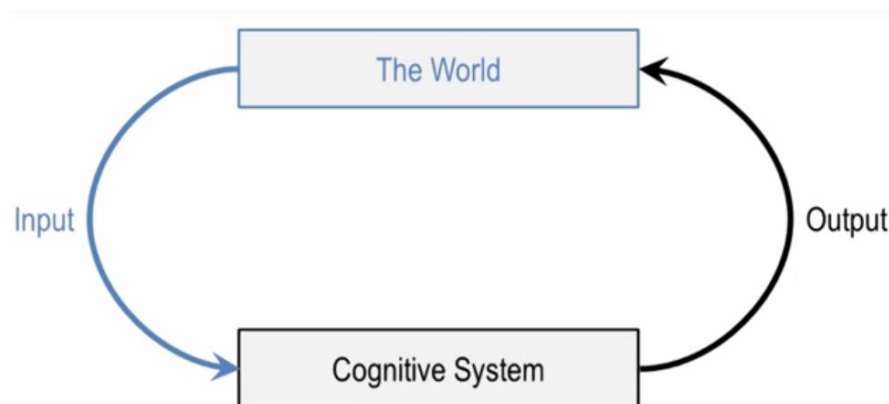
And we'll also look at applications of these in multiple areas of our everyday lives.

(c) 2017 by David Joyner, all rights reserved.

Feedback Cycles are Fundamental

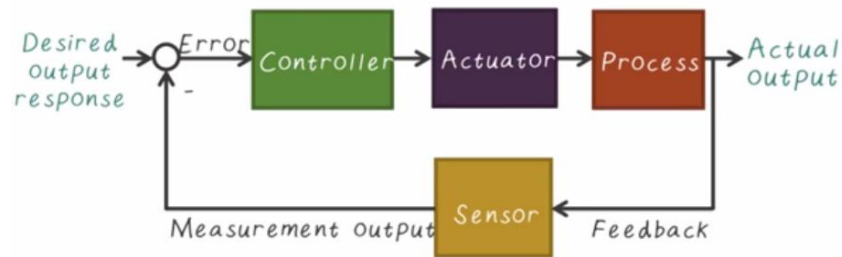


Feedback cycles are incredibly ubiquitous, whether or not there's a computational interface involved. Everything from reading to driving a car to interacting with other people could be an example of a feedback cycle in action. They're how we learn everything, from how to walk to how to solve a Rubik's cube to how to take the third order partial derivative of a function. I assume, I've never done that. We do something, we see the result, and we adjust what we do the next time accordingly. You may have even seen other examples of this before, too.



If you've taken Ashok's and mine knowledge-based AI class, we talk about how agents are constantly interacting with, learning from, and affecting the world around them. That's a feedback cycle.

CLOSED-LOOP CONTROL SYSTEM



If you've taken Rahim Baez's cyber physical systems course, you've seen this without human involved at all, as a system can autonomously read input and react accordingly. Under some definitions, some people would even call this the artificial intelligence, specifically because it mimics what a human actually does. They act in the world and they evaluate the result.

A person possesses intelligence insofar as he has learned, or can learn, to adjust himself to his environment. S. S. Colvin

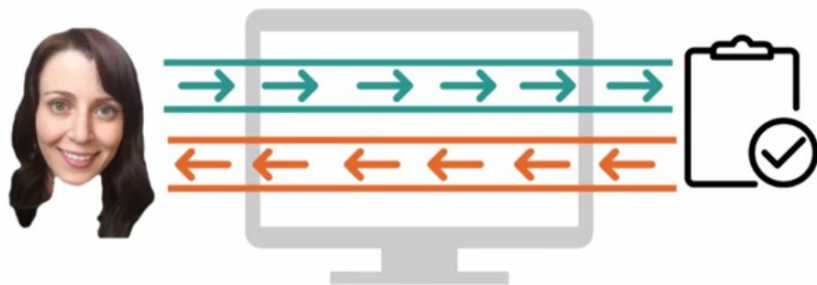
The capacity to learn or to profit by experience. W. F. Dearborn

Any system that generates adaptive behavior to meet goals in a range of environments. D. Fogel

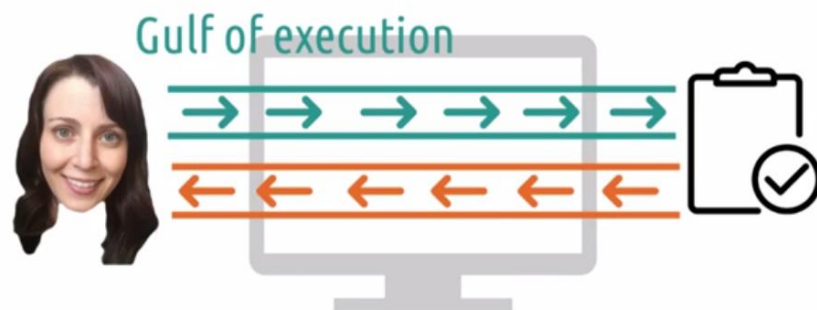
Intelligence means getting better over time. R. Schank

In fact, if you look at some of the definitions of intelligence out there, you'll find that many people actually define feedback cycles as the hallmark of intelligent behavior. Or they might define intelligence as abilities that must be gained through feedback cycles. Colvin's definition, for example, involves adjusting to one's environment, which means acting in it and then evaluating the results. Dearborn's definition of learning or profiting by experience is exactly this as well. You do something and experience the results, and learn from it. Adaptive behavior in general can be considered an example of a feedback cycle. Behavior means acting in the world. And adapting means processing the results and changing your behavior accordingly. And most generally, Schank's definition is clearly an ability gained through feedback cycles, getting better over time based on evaluation of the results of one's actions in the world. And Schank's general definition, getting better over time, is clearly something that can happen as a result of participation in a feedback cycle. We find that nearly all of HCI can be interpreted in some ways as an application of feedback cycles, whether between a person and a task, a person and an interface, or systems comprised of multiple people and multiple interfaces.

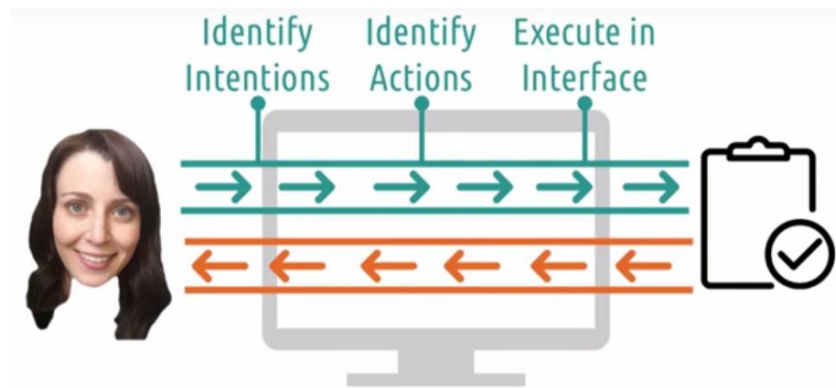
Gulf of Execution



In our feedback cycle diagram, we have on the left, some user and on the right, some task or system. The user puts some input into the system through the interface and the system communicates some output back to the user again through the interface. Incumbent on this are two general challenges, the user's interaction with the task through the interface and the task's return to the user of the output via the interface.



The first is called the Gulf of execution. The Gulf of execution can be defined as how do I know what I can do. The user has some goals. How do they figure out how to make those goals a reality? How do they figure out what actions to take to make the state of the system match their goal state? This is the Gulf of execution. How hard is it to do in the interface what is necessary to accomplish the users' goals? Or alternatively, what's the difference between what the user thinks they should have to do and what they actually have to do.



Now there are a number of components of this. The first component, they need to be able to identify what their goal is in the context of the system. There might be a mismatch between their own understanding and the system's structure. Think of transitioning from an old-fashioned VCR to a more modern DVR or from a DVR to watching things on-demand. The user needs to think of their goal in terms of their current system. Second, they need to be able to identify the actions necessary to accomplish their goals. Now that they know what their goal is in the context of the system, they need to identify the actions that it will take to make that goal a reality. And third, once I've identified those actions, they need to actually execute the actions within the interface. Again, imagine someone who's learning to use an on demand video interface, when they're used to using things like VCRs and DVRs. Their goal hasn't changed. They want to watch some program that's already aired. But in the context of a VCR or a DVR, their intention might be to record that program. In the context of an on demand video interface, their intentions instead are to call up the existing version of that program. That's a mismatch between what they think their goal is and what their goal is in the context of this new system. But once they understand what the goal means in their current system, they now need to know how to pull up that program. They need to know how to navigate the menus and find the program that they want to watch and then start it playing. And then once they know what to do, they need to actually execute that series of button presses. For example, they might know what actions to perform but they might not know where to find them. That would present a difficulty in executing those actions. So the gulf of execution takes the user from understanding their own goals to understanding their goals in the context of the system, to understanding the actions necessary to realize those goals, to actually executing those actions. And each of these presents some difficulties.

Gulf of Execution Example



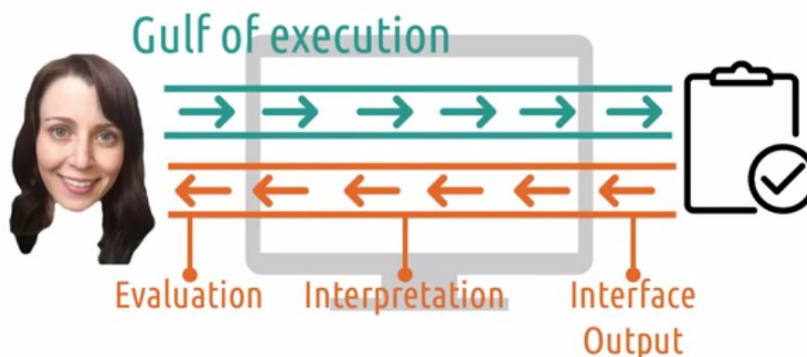
Let's take a simple example of the gulf of execution. I'm making my lunch, I have my bowl of chili in the microwave. My goal is simple, I want to heat it up. How hard is that? Well, typically when I've been cooking in the past, cooking is defined in terms of the amount of time it takes. So, in the context of this system, I specify my intent as to microwave it for one minute. Now what are the actions necessary to do so? I press Time Cook to enter the time-cooking mode, I enter the time, one minute, and I press Start. I didn't press Start just now, but I would press Start. I specified my intent, microwave for one minute. I specified my actions, pressing the right sequence of buttons, and I executed those actions. Could we make this better? There were a lot of button presses to microwave for just one minute. If we think that's a common behavior, we might be able to make it simpler. Instead of pressing Time Cook one, zero, zero and Start, I might just press one and wait. Watch. [NOISE] So I've narrowed the gulf of execution by shrinking the number of actions required, but I may have enlarged it by making it more difficult to identify the actions required. When I look at the microwave, Time Cook gives me an idea of what that button does. So if I'm a novice at this, I can discover how to accomplish my goal. That's good for the gulf of execution. It's easier to look at the button and figure out what to do than to have to go look, read a manual, or anything like that and find out on your own. But once you know that all you have to do is press one, that's much easier to execute. That's something nice about this interface, it caters to both novices and experts, there's a hard and discoverable way and a short and visible way. But let's rewind all the way back to the goal I set up initially, my goal was to heat up my chili. I specified my intent in terms of the system as microwaving it for one minute. But was that the right thing to do? After one minute, my chili might not be hot enough, this microwave actually has an automatic reheat function that senses the food's temperature and stops when the time seems right. So the best bridge over the gulf of execution might also involve helping me reframe my intention. Instead of going to microwave for one minute, it might encourage me to reframe this as simply heating until ready and letting the microwave do the rest.

5 Tips: Gulfs of Execution



Here are five quick tips for bridging gulfs of execution. Number 1, make functions discoverable. Imagine a user is sitting in front of your interface for the very first time. How would they know what they can do? Do they have to read the documentation, take a class? Ideally the functions of the interface would be discoverable, meaning that they can find them clearly labelled within the interface. Number 2, let the user mess around. You want your user to poke around and discover things so make them feel safe in doing so. Don't include any actions that can't be undone, avoid any buttons that can irreversibly ruin their document or setup. That way the user will feel safe discovering things in your interface. Number 3, be consistent with other tools. We all want to try new things and innovate, but we can bridge gulfs of execution nicely by adopting the same standards that many other tools use. Use CTRL+C for copy and CTRL+V for paste. Use a diskette icon for save even though no one actually uses floppy disks anymore. This makes it easy for users to figure out what to do in your interface. Number 4, know your user. The gulf of execution has a number of components, identifying intentions, identifying the actions to take, and taking the actions. For novice users, identifying their intentions and actions are most valuable, so making commands discoverable through things like menus is preferable. For experts though, actually doing the action is more valuable. That's why many experts prefer the command line. Although it lacks many usability principles targeted at novices, it's very efficient. Number 5, feed forward. We've talked about feed back, which is a response to something the user did. Feed forward is more like feed back on what the user might want to do. It helps the user predict what the result of an action will be. For example, when you pull down the Facebook newsfeed on your phone, it starts to show a little refresh icon. If you don't finish pulling down, it doesn't refresh. That's feedforward, information on what will happen if you keep doing what you're doing. Many of these tips are derived from some of the fundamental principles of design pioneered by people like Don Norman and Jakob Nielsen, and we'll cover them more in another lesson.

Gulf of Evaluation



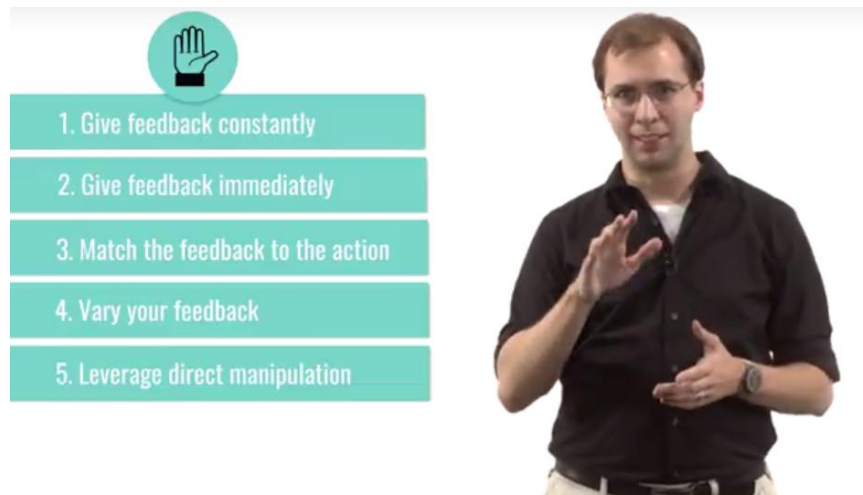
The second challenge is for the task to express to the user through the interface the output of the actions that the user took. This is called the gulf of evaluation because the user needs to evaluate the new state of the system in response to the actions they took. Like the gulf of execution, we can think of this in terms of three parts. There's the actual physical form of the output from the interface. What did it actually do in response? There might be something visual, there might be a sound, a vibration, some kind of output. The second is interpretation. Can the user interpret the real meaning of that output? You might think of this in terms of a smartphone. If a smartphone vibrates in your pocket, can you interpret what the meaning of that output was or you have to pull the phone out and actually see? And then the third phase is evaluation. Can the user use that interpretation to evaluate whether or not their goals were accomplished? You can imagine submitting a form online. It might give you output that you interpret to mean that the form was received, but you might not be able to evaluate whether or not the form was actually accepted. Once they've received and interpreted that output, the final step is to evaluate whether or not that interpretation means that their goals were actually realized within the system. Take our on demand video service example again, imagine that the user has gotten all the way to finding the program that they want to watch and they've pressed the play button on their remote. Imagine the interface responds by hiding the menus that they were using to navigate amongst the service. Can they interpret the meaning of that output? And can they evaluate whether or not that interpretation means that their goals were realized? If they're a novice user, maybe not. An expert might correctly interpret that the screen blacking out is because the service is trying to load the video. They then evaluate that interpretation and determine that their goals have been realized. The service is trying to play the show they want to watch. But, a novice user might interpret that output to mean that the service has stopped working at all, like when your computer just shuts down and the screen goes black. They then incorrectly evaluate that their goals were not actually realized in the system. We might get over this by showing some kind of buffering icon. That's a different kind of output from the system that helps the user correctly interpret that the system is still working on the actions that they put in. They then can evaluate that maybe their goals were correctly realized after all because the system is still working to bring up their show. So, as you can see, each of these three stages presents some unique challenges.

Gulf of Evaluation Example



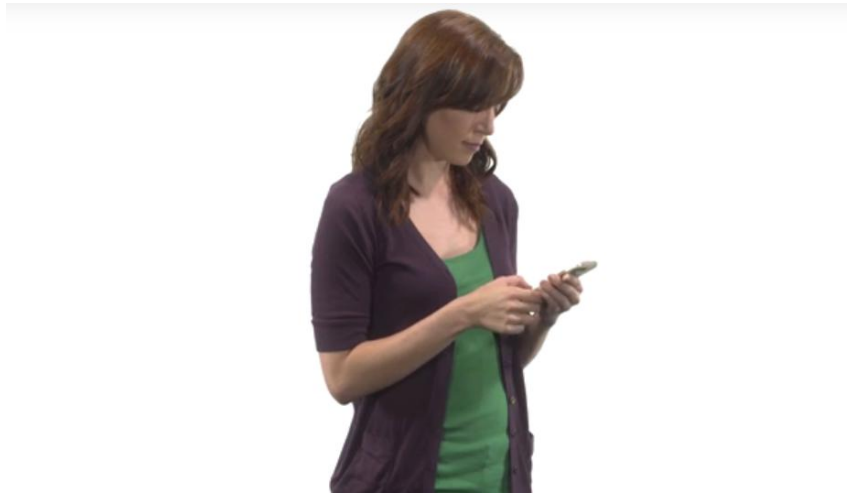
Let's take a thermostat, for example. I have a goal to make the room warmer, so I do something to my thermostat with the intention of making the room warmer. What does the system do as a result? Well, it turns the heat on, that would be the successful result of my action. But how do I know that the heat was turned on? Well, maybe I can hear it, I might hear it click on. But that's a one time kind of thing and it might be quiet. And if I'm mishearing it, I have no way of double checking it. So I'm not sure if I heard it, and I have to go find a vent and put my hand on it and try to feel the heat coming out. And there's more going on in a heater, it might have worked, but the heater doesn't immediately turn on for one reason or the other. These are signs of a large gulf of evaluation. Neither the sound or the vent are optimal displays because they're either hard to reach or possible to miss. Feeling the heat might be easy to interpret, but hearing the heater turn on might not. So either way, I have to do a lot to evaluate whether or not my action was successful. And this is all for a very small piece of feedback. Ideally if I wasn't successful, we want the system to also tell me why I wasn't successful so I can evaluate what I did wrong and respond accordingly. There's a very large gulf of evaluation if there's no indicator on the actual thermostat. So how can we resolve that? Well, simple. We just mark on the thermostat that the heat is on. That sounds trivial, but nothing in the fundamental design of this system demanded a note like this. It's only in thinking about the system from the perspective of the user that we find that need. I can let you know as well, this system still isn't very ideal. For various reasons, it'll turn the heater on or the air conditioning off even when it hasn't reached the temperature I put in. And it gives me no indication of why. I can look at the system and evaluate that the temperature is set to lower than the current temperature in the room. But at the same time, I can see that the heater isn't on. Under those circumstances, I have no way of knowing if the heater's malfunctioning, if the switch is wrong, or I don't even know. In this case, it might just be that it's set to the wrong mode. The mode is visible, but after I remembered to check it, it appears to be malfunctioning. We can imagine an alternative message on the screen indicating the direction of the relationship or something similar that would give some sign that it's currently set incorrectly.

5 Tips: Gulfs of Evaluation



Here are five quick tips for bridging gulfs of evaluation. Number one, give feedback constantly. Don't automatically wait for whatever the user did to be processed in the system before giving feedback. Give them feedback that the input was received. Give them feedback on what input was received. Help the user understand where the system is in executing their action by giving feedback at every step of the process. Number two. Give feedback immediately. Let the user know they've been heard even when you're not ready to give them a full response yet. If they tap an icon to open an app, there should be immediate feedback just on that tap. That way, even if the app takes awhile to open, the user knows that the phone recognized their input. That's why icons briefly grey out when you tap them on your phone. Number three. Match the feedback to the action. It might seem like this amount of constant immediate feedback would get annoying and if executed poorly it really would. Subtle actions should have subtle feedback. Significant actions should have significant feedback. Number four, vary your feedback. It's often tempting to view our designs as existing solely on a screen and so we want to give the feedback on the screen. But the screen is where the interaction is taking place, so visual feedback can actually get in the way. Think about how auditory or haptic feedback can be used instead of relying just on visual feedback. Number five, leverage direct manipulation. We talk about this a lot more, but whenever possible, let the user feel like they're directly manipulating things in the system. Things like dragging stuff around or pulling something to make it larger or smaller are very intuitive actions. Because they feel like you're interacting directly with the content. Use that. Again, we talk far more about this in another lesson, but it's worth mentioning here as well. By loading these things into your short-term memory several times, we hope to help solidify them in your long-term memory. And that relationship is actually something we also talk about elsewhere in this unit.

Good Design, Bad Design: Feedback Cycles



[MUSIC] Good design. A phone that quietly clicks every time a letter is successfully pressed to let you know that the press has been received. Bad design. A phone that loudly shouts every letter you type.

>> P. I. C.

Remember small actions get small feedback. The only time you might want your device to yell a confirmation at you is, if you'd just ordered a nuclear launch or something.

Quiz: Reflections: Feedback Cycles



Let's pause for a second, and reflect on the roles of gulfs of execution and gulfs of evaluation in our own lives. So try to think of a time when you've encountered a wide gulf of execution, and a wide gulf of evaluation. This doesn't have to be a computer, it could be any interface. In other words, what was a time when you were interacting with an interface, but couldn't think of how to accomplish what you wanted to accomplish? What was a time when you were interacting with an interface and couldn't tell if you'd accomplished what you wanted to accomplish?



It's not a coincidence that I'm filming this in my basement. This actually happened to me a few weeks ago. The circuit to our basement was tripped, which is where we keep our modem, so our internet was out. Now this is a brand new house and it was the first time we tripped a breaker, so I pulled out my flashlight and I opened the panel. And none of the labels over here clearly corresponded to the breaker I was looking for over here. I ended up trying every single one of them and still it didn't work. I shut off everything in the house. Why didn't it work? In reality, there was a reset button on the outlet itself that had to be pressed. The only reason we noticed it was because my wife noticed something out of

the corner of her eye turning on and off as I switched these. That was a terribly large gulf of execution. I knew what I wanted to accomplish, I could translate it into the system's terms easily, reset a breaker. But figuring out the actions to accomplish that goal was very difficult. That's a large gulf of execution. How was that? >> [SOUND] What? Sorry, I wasn't paying attention. >> You weren't watching? [LAUGH] So I have no way of knowing if that was good or not? Isn't that a terrible gulf of evaluation? I joke, but a lack of feedback on your performance at a task, whether it be filming, like I'm doing now or doing a project like you'll do later in our material, presents the same kind of poor gulf of evaluation.

Quiz: Feedback Cycles in David's Car 1



Why is the start button located here?

- It's closest to the engine, meaning the car will start faster.
- It's the only open place on the car after the other devices are inserted.
- It's where the user expects the button to be.
- This location avoids the car getting accidentally turned off or on.

15 years ago we might not have talked about cars in the context of discussing HCI, but nowadays this is basically a computer on wheels. So let's talk a little bit about how feedback cycles apply here. Let's start with the ignition. The button that I start my car is right here. Why is it located there?



Before cars had push button starts, this is where you inserted the key to turn on the ignition. Why? I have no idea. But I do know that now, the start button can be placed in any number of different locations. So why do we put it where we've always put it? Well, the reason is, that's where the driver expects it to be placed. We help them across the gulf of execution by designing a system that's consistent with their expectations about how it should work. It makes it easier for them to translate their intentions into actions. Now, other times we might violate this principle because of some other benefits we hope to gain. But generally speaking, when all else is equal, we want to stay consistent with the way users expect our systems to work.

Quiz: Feedback Cycles in David's Car 2



Do you think the car turned on?

- Yes
- No

So we know where the ignition button is. Let's press it. [NOISE] Do you think the car turned on?



Well, what do we know? We know the car was off. We know this is clearly the power button based on how it's labeled and where it's located. And most importantly, when I pressed it we heard kind of a

happy confirmation-y sound. So did the car turn on? Actually, it didn't. To turn this car on you have press the brake pedal while pressing the on button. The car doesn't do a great job of helping us across that goal of execution. There's no indicator that you're doing it wrong until you've actually already done it wrong. But the car does give us a short gulf of evaluation. If you do it incorrectly, an alert pops up on the dashboard letting you know you need to press the brake pedal and then press the on button. The output presented is easy to interpret. As presented in the context of when you need to know that information, so you kind of understand that it's a response to what you just did. So here we have some trouble with the gulf of execution but the gulf of evaluation is still pretty short. So now that I see this message I press down the brake pedal, press the on button [SOUND] and now the car is on.

Quiz: Feedback Cycles in David's Car 3



What are some ways this feedback cycle could be improved?

So now that we've seen the way this feedback cycle currently works, let's talk about improving it. How might we make this feedback cycle even better? How might we narrow the gulf of execution and the gulf of evaluation?



So here are a few ideas that I had. We know that the screen can show an alert when you try to turn the car on without pressing the brake pedal down. Why not show that alert as soon as the driver gets in

the car every time? That doesn't widen the gulf of execution for an expert user, but it does narrow it for a novice user, because even a novice can see that alert the first time they get in the car. But what still throws me off to this day is the sound the car makes when you try and turn it on. Watch. [SOUND] Did the car turn on? No. It didn't that time. [SOUND] Now it turned on. So it plays the same sound initially when you press the button and then plays a different follow up sound to confirm that the car actually turned on. I know why they do this, that one sound just confirms that you pressed the button successfully while the other sound confirms that the car turned on. But for me, I would just as soon have two different sounds confirm whether or not you just pressed the button or whether the car turned on. That way, just the presence of a sound confirms the fact that the button was pressed and the nature of the sound confirms the effect of that press.

Seven Questions for Bridging Gulfs

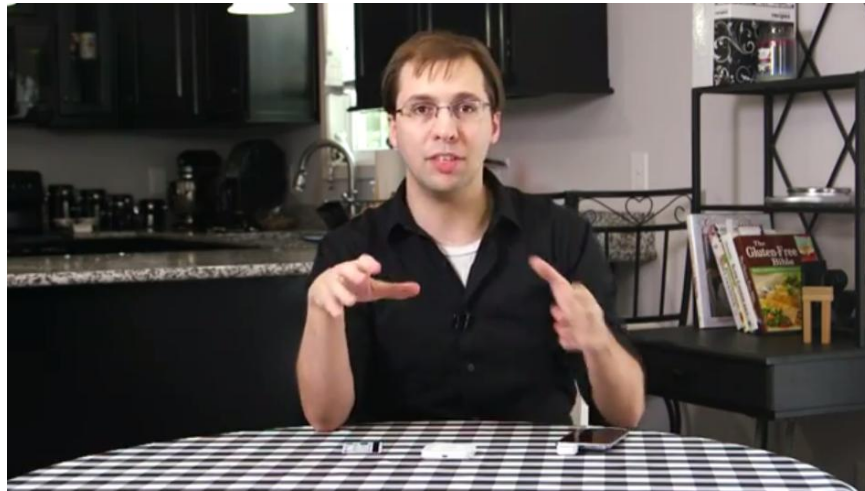
Interpret	How easily can one determine the function of the device?
Discern	How easily can one tell what actions are possible?
Identify	How easily can one determine the mapping from intention to physical movement?
Perform	How easily can one actually perform that physical movement?
Observe	How easily can one tell what state the system is in?
Compare	How easily can one tell if the system is in the desired state?
Interpret	How easily can one determine the mapping from system state to interpretation?

In his book, *Design of Everyday Things*, Don Norman outlines seven questions that we should ask when determining how usable a device is. Number one, how easily can one determine the function of the device? Number two, how easily can one tell what actions are possible? Number three, how easily can one determine the mapping from intention to physical movement? Number four, how easily can one actually perform physical movement? Number five, how easily can one tell what state the system is in? Number six, how easily can one tell if the system is in the desired state? And number seven, how easily can one determine the mapping from system state to interpretation? You'll notice these match our stages from earlier. Interpret your goal on the context of the device's function. Discern what actions are possible on the device. Identify how to perform an action or what action you want to perform. Perform the action. Observe the system's output. Compare the output to the desired state. And interpret the difference between the output and the desired state.

Exploring HCI: Feedback Cycles

I asked you earlier to pick an area of HCI in which you're interested, and reflect on it as you go through this course. Depending on the area you selected, feedback cycles can play a huge number of different roles. In healthcare, for example, feedback cycles are critical to helping patients manage their symptoms, and that relies on the results of certain tests being easy to interpret and evaluate. Feedback cycles are also present in some of the bigger challenges for gesture based interactions. It can be difficult to get feedback on how a system interpreted a certain gesture and why it interpreted it that way. Compare that to touch, where it's generally very easy to understand where you touched the screen. So think for a moment about how feedback cycles affect the area you chose to keep in mind.

Quiz: Design Challenge: Credit Card Readers 1



What are some ways inserting the card gives a better feedback cycle than sliding it?

Lately I've encountered another interesting example of feedback cycles in action. You may have actually seen this before as well. They're the new credit card readers. My wife sells arts and crafts at local events, and so she has these Square readers that can scan credit cards on her phone. One version lets you swipe, and the new version lets you insert the card. So let's check this out real quick. With the swipe version you just insert the card and pull it through, just like a traditional card reader. The problem is there's typically no feedback on whether you're swiping correctly. And what's more is you can be wrong in both directions. You can be both too fast or too slow. So you may have had a time when you were trying to swipe a credit card on some kind of reader, and you kept doing it more and more slowly and deliberately, thinking that the problem was that you had done it too fast originally. And then you discover that you've actually been going too slowly all along and your slowing down was actually counterproductive. There's no feedback here and the space and acceptable input is bounded on both sides. You have to go above one speed and below another speed. But now credit card readers are moving to this model where you just insert the card. You try, at least. In terms of feedback cycles, in what ways is this actually better?

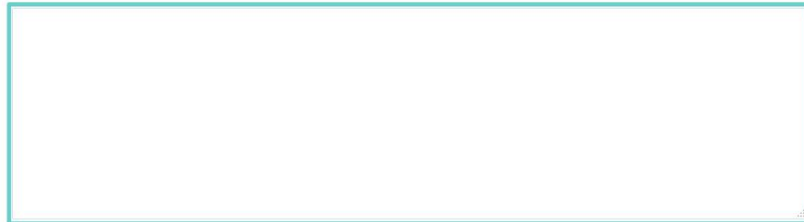


First, in terms of the gulf of execution, the insertion method is actually physically easier to do. While you can be both too fast and too slow with the sliding method, you can't push it too far in with the insertion method. So you know if there's an error, it's because the card isn't far enough into the reader. And second, there's rich feedback with the insertion method. It doesn't even have to come from the screen telling you that you didn't do it correctly. You feel the card stop when it's far enough into the reader. You have immediate physical feedback on whether you're putting it in the right place, and whether you've actually put it far enough in, rather than delayed feedback asking you to try again after some kind of waiting period.

Quiz: Design Challenge: Credit Card Readers 2



How can we build feedback into this system to prevent customers from walking away without their credit cards?



So, using the insertion method is significantly easier. However, the insertion method introduces a new problem. With the sliding method, I never had to actually physically let go of my card, so there was little chance of me walking away without it. With the insertion method, I insert the card and I wait. I'm not used to having to remember to retrieve my card from the card reader. Now this isn't quite as big a deal with these new portable readers, but for the mounted ones you see in stores it can be far more problematic. So how can we build some feedback into the system to make sure people remember their cards when they walk away?



There are a few things we could do here. We might build some kind of buzzer into the card reader to let the customer know when they can take their card out. That would make sure that they don't leave without it. ATM machines often do this, actually. They'll ring a buzzer until the card and the cash are removed. But that's noisy and potentially irritating. It would mess with the ambiance of a restaurant or something like that. We could do something super complicated, like pair the credit card with a smartphone and ring the phone when it gets too far away from the credit card. But that requires adding some new technology to every single credit card, which could be a pretty big expense. So what about something simpler? Why not force a customer to remove the credit card in order to get the receipt and their goods? Unless they're going to walk away without what they came to buy, that'll ensure that they remember their card.

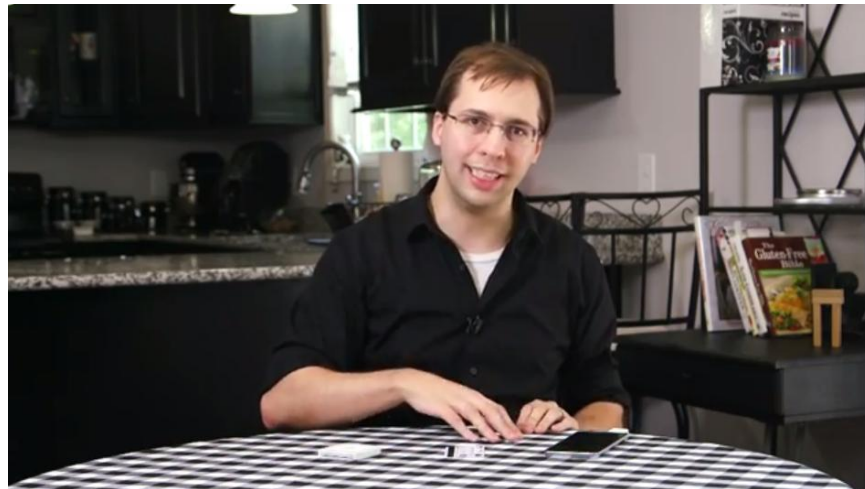
Quiz: Design Challenge: Credit Card Readers 3



What's wrong with asking how to make the process of sliding a credit card easier?

- Security is more important than HCI in this setting.
- It forces us to think only in terms of the credit card itself.
- Credit cards will soon be obsolete anyway.
- It's more important to stick to what people are used to than to try to make small improvements.

Now notice one last thing about this example. We've been discussing how to make the process of sliding or swiping a credit card easier. What's wrong with that question?

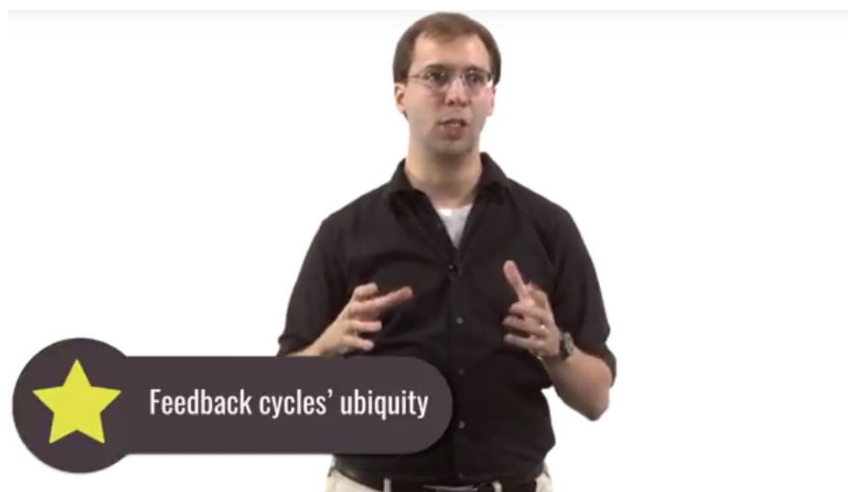


The problem is that we're not focused on the right task. Our task shouldn't be to swipe a credit card, or insert a credit card, or anything like that. Our task should be how to most easily pay for purchases. And possibly the easiest way to do that would be to design a system that lets you just tap your phone against the reader, this reader actually does that. That way, we can use the thing that people have on them at all times. Now maybe that's isn't the best option for various other reasons, but the important thing is we need to focus on what we're really trying to accomplish. Not just how we've done it in the past. We can make incremental improvements just sliding or swiping or inserting a credit card all we want. But we should always keep our eyes on the underlying task that the user needs to accomplish.

Conclusion to Feedback Cycles



Today we've talked about arguably the most fundamental concept of human computer interaction, feedback cycles. We describe feedback cycles for our purposes as the exchange of input and output between a user and a system to accomplish some goal.



We discussed feedback cycles' incredible ubiquity in other fields in discussions.



We talked about gulfs of execution, the distance between knowing what they want to accomplish and actually executing the steps necessary to accomplish it.



We talked about gulfs of evaluation, the distance between making some change in the system and evaluating whether or not the goal was accomplished.



We introduced the seven questions we need to ask ourselves to bridge those goals. Now that we understand these goals, our next goal is to understand methods for crossing them.

2.3 Direct Manipulation

Compiled by Shipra De, Summer 2017

Introduction to Direct Manipulation and Invisible Interfaces



[MUSIC] Today we'll talk about two applications of good feedback cycles. Direct manipulation and invisible interfaces. Direct manipulation is the principle that the user should feel as much as possible like they're directly controlling the object of their task. So for example, if you're trying to enlarge an image on your phone, it might be better to be able to drag it with your fingers rather than tapping a button that says, zoom in. That way you're really interacting directly with the photo. New technologies like touch screens are making it more and more possible to feel like we're directly manipulating something, even when there's an interface in the way.



At their best, the interface actually disappears, which is what we mean by an invisible interface. With an invisible interface the user has to spend no time thinking about the interface that they're using. All their time is dedicated to thinking about the task that they're performing.

Direct Manipulation: The Desktop Metaphor

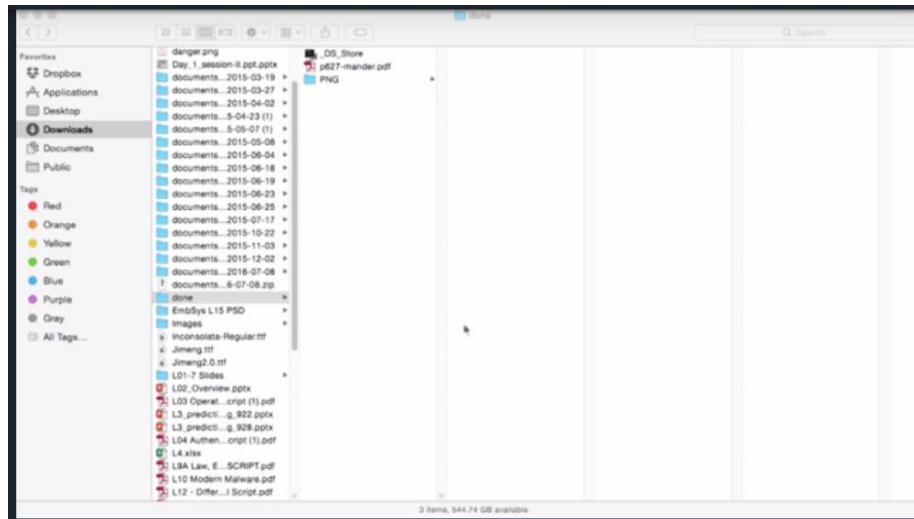


Our goal is to narrow the gulf of execution and the gulf of evaluation as much as possible. And arguably the ultimate form of this is something called direct manipulation. Now today direct manipulation is a very common interaction style. But in the history of HCI it was a revolutionary new approach. Now to understand direct manipulation, let's talk about the desktop metaphor. The files and folders on your computer are meant to mimic physical files and folders on a desktop. So, here are on my physical desktop, I have some files. What do I do if I want to move them? Well, I pick them up and I move them. What do I do if I want to put them in a folder? I pick them up and put them in the folder. I'm physically moving the files from where they are to where I want them to be. If files and folders on a computer are meant to mimic files and folders on a physical desk, then shouldn't the act of moving them also mimic the real world action of moving them? Wouldn't it narrow the gulf execution to leverage that real world experience and that real world expectation?

Direct Manipulation: The Desktop Metaphor

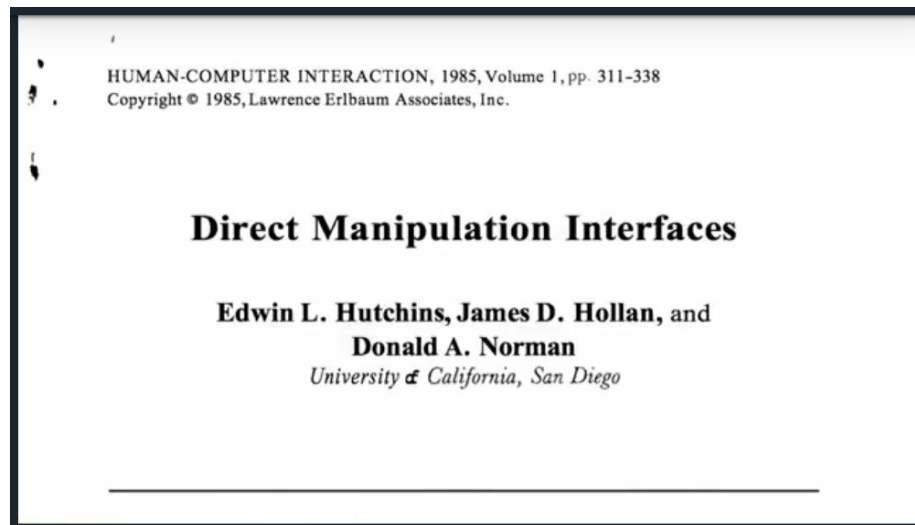
```
lawn-128-61-27-135:Downloads Calculon$ ls
Advanced Web Security- Part 1_FINAL.pptx
Advanced Web Security- Part 2_FINAL.pptx
Advanced Web Security- Part 3_FINAL.pptx
Day_1_session11.ppt.pptx
EmbSys L15 PSD
Images
Inconslata-Regular.ttf
Jineng.ttf
Jineng2.0.ttf
L01-7 Slides
L02_Overview.pptx
L03 Operating Systems Security - Visual Script (1).pdf
L04 Authentication - Script (1).pdf
L10 Modern Malware.pdf
L12 - Differential Register Allocation - Visual Script.pdf
L12 Intrusion Detection part 4.pdf
L3_predictive_modeling_920.pptx
L3_predictive_modeling_920.pptx
L4.xlsx
L04 Lw, Ethics, and Privacy - VISUAL SCRIPT.pdf
MoreLetters.ttf
PSD
PSD IMGs
PSD-2016-02-18.zip
PSD-2016-02-29.zip
PSDs
PSDs 2
PSDs 3
PSDs 4
lawn-128-61-27-135:Downloads Calculon$ mv p627-mander.pdf done
lawn-128-61-27-135:Downloads Calculon$ cd done
lawn-128-61-27-135:done Calculon$ ls
PNG
QC_Store
p627-mander.pdf
lawn-128-61-27-135:done Calculon$
```

Files and folders on my computer are meant to mimic files and folders on my physical desk. So we ideally want the action of moving them around on my computer to mimic the action of moving them around on my physical desk. But it wasn't always like this. Before graphical user interfaces were common, we moved files around using command line interfaces like this. The folder structure is still the same on the operating system. But instead of visualizing it as folders and icons, I'm interacting with a text-based command line. To view the files, I might need to type a command like `ls`, which I just have to remember. If I don't remember that command, I don't have much recourse to go find out what I'm supposed to be doing. To move a file, I need to type something like this. I have to type the command, the file I want to move, and the folder I want to move it to. Again, if I forget the name of that command, or the order of the parameters to provide, there's not a whole lot I can do to recover from that. I need to run off to Google and find out what the correct order of the commands was. Which is actually what I did before filming this video because I don't actually use the terminal very often. Then when I execute that command, there's not a lot of feedback to let me know if it actually executed correctly. I might need to change folders to find out. There I see the files present in that folder but I had to go and look manually. There's nothing really natural about this. Now don't get me wrong, once you know how to interact with this interface, it's very efficient to use. But when you're a novice at it, when you've never used it before, this is completely unlike the task of managing physical files on your real desk.

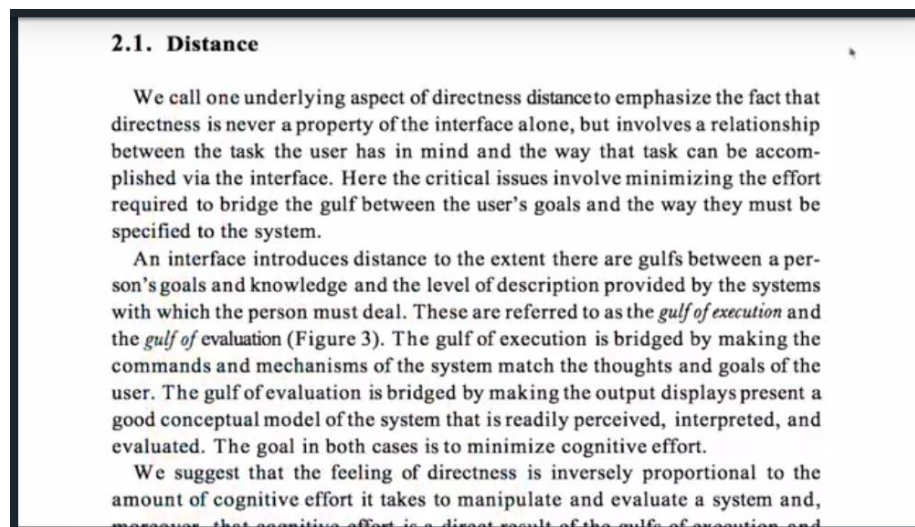


Then, the computer mouse came along. And with it came the ability to move a mouse around the screen. Equipped with this, my action in moving files and folders becomes much more direct. I can actually just click the file I want to move and drag it into the new folder. I get instant feedback by the fact that the file disappeared as soon as I dragged it over. And there was a sound effect that you may or may not have been able to hear. So now instead of typing in some cryptic command that I just have to be able to remember, I can just click on the file I want to move, and physically drag it to the folder in which I want to have it. That's a very natural interaction, it mimics what I do on my physical desk. Moving the mouse around is a lot better than having to type in those commands, but the gulf of execution and evaluation are still present, especially for some novice users. There's still some interpretation that has to happen to understand that when I move my hand a little left on the mouse, the cursor on screen will move to the left as well. And while clicking feels kind of like grabbing, there's still some interpretation there. It's more direct than the command line, but there's still a gap. The modern touchscreens made direct manipulation more direct than ever. Let's say I want to put an icon into a folder on my screen. How do I do it? I hold down the icon, and I drag it to the screen. The fact that if I wanted to move something around my desk, I would have to hold it down, means that this is almost entirely direct manipulation. I don't need any prior knowledge to attempt to do what feels natural for moving that icon into that folder. That gives us a nice, general heuristic to keep in mind. How do we help the user interact most closely with the target of their task? How do we make it so they're manipulating it as directly as possible?

Paper Spotlight: "Direct Manipulation Interfaces"

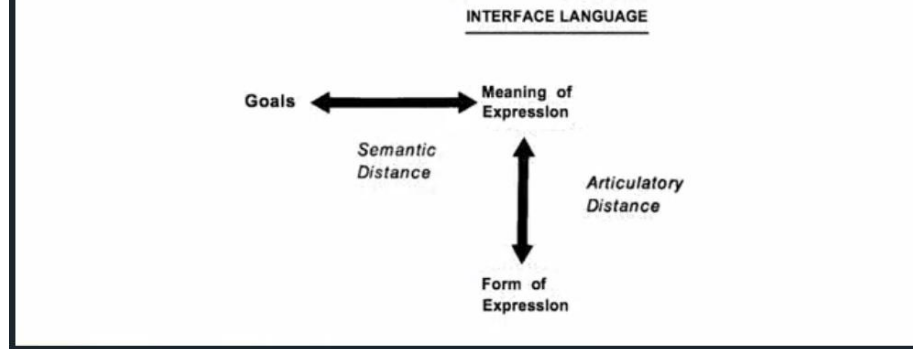


The Seminal Paper on Direct Manipulation Interfaces came out in 1985 coauthored by Edwin Hutchins, James Hollan, and Don Norman. We'll talk a lot more about Hutchins and Norman later in our conversations. Hutchins coauthored the foundational paper on distributed cognitive and Norman created one of the most accepted sets of design principles in his seminal book, the Design of Everyday Things. But in 1985, direct manipulation was starting to become a more common design strategy. Hutchins, Hollan, and Norman identified two aspects of directness.



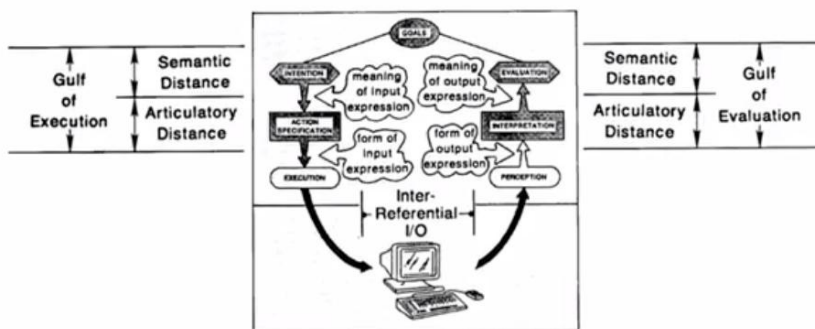
The first was distance. Distance is the distance between the users goals and the system itself. This is the idea of gulfs of execution and evaluation that we talked about in the context of feedback cycles. They write that "the feeling of directness is inversely proportional to the amount of cognitive effort it takes to manipulate and evaluate the system." In other words, the greater the cognitive load required to use the system, the less direct the interaction with the system actually feels.

Figure 4. Every expression in the interface language has a meaning and a form. Semantic distance reflects the relationship between the user intentions and the meaning of expressions in the interface languages both for input and output. Articulatory distance reflects the relationship between the physical form of an expression in the interaction language and its meaning, again, both for input and output. The easier it is to go from the form or appearance of the input or output to meaning, the smaller the articulatory distance.



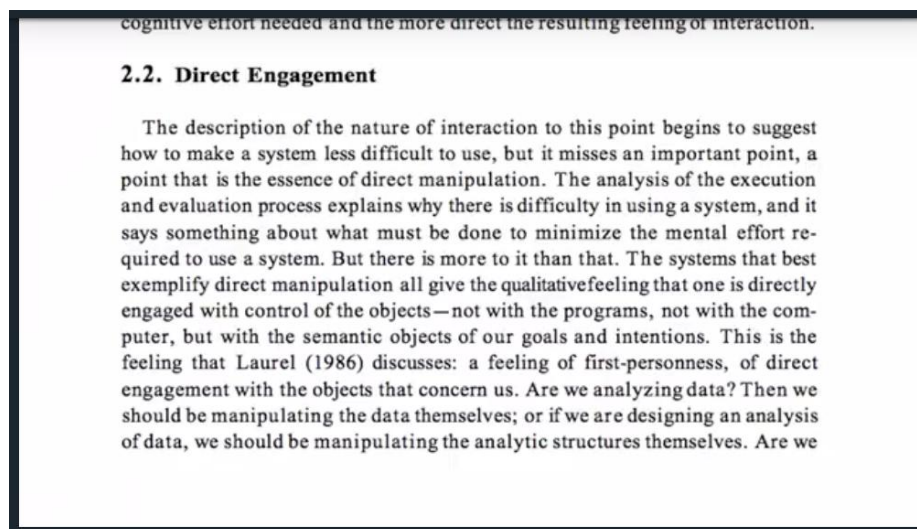
The authors break distance into two components, semantic distance and articulatory distance. Semantic distance refers to the distance between the user's goals and their expression in the system. In other words, it's how hard it is to know what to do. Articulatory distance is the distance between expression and its execution. In other words, it's how hard it is to actually do what you know to do. You might notice that semantic distance encompasses our identify intentions and identify actions part of our gulf of execution. And articulatory distance comprises that execute actions phase.

Figure 6. Forming an intention is the activity that spans semantic distance in the gulf of execution. The intention specifies the meaning of the input expression that is to satisfy the user's goal. Forming an action specification is the activity that spans articulatory distance in the gulf of execution. The action specification prescribes the form of an input expression having the desired meaning. The form of the input expression is executed by the user on the machine interface and the form of the output expression appears on the machine interface, to be perceived by the user. When some part of the form of a previous output expression is incorporated in the form of a new input expression, the input and output are said to be inter-referential. Interpretation is the activity that spans articulatory distance in the gulf of evaluation. Interpretation determines the meaning of the output expression from the form of the output expression. Evaluation is the activity that spans semantic distance in the gulf of evaluation. Evaluation assesses the relationship between the meaning of the output expression and the user's goal.

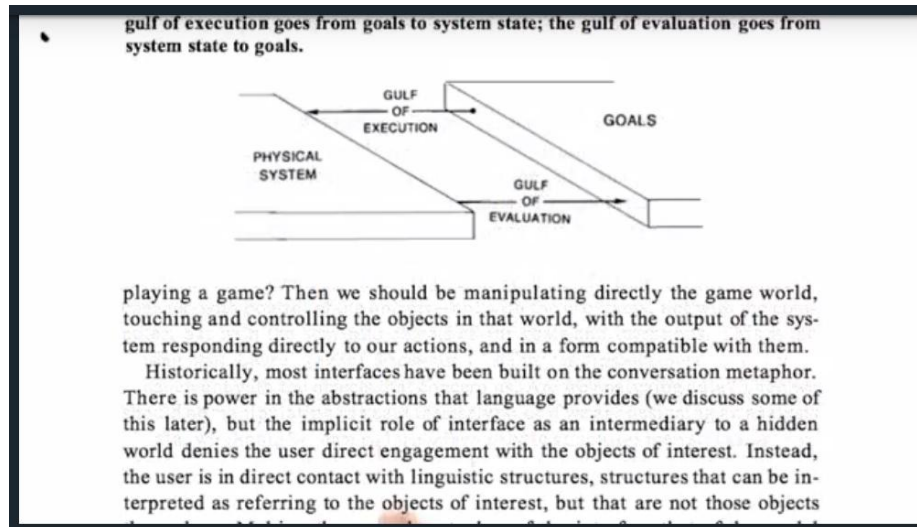


This is brought together here in figure 6 from this paper. The user starts with some goals and translates those goals into the intentions in the context of the interface. They then translate those intentions into the form of the input, the actions, and execute those actions. The system then does something and then gives back some form of output. The user then interprets the form of that output to discern the meaning of that output, and then evaluates whether or not that meaning matches their goals. So to take an example, when I brought up this figure, I needed to rotate the paper to display it correctly. That was my goal. I translated that goal into the context of the application, a rotate option that was probably hidden somewhere. I then identified the action, which was pressing the rotate button, and I executed

that action. The system then did something and returned the output to me. The output specifically was the paper turned upside down, instead of turned the way I wanted it. That was the form of the output. I then interpreted that form to discern the meaning that it had rotated it the wrong way. I evaluated that my goals weren't accomplished and now I knew what to do next. I then pressed the button two more times to rotate it twice again, and the system then returned that output to me. I interpreted the form of that output to mean that the figure was now right side up, and I evaluated that that matched my initial goals. You might be able to see that this cycle is happening constantly whenever you're interacting with a computational interface. You could think of it in terms of broad tasks like searching a document for some key word or you could think of it in terms of each individual little tasks like interacting with the menus and pulling up the right prompts. But distance is only one component of direct manipulation. It's possible to have interfaces with very little distance that nonetheless are not examples of this kind of direct interaction. Everything we've talked about so far is true of feedback cycles in general, not just of direct manipulation.

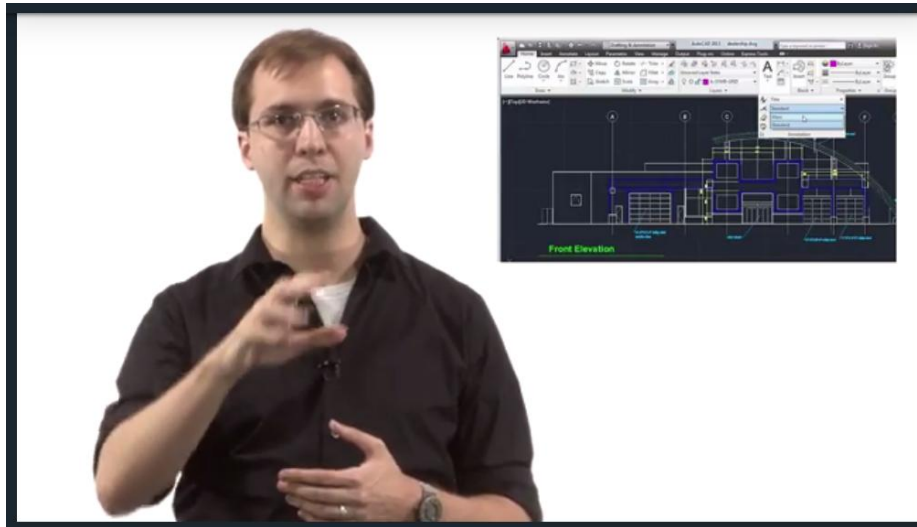


That's why the second component of this is direct engagement. What sets direct manipulation apart is this second component. The authors of the paper write that "the systems that best exemplify direct manipulation all give the qualitative feeling that one is directly engaged with the control of the objects—not with the programs, not with the computer, but with the semantic objects of our goals and intentions." If we're moving files we should be physically moving the representations of the files. If we're playing a game, we should be directly controlling our characters. If we're navigating channels, we should be specifically selecting clear representations of the channels that we want. That's what takes a general, good feedback cycle and makes it an instance of direct manipulation.



We can shorten the gulfs of execution and evaluation in a number of ways without direct manipulation, but direct manipulation is a powerful method for shortening that distance.

Exploring HCI: Direct Manipulation



Virtual reality right now is making some incredible strides in facilitating direct manipulation in places where it just hasn't been possible before. Traditionally, when designers are designing stuff in 3D, they're forced to use a 2D interface. That translation from 2D to 3D really gets in the way of directly manipulating whatever is being designed. Through virtual reality though, designers are able to view what they're designing in 3D as if it's in the room there with them. They can rotate it around with the same hand motions you'd use to rotate a physical object. They can physically move around the object to get different angles on it. So virtual reality is allowing us to bring the principle of direct manipulation to tasks it hasn't been able to touch before. But there's still a lot of work to do. Gesture interfaces like those used in virtual reality struggle with some feedback issues. We aim to make the user feel like they're physically manipulating the artifact. But when you're working with something with your hands, it pushes back against you. How do we recreate that in virtual reality?

Quiz: Reflections: Direct Manipulation



What is an example of a task you perform regularly that does not leverage the principle of direct manipulation?

Take a moment real quick and reflect on some of the tasks you perform with computers day-to-day. What are some of the places where you don't interact through direct manipulation? If you're having trouble thinking of one, think especially about places where technology is replacing things you used to do manually. Chances are, the physical interface was a bit closer to the task than the new technical one. How can the technical interface better leverage direct manipulation?



When I was writing the script for this exact video, I was interrupted by a text message from a friend of mine. And in the reply I was writing, I wanted to include a smiley face. We know that using emojis and emoticons tends to humanize textual communication. On my phone, the interface for inserting an emoji is to tap an icon to bring up a list of all the emojis and then select the one that you want. When I'm reacting to someone in conversation, I'm not mentally scrolling through a list of all my possible emotions and then choosing the one that corresponds. I'm just reacting naturally. Why can't my phone capture that? Instead of having to select smiling from a list of emotions, maybe my phone could just have a button to insert the emotion corresponding to my current facial expression. So to wink, I would just wink. To frown, I would just frown. It wouldn't be possible to capture every possible face, but for some of the most commonly used ones, it might be more efficient.

Babies and Direct Manipulation



There may be no better example of the power of direct manipulation than watching a baby use an interface. Let's watch my daughter, Lucy, try and use her Kindle Fire tablet. My daughter Lucy is 18 months old, yet when I give her an interface that uses direct manipulation, she's able to use it. She wouldn't be able to use a keyboard or a mouse yet, but because she's directly interacting with the things on the screen, she can use it.



Actually, there might be an even better example of direct manipulation in action. There are games made for tablet computers for cats. Yes, cats can use tablet computers when they use direct manipulation.

Quiz: Exercise: Direct Manipulation



Which of these Mac touchpad gestures do you think are examples of direct manipulation?

- ☐ Pressing down to click.
- ☐ Pressing two fingers down to right-click.
- ☐ Dragging two fingers down to scroll.
- ☐ Double-tapping to zoom in and out.
- ☐ Pinching to zoom in and out.

Let's try a quick exercise on direct manipulation. The Mac touchpad is famous for facilitating a lot of different kinds of interactions. For example, I can press on it to click, press the two fingers to right-click. I can pull up and down with two fingers to scroll up and down. I can double tap with two fingers to scroll in and out a little bit. And I can pinch to zoom in and out a lot more. Which of these are good examples of direct manipulation in action?

Now there's some room for disagreement here, but I think these five seem to be pretty cut and dry. We can think about whether or not these are direct manipulation by considering whether or not what we're doing to the touchpad is what we'd like to do to the screen itself. For clicking, I would consider that direct manipulation because just as we press directly on the screen we're pressing directly on a touchpad. Right-clicking though, the two finger tap, doesn't really exemplify direct manipulation, because there's nothing natural about using two fingers to bring up a context menu as opposed to using one to click. We have to kind of learn that behavior. Scrolling makes sense because with scrolling it's like I'm physically pulling the page up and down to see different portions of it. The two finger tap for

zooming in and out a little bit though isn't really direct manipulation, because there's no real clear reason that needs to zoom in, zoom out. Pinching on the other hand though, makes sense. Because it's as if I'm physically grabbing the page and shrinking and enlarging it. So some of these I would say are pretty good examples of direct manipulation. While others are things that we kind of have learn to do.

Making Indirect Manipulation

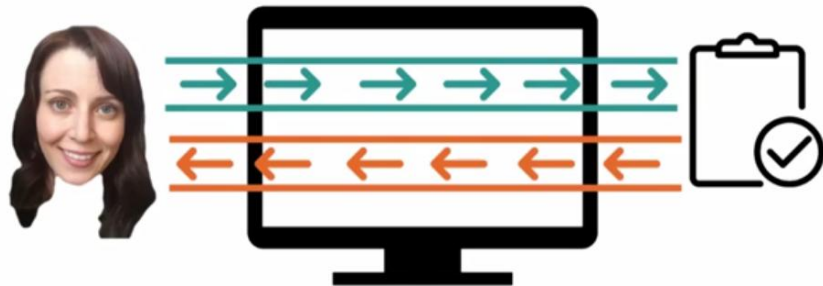


The Mac Touchpad has some interesting examples of how you can make indirect manipulation feel more direct. For example, if I swipe from the right to the left on the touchpad with two fingers, it pulls up this notification center over on the right. This feels direct because the notification center popped up in the same place on the screen that I swiped on the touchpad. The touchpad is almost like a miniature version of the screen. But they could have placed a notification center anywhere and used any kind of interaction to pull it up. This isn't like scrolling where there is something fundamental about the content that demands a certain kind of interaction. They could have designed this however they wanted. But by placing the notification center there and using that interaction to pull it up, it feels more direct. Now, animation can also help us accomplish this. On the Touchpad, I can clear the windows off my desktop by kind of spreading out my fingers on the touchpad, and the animation shows them going off to the side. And while that's kind of like clearing off your desk, I'd argue it's not close enough to feel direct except that the animation on screen mimics that action as well. The windows could have faded away or they could have just slid to the bottom and still accomplish the same function of hiding what's on my desktop. But the animation they chose reinforces that interaction. It makes it feel more direct. The same thing actually applies with Launchpad, which we bring up with the opposite function by pinching our fingers together. The animation looks kind of like we're pulling back a little bit or zooming out and we see the launch pad come into view, just as the gesture is similar to zooming out on the screen. So direct manipulation isn't just about designing interactions that feel like you're directly manipulating the interface. It's also about designing interfaces that lend themselves to interactions that feel more direct.

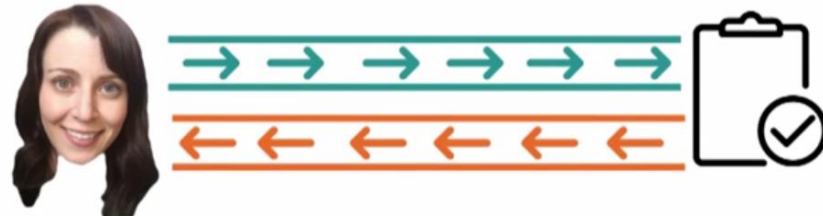
Exploring HCI: Direct Manipulation

Depending on the area of HCI that you chose to explore, direct manipulation might be a big open-question. So for gesture based interaction for example, you're generally not actually touching anything. Direct manipulation is contingent on immediate feedback that maps directly to the interaction. So how do you create that in a gesture-based interface? This is a big challenge for virtual reality as well. Virtual reality thrives on making you feel like you're somewhere else, both visually and auditorily, but it has a long way to go kinesthetically. How do you create the feeling of direct manipulation based on physical action when you can only give feedback visually or auditorily? So take a second and reflect on how these principals of direct manipulation apply to your chosen area of HCI.

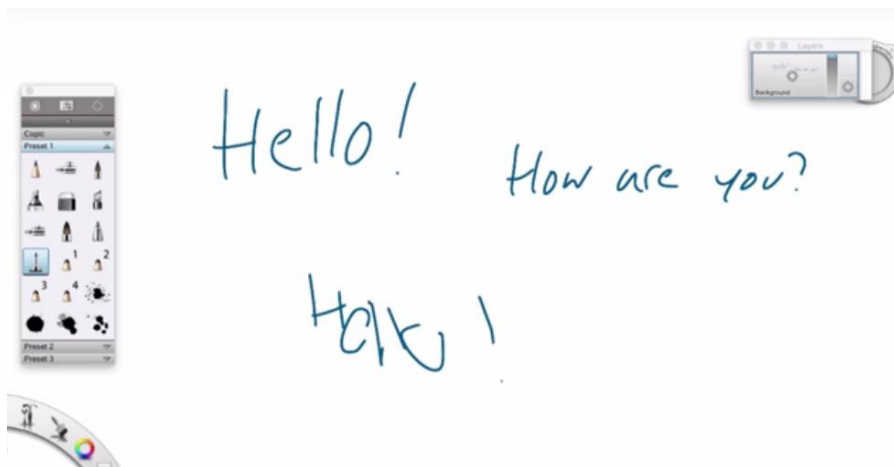
Invisible Interfaces



Whether through using direct manipulation, through innovative approaches to shrinking these gulfs or through the user's patience and learning, our ultimate goal is for the interface between the user and the task to become invisible.



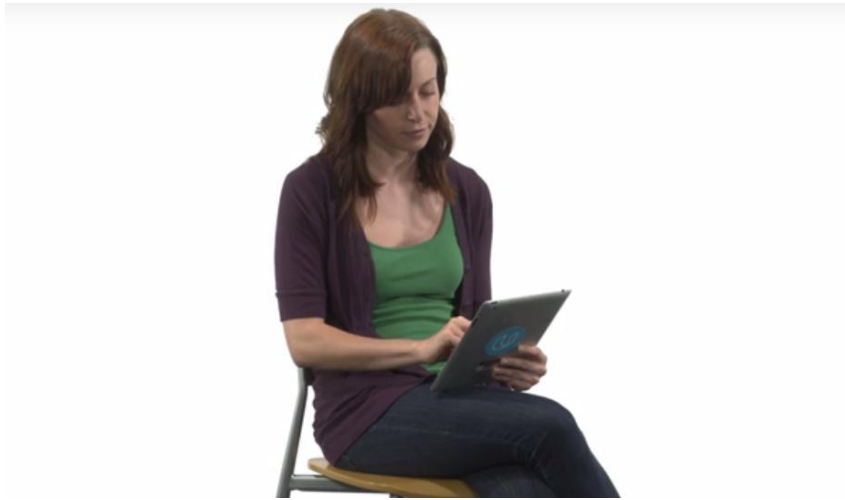
What this means is that even though there is an interface in the middle, the user spends no time thinking about it. Instead, they feel like they're interacting directly with the task rather than with some interface.



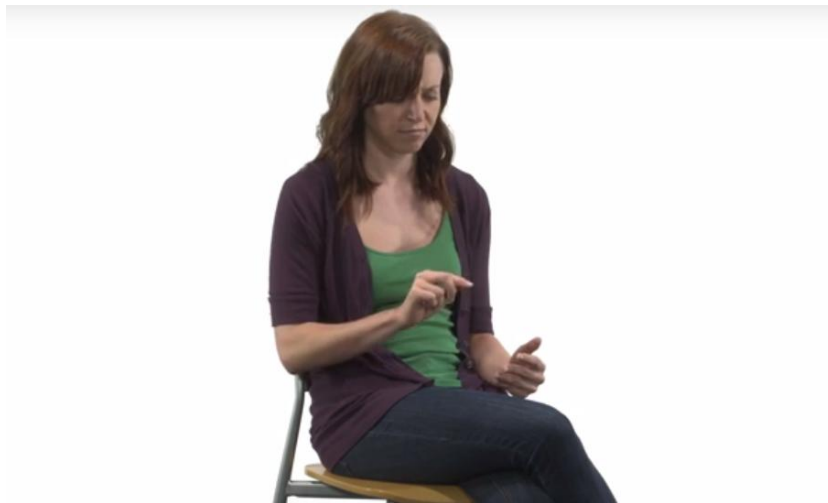
So for example, I have a stylus and I'm going to write on this tablet computer. I'm interacting with an interface just translating my drawing into data in the system. But for me, this feels just like I'm writing on a normal page. That feels just like writing on paper. This interface between me and the data representation of my drawing underneath is pretty much invisible. I feel like I'm writing on paper.

Contrast that with trying to draw with a mouse. That feels extremely unnatural. I'm very well aware of the mouse as the interface between myself and this drawing task. So the direct manipulation facilitated by the stylus gets me much closer to my task and helps the interface disappear between me and what I'm trying to accomplish.

Good Design, Bad Design In



[MUSIC] Good design. Interfaces that are metaphorically invisible.



Bad design, interfaces that are literally invisible. Well, kind of, just your base interfaces are in one sense literally invisible, that's actually why it's so important to give great feedback. Because otherwise, it's tough to gauge the success of a gesture interaction.

Invisibility by Learning

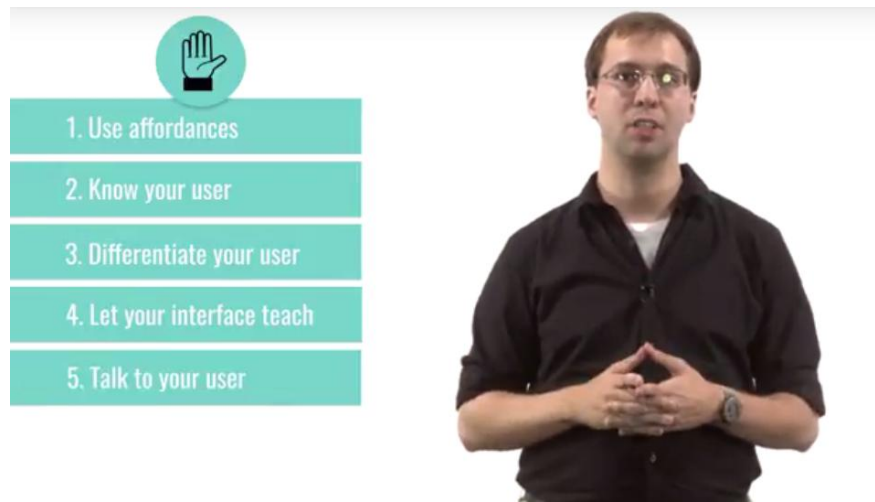


We shouldn't fall into the trap of assuming that just because an interface has become invisible, the design is great. Interfaces become invisible not just through great design, but also because users learn to use them. With enough practice and experience, many users will become sufficiently comfortable with many interfaces to feel invisibly integrated in the task. So take driving for example. Lets say I'm driving a car and I discover I'm headed right for someone. What's my reaction? Well I turn the wheel to the side and I press my brake. It's instinctive. I do it immediately, but think about that action. If I was just running down the street and suddenly I saw someone in front of me would it be natural for me to go like that? Of course not. The steering wheel was an interface I used to turn to the left. But it's become invisible during the task of driving because of all my practice with it. But just because the interface has become invisible doesn't mean it's great interface. People spend months learning to drive, they pay hundreds of dollars for classes. And they have to pass a complicated test. Driving is important enough that it can have that kind of learning curve. But for the interfaces that we design, we generally can't expect users to give us an entire year just to learn to use them. We'll be lucky if they give us an entire minute to learn to use them. So our goal is to make our interfaces invisible by design.

Invisibility by Design

Our goal is to create interfaces that are invisible from the moment the user starts using them. They should feel immediately as if they're interacting with the task underlying the interface. Now this is an extremely tall order and one we honestly probably won't meet very often, but it's the goal. In fact, in my opinion, this is why people tend to underestimate the complexity of HCI. When you do things right, people won't be aware that you've done anything at all. So how do we create interfaces that are invisible from the very first moment the user starts using them? That's precisely what we'll discuss in a lot of our conversations about HCI. We'll talk about principles for creating interfaces that disappear like leveraging prior expectations and providing quick feedback. We'll also talk a lot about how to get inside the user's head and understand what they're seeing when they look at an interface that we can make sure that they're internal mental model matches the system. In fact, if we consider invisibility to be a hallmark of usable design, then this entire course could be retitled Creating Invisible Interfaces.

5 Tips: Invisible Interfaces



Here are five tips for designing invisible interfaces. Number 1, use affordances. We talk more about affordances when we discuss design principles and heuristics. But affordances are places where the visual design of the interface is just how it's supposed to be used. Buttons are for pressing. Dials are for turning. Switches are for flicking. Use these expectations to make your interface more usable. Number 2, know your user. Invisibility means different things to different people. Invisibility to a novice means the interactions are all natural. But invisibility to an expert means maximizing efficiency. Know for whom you're trying to design. Number 3. Differentiate your user. Maybe you're designing something for both novices and experts. If that's the case, provide multiple ways of accomplishing tasks. For example, having copy and paste under the Edit menu keeps those options discoverable. But providing Ctrl C and Ctrl V as shortcuts keep those actions efficient. Number 4. Let your interface teach. When we think of teaching users how to use our software we usually think of tutorials or manuals. But ideally the interface itself will do the teaching. For example, when users select Copy and Paste from the Edit menu, they see the hotkey that corresponds to that function. The goal is to teach them a more efficient way of performing the actions without requiring them to already know that in order to do their work. Number 5. Talk to your user. We'll say this over and over again, but the best thing you can do is talk to the user. Ask them what they're thinking while they use an interface. Note especially whether they're talking about the task or the interface. If they're talking about the interface, then it's pretty visible.

Quiz: Reflections: Invisibility by Design



What is a time when you picked up a new interface and knew immediately how to use it to accomplish your goal?

Reflecting on where we've encountered invisible interfaces is difficult, because they were invisible. What makes them so good is the fact that we didn't have to notice them. But give it a try anyway. Try to think of a time where you picked up a new interface for the very first time and immediately knew exactly how to use it to accomplish the task you had in mind.



One of my favorite examples of an interface that is invisible by design comes from a video game called Portal 2. In lots of video games, you use a control stick to control the camera in game, but different people have different preferences for how the camera should behave. Some feel if you press up, you should look up. Others like myself, feel if you press down you should look up, more like you're controlling an airplane with a joystick. In most games you have to set this manually by going to options, selecting camera controls, and enabling or disabling a y axis and it's just a chore. But in portal two, watch what happens. >> You will hear a buzzer. When you hear the buzzer, look up at the ceiling. [SOUND] Good. You will hear a buzzer. When you hear the buzzer, look down at the floor. [SOUND] Good. This completes the gymnastic portion of your mandatory physical and mental wellness exercise. >> Did you see that? It was so subtle you might not have even noticed it. A character in the game asked

me to look up. The game assumed whichever direction I pressed was the way I would want to press when I want to look up. And set my preference accordingly. No option screen, no changing settings. The game automatically and invisibly had me complete my goal of correctly setting my camera preference.

Quiz: Design Challenge: The



Take a moment to reflect. Then, come share your reflections with the rest of us over on the class forums.

For this design challenge, let's tackle one of the most common problems addressed in undergraduate HCI classes, designing a better remote control. Now, these probably aren't very good interfaces. And that's not to say that they're poorly designed, but the constraints on how many different things they have to do and how limiting the physical structure can be, make these difficult to use. You might have seen humorous images online of people putting tape over certain buttons on the controls to make them easier to use for their parents or their kids. How would we design an invisible interface for universal remote control, one that doesn't have the learning curves that these have?



Personally I think this is a great candidate for a voice interface. And in fact, Comcast, Amazon and others have already started experimenting with voice interfaces for remote controls. One of the challenges with voice interfaces is that generally the commands aren't very discoverable. Generally, if you don't know what you can say, you have no way of finding out. But watching TV and movies is such a normal part of our conversations that we already have a vocabulary of how to say what we want to do. The challenge is for us designers to make a system that can understand that vocabulary. That way when I say, watch Community, it understands that Community is a TV show and it tries to figure out, do I grab it from the DVR, do I grab it from On Demand, do I see if it's on live? The vocabulary for the user was very natural. So for example, watch Conan.

>> Well tonight, a fan named David Joiner thinks he caught a mistake. He says it happened when I was telling a joke recently about Rand Paul.

>> Hey Conan, I was watching your episode on April 17th, and you said that Rand Paul wanted to run for president.

I had to put that in there somewhere.

Conclusion to Direct Manipulation



Today we have talked about two applications of effective feedback cycles. Direct manipulation, and invisible interfaces. We talked about how interfaces are most effective when the user has a sense that they're directly manipulating the object in their task. We talked about how modern technology like touch screens and virtual reality are making it possible for manipulation to feel more and more direct. We talked about how the most effective interfaces become invisible between the user and their task. It's just that the user spends no time at all thinking about the interface. And we talked about how interfaces can become invisible via either learning or design and are most interested in designing them to become invisible. To a large extent, that's the definition of usable design. Designing interfaces that disappear between the user and their task.

2.4 Human Abilities

Compiled by Shipra De, Summer 2017

Introduction to Human Abilities



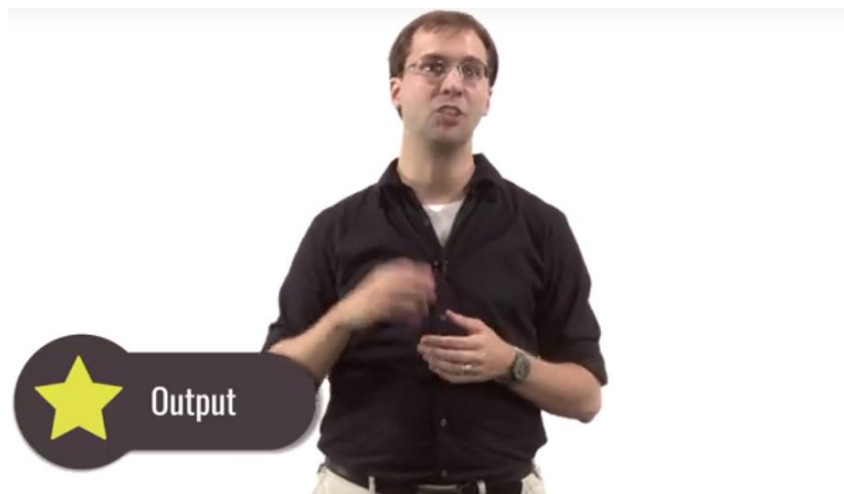
[MUSIC] Human computer interaction starts with human. So it's important that we understand who the human is, and what they're capable of doing. In this lesson, we're going to bring up some psychology of what humans can do. We'll look at three systems. Input, processing, and output.



Input is how stimuli are sent from the world, and perceived inside the mind.

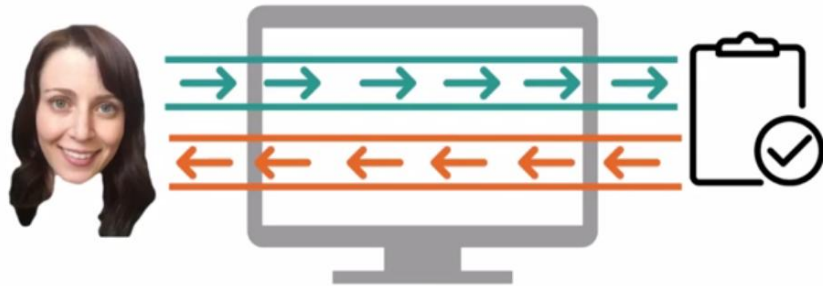


Processing is cognition, how the brain stores, and reasons over the input it's received.



Output is how the brain then controls the individual's actions out in the world. Now, we're going to cover a lot of material at a very high level. If you're interested in hearing more, I recommend taking a psychology class, especially one focusing on sensation and perception. We'll put some recommended courses in the notes.

Information Processing



In discussing human abilities, we're going to adopt something similar to the processor view of the human. For now we're interested in what they can do, physically, cognitively, and so on.



So we're going to focus exclusively on what's going on over here. We're going to look at how the person makes sense of input, and how they then act in the world. And right now, we're not going to worry too much about where that input came from, or what their actions in the world actually do. Notice that in this lesson we're discussing the human, almost the same way we discuss the computer, or the interface, in most lessons. The human is something that produces output and consumes input, just like a computer might be otherwise. But for now, we're only interested in how the human does this.

Sensation and Perception: Visual



Let's start by talking a bit about what the average person can sense and perceive. So, here we have Morgan again. Morgan has eyes. Morgan's eyes are useful for a lot of things.



The center of Morgan's eye is most useful for focusing closely on color or tracking movement. So, we can assume that the most important details should be placed in the center of her view. Morgan's peripheral vision is good for detecting motion, but it isn't as good for detecting color or detail. So while we might use her periphery for some alerts, we shouldn't require her to focus closely on anything out there.



As a woman, Morgan is unlikely to be colorblind, she has about a 1 in 200 chance. Men have a much greater prevalence of color blindness at about 1 in 12. Either way, that's a significant body of people. So we want to avoid relying on color to understand the interface. We can use it to emphasize knowledge that it's already present in the system, but using the system shouldn't rely on perceiving color. Sight is directional. If Morgan's looking the wrong way or has her eyes closed, she'll miss visual feedback.



As Morgan gets older, her visual acuity will decrease. So if we're designing something with older audiences in mind, we want to be careful of things like font size. Ideally, these would be adjustable to meet the needs of multiple audiences. All together though, Morgan's visual system is hugely important to her cognition. The majority of concepts we cover in HCI are likely connected to visual perception. [SOUND]

Sensation and Perception: Auditory



Morgan also has ears.



Morgan can discern noises based on both their pitch and their loudness.

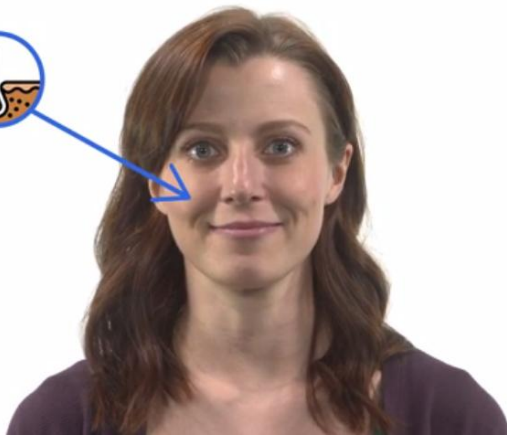


Her ears are remarkably good at localizing sound as well. In fact, she can tell the difference between a nearby quiet sound and a far away loud sound. Even if their relative pitches and loudnesses are the same when they reach her ear.



Unlike vision, hearing isn't directional. Morgan can't close her ears or point her ears the wrong direction so she can't as easily filter out auditory information. That might be useful for designing alerts, but it's problematic for overwhelming her or sharing too much information with the people around her.

Sensation and Perception: Haptic



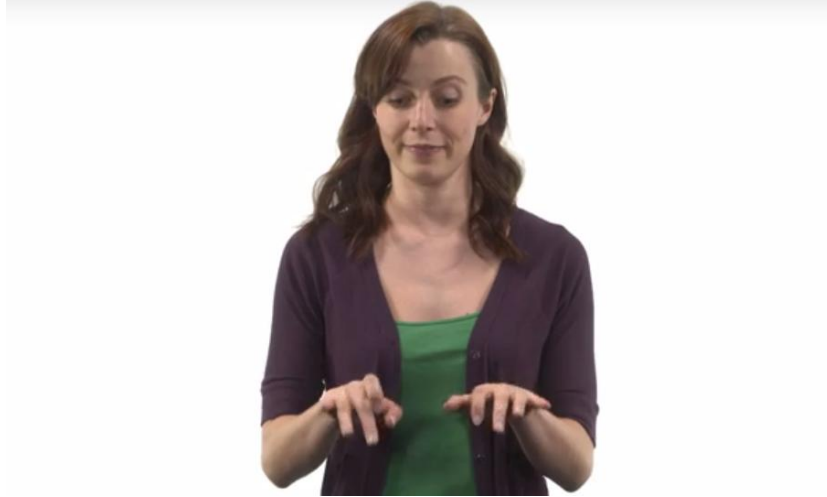
Morgan's skin can feel things.



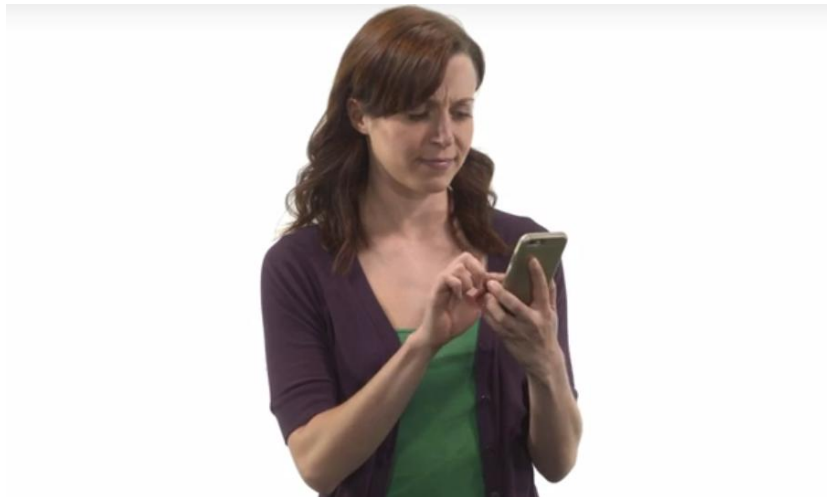
It can't feel at a distance but it can feel when things are touching right up against it.



It can feel a variety of different types of input, like pressure, vibration and temperature. Like listening, Morgan can't easily filter out touch feedback. But unlike listening, touch feedback is generally only available to the person it's touching, so it can be used to create more personal feedback. Traditionally, touch feedback, or haptic feedback, has been very natural.

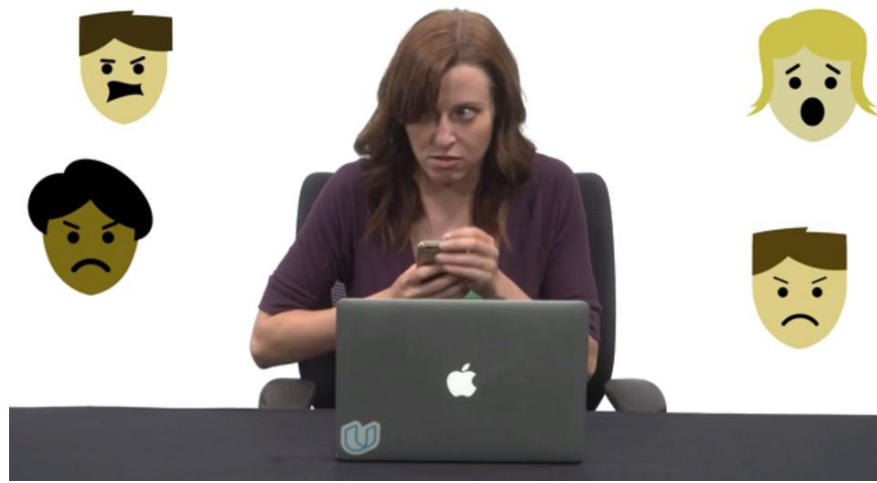


Morgan feels the keys go down as she presses them on keyboard.



But with touch screens, motion controls and virtual reality, touch needs to be more and more designed explicitly into the system if we're to use it.

Quiz: Design Challenge: Message Alerts

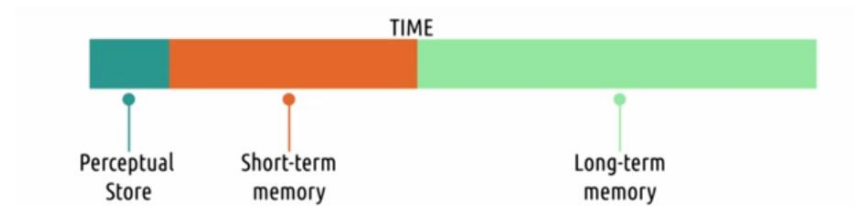


Let's design something for Morgan real quick. Let's tackle the common problem of being alerted when you've received a text message. Here are the constraints on our design for Morgan. It must alert her whether the phone is in her pocket or on the table. It cannot disturb the people around her. [SOUND] And yes, vibrating loudly against the table counts as disturbing the people around her. You're not restricted to just one modality, though, but you are restricted to the sensors that the phone has available.

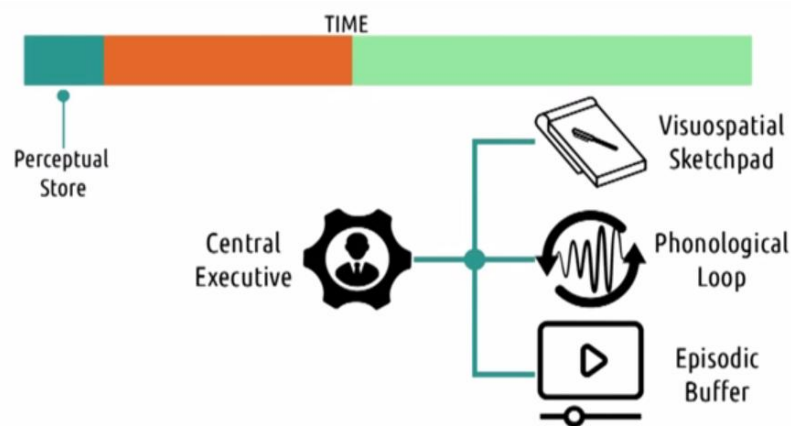


Here's one possible design. We know that smart phones have cameras and light sensors on them. We can use that to determine where the phone is, and what kind of alert it should trigger. If the sensor detects light, that means the screen is visible, so it might alert her simply by flashing its flashlight or illuminating the screen. If the sensor does not detect light, it would infer that the phone is in her pocket and thus, would vibrate instead. Now, of course, this isn't perfect. It could be in her purse, or she could have put it face down. That's why we iterate on a design like this, to improve it based on the user's experiences.

Quiz: Memory: Perceptual Store



After the perception portion of this model comes the cognition portion, starting with memory. There are lots of different models of memory out there. For our purposes we're going to talk about three different kinds of memory, the perceptual store or working memory, the short-term memory, and the long term memory. Some scientists argue that there are other types of memory as well, like an intermediate sort of back of the mind memory. But the greatest consensus is around the existence of at least these three kinds.



So let's start with the first, the perceptual store or the working memory. The perceptual store is a very short term memory lasting less than a second. One of the most common models of working memory came from Baddeley and Hitch in 1974. They described it as having three parts. First, there's the visuospatial sketchpad which holds visual information for active manipulation. So for example, picture a pencil. The visuospatial sketchpad is where you're currently seeing that pencil. A second part is the phonological loop. The phonological loop is similar, but for verbal or auditory information. It stores the sounds or speech you've heard recently, such as the sound of me talking to you right now. A third part is the episodic buffer. The episodic buffer takes care of integrating information from the other systems as well as chronological ordering to put things in place. Finally all three of these are coordinated by a central executive.



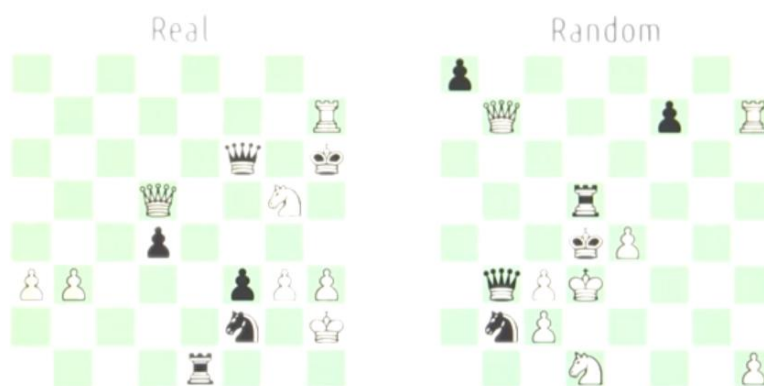
So let's try an example of this. I'm going to very quickly show you a picture and ask you a question about it. Don't focus on any particular portion of the picture, try to view it as a whole.



What was the score on the scoreboard?

- ☒ 0 - 0
- ☒ 2 - 2
- ☒ 5 - 2
- ☒ 5 - 3

What was the score on the scoreboard?



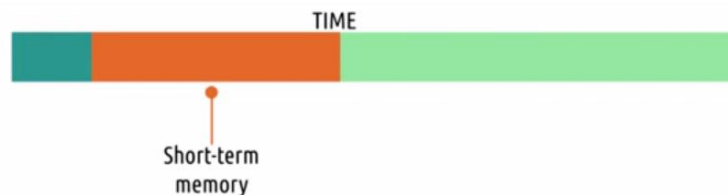
As you can now see, the score was 0 to 0. Now, as you tried to reason over that, what you probably did was picture the image in your mind. That was trying to reason over what was stored in the perceptual buffer, and it decayed very quickly. However, if you're a fan of baseball, you probably had a better chance of getting that right. That's because you have some domain expertise. You're better able to process images about that domain more quickly. You might, for example, have recognized that most of the innings weren't marked. So that increases the odds that the score of the game was pretty low. This idea is actually the foundation of a fascinating study about chess experts versus novices and recognizing the configuration of chess boards. The study found that experts were far better than novices at remembering realistic chess boards that were only flashed for a short period of time, like the one on the left. But experts were no better than novices at remembering random chessboards. So expertise, or rehearsal, delays the decay of the perceptual buffer.

Quiz: Memory: Short Term and Chunking



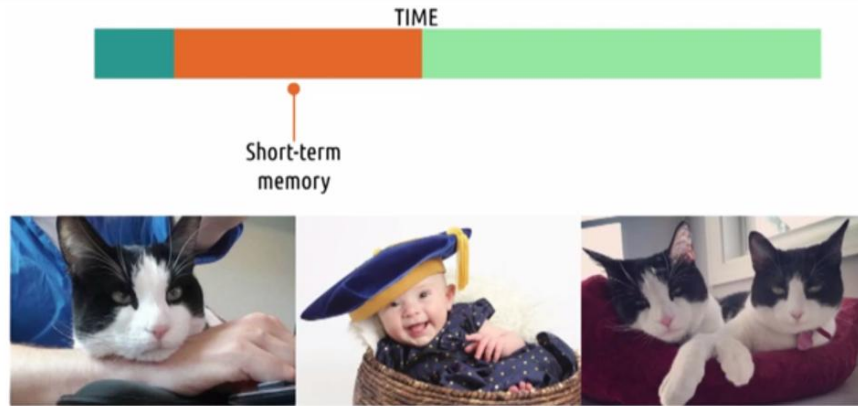
“Four to five chunks at a time”

When we're designing interfaces, short-term memory is one of our biggest concerns. It's important we avoid requiring the user to keep too much stored in short-term memory at a time. Current research shows that users can really only store four to five chunks of information at a time. For a long time, there was a popular idea that people could store seven, plus or minus two, items in memory at a time. But more recent research suggests that the number is really four to five chunks. There are two principals we need to keep in mind here though. The first, is the idea of chunking. Chunking is grouping together several bits of information into one chunk to remember.



FSDVW	PODAP	PUPPY
POBXC	REXIC	WATCH

So to illustrate this, let's try it out. I'm about to show you six combinations of letters. Try to memorize them and then enter them into the exercise that pops up. Are you ready? Now to keep you from just rehearsing it in your perceptual store until you can reenter it, I'm going to stall and show you some pictures of my cats.



There's one. There's both of them, and here's my daughter.

Okay, now fill in the words.



What were the six words in the previous video?

So what happened? Well, what likely happened is that you maybe had only a little bit of trouble remembering the two real words that were listed on the right. You might have had some more trouble remembering the two words that were listed in the middle that were fake words, but did nonetheless look like real words. And you probably had a lot of trouble remembering the two series of letters over on the left. Why is all of that? Well, when it came to memorizing these two words, you were just calling up a chunk that you've already had. You didn't see these as arbitrary collections of letters, you just saw them as words. For the ones in the middle, you've never seen those combinations of letters before, but you could pronounce them as if they were words. So you likely saw them as fake words rather than just random collection of letters. For these over the left though, you had to memorize five individual characters. So that's means that while these four were able to jump in the words or pseudo words. These likely had to be remembered as five chunks each. That's makes this much more difficult to remember than these.

Quiz: Memory: Short Term and Recognition



Which of these words did you see previously?

- FSDVW
- SOMIP
- PUPPY
- RTPVP
- REXIC
- TABLE

However, there is a way we can make this easier. So let's ask a different question. Which of these six words did you see in the exercise previously?



Which of these words did you see previously?

- FSDVW
- SOMIP
- PUPPY
- RTPVP
- REXIC
- TABLE

Even if you had trouble just naming these series of letters in the exercise previously. You probably were much more successful at this exercise. Why is that? That's because it's far easier to recognize something you know than to recall it independently. And that's a useful take away for us as we design interfaces. We can minimize the memory load on the user by relying more on their ability to recognize things than to recall them.

Memory: Short Term Takeaways

8 5 3 3 7 1 6 2 2 7



So what are the implications of short term memory for HCI? We don't want to ask the user to hold too much in memory at a time, four to five chunks is all. Asking the user to hold ten numbers in short term memory, for example, would probably be too much. But we can increase the user's effective short term memory capacity by helping them chunk things.

853 371 6227



For example, this is probably by far easier to remember, even though it's the same content. We've shrunk ten items into three. And we've used a format for phone numbers with which you're probably familiar if you're in the United States. If you're from outside the U.S., you might be familiar with a different grouping. But the same principle applies. And finally, when possible, we should leverage recognition over recall. For example, if I ask you to recite the number, maybe you could. In fact, go ahead. Try it.

749 277 4924
853 371 6227
211 960 3274
637 859 0226



Whether or not you could do that, you almost certainly, though, can pick it from this list. This is one of the reasons why menu bars and tool strips are so ubiquitous in software design. The user doesn't have to remember the icon for a command or the name of an option. They just have to recognize it when they see it.

Memory: Long Term Memory

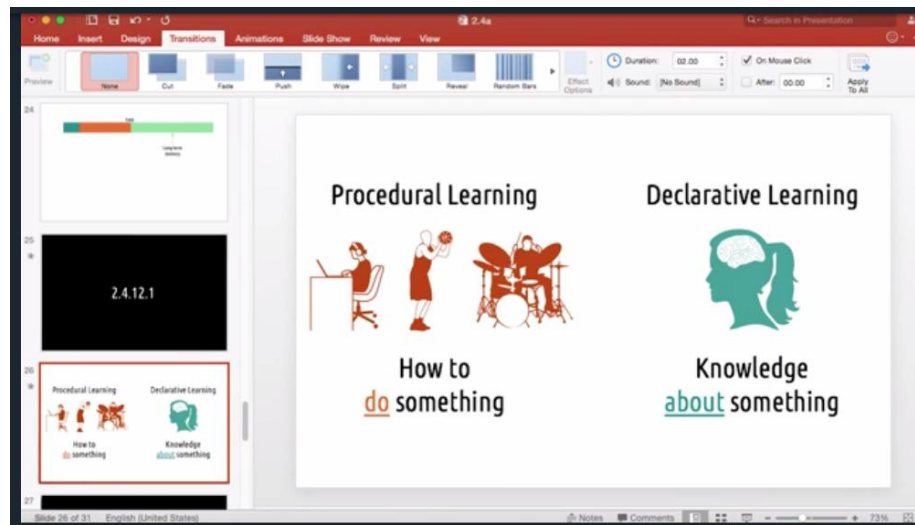


Finally, we have long-term memory. Long-term memory is a seemingly unlimited store of memories, but it's harder to put something into long-term memory, than to put it into short-term memory. In fact, to load something into long-term memory, you generally need to put it into short-term memory several times.

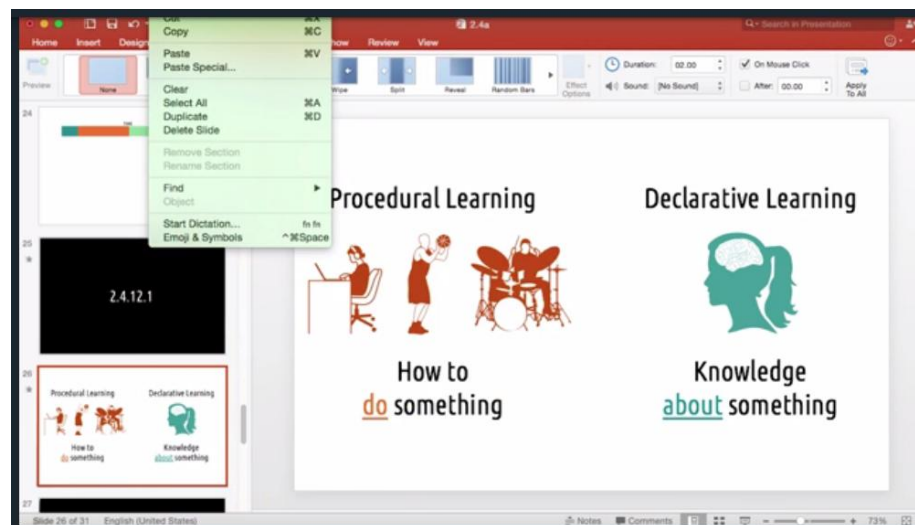


To demonstrate this, I'm going to describe something called lightner system. The lightner system is a way of memorizing key value pairs, or in other words memorizing flashcards. Those can be words and their definitions, countries and their capitals, laws and their formulas, anything where you're given a key and asked to return a value. So I have some flashcards here that have the capitals of the world. What I do is go through each one, read the country, check to see if I remember the capital. And if I do, I'll put it in the pile on the right. I read the country and don't know the capital, I'll put it in the pile on the left. So let me do that real quick. [MUSIC] Now tomorrow, I will just go through the pile on the left. Any that I remember from the pile on the left tomorrow, I'll move to the pile on the right. Any that I still don't remember, will stay in the pile on the left. So I'm focusing my attention on those that I don't yet know. In four days, I'll go through the pile on the right. And any that I don't remember then, I'll move back to my pile on the left, to remind me to go through them each day. So the things that I remember least, are most often loaded in the short-term memory, solidifying them in my long-term memory. Now in practice, you wouldn't just do this with two piles, you'd do a three, or four, or five. And the long restoration pile you'd might only go through yearly, just to see if it's decayed yet.

Cognition: Learning



Now let's talk a little bit about cognition. One of the most important cognitive processes to consider is learning. When we design interfaces, we are in some ways hoping the user has to learn as little as possible to find the interface useful. However, our interfaces should also teach the user over time how to use them most efficiently. We can take PowerPoint as an example of this. Let's pretend this is the first time I've ever used PowerPoint and I want to copy something. This application doesn't assume I know anything yet. If I poke around, I'll find the Copy button under the Edit menu, which is off-screen above the slide.



When I bring up that menu, it also shows the hot key that will allow me to actually copy something. That's helping me to learn to interact with the application more efficiently through hot keys instead of through menus. But yet, it's also not assuming that I already knew that, because I was still able to find this under that menu.

Procedural Learning



How to
do something

Declarative Learning

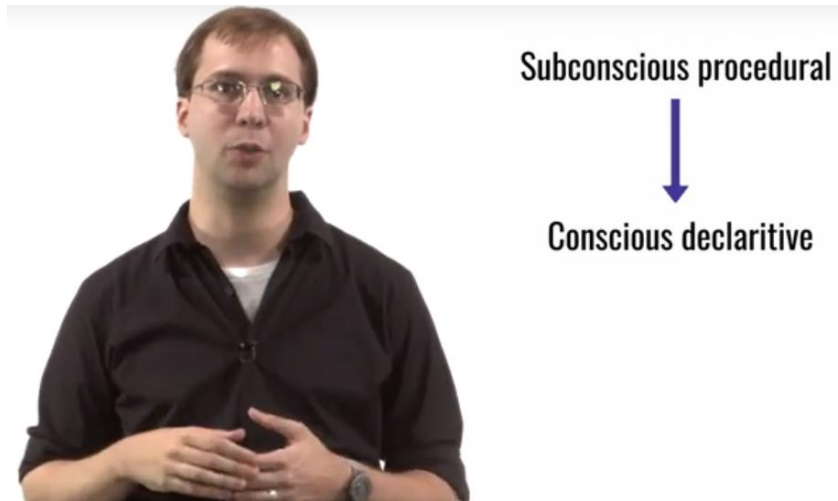


Knowledge
about something

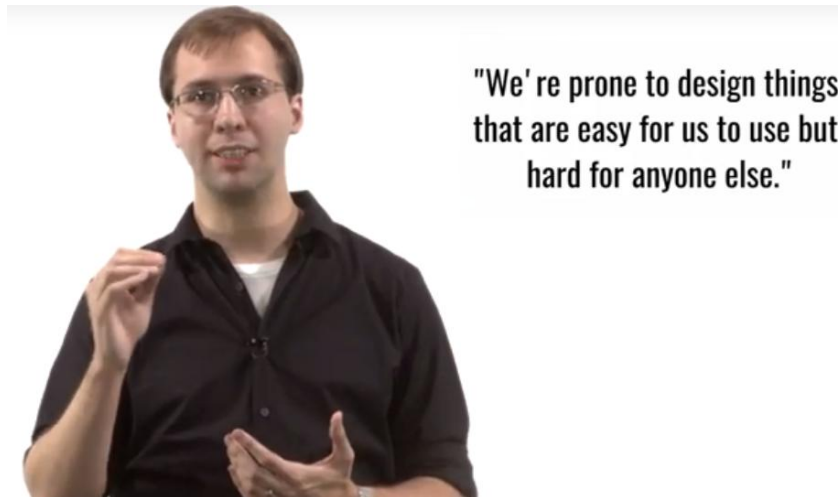
There are two kinds of learning we're most interested in, procedural and declarative. Procedural is how to do something. That can be doing work on a computer, playing sports, playing musical instrument. It's a task in which you're engaged or an activity you are performing, it's something that you do. Declarative learning is knowledge about something. It's what you know in your head. It's what you can answer when asked. So if I asked you, what's the hotkey for paste? I'm asking for a declarative knowledge. If I asked you, paste your clipboard here, I'm asking for procedural knowledge. If you're an expert computer user, you'd probably find it easier to just actually paste your clipboard than to answer me when I say, what's the hotkey for paste? And in all likelihood, when I ask you, what's the hotkey for paste, the way you remember it is you mentally simulate doing it, and then you look at what you simulated yourself doing. What's interesting is that while declarative knowledge is how we generally communicate with one another, procedural knowledge is generally what we do in HCI. When you have strong procedural knowledge, you may forget how you're doing what you're doing, because it's so second nature.



You're unconsciously competent with what you're doing. When you're in that state, it can be difficult to explain to someone who lacks that competence, because you aren't sure what makes you so good at it.

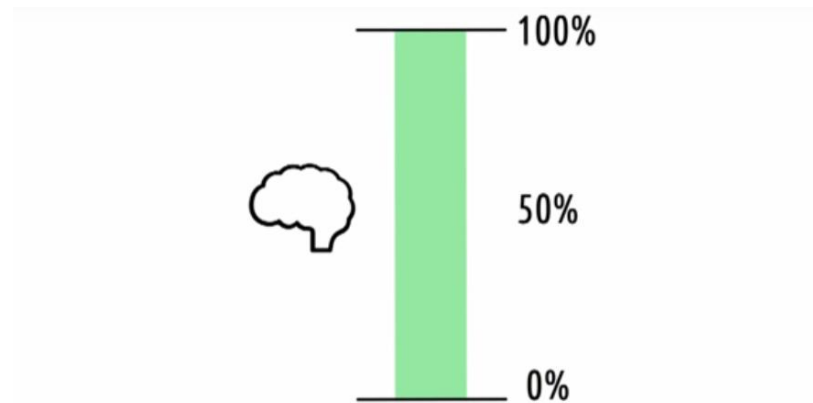


It's difficult to translate your subconscious procedure analogy into explicit declarative knowledge. A declarative knowledge is how we communicate, that's how we communicate with novice users.

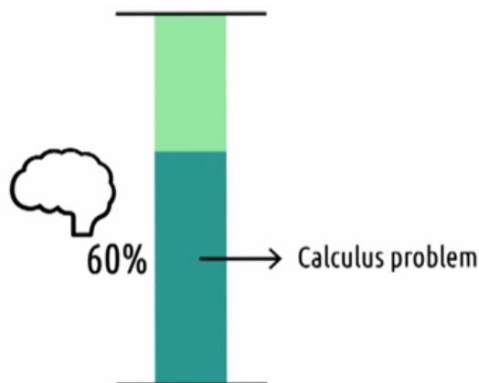


This is important, because as the designers of interfaces, we're the experts in our domains. That means we're prone to design things that are easy for us to use but hard for anyone else.

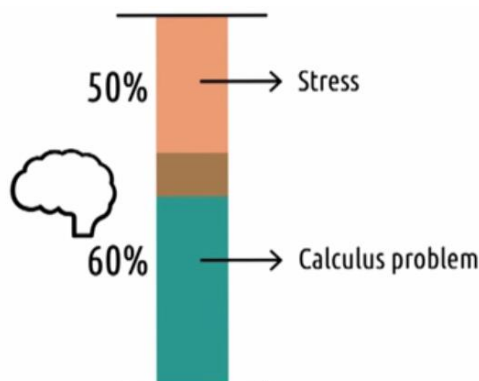
Cognition: Cognitive Load



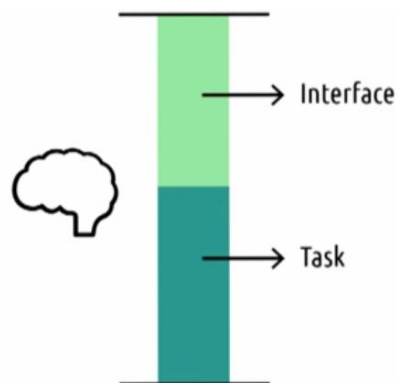
To talk about cognitive load, let's think for a moment of the brain like it's a computer. The community is actually divided on whether or not the brain actually operates this way, but for the purposes of this explanation, it's a useful metaphor. So your brain has a certain number of resources available to it, the same way your computer has a certain number of processor resources available to it. Each thing that the brain is working on takes up some of those resources.



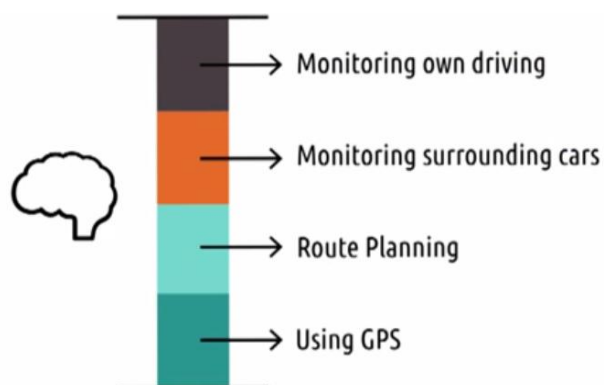
Let's say you're at home in a quiet area, working on a calculus problem that requires 60% of your cognitive resources. In that setting, you have plenty of resources to solve that problem.



However, then you go to take a calculus test. Now you have some stress in there. Now you're stressing about the impact this test is going to have on your grade. You're stressing about how well other people seem to think they are doing on it. Whether or not other people seem to be struggling while you struggle. This is taking up a lot of your cognitive resources. Here we see the stress taking up 50% of the cognitive resources you have. Now you don't have sufficient resources to complete the problem successfully. I hypothesize that's why test taking anxiety can have such a negative effect. It takes resources away from actually working on the test. You can apply these same principles to the presence of distractions, anxiety disorders and more. Cognitive load has two major applications to our working design interfaces.



One, we want to reduce the cognitive load posed by the interface, so that the user can focus on the task.



Second, we want to understand the context of what else is going on while users are using our interface. We need to understand what else is competing for the cognitive resources users need to use our interface. If we're designing a GPS or navigation system for example, we want to be aware that the user will have relatively few cognitive resources because they're focusing on so many things at once.


Quiz: Reflections: Cognitive Load




Let's take a second and reflect on cognitive load. Try to think of a task where you've encountered a high cognitive load. What different things did you have to keep in mind at the same time? And how could an interface have actually helped you with this problem?

Computer programming is one task with an incredibly high cognitive load. At any given time, you're likely holding in working memory your goals for this line of code, your goals for this function, your goals for this portion of the program as a whole, the variables you've created and a lot more. That's why there's so many jokes about how bad it is to interrupt a programmer, because they have so much in working memory that they lose when they transition to another task. But there are ways good IDEs can help mitigate those issues. For example, inline automated error checking is one way to reduce the cognitive load on programmers, because it lets them focus more on what they're trying to accomplish rather than the low level syntax mistakes. In that way, the IDE offloads some of the responsibility from the user to the interface. Now we could phrase that a little bit differently too. We could describe this as distributing the cognitive load more evenly between the different components of the system, myself and the computer. That's a perspective we discuss when we talk about distributed cognition.

5 Tips: Reducing Cognitive Load



1. Use multiple modalities
2. Let the modalities complement each other

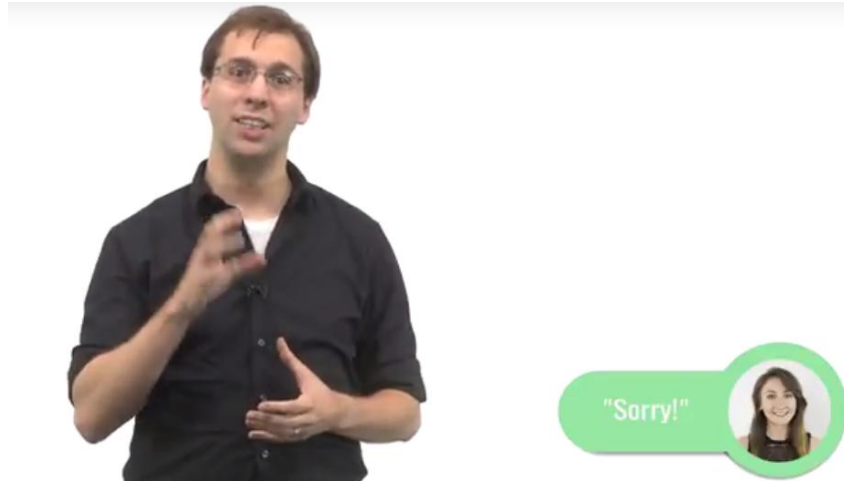


Here are five quick tips for reducing cognitive load in your interfaces. Number 1, use multiple modalities. Most often, that's going to be both visual and verbal. But when only one system is engaged, it's natural for it to become overloaded while the other one becomes bored. So describe things verbally and also present them visually. Number 2, let the modalities complement each other.



"These are my cats, Louise and Larry!"

Some people will take that first tip and use it as justification to present different content of the two modalities. That actually increases cognitive load because the user has to try to process two things at once, as you just noticed when Amanda put something irrelevant up while I said that.



Instead, focus on letting each modality support, illustrate or explain the other, instead of competing with the other.

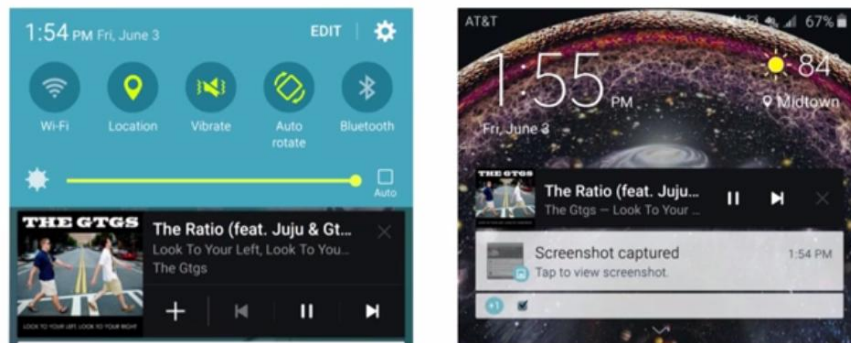


Number 3, give the user control of the pace. That's more pertinent in educational applications of cognitive load, but oftentimes interfaces have built-in timers on things like menus disappearing or selections needing to be made. That dictates the pace, induces stress, and it raises cognitive load. Instead, let the user control the pace. Number 4, emphasize essential content and minimize clutter. The principal of discoverability says we want the user to be able to find the functions available to them, but that could also raise cognitive load if we just give users a list of 500 different options. To alleviate that, we can design our interfaces in a way that emphasizes the most common actions, while still giving access to the full range of possible options. Number 5, offload tasks. Look closely at what the user has to do or remember at every stage of the interface's operation. And ask if you can offload part of that task on to the interface. For example, if a user needs to remember something that they entered on a previous screen, show them what they entered. If there's a task they need to do manually, see if you can trigger it automatically.

Motor System



So our user has received some input. It's entered her memory, she cognitively processed it. Now it's time to act in the world in response. In designing interfaces, we're also interested in what is physically possible for users to do. This includes things like, how fast they can move, or how precisely they can click or tap on something.



For example, here are two versions of the Spotify control widget that appears on Android phones. On the left is the version that's available in the tray of the phone that you can access at any given time by swiping down on the phone screen. And on the right is the version that appears on the lock screen when you turn on a locked phone while it's playing music. In each case, the X closes the app, which is consistent with a lot of other applications. The forward, back and pause buttons are similarly consistent with their usual meanings. I don't actually know what the plus sign here does. It doesn't have a clear mapping to some underlying function. Now note on the left, we have the close button, in the top right corner. It's far away from anything else in the widget. On the right, the close button is right beside the skip button. I can speak from considerable personal experience, and say that the level of specificity or the level of precision required to tap that X, instead of tapping the skip button, is pretty significant. Especially if you're using this while running or driving, or anything besides just sitting there, interacting directly with your phone. The precision of the user's ability to tap on a button is significantly reduced in those situations. And in this case, that can lead to the quick error of closing the application when all you're trying to do is skip forward to the next song. This isn't an error in the perception of the screen. It's not an error in their memory of the controls. They're not thinking that the X button actually is the skip button. This is just an error in what they're physically able to perform at a given time. The interface relies on more precision than they would have in many circumstances. So this design doesn't take into consideration the motor system of the user or the full context surrounding usage of this

application. This isn't as significant in the design on the left, because there's more room around that close button. If I aim for the forward button and miss, the worst that's going to happen is I might pause it. I'm not going to close it by accident. This is one example of how we need to be aware of the constraints on the user's motor system. What they can physically do, how precise or accurate they can be, and so on. And we have to be aware of that in the context where the application is going to be used as well. These buttons are no smaller than the keys on a smart phone keyboard but we expect more specificity when they're sitting there typing with their thumbs, as opposed to reaching over and interacting real quick on something on the lock screen. Now of course, there might be other constraints around this. There might be a reason why this button's placed there. There might be some constraint in the Android system that doesn't let them use more than one row of the lock screen. In that case, we would need to make our interface more tolerant of errors. Maybe require a double tap to close the app, or maybe we mute it when it's pressed and then gives the user five seconds to confirm that that's actually what they want to do. Those are ways of reducing the penalty for errors.

Exploring HCI: Human Abilities

We've talked a lot about different human abilities in this lesson. Depending on the demand you chose, the human abilities in which you're interested, may vary dramatically. If you looking at gestural interfaces or wearable devices than the limitations of the human motor system might be very important. On the other hand, if you're interested in educational technology, you're likely more interested in some of the cognitive issues surrounding designing technology. For virtual reality, you're main concern will likely be perception. Although, there are interesting open questions about how we physically interact with virtual reality as well. So take a few moments and reflect on what the limitations of human ability are in the domain of HCI that you chose to explore.

Conclusion to Human Abilities



Today, we've gone through a crash course on human abilities and perception. We started off by talking about the main ways people perceive the world around them through sight, sound and touch.



Then we discussed some of the components of cognition, especially memory and learning.

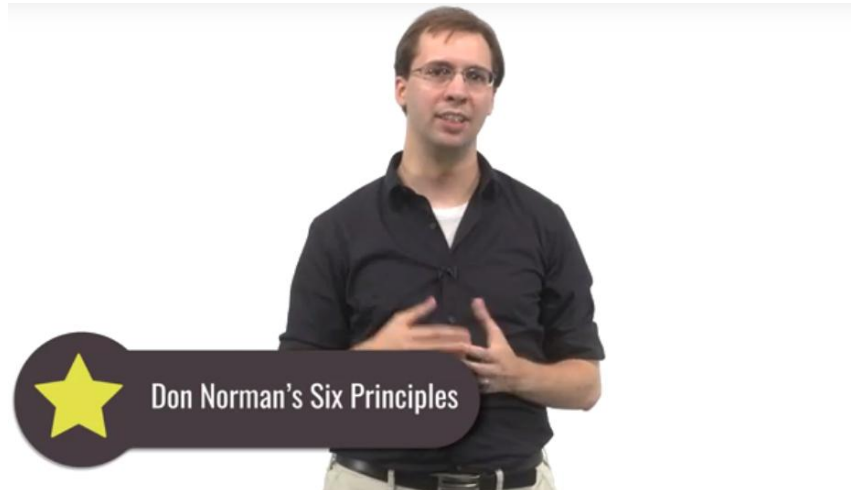


Then we discussed the motor system, how the person then interacts with the world around them. In this single lesson, we've only scratched the surface of human perception. There are entire courses, even entire degree programs, that focus on these principles. We'll give you some suggestions on some of the notes. So don't think we've given you a full view of the field. Instead, we hope we've given you just enough to start to keep human abilities in mind, and enough to know what to research as you start to learn to design interfaces.

2.5 Design Principles and Heuristics

Compiled by Shipra De, Summer 2017

Introduction to Design Principles



[MUSIC] Over the many years of HCI development, experts have come up with a wide variety of principles and heuristics for designing good interfaces. None of these are hard and fast rules like the law of gravity or something. But they're useful guidelines to keep in mind when designing our interfaces. Likely, the most popular and influential of these is Don Norman's six principles of design.



Larry Constantine and Lucy Lockwood have a similar set of principles of user interface design, with some overlaps but also some distinctions.



Jacob Nielsen has a set of Ten Heuristics for user interface design that can be used for both design and evaluation.



And while those are all interested in general usability, there also exists a set of seven principles called Principles of Universal Design. These are similarly concerned with usability, but more specifically for the greatest number of people. Putting these four sets together, we'll talk about 15 unique principles for interaction design.

The Sets



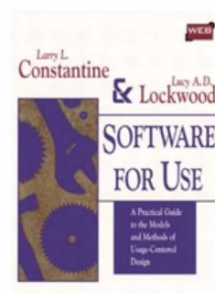
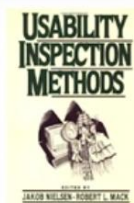
Don Norman's Six Design Principles

In this lesson we're going to talk about four sets of design principles. These aren't the only four sets, but they're the ones that I see referenced most often, and we'll talk about what some of the others might be at the end of the lesson. In this book, *The Design of Everyday Things*, Don Norman outlined his famous six design principles. This is probably the most famous set of design principles out there. The more recent versions of the book actually have a seventh principle. But that seventh principle is actually one of our entire lessons.



Jakob Nielsen's Ten Design Heuristics

Jakob Nielsen outlines ten design heuristics in his book, *Usability Inspection Methods*. Many of Norman's principles are similar to Nielsen's but there's some unique ones, as well. What's interesting, is Norman and Nielsen went into business together and formed the Nielsen Norman Group, which is for user experience training, consulting, and HCI research.



Larry Constantine's and Lucy Lockwood's Six Principles
















In their book *Software for Use*, Larry Constantine and Lucy Lockwood outline an additional six principles. Again, many of them overlap with these two, but some of them are unique.



Finally, Ronald Mace of North Carolina State University proposed Seven Principles of Universal Design. The Center for Excellence in Universal Design, whose mobile site is presented here, has continued research in this area. These are a little bit different than the heuristics and principles presented in the other three. While these three are most concerned with usability in general, universal design is specifically concerned with designing interfaces and devices that can be used by everyone regardless of age, disability, and so on. To make this lesson a little easier to follow, I've tried to merge these four sets of principles into one larger set capturing the overlap between many of them.



In this lesson, we'll go through these 15 principles. These principles are intended to distill out some of the overlap between those different sets.

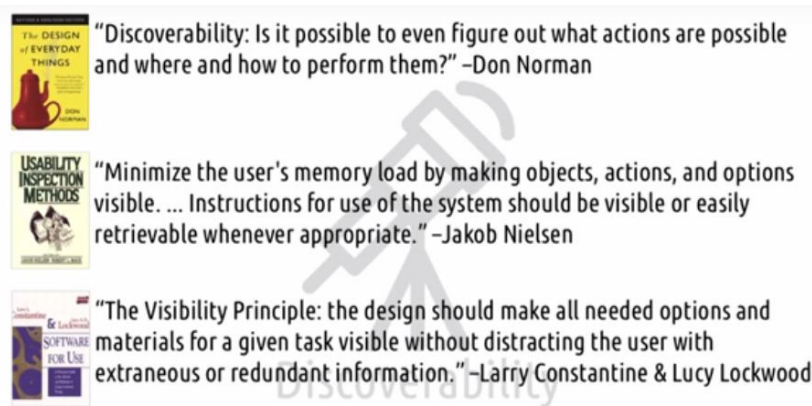
	Don Norman	Jakob Nielsen	Constantine & Lockwood	Universal Design
 Discoverability	✓	✓	✓	
 Simplicity		✓	✓	✓
 Affordances	✓			
 Mapping	✓	✓		
 Perceptibility	✓	✓		✓
 Consistency	✓	✓	✓	
 Flexibility		✓		✓
 Equity				✓
 Ease				✓
 Comfort				✓
 Structure			✓	
 Constraints	✓	✓		
 Tolerance		✓	✓	✓
 Feedback	✓	✓	✓	
 Documentation		✓		

This table shows those 15 principles, my names for each of them, and which sets they come from. Note that my 15 principals are just an abstraction or summary of these sets of principles. And you should make sure to understand the sets themselves as well. There are some subtle differences between the principles I've grouped together from different sets. And we'll talk about those as we go forward. And again, note that these aren't the only four sets of design principals out there. At the end of the lesson, we'll chat about a few more. And we'll also mention when others apply within this lesson as well.

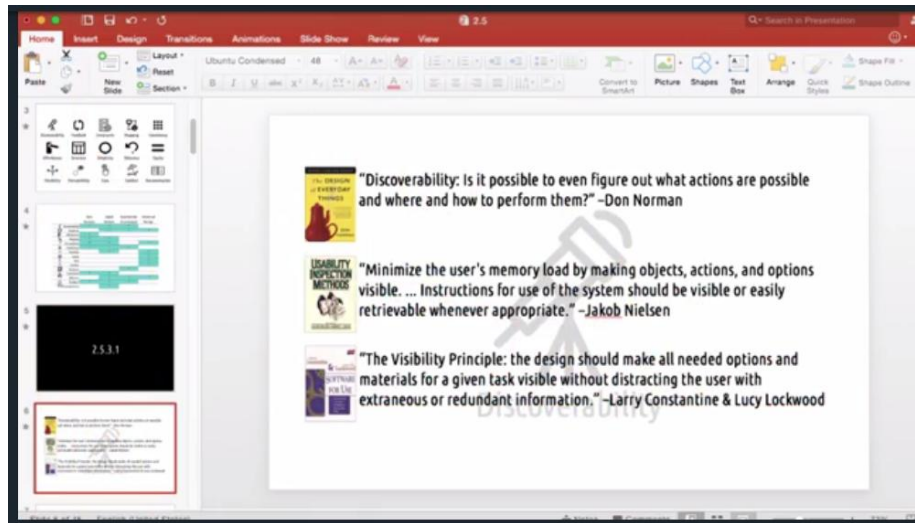
Discoverability



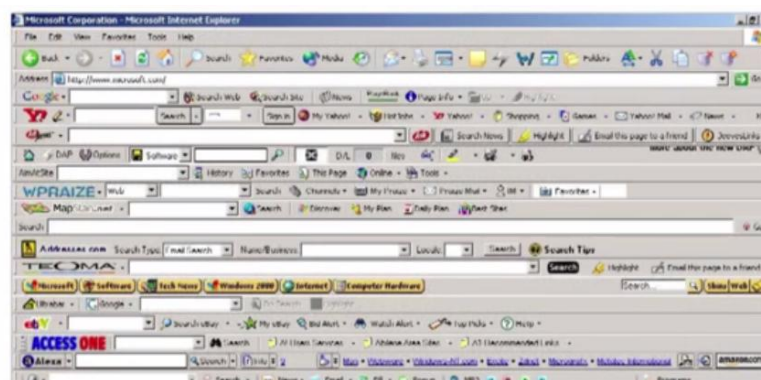
Our first principle is discoverability.



Don Norman describes it by asking, is it possible to even figure out what actions are possible and where and how to perform them? Nielsen has a similar principle. He advises us to minimize the user's memory load by making objects, actions, and options visible. Instructions for use of the system should be visible or easily retrievable whenever appropriate. In other words, when the user doesn't know what to do, they should be able to easily figure out what to do. Constantine and Lockwood have a similar principle called the visibility principle. The design should make all needed options and materials for a given task visible without distracting the user with extraneous or redundant information. The idea behind all three of these principles is that relevant function should be made visible, so the user can discover them as opposed to having to read about them in some documentation or learn them through some tutorial.



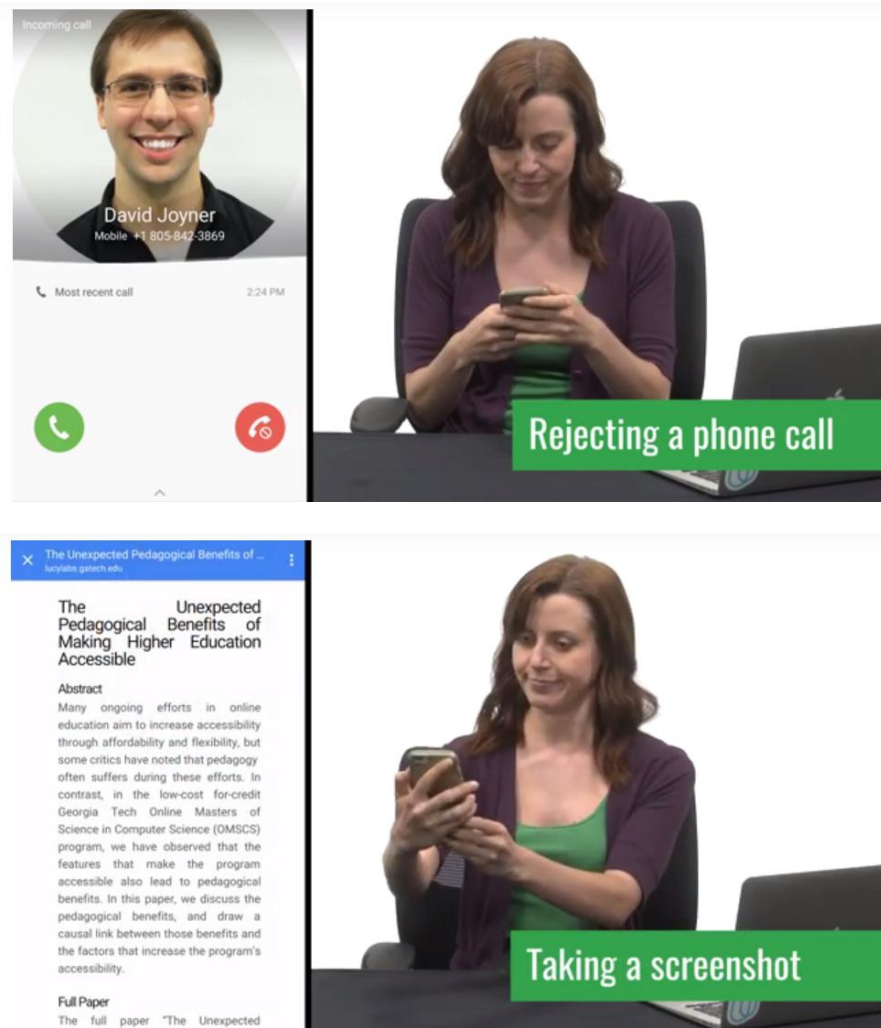
Let's take an example of this real quick. Here in PowerPoint, there are a number of different menus available at the top, as well as some toolbars. The effect here is that I can browse the different functions available to me. I can discover what's there. For Nielsen, this means that I don't have to remember all of these. I just have to recognize them when I see them in the tool bars. For example, I don't have to remember Arrange as some keyboard I have to type in manually to bring up some ideas about how I might arrange things. All I have to do is recognize Arrange as the right button when I see it. Now while this might be true at the application level, it's not often true at the operating system level, because the operating system doesn't command so much screen real estate all the time and probably for good reason. So for example, on a Mac, I can use Command Shift 4 to take a screen shot only of a certain area of my screen. However, the only way I know of to find that is to Google it or read it in a manual. It isn't discoverable or visible on it's own. And you might never even realize it's possible. So the principle of discoverability advocates that functions be visible to the user, so that they can discover them, rather than relying on them learning them elsewhere. I actually use a PC more than a Mac. And whenever I come back to my Mac after not using it for awhile, I have to Google that again. I know it's possible, but I never remember the command that actually makes it happen.

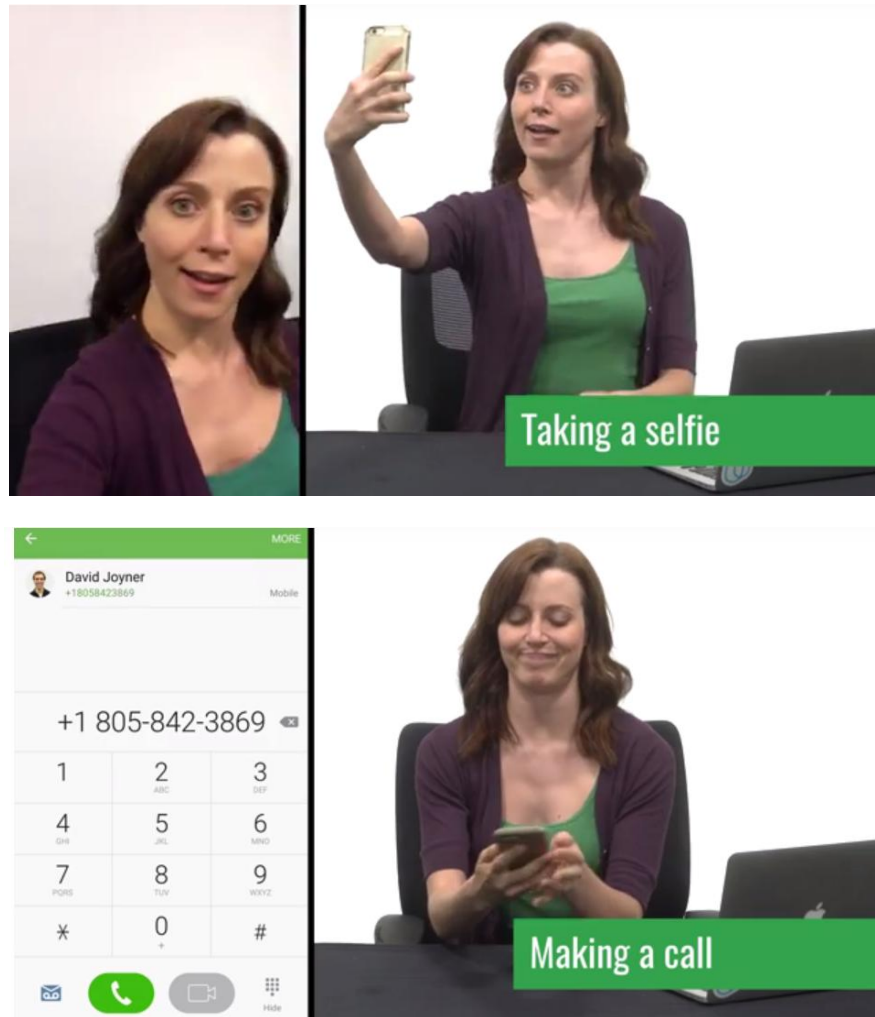


Constantine and Lockwood's principle of visibility would add on to this that we shouldn't get too crazy. We want to make functions discoverable, but that doesn't mean just throwing everything on the screen. We want to walk a line between discoverability, and simplicity

Quiz: Design Challenge: Discovering Gestures

Discoverability is one of the challenges for designing gesture-based interfaces. To understand this, let's watch Morgan do some ordinary actions with her phone.

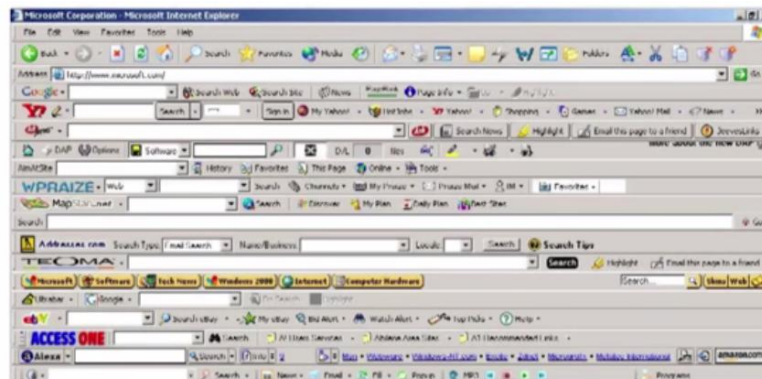




[MUSIC] [SOUND] [MUSIC] We just saw Morgan do four things with the phone. Reject a call, take a screenshot, take a selfie, and make a phone call. For each of those, this phone actually has a corresponding gesture that would have made it easier. She could have just turned the phone over to reject the call or said, shoot, to take the selfie. The problem is that these are not discoverable. Having a menu of voice commands kind of defeats the purpose of saving screen real estate and simplicity through gestures and voice commands. So, brainstorm a bit. How would you make these gesture commands more discoverable?

There's a lot of ways we might do this, from giving her a tutorial in advance, to giving her some tutoring in context. For example, we might use the title bar of the phone to just briefly flash a message letting the user know when something they've done could have been triggered by a gesture or a voice command. That way, we're delivering instruction in the context of the activity. We could also give a log of those so that they can check back at their convenience and see the tasks they could have performed in other ways.

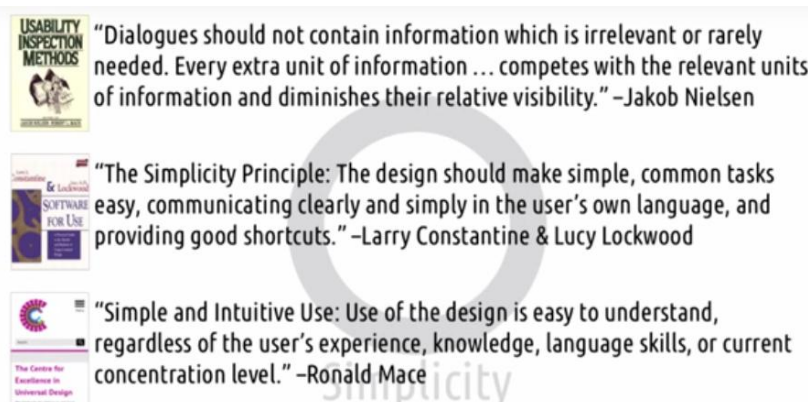
Simplicity



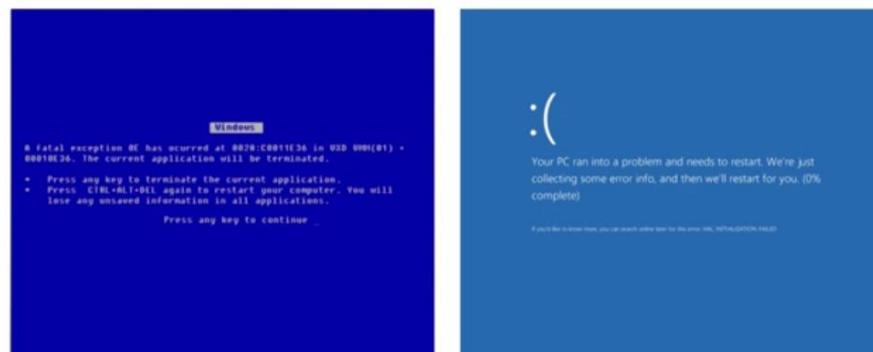
There often exists a tension between discoverability and simplicity. On the one hand, discoverability means you need to be able to find things. But how can you find them if they're not accessible or visible? That's how you get interfaces like this with way too many things visible. And ironically as a result, it actually becomes harder to find what you're looking for because there's so many different things you have to look at.



This is where the principle of simplicity comes in. Simplicity is part of three of our sets of principles, Nielsen's, Constantine and Lockwood's, and the universal design principles.



Nielsen writes specifically about dialogues. He says that the dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information competes with the relevant units of information, and diminishes their relative visibility, which we see with all those toolbars Constantine and Lockwood establishes as their simplicity principle. They say the design should make simple common tasks easy. Communicating clearly and simply in the user's own language and providing good shortcuts. Universal design is concerned with simplicity as well. Their principle of simple and intuitive use advocates the use of design easy to understand regardless of the user's experience, knowledge, language skills, or current concentration level. And in this principle you can see universal design's special concern with appealing to users of a variety of different levels of expertise, ages, disabilities, and so on.



Now in some ways, these principles are about designing interfaces but they cover other elements as well. One example of this is the infamous blue screen of death from the Windows operating systems. On the left we have the blue screen of death as it appeared in older versions of Windows. And on the right we have how it appears now on Windows 10. There are a lot of changes here. The blue is softer and more appealing. The description of the error is in plain language. But the same information is still provided, it's just de-emphasized. This is a nice application of Nielsen's heuristic. The user should only be given as much information as they need. Here, the information that most users would need, which is just that a problem occurred and here's how close I am to recovering from it, are presented more prominently than the detailed information that might only be useful to an expert.



Another interesting application of simplicity in action came when New York tried to create simpler signs to represent its allowed parking schedule.



Navigating the sign on the left is pretty much impossible. But it's pretty easy to interpret the one on the right. The universal design principle of simplicity is particularly interested in whether or not people of different experiences, levels of knowledge, or languages can figure out what to do. Navigating this sign requires a lot of cognitive attention and some language skills. Whereas I would hypothesize that even someone who struggles with English might be able to make sense of the sign on the right. These two signs communicate the same information, but while the one on the left requires a lot of cognitive load and language skills, the one on the right can probably be understood with little effort and little experience.

Affordances



One way to keep design both simple and usable is to design interfaces that by their very design tell you how to use them. Don Norman describes these as affordances. The design of a thing affords or hints at the way it's supposed to be used. This is also similar to the familiarity principle from Dicks, et al. This is extremely common in the physical world because the physical design of objects is connected to the physical function that they serve. Buttons are meant to be pressed. Handles are meant to be pulled. Knobs are meant to be turned. You can simply look at it and understand how you're supposed to use it.



Our next principle is the principle of affordances.

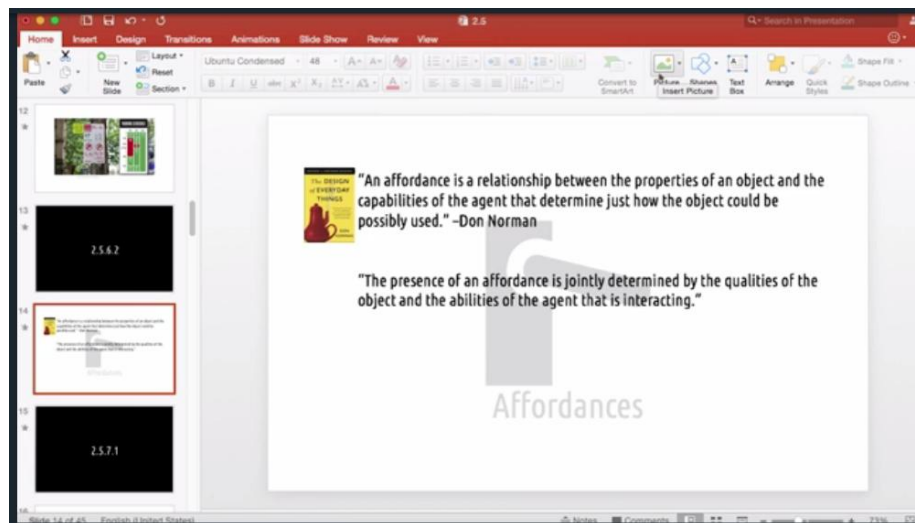


"An affordance is a relationship between the properties of an object and the capabilities of the agent that determine just how the object could be possibly used." –Don Norman

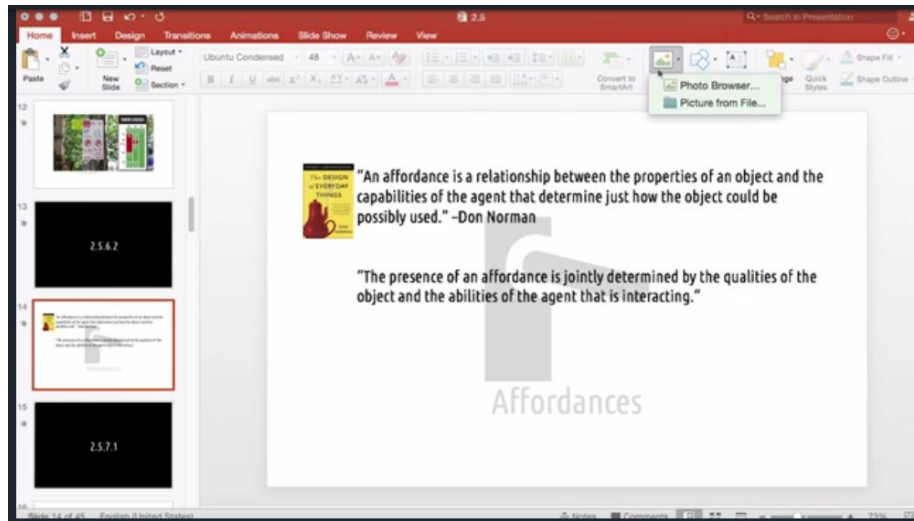
"The presence of an affordance is jointly determined by the qualities of the object and the abilities of the agent that is interacting."

Affordances

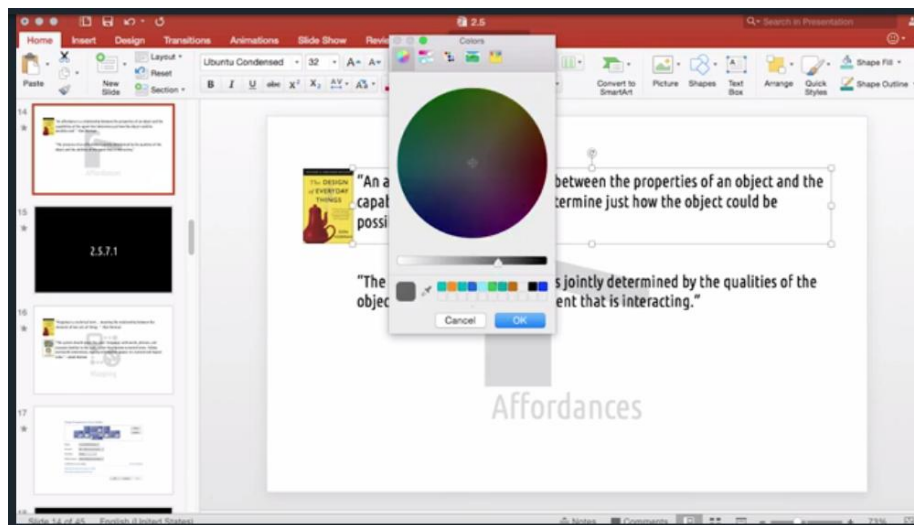
Norman writes that an affordance is the relationship between the properties of an object and the capabilities of the agent that determine just how the object could possibly be used. In other words, an object with an affordance basically tells the user, by its very design, how it's meant to be used. I use a door handle as the icon for this because a door handle is a great example of an affordance. You can look at it and understand that you're probably supposed to pull it down or push it up. The very design of it tells you how you're supposed to use it. Norman goes on to say, the presence of an affordance is jointly determined by the qualities of the object and the abilities of the agent that is interacting. In other words, an affordance for one person isn't an affordance for everyone. If you didn't grow up around door handles than maybe that door handle doesn't have an affordance for you the way it would for someone who grew up around that. Our affordances are defined by who the user is.



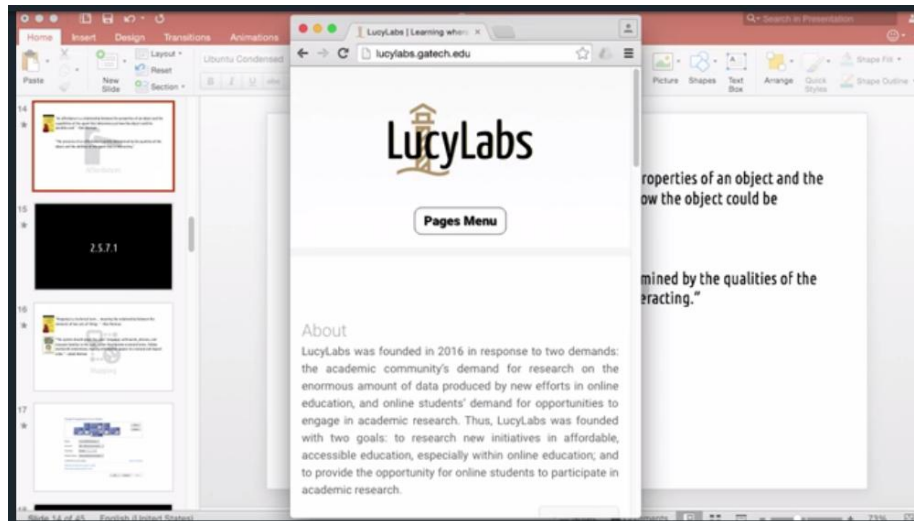
The challenge is that in the virtual computer world, there's no such inherent connection between the physical design and the function of an interface, the way you might often find in the real world. For example, when I mouse over a button in my interface, the style that appears around it makes it look like it's elevated. Makes it look like it's popping out of the interface. That affords the action of then pushing it down, and I know that I need to click it to push it down.



When I click it, it depresses, it gets darker, it looks like it's sunk into the interface. So here we've manually created an affordance that would exist in the real world. The design of this button hints at how it's supposed to be used. It hints at the fact that it's supposed to be pressed. So we have to create that naturalness manually. We can do that in a number of different ways. We could, for example, visualize the space of options.



Here this color picture does a good job of this. The horizontal line effectively shows us the list of options available to us. The placement of the dial suggests where we are now. And there's this kind of implicit notion that I could drag this dial around to change my color. We can also leverage **metaphors** or analogies to physical devices. You can imagine that if this content was presented like a book, I might scroll through it by flicking to the side, as if it's a page. You may have seen interfaces that work exactly like that. There's no computational reason why that should mean go to the next page or that should mean go back a page. Except that it makes sense in the context of the physical interface it's meant to mimic. We swipe in a book so let's swipe in a book-like interface.

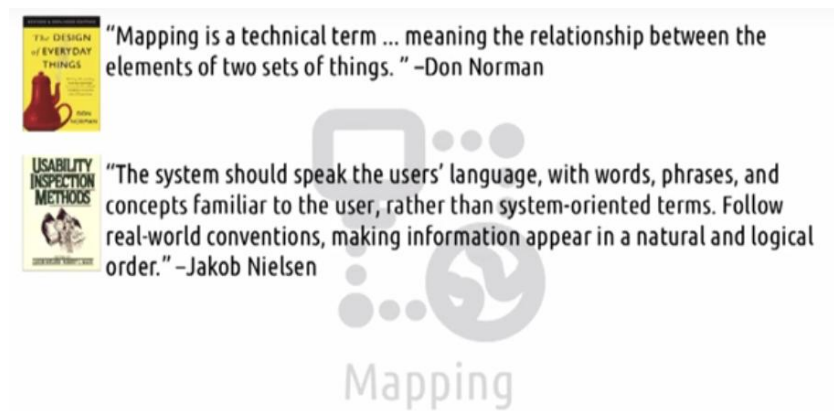


Of course, there are also actions in the virtual world that have no real world analogy like pulling up a menu on a mobile site. In that case we might use signifiers. **Signifiers** are a principle in Norman's more recent editions. Signifiers are in context instructions like arrows to indicate which way to swipe for a particular action. Or in this case a button labelled Menu to indicate how to pull up a menu. In this way we can kind of create our own affordances by creating an intuitive mapping between controls and their effects in the world, being consistent with what others have done in the past.

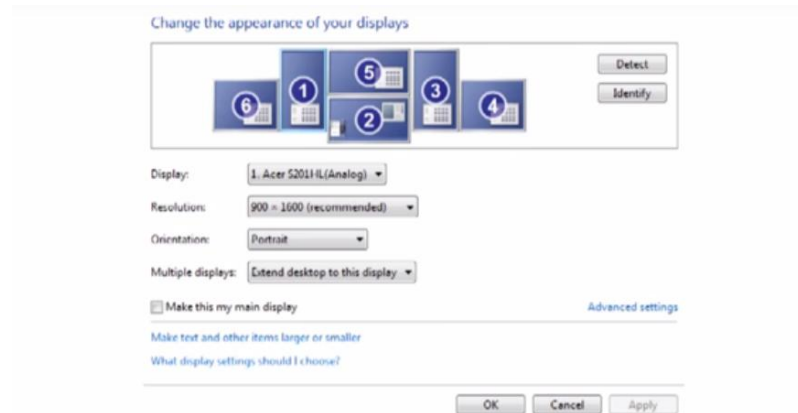
Mapping



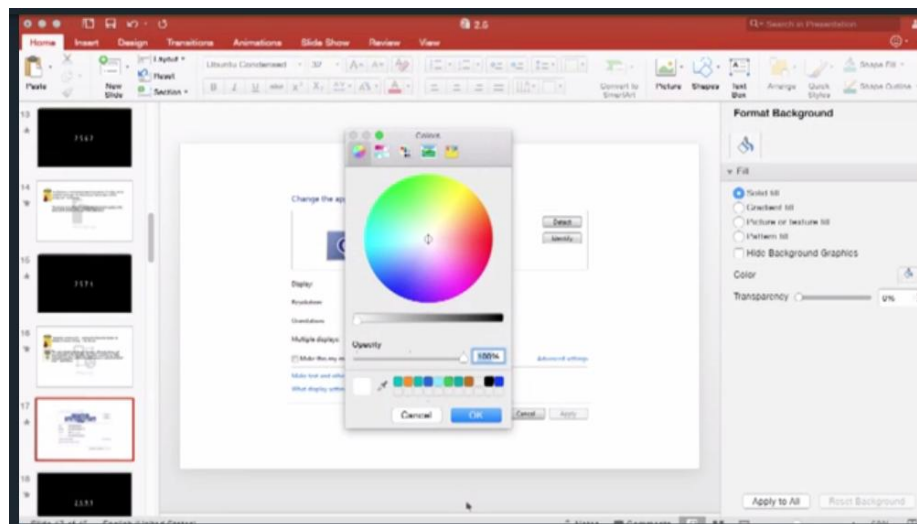
Norman and Nielsen both talk about the need for a mapping between interfaces and their effects in the world.



Norman notes that mapping is actually a technical term coming from mathematics that means a relationship between the elements of two sets of things. In this case, our two sets are the interface and the world. For example, these book icons might help you map these quotes to the books from which they were taken. Nielsen describes mapping by saying the system should speak the users' language, with words, phrases, and concepts that are familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order. A great example of this, is the fact that we call cut, copy, and paste, cut, copy and paste. Surely there could have been more technical terms like duplicate instead of copy. But using cut, copy, and paste forms a natural mapping between our own vocabulary and what happens in the system. Note that these two principles are subtly different, but they're actually strongly related. Nielsen's heuristic describes the general goal, while Norman's principle describes one way to achieve it. Strong mappings help make information appear in natural and logical order.



A great example of this is setting the arrangement of different monitors. This actually comes from my display in my home office. This visualization creates a very natural mapping between the way the system treats the monitors, and the way they're actually laid out in the world. If there's a mismatch, or if something doesn't make sense, I can easily look at this and map it with the arrangement of the monitors in front of me and figure out what's going on. This could instead be shown as just a list of pixel locations. And that would still present all the exact same information, but in a way that isn't as easily mapped out to the real world. Now, mappings and affordances are similar principles, but they have a clear and important difference.



We can see that difference in our color meter again. Affordances were about creating interfaces where their designs suggested how they're supposed to be used. The placement of this notch along this horizontal bar, kind of affords the idea that it could be dragged around. The horizontal bar visualizes the space which makes it seem like we could move that notch around to set our color. However, that design on its own wouldn't necessarily create a good mapping. Imagine, if instead of the bar fading from white to black, it was just white the entire way. It would still be very obvious how you're supposed to use it. But it wouldn't be obvious what the effect of using it would actually be. It's the present of that fade from white to black that makes it easier to see what will happen if I actually drag

that around. I can imagine it's going to make the colors fade from white to black. That creates the mapping to the effect of dragging it around on that meter. So mapping refers to creating interfaces where the design makes it clear what the effect will be when using them, not just creating interfaces where it's clear how you're supposed to use them. With this color meter, the arrangement of the controls makes clear what to do and the visualization underneath, makes it clear what will happen when I do it.

Quiz: Design Challenge: Mapping and Switches



A good example of the difference between affordances and mappings is a light switch. A light switch very clearly affords how you're supposed to use it. You're supposed to flip it. But these switches have no mapping to what will happen when I switch them. I can look at it and clearly see what I'm supposed to do. But I can't tell what the effect is going to be in the real world.



Contrast with the dials on my stove. There are four dials and each is augmented with this little icon that tells you which burner is controlled by that dial. So there's a mapping between the controls and the effects. So how would you redesign these light switches to create not only affordances but also mappings. If relevant, this one turns on the breakfast room light, this one turns on the counter light and this one turns on the kitchen light.

There are a few things we could do actually. Maybe we could put a small letter next to each light switch that indicates which light in the room that switch controls. K for kitchen, C for counter top, B for




breakfast room. Maybe we actually put icons that demonstrates which kind of light is controlled by each switch. So the counter top lights are kind of scone lights, so maybe we put an icon that looks like the counter top lights. But likely the easiest thing is actually the way the system really was designed. I just didn't notice it until I started writing this video. The lights from left to right in the room are actually controlled by the light switches from left to right on the wall. This switch controls the light over there. This switch controls the light right here. And this switch controls the light back there. That actually forms a pretty intuitive mapping.

Perceptability



Perceptability

Our next principle is perceptability. Perceptability refers to the user's ability to perceive the state of the system.

	<p>"The system should always keep users informed about what is going on, through appropriate feedback within reasonable time." –Jakob Nielsen</p>
	<p>"The design communicates necessary information effectively to the user, regardless of ambient conditions or the user's sensory abilities." –Ronald Mace</p>
	<p>"Feedback must be immediate. ... Feedback must also be informative. ... Poor feedback can be worse than no feedback at all, because it is distracting, uninformative, and in many cases irritating and anxiety-provoking" –Don Norman</p>

Nielsen states that the system should always keep users informed about what is going on, through appropriate feedback within reasonable time. That allows the user to perceive what's going on inside the system. Universal design notes that the design should communicate necessary information effectively to the user, regardless of ambient conditions or the user's sensory abilities. In other words, everyone using the interface should be able to perceive the current state. Note that this is also similar to Norman's notion of feedback. He writes that feedback must be immediate, must be informative, and that poor feedback can be worse than no feedback at all. But feedback is so ubiquitous, so general, that really, feedback could be applied to any principle we talk about in this entire lesson. So instead we're going to reserve this more narrow definition for when we talk about errors. And our lesson on feedback cycles covers the idea of feedback more generally. Things like light switches and oven dials, actually do this very nicely. I can look at a light switch and determine whether the system it controls is on or off, based on whether the switch is up or down. Same with the oven dial. I can immediately see where the dial is set. But there's a common household control, that flagrantly violates this principle of perceptability.



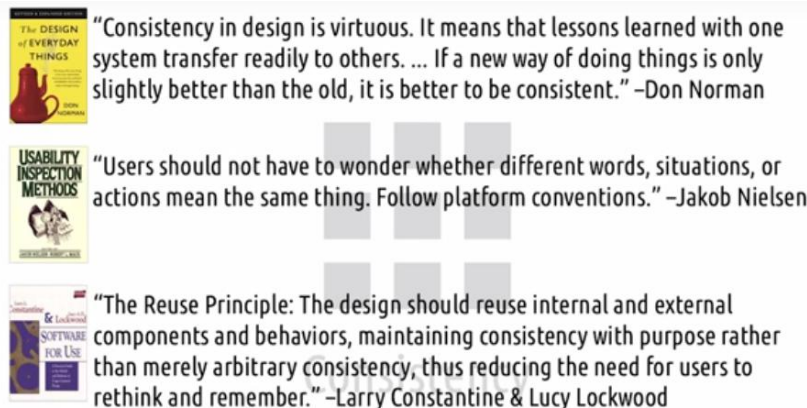
Here's our ceiling fan, you might have one just like it. It has two chains. One controls the light, one controls the fan speed. But both only when the switch on the wall is on. Now first, the mapping here is awful. There's no indication which control is which. But worse, the fan chain, which is this one, doesn't give any indication of which setting the fan is on currently. I don't honestly even know how many settings it has. [SOUND] I don't know if pulling it makes it go up and then down, up and then off, down and then off. Whenever I use it, I just pull it, wait ten seconds and see if I like the speed, and then pull it again. And this is all only if the wall switch is on. Now, of course people have resolved this with dials or other controls, and yet these dang chains still seem to be the most common approach despite this challenge of perceptibility.

Consistency



Consistency

Consistency is a principle from Norman, Nielsen, and Constantine and Lockwood. It refers to using controls, using visualizations, using layouts, using anything we use in our interface design consistently, across both the interfaces that we design and what we design more broadly as a community.

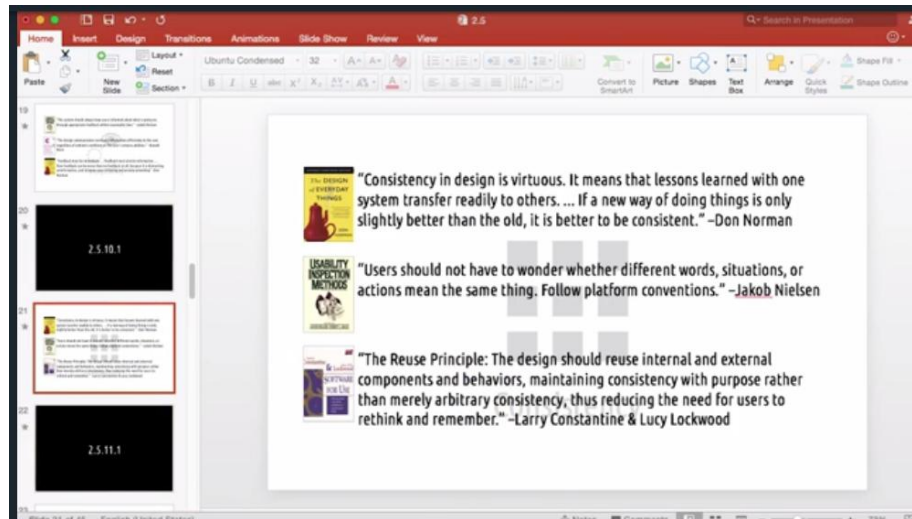


Norman writes that consistency in design is virtuous. It's a powerful word, there. It means that lessons learned with one system transfer readily to others. If a new way of doing things is only slightly better than the old, it's better to be consistent. Of course there will be times when new ways of doing things will be significantly better than the old. That is how we actually make progress, that is how we advance. If we are only making tiny little iterative improvements, it might be better to stick to the old way of doing things, because users are used to it. They are able to do it more efficiently. Nielsen writes that users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions. In other words, be consistent with what other people have done on the same platform, in the same domain, and so on. Constantine and Lockwood describe consistency as reuse. They say the design should reuse internal and external components and behaviors, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember. That means that we don't have to be consistent with things that don't really impact what the user knows to do. The color of the window, for example, isn't going to change whether the user understands what the word copy means in the context of an interface. But changing the word copy to duplicate might force users to actually rethink and remember what that term means. In some cases, that might be a good thing. If duplicate actually does something slightly different than copy, then

changing that would force our users to understand that they're doing something different. But if we're doing the same thing, it's important to maintain consistency, so the user doesn't have to think as much and can focus on the task at hand, instead of on our interface. The general idea across all of these is we should be consistent both within and across interfaces to minimize the amount of learning the user needs to do to learn our interface. In this way, we create affordances on our own. Unlike traditional physical affordances, there's no physical reason for the interface to be designed a certain way, but by convention, we create expectations for users and then fulfill those expectations consistently.

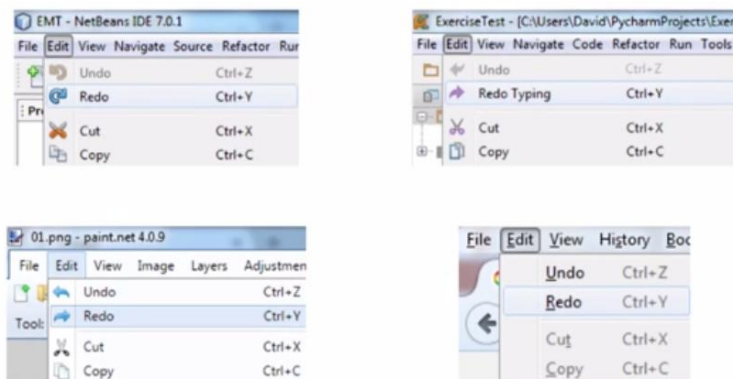


One great example of following these conventions are the links we use in text on most websites. For whatever reason, an early convention on the internet was for links to be blue and underlined. Now when we want to indicate to users that some text is clickable, what do we do? Generally, we might make it blue and underline it. Sometimes we change this, as you can see here. Underlining has actually fallen out of fashion in a lot of places and now we just use the distinct text color to indicate a link that can be clicked. On some other sites, the color itself might be different. It might be red against blue text instead of blue against black. But the convention of using a contrasting color to mark links has remained and the most fundamental convention is still blue underlines. Again, there's no physical reason why links need to be blue, or why they even need to be a different text color at all. But that convention helps users understand how to use our interfaces. If you've used the internet before and then visit Wikipedia for the first time, you'll understand that these are links without even thinking about it. Most of the interfaces we design will have a number of functions in common with other interfaces. So by leveraging the way things have been done in the past, we can help users understand our interfaces more quickly.

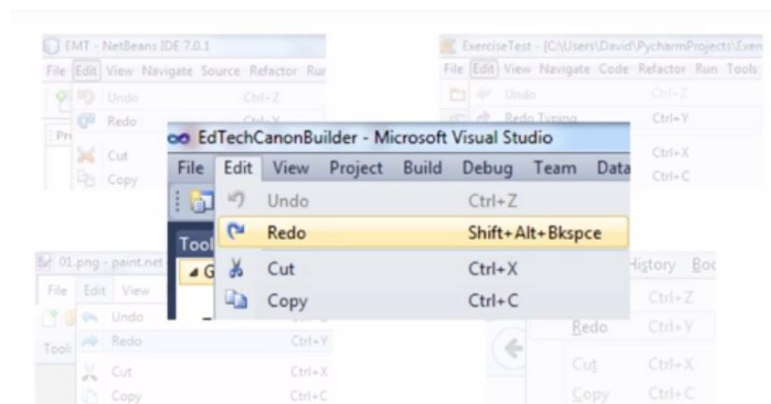


Other common examples of consistency in interface design would include things like using consistent hot keys for things like copy, paste, and select all. Ordering the menus, file, edit, view, etc. Putting options like save and open under file. We don't even tend to think about these things when we're using an interface until we encounter one that defies our conventions. And yet, someone has to consciously decide to be consistent with established norms. This is an example of design becoming invisible. When people do it right, we don't notice they did it at all. When people put those options in the right places, we don't even think about it. But when people put them in the wrong places, it's pretty jarring and startling.

Consistency: The Curious Case of Ctrl+Y



One of my favorite examples of how consistency matters comes from Microsoft's Visual Studio development environment. And to be clear: I adore Visual Studio, so I'm not just piling onto it. As you see here, in most interfaces, Ctrl+Y is the 'redo' hotkey. If you hit undo one too many times, you can hit Ctrl+Y to 'redo' the last 'undone' action.

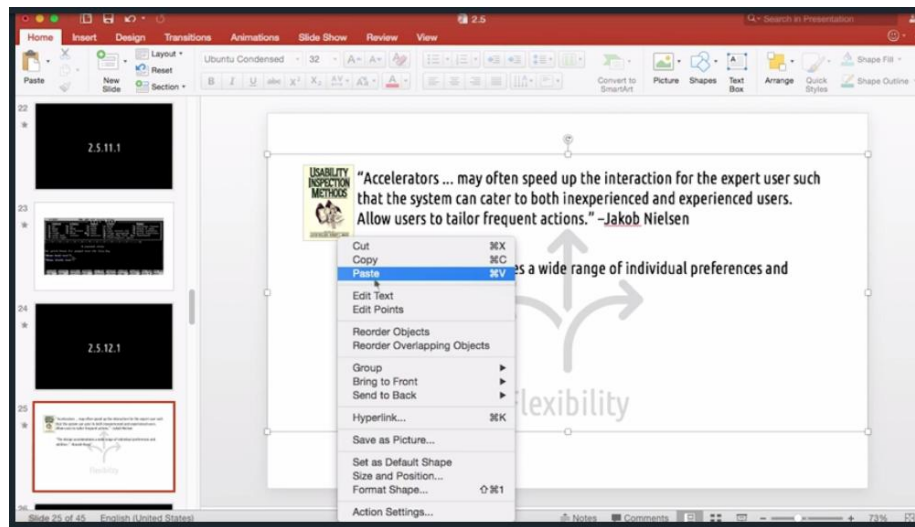


But in Visual Studio, by default it's... Shift+Alt+Backspace. What?! And what's worse than this is Ctrl+Y is the 'delete line' function, which is a function I had never even heard of before Visual Studio. So, if you're pressing Ctrl+Z a bunch of times to maybe rewind the changes you've made lately, and then you press Ctrl+Y out of habit because that's what every other interface uses for redo, the effect is that you delete the current line instead of redoing anything and that actually makes a new change, which means you lose that entire tree of redoable actions. Anything you've undone now can't be recovered. It's infuriating. And yet, it isn't without its reasons. The reason: consistency.



Ctrl+Y was the hotkey for delete function in WordStar, one of the very first word processors, before Ctrl+Y was the hotkey for the more general redo function. There wasn't even a redo function back then. I've heard that Y in this context stood for 'Yank', but I don't know how true that is. But Ctrl+Y had been used to delete a line for from WordStar all the way through Visual Basic 6, which was the predecessor of Visual Studio. So, in designing Visual Studio, Microsoft had a choice: be consistent with the convention from WordStar and Visual Basic 6, or be consistent with the convention they were using in their other interfaces. They chose to be consistent with the predecessors to Visual Studio, and they've stayed consistent with that ever since. So in trying to maintain the consistency principle in one way, they actually violated it in another way. So if you try to leverage the consistency principle you're going to encounter some challenges. There may be multiple conflicting things with which you want to be consistent. There may be questions about whether a certain change is worth the drop in consistency. These are things to test with users, which we'll talk about in the next unit.

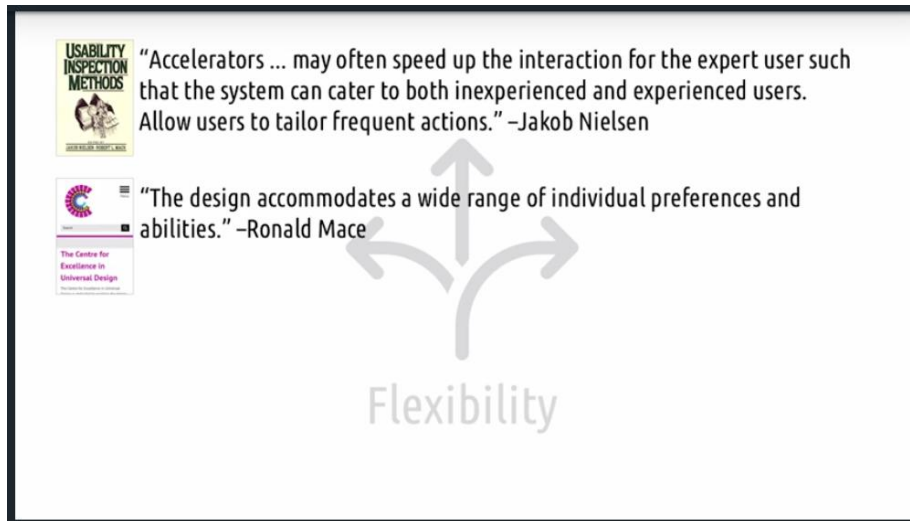
Flexibility



Depending on your expertise with the computers, there's a strong chance you've found yourself on one side or the other of the following exchange. Imagine one person is watching another person use a computer. The person using the computer repeatedly right-clicks and selects Cut to cut things. And then right-clicks and selects Paste to paste them back again. The person watching insists that they can just use Ctrl+X and Ctrl+V. The person working doesn't understand why the person watching cares. The person watching doesn't understand why the person working won't use the more efficient method. And in reality, they're both right.



This is the principal of flexibility. These two options are available because of the principle of flexibility from both Nielsen's heuristics and the principles of universal design.



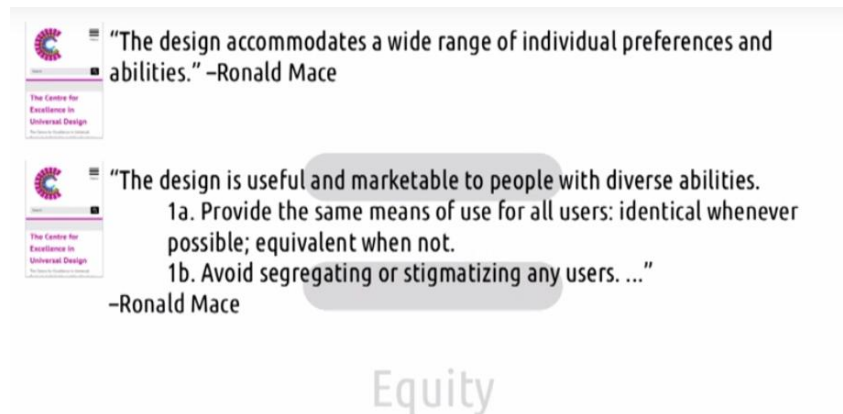
Nielsen specifically comments on the use of accelerators, which are hot keys. He says that accelerators may often speed up the interaction for the expert user, such that the system can cater to both inexperienced and experienced users. He advises that we allow users to tailor frequent actions. Universal design says something similar. They noted the design should accommodate a wide range of individual preferences and abilities. Another set of design principles from Dix et al also have a category of principles called flexibility principles, that advocate user customizability in supporting multiple designs for the same task. Here, Nielsen is most interested in catering to both novice and expert users, while the principles of universal design are more interested in accommodating users of various abilities and preferences. But the underlying principle here is the same, flexibility. Wherever possible, we should support the different interactions in which people engage naturally, rather than forcing them into one against their expertise or against their preference.

Equity

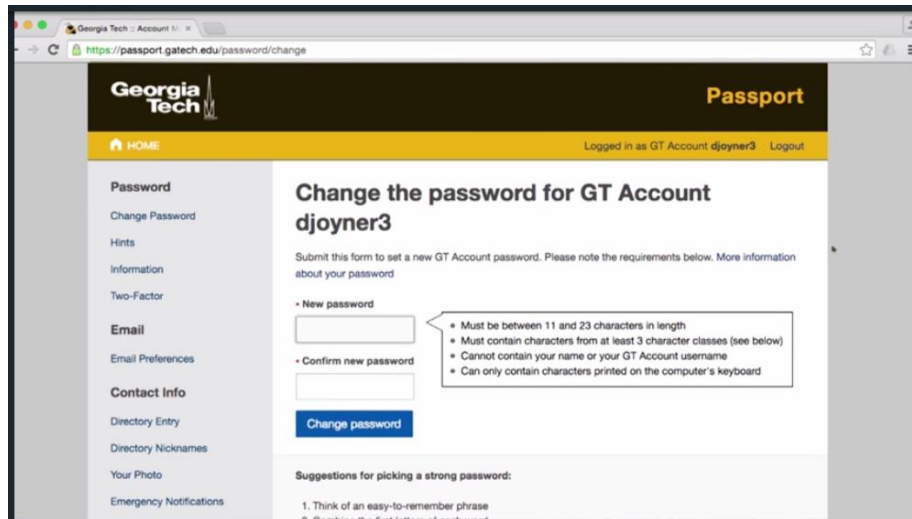


Equity

The principle of flexibility in some ways appears to clash with the principle of equity. But both come from the principles of universal design.



The principle of flexibility said the design should accommodate a wide range of individual preferences and abilities. But the principle of equity says the design is useful and marketable to people with diverse abilities, and it goes on to say we should provide the same means for all users, identical whenever possible and equivalent when not. And we should avoid segregating or stigmatizing any users. Now, in some ways, these systems might compete. This says we should allow every user to use the system the same way, whereas this one says that we should allow different, flexible methods of interacting with the system. In reality, though, these are actually complementary of one another. Equity is largely about helping all users have the same user experience, while flexibility might be a means to achieve that. For example, if we want all our users to enjoy using our interface, keeping things discoverable for novice users and efficient for expert users allows us to accommodate a wide range of individual preferences and abilities. User experience in this instance means treating every user like they're within the target audience and extending the same benefits to all users, including things like privacy and security. We might do that in different ways, but the important note is that the experience is the same across all users. That's what equity is about.



One good example of equity in action are the requirements for password resets. We want to design a system so that both expert and novice users experience the same level of security. Security is part of the user experience. Now, experts, we would assume, understand the value of a complex password. Novices might not. So if we don't have requirements around passwords, novices might not experience the same level of security as experts. So password requirements can be seen as a way of making sure the user experience across novices and experts is the same with regard to security. In the process, we might actually frustrate novice users a little bit. You could actually see this as a violation of the flexibility principle, that we're not flexibly accommodating in the kind of interaction that novices want to have. But the important thing, is we're extending the same security benefits to everyone, and that's equitable treatment. And that's also a good example of how at times, the different design principles will appear to compete, and you have to decide what the best approach is going forward.

Ease and Comfort

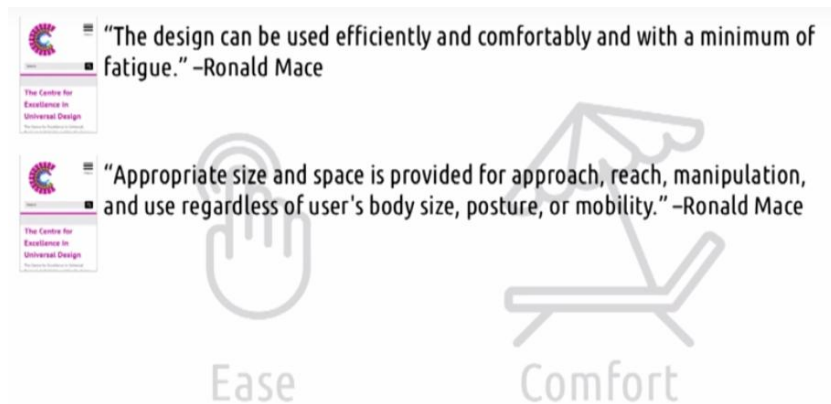


Ease

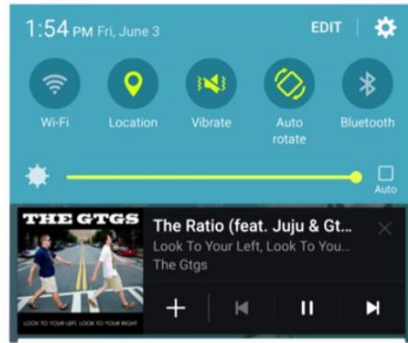


Comfort

Ease and comfort are two similar ideas that come from the principles of universal design. And they also relate to equitable treatment, specifically in terms of physical interaction.



The ease principle, which interestingly uses the word comfort, says the design can be used efficiently and comfortably and with a minimum amount of fatigue. The comfort principle notes that appropriate size and space is provided for approach, reach, manipulation and use regardless of the user's body size, posture or mobility. Now, in the past, these principles didn't have an enormous amount of application to HCI. Because we generally assume that the user was sitting at their desk with a keyboard and a monitor. But as more and more interfaces are becoming equipped with computers, we'll find HCI dealing with these issues more and more. For example, the seat control in your car might now actually be run by a computer that remembers your settings and restores them when you get back in the car. That's an instance of HCI trying to improve user ease and comfort in a physical area.



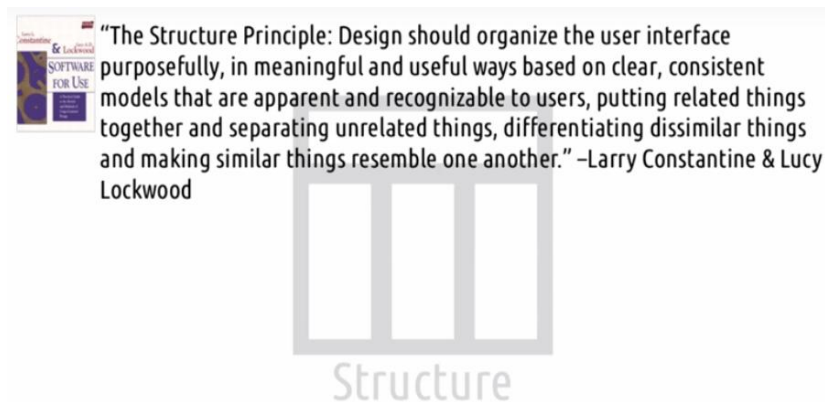
Mobile interfaces are great examples of this as well. When deciding the size of buttons on a mobile interface, we should take into consideration that some users might have tremors that make it more difficult to interact precisely with different buttons. As we get into areas like wearable computing and virtual reality, these issues of ease and comfort are going to become more and more pertinent.

Structure



Structure

The structure principle is concerned with the overall architecture of a user interface. In many ways, it's more closely related to the narrower field of user interface design than HCI more generally.



It comes from Constantine and Lockwood and they define it as their structure principle which says that design should organize the user interface purposefully, in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another. That's a long sentence. But what it really says is we should organize our user interfaces in ways that helps the user's mental model match the actual content of the task. What's interesting to me about the structure principle is that it borrows from a form of UI design that predates computers considerably. We find many of the principles we learned in designing newspapers and textbooks apply nicely to user interfaces as well.



For example, this is the Wall Street journal print edition from several years ago. And here's the Wall Street Journal web site. Notice many of the structural principles present in the print version are present in the website as well. Now part of that is for brand consistency. But part of it is because the very same ideas we use in developing magazines and newspapers still apply to the development of websites. Lines and spacing set apart different categories of articles. Headlines are still in bold while the article text is smaller. Now there are of course differences because web sites can for example link to articles while physical papers cannot, which is why these are all shorter than the actual paper are. But we see a lot of the same principles at work in the website that were at work in the physical layout. Those are largely parts of structure: organizing things in intuitive ways that groups similar parts, separates dissimilar ones, and helps the user navigate what they're consuming.

Constraints



In designing user interfaces our goal is typically to make the interface usable. And a big part of usability is accounting for user error.



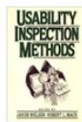
"There's no such thing as user error!"



Many design theorists argue that there's actually no such thing as user error. If the user commits an error it was because the system was not structured in a way to prevent or recover from it. And I happen to agree with that. Now, one way we can avoid error is by preventing the user from performing erroneously in the first place. This is the idea of constraints. Constraining the user to only performing the correct actions in the first place.



"Constraints are powerful clues, limiting the set of possible actions. The thoughtful use of constraints in design lets people readily determine the proper course of action, even in a novel situation." –Don Norman



"Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action." –Jakob Nielsen

Constraints

On constraints, Norman writes, the constraints are powerful clues, limiting the set of possible actions. The thoughtful use of Constraints in design lets people readily determine the proper course of action, even in a novel situation. And remember, designing so that users are immediately comfortable in novel situations, is one of the goals of good user interface design. Nielsen notes that even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action. For example, if our application was prone to users accidentally closing it when they don't mean to, ask them when it's about to close if that is actually what they meant to do. Both of these approaches refer to the need to stop faulty user input before it's even received. This is a principle you might already encounter a lot.

Georgia Tech Passport

HOME Logged in as GT Account djoyner3 Logout

Password

- Change Password
- Hints
- Information
- Two-Factor

Email

- Email Preferences

Contact Info

- Directory Entry
- Directory Nicknames

Change the password for GT Account djoyner3

Submit this form to set a new GT Account password. Please note the requirements below. More information about your password

• New password

• Confirm new password

Change password

- Must be between 11 and 23 characters in length
- Must contain characters from at least 3 character classes (see below)
- Cannot contain your name or your GT Account username
- Can only contain characters printed on the computer's keyboard

Our password reset screen actually does this pretty well. First, it shows us the constraints under which we're operating right there visibly on the screen so we're not left guessing as to what we're supposed to be doing. Then, as we start to interact, it tells us if we're violating any of those constraints.

Georgia Tech Passport

HOME Logged in as GT Account djoyner3 Logout

Password

- Change Password
- Hints
- Information
- Two-Factor

Email

- Email Preferences

Contact Info

- Directory Entry
- Directory Nicknames

Change the password for GT Account djoyner3

Submit this form to set a new GT Account password. Please note the requirements below. More information about your password

• New password

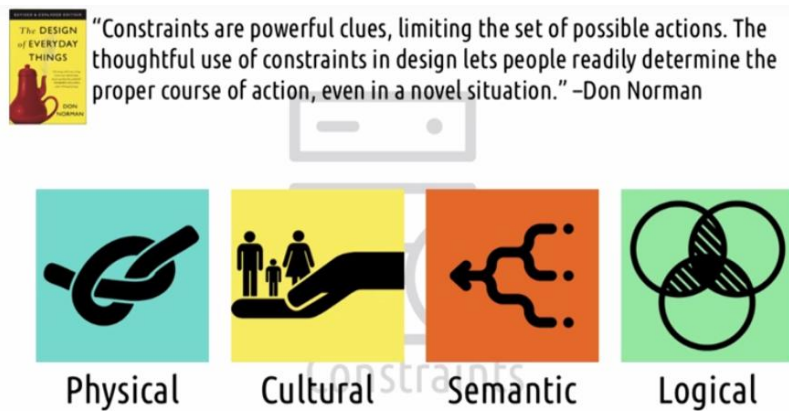
• Confirm new password

Change password

- ❌ Must be between 11 and 23 characters in length
- ❌ Must contain characters from at least 3 character classes (see below)
- ✅ Cannot contain your name or your GT Account username
- ✅ Can only contain characters printed on the computer's keyboard

So if I were just try to make my password the incredibly common, 1234, it immediately tells me that password isn't long enough and it doesn't represent enough character classes. Now, obviously, it can't prevent me from entering 1234 in the first place, since maybe that's along the way to a longer more valid password. But it's visualizing those constraints so that instead of submitting and getting frustrated when it didn't tell me I was doing it wrong until I'd actually done it, it actually tells me in right in the context of doing it that I'm not on the right track. This is kind of a soft constraint. It doesn't prevent me from doing something but it tells me while I'm doing it that I'm doing it wrong. A harder constraint goes along with that last bullet, can only contain characters printed on the computer's keyboard. Right now, I'm trying to paste in a character that isn't on the computer keyboard and it's just not showing it all together. It's a hard constraint against the inputting characters that aren't allowed. So it's preventing me from putting an invalid input in the first place. So in their simplest form, constraints can be described as preventing the user from putting in input that wasn't going to work anyway.

Norman's Four Types of Constraints



Norman takes us a step further though, when he breaks down constraints into four sub-categories. These aren't just about preventing wrong input. They're also about insuring correct input. They're about making sure the user knows what to do next. Physical constraints are those that are literally physically prevent you from performing the wrong action. A three-prong plug, for example, can only physically be inserted in one way, which prevents mistakes. USB sticks can only be physically inserted one way all the way. But the constraint doesn't arise until you've already tried to do it incorrectly. You can look at a wall outlet and understand if you're trying to put it incorrectly. But it's harder to look at a USB and know whether you're trying to insert it the right way. A second kind is a cultural constraint. These are those rules that are generally followed by different societies, like facing forward on escalators, or forming a line while waiting. In designing we might rely on these, but we should be careful of intercultural differences. A third kind of constraint is a semantic constraint. Those are constraints that are inherent to the meaning of a situation. They're similar to affordances in that regard. For example, the purpose of a rear view mirror is to see behind you. So therefore, the mirror must reflect from behind, it's inherent to the idea of a rear view mirror, that it should reflect in a certain way. In the future that meaning might change, autonomous vehicles might not need mirrors for passengers, so the semantic constraints of today, might be gone tomorrow. And finally the fourth kind of constraint is a logical constraint. Logical constraints are things that are self-evident based on a situation, not just based on the design of something like a semantic constraint, but based on the situation at hand. For example, imagine building some furniture. When you reach the end, there's only one hole left, and only one screw. Logically, the one screw left is constrained to go in the one remaining hole. That's a logical constraint.

Quiz: Reflections: Constraints

A lot of the principles we talk about are cases where you might never even notice if they've been done well. There are principles of invisible design, where succeeding allows the user to focus on the underlying tasks. But constraints are different. Constraints actively stand in the user's way and that means they've become more visible. That's often a bad thing, but in the case of constraints it serves the greater good. Constraints might prevent users from entering invalid input or force users to adopt certain safeguards. So of all the principles we've discussed, this might be the one you've noticed. So take a second, and think. Can you think of any times you've encountered interfaces that had constraints in them?

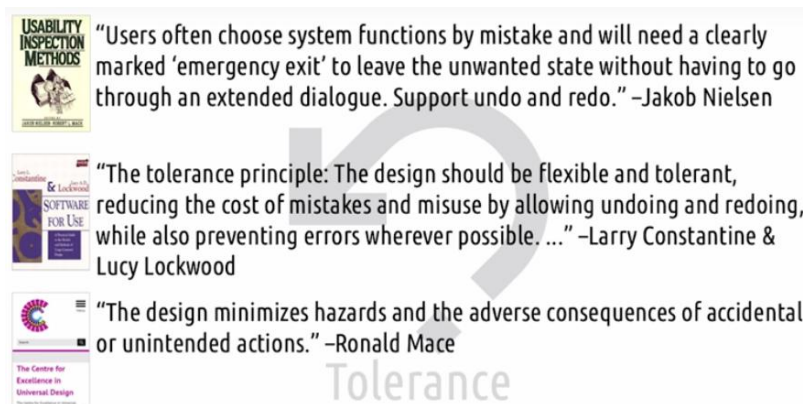


I have kind of an interesting example of this. I can't demonstrate it well because the car has to be in motion, but on my Leaf there's an option screen, and it lets you change the time and the date, and some other options on the car. And you can use that option screen until the car starts moving. But at that point, the menu blocks you from using it, saying you can only use it when the car is at rest. That's for safety reasons. They don't want people fiddling with the option screen while driving. What makes it interesting, though, is it's a constraint that isn't in the service of usability, it's in the service of safety. The car is made less usable to make it more safe.

Tolerance



We can't constrain away all errors all the time though. So there are two principles for how we deal with errors that do occur, feedback and tolerance. Tolerance means that users shouldn't be at risk of causing too much trouble accidentally.



For this, Nielsen writes that users often choose system functions by mistake. And will need a clearly marked emergency exit to leave the unwanted state without having to go through an extended dialogue. Support undo and redo. For Constantine and Lockwood this is the tolerance principle. They write the design should be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing, while also preventing errors wherever possible. It should be becoming clearer, why that control Y issue with Visual Studio, was so significant. Undo and redo are fundamental concepts of tolerance. And that control Y issue, where control Y removes the line in Visual Studio gets in the way of redo allowing us to recover from mistakes. For Dix et al, this is the principle of recoverability. Now, Nielsen's definition is most interested in supporting user experimentation. The system should tolerate users poking around with things. Universal Design simply says, the design minimizes hazards and the adverse consequences of accidental or unintended actions. Dix et al also refers to this as the principle of recoverability. Now Nielsen's definition is most interested in supporting user experimentation. They system should tolerate users poking around with things. That actually enhances the principle of discoverability because if the user feels safe experimenting with things they're more likely to discover what's available to them. The principles from Constantine and Lockwood and the principles of Universal Design are more about recovering from traditional mistakes.

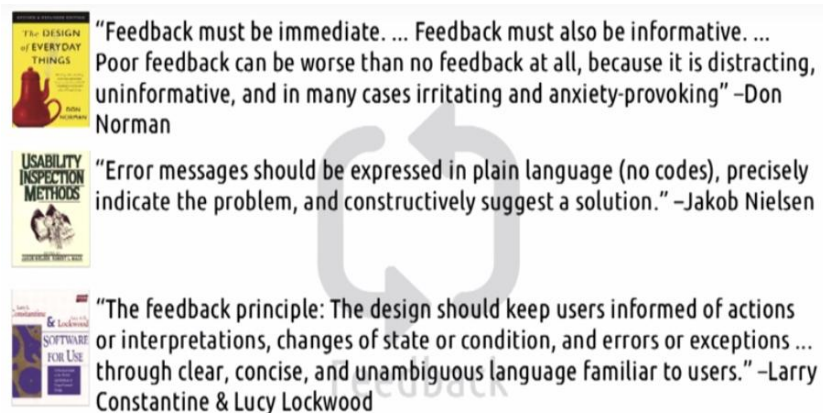
“A computer shall not harm your work or, through inactivity, allow
your work to come to harm.” –Jef Raskin

Jef Raskin poses this as a more humorous law of interface design. A computer shall not harm your work or through inactivity, allow your work to come to harm. So we first have to make sure that the system prevents the user from doing too much damage accidentally. Either by constraining them away from making those mistakes, or allowing an easy way to recover after those mistakes have been made.

Feedback



Second, the system should give plenty of feedback so that the user can understand why the error happened and how to avoid it in the future.



Norman writes that feedback must be immediate and it must be informative. Poor feedback can be worse than no feedback at all. Because it's distracting, uninformative, and, in many cases, irritating and anxiety-provoking. If anything has ever described the classic Windows Blue Screen of Death, it's this. It's terrifying. It's bold. It's cryptic. And it scares you more than it informs you. Nielsen writes that error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution. Note this tight relationship with recoverability. Not only should it be possible to recover from an error, the system should tell you exactly how to recover from an error. That's feedback in response to errors. For Constantine and Lockwood, this is the feedback principle. The design should keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions... through clear, concise, and unambiguous language familiar to users. Again, the old Windows blue screen of death doesn't do this very well. Because the language is not familiar, it's not concise, and it doesn't actually tell you what the state or condition is. The new one does a much better job of this. Notice as well that Norman, Constantine, and Lockwood are interested in feedback more generally, not just in response to errors. That's so fundamental that we have an entire lesson on feedback cycles that really is more emblematic of the overall principle of feedback. Here we're most interested in feedback in response to errors, which is a very important concept on its own.

Documentation



Documentation

Finally, Nielsen has one last heuristic regarding user error, documentation. I put this last for a reason, one goal of usable design is to avoid the need for documentation altogether. We want users to just interact naturally with our interfaces. In modern design, we probably can't rely on users reading our documentation at all unless they're being required to use our interface altogether.



“Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.” –Jakob Nielsen



Documentation

And Nielsen generally agrees. He writes that even though it's better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on user's task, list concrete steps to be carried out, and not be too large. I feel modern design as a whole has made great strides in this direction over the past several years. Nowadays, most often, when you use documentation online or wherever you might find it, it's framed in terms of tasks. You input what you want to do, and it gives you a concrete list of steps to actually carry it out. That's a refreshing change compared to older documentation, which was more dedicated to just listing out everything a given interface could do without any consideration to what you were actually trying to do.

Exploring HCI: Design Principles and Heuristics

We've talked about a bunch of different design principles in this lesson. How these design principles apply to your design tasks will differ significantly based on what area you're working in. In gestural interfaces, for example, constraint present a big challenge because we can't physically constrain our users movement. We have to give them feedback or feedforward in different ways. If we're working in particularly complex domains, we have to think hard about what simplicity means. If the underlying task is complex, how simple can and should the interface actually be? We might find ourselves in domains with enormous concerns regarding universal design. If you create something that a person with a disability can't use, you risk big problems, both ethically and legally. So take a few moments and reflect on how these design principles apply to the area of HCI that you've chose to investigate.

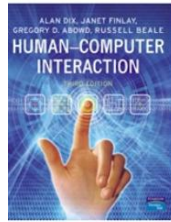
Other Sets of Principles



So I've attempted to distill the 29 combined principles from Norman, from Nielsen, Constantine, Lockwood, and the Institute for Universal Design into just these 15.

	Don Norman	Jakob Nielsen	Constantine & Lockwood	Universal Design
Discoverability	✓	✓	✓	
Simplicity		✓	✓	✓
Affordances	✓			
Mapping	✓	✓		
Perceptibility	✓*	✓		✓
Consistency	✓	✓	✓	
Flexibility		✓		✓
Equity				✓
Ease				✓
Comfort				✓
Structure			✓	
Constraints	✓	✓		
Tolerance		✓	✓	✓
Feedback	✓	✓	✓	
Documentation		✓		

Here you can see where each of these principles comes from. I do recommend reading the original four lists to pick up on some of the more subtle differences between these principles that I grouped together, especially perceptibility, tolerance, and feedback. Note also that in more recent editions, Norman has one more principle: conceptual models. That's actually the subject of an entire lesson in this course. These also certainly aren't the only four sets of design principles. There are several more.

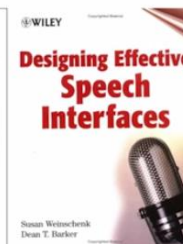
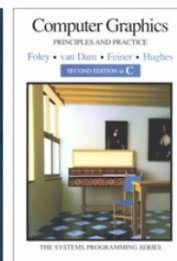
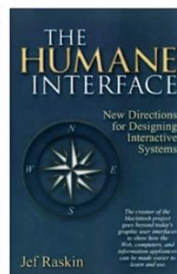


1. Learnability

2. Flexibility

3. Robustness

For example, Dix, Finlay, Abowd, and Beale propose three categories of principles: Learnability for how easily a user can grasp an interface, Flexibility for how many ways an interface can be used, and Robustness for how well an interface gives feedback and recovers from errors.



...and many more!

We'll talk about their learnability principles when we talk about mental models. Jill Gerhardt-Powels has a list of principles for cognitive engineering and especially at reducing cognitive load. Her list has some particularly useful applications for data processing and visualization. In the Human Interface, Jeff Raskin outlines some additional revolutionary design rules. I wouldn't necessarily advocate following them, but they're interesting to see a very different approach to things. In Computer Graphics Principles and Practice, Jim Foley and others give some principles that apply specifically to 2D and 3D computer graphics. And finally Susan Weinschenk and Dean Barker have a set of guidelines that provide an even more holistic view of interface design, including things like linguistic and cultural sensitivity, tempo and pace, and domain clarity. And even these are only some of the additional lists. There are many more that I encourage you to look into.

Conclusion to Design Princip

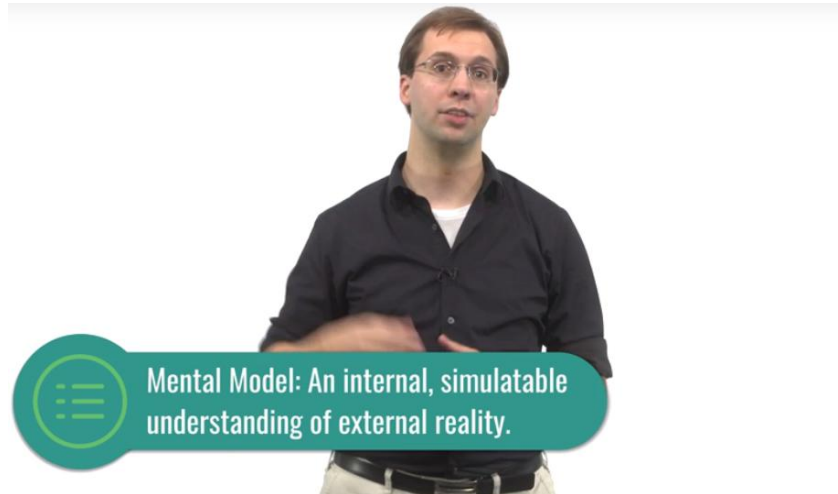


In this lesson, I've tried to take the various different lists of usability guidelines from different sources and distill them down to a list you can work with. We combined the lists from Don Norman, Jakob Nielsen, Larry Constantine, Lucy Lockwood, and the Institute for Universal Design into fifteen principles. Now remember, though, these are just guidelines, principles, and heuristics: none of them are unbreakable rules. You'll often find yourself wrestling with tensions between multiple principles. There will be something cool you'll want to implement, but only your most expert users will be able to understand it. Or, there will be some new interaction method that you want to test, but you aren't sure how to make it visible or learnable to the user. These principles are things you should think about when designing, but they only get you so far. You still need needfinding, prototyping, and evaluation to find what actually works in reality.

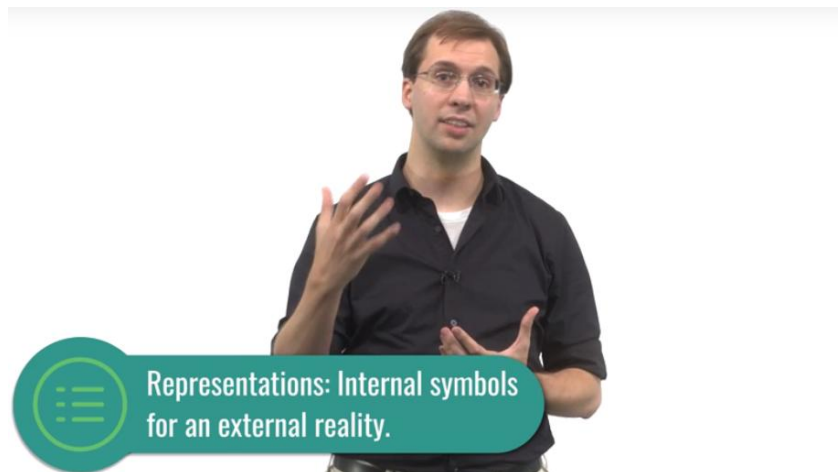
2.6 Mental Models and Representations

Compiled by Shipra De, Summer 2017

Introduction



[MUSIC] Today we're going to talk about mental models and representations. A mental model is the understanding you hold in your head about the world around you. Simulating a mental model allows you to make predictions and figure out how to achieve your goals out in the real world. A good interface will give the user a good mental model of the system that it presents.

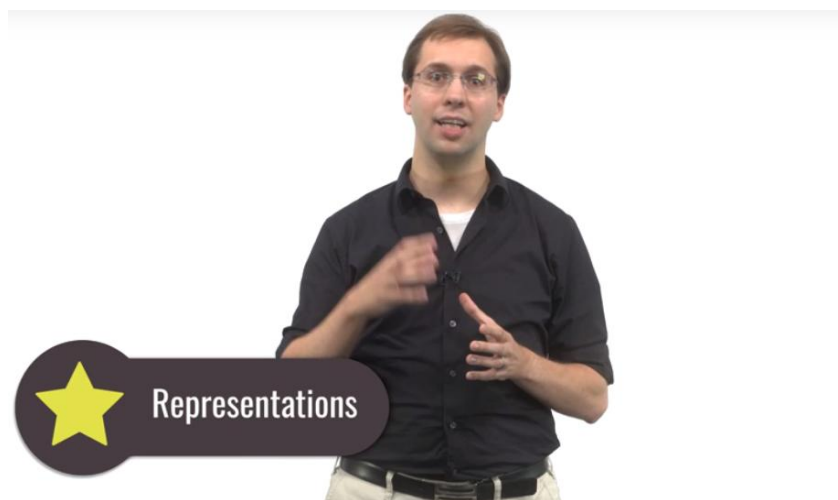


In order to develop good mental models we need to give users good representations of the system with which they're interacting. In that way, we can help users learn how to use our interfaces as quickly as

possible. So that's what we'll talk about in this lesson, creating representations that help users develop accurate mental models of our systems.



We'll start by talking about mental models in general and how they apply to the interfaces with which we're familiar.



Then we'll talk about how representations can make problem solving easier or harder.



After that, we'll talk about how metaphors and analogies can be useful tools to create good representations that lead to accurate mental models.

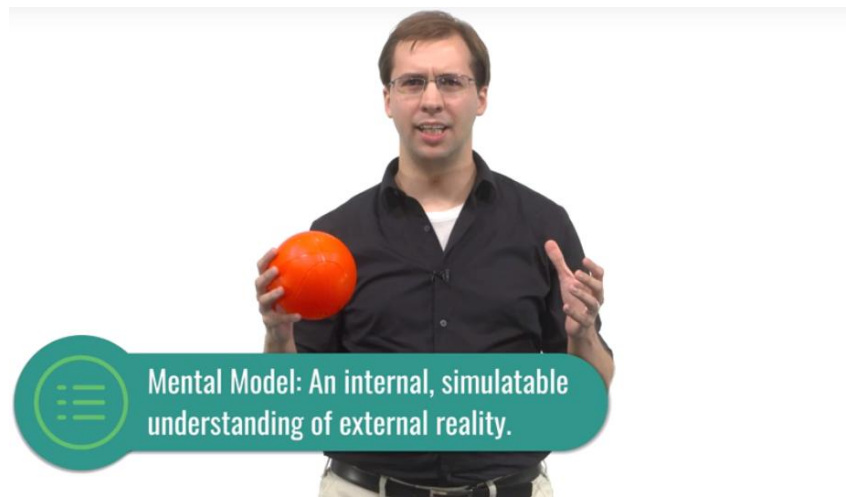


Then we'll discuss how user error can arise either from inaccuracies or mistakes in the user's mental model, or just from accidental slips, despite an accurate mental model.

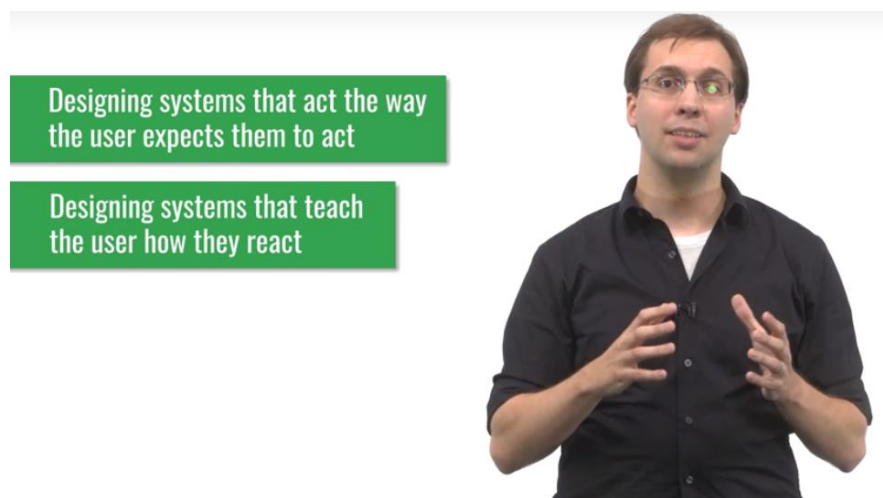


Finally, we'll close by discussing learned helplessness, one of the repercussions of poor interface design, as well as expert blindspot, which is one of the reasons why poor design can occur.

Mental Models



A mental model is a person's understanding of the way something in the real world works. It's an understanding of the processes, relationships and connections in real systems. Using mental models we generate expectations or predictions about the world and then we check whether the actual outcomes match our mental model. So I'm holding this basketball because generally, we all probably have a model of what will happen if I try to bounce this ball. [SOUND] It comes back up. [SOUND] You didn't have to see it come up to know what would happen. You use your mental model of the world to simulate the event. And then you use that mental simulation to make predictions. When reality doesn't match with our mental model, it makes us uncomfortable. We want to know why our mental model was wrong. Maybe it makes us curious. But when it happens over and over, it can frustrate us. It can make us feel that we just don't and never will understand. As interface designers, this presents us with a lot of challenges. We want to make sure that the users mental model in our systems matches the way our systems actually work.



We can do that in two primary ways, one by designing systems that act the way people already expect them to act. And two, by designing systems that, by their very nature, teach people how they'll act. That way we can minimize the discomfort that comes from systems acting ways that users don't expect.

Mental Models and Education

Mental models are not a uniquely HCI principle. In fact, if you search for mental models online, you'll probably find just as much discussion of them in the context of education as the context of HCI. And that's a very useful analogy to keep in mind. When you're designing an interface, you're playing very much the role of an educator. Your goal is to teach your user how the system works through the design of your interface. But unlike a teacher, you don't generally have the benefit of being able to stand here and explain things directly to your user. Most users don't watch tutorials or read documentation. You have to design interfaces that teach users while they're using them. That's where representations will come in. Good representations show the user exactly how the system actually works. It's an enormous challenge, but it's also incredibly satisfying when you do it well.

Mental Models in Action



So let's talk a little bit about mental models in the context of the climate control systems we see on automobiles. So this is my old car, it's a 1989 Volvo. It, sadly, does not run anymore. But let's talk about how the climate control system would work back when it did run.



So, it's a hot day outside. Looking at these controls, how would you make the air temperature colder and the air come out faster? The natural thought to me would be to turn the fan speed up over on the right, and the air temperature to the blue side over on the left. But this doesn't actually make the temperature any colder, it just disables the heat. This dial over here in the top right, has to be turned on to make the air conditioning actually work. So just turning this thing over to the blue side doesn't actually turn on the air conditioning. So to make it colder, you have to both slide this lever over to the left, and turn this dial to the red area. It's kind of hard to see in this, but this little area over here on the left side of this dial is actually red. The red area on the air conditioning control designates the maximum coldness. What? This also means you can turn on both the heat and the air conditioning at

the same time and have neither of them blowing out if your fan is turned off. None of these really match my mental model of how this system works and the colors used here do nothing to correct my mental model. There's a control here that if you turn it to blue doesn't make the car any colder. And if you turn this other control to red, it does make the car colder. What? So back in 1989, there was a lot of room for improvement on designing the climate control system for a car like this. Let's see if we actually did improve that by talking about my new car, which is a 2015 Nissan Leaf. So in the 26 years since my old Volvo came out, have we gotten better at this? Well yeah, we've gotten a good bit better, although there's still a good bit of room for improvement.



So, here's the climate control system from my Leaf. I have one dial, that turns the fan speed up and down. One dial that turns the temperature of the air coming out up and down. And so as far as that's concerned, it's pretty simple. But this interface still has some things that are pretty confusing. So for example, it has an automatic mode, where it tries to adjust the air temperature and the fan speed to bring the temperature of the car to the temperature that I want. So I press auto. Now it's going to change the fan speed, and change the air temperature, if I didn't already have it at the lowest, to try and get the car cooler faster. The problem is that I want to turn auto off, I don't actually know how to do it. Pressing auto doesn't actually turn it off. If I turn it so that it doesn't only circulate air inside the car, then it turns auto off. But that might not be what I wanted. Maybe I wanted it to go to a certain air temperature without just circulating the air in the car. I turn auto back on, it's going to turn that back on. Don't know why. So right now, as far as I know, the only way to turn auto off is to turn the circulation mode off. It also lets me turn AC and heat on at the same time, which I don't understand at all. Why would I ever need that? So there are some things that the system really should do better, or some things that it should constrain so the user doesn't do things that don't make sense in the context of wanting to set the temperature.

5 Tips: Mental Models for Learnable Interfaces



Matching our interface design to users' mental models, is a valuable way to create interfaces that are easily learnable by users. Here are five tips, or in this case, principles to leverage for creating learnable interfaces. These principles of learnability were proposed by Dix, Finlay, Abowd and Beale, in their book, Human-Computer Interaction. Number one, predictability. Look at an action. Can the user predict what will happen? For example, graying out a button is a good way to help the user predict that clicking that button, will do nothing. Number two, synthesizability. Not only should the user be able to predict the effects of an action before they perform it, they should also be able to see the sequence of actions that led to their current state. That can be difficult in graphical user interfaces, but something like the log of actions that they can see in the undo menu can make it easier. Command line interfaces are actually good at this, they give a log of commands that have been given in order. Number three, familiarity. This is similar to Norman's principle of affordances. The interface should leverage actions with which the user is already familiar from real world experience. For example, if your trying to indicate something is either good or bad. You'd likely want to use red and green instead of blue and yellow. Number four, generalizability. Similar to familiarity and to Norman's principle of consistency, knowledge of one user interface should generalize to others. If your interface has tasks that are similar to other interface's tasks, like saving, and copying, and pasting, it should perform those tasks in the same way. Number five, consistency. This is slightly different than Norman's principle of consistency. This means that similar tasks or operations within a single interface, should behave the same way. For example, you wouldn't want to have Ctrl+x cut some text if text is selected, but close the application if there is no text selected. The behavior of that action, should be consistent across the interface. Using these principles can help the user leverage their existing mental models of other designs, as well as develop a mental model of your interface as quickly as possible.



Quiz: Representations for Problem Solving 1

A hiker starts climbing a mountain at 7:00am.



A good representation for our problem will make the solution self-evident. Let's take a classic example of this. A hiker starts climbing a mountain at 7 AM. He arrives at the cabin on top at 7 PM.

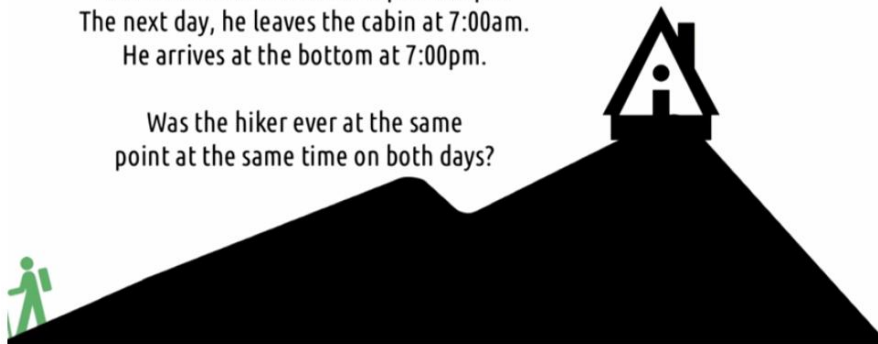
A hiker starts climbing a mountain at 7:00am.

He arrives at the cabin on top at 7:00pm.

The next day, he leaves the cabin at 7:00am.

He arrives at the bottom at 7:00pm.

Was the hiker ever at the same point at the same time on both days?



The next day, he leaves the cabin at 7 AM and arrives at the bottom at 7 PM.

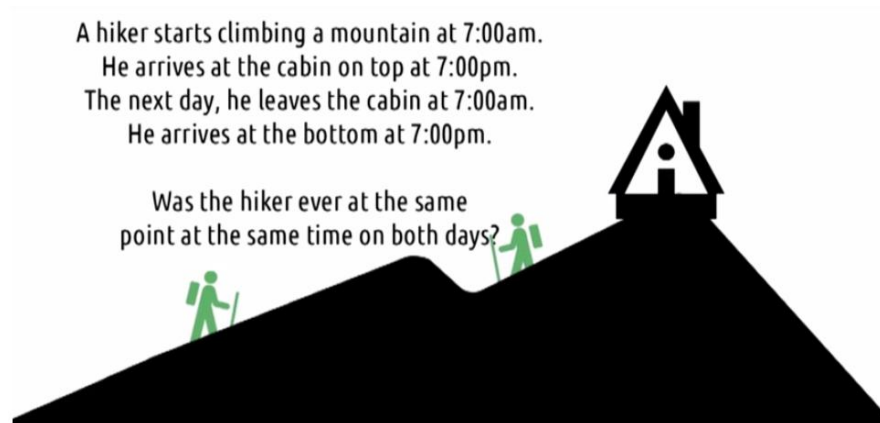


Was the hiker ever at the same point at the same time on both days?

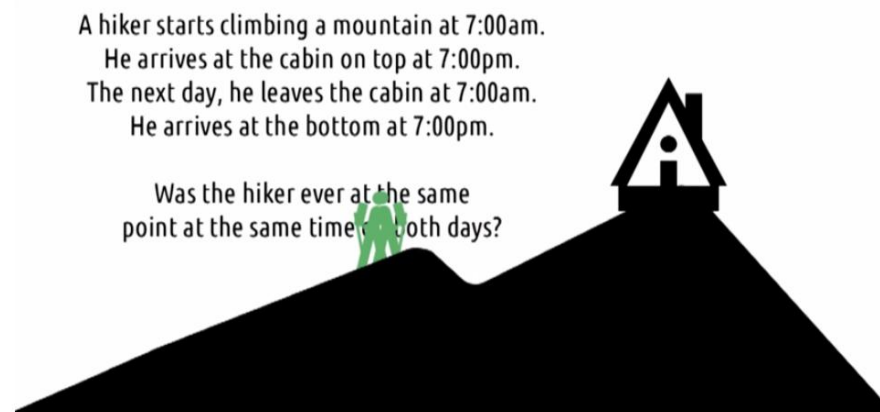
☒ Yes

☐ No

The question, was the hiker ever at the same point at the same time on both days?



Let's watch that animation again. The hiker goes up the hill on one day, stays the night. And then goes back down the hill the next day. And we want to know, was the hiker ever at the same point at the same time on both days? And the answer is yes. Describe the way we describe it right here, it might actually seem odd that there is a point where the hiker is in the same place at the same time on both days. That seems like a strange coincidence, but what if we tweak the representation a little bit? Instead of one hiker going up and then coming down the next day. Let's visualize the two days at the same time. If we represent the problem like this, we'll quickly see the hiker has to pass himself.



To show it again, we know the answer is yes, because there's a time when the hiker would have passed himself if he was going in both directions on the same day. And to pass himself, he has to be in the same point at the same time. That representation took a hard problem and made it very easy.

Quiz: Representations for Problem Solving 2



How many moves does it take to solve the circles and squares problem?

For simple problems, identifying a good representation can be easy. But what about for more complex problems? For those problems, we might need some examples of what makes a good representation. So let's try a complex example. We'll use a problem with which you might be familiar. For now, I'll call it the circles and squares problem. On one side of a table, I have three circles and three squares. My goal is to move the three circles and three squares to the other side of the table. I can only move two shapes at a time, and the direction of the moves must alternate, starting with a move to the right. The number of squares on either side can never outnumber the number of circles unless there are no circles at all on that side. How many moves does it take to accomplish this? Try it out and enter the number of moves it takes in the box. Or just skip if you give up.

If you solved it, well done on figuring it out despite such a terrible representation. Or congratulations on recognizing by analogy, that it's the same as a problem you've seen in another class. If you skipped it, I don't blame you. It's not an easy problem. But it's even harder when the representation is so poor. There were lots of weaknesses in that representation. Let's step through how we would improve it. The first thing we could do is simply write the problem out. Audio is a poor representation of complex problems.

The Circles and Squares Problem

On the left side of a table are three circles and three squares. Your goal is to move all six shapes to the right side of the table, following these rules:

- You may only move one or two shapes at a time.
- The direction of your moves must alternate, starting with a move to the right.
- Squares must never outnumber circles on either side, unless there are no circles on that side.

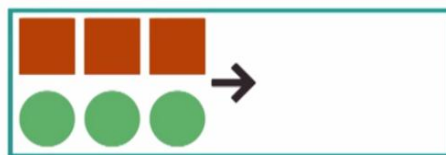
So here's a written representation of the problem. If the trouble you were having solving the exercise was just remembering all the rules, having this written down would be a huge help. But we can still do a lot better than this. Instead, we can represent the problem visually.



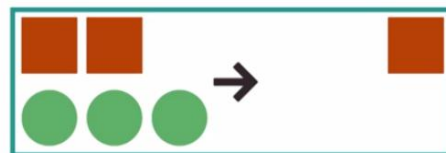
Here we have the shapes, the three circles and the three squares. And we can imagine actually moving them back and forth.



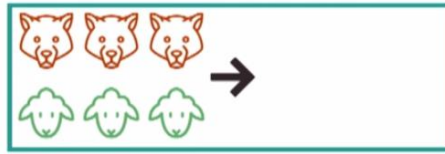
That arrow in the center, represents the direction of the next move.



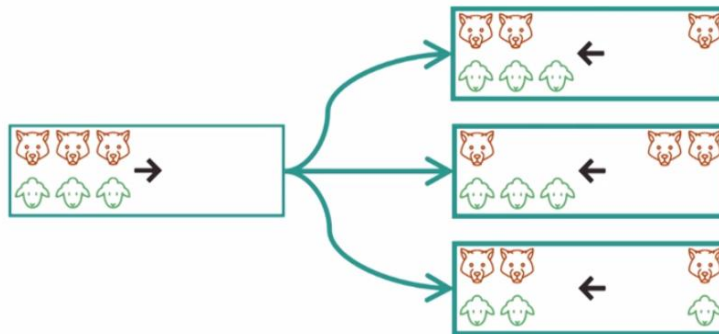
But we can still do better. Right now we have to work to compare the number of squares and circles, so let's line them up. This makes it very easy to compare and make sure that the circles always outnumber the squares. And we can still do the same manipulation, moving them back and forth.



Now the only remaining problem is that we have to keep in working memory, the rule that squares may not outnumber circles. There is no natural reason why need more squares than circles, it's just kind of an arbitrary rule.

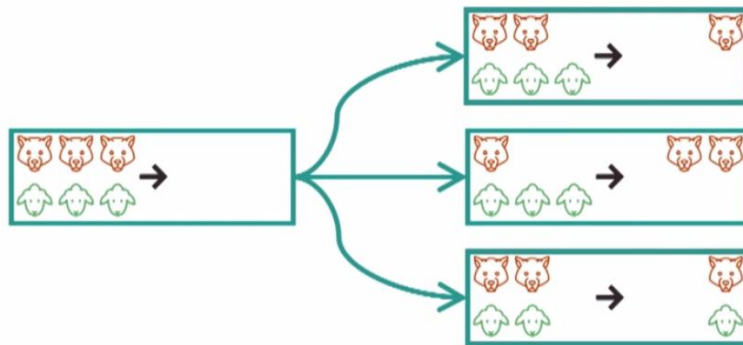


So let's make it more self evident. Let's make the squares wolves, and the circles sheep. As long as the sheep outnumber the wolves, the sheep can defend themselves, kind of. But if the wolves ever outnumber the sheep, they'll eat them. But if there are no sheep, then there's nothing for the wolves to eat, so that's okay. So now we have a new representation of the problem, one that will make the problem much easier to solve. The rules are more obvious, and it's easier to evaluate whether or not they're being met.



Finally, we can make this visualization even a little bit more useful, by actually showing the movements between different states. That way we can see that for any state in the problem, there's a finite number of next legal states. This would also allow us to notice when we've accidentally revisited an earlier state, so we can avoid going around in circles. So for example, from this state, we might choose to move the wolf and the sheep back to the left, but we'll immediately notice that would make the state the same as this one. And it's not useful to backtrack and revisit an earlier state. So we know not to do that. So these representations have made it much easier to solve this problem, than just the verbal representation we started with.

Characteristics of Good Representations

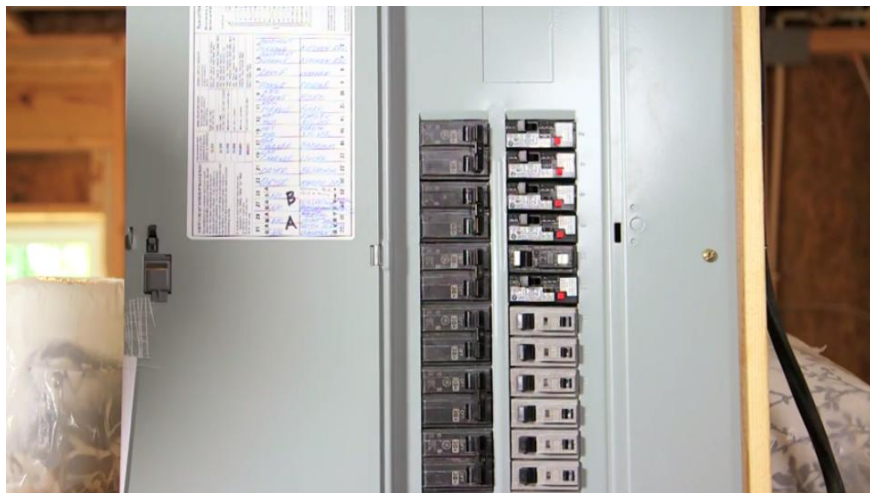


What are the characteristics of a good representation? First, good representations make relationships explicit. Laying things out like this makes it easy to tell that there are more sheep than wolves. Second, good representations bring objects and relationships together. Representing these as wolves and sheep, makes that relationship that the sheep must outnumber the wolves much more salient than using squares and circles. That brings the objects together with the relationships between them. Third, a good representation excludes extraneous details. For example, sometimes this problem is described in the form of having a river and a boat. But those details aren't actually relevant to solving the problem at all. So, we've left them out of here. All we need to know is they need to move from the left to the right. Doesn't matter if it's a river, doesn't matter if it's a boat, this is all the information that we need. So we left out the extraneous information. Fourth, good representations expose natural constraints. We describe these as sheep and wolves because it makes it easier to think about the rule that wolves may never outnumber sheep. Now of course, this isn't the best rule because we know that sheep can't actually defend themselves against wolves. Three sheep and one wolf, the wolf would still win. However, if we visualize these as guards and prisoners instead, it involves holding and working memory the idea that prisoners inexplicably won't flee if they're left without any guards. So personally, I think the wolves and sheep metaphor is better. But perhaps the original name of the problem is even better. This was originally described as the cannibals and missionaries problem. It makes more sense that a missionary could defend themselves against a cannibal than a sheep could defend themselves against a wolf. But the cannibals and missionaries problem makes it a little bit dark. So let's stick with sheep and wolves.

Quiz: Design Challenge: Representations



So let's take an example of redesigning a representation to create a better mapping with a task. Here we have my circuit breaker.

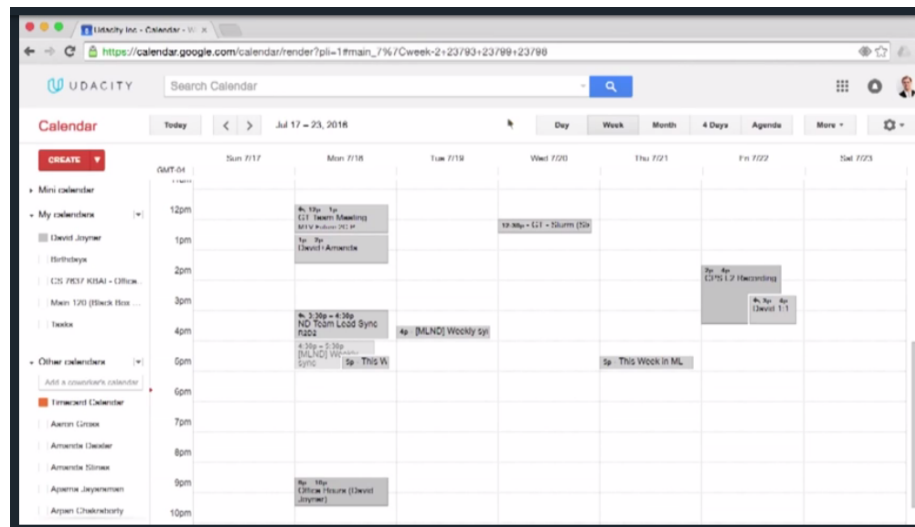


On the left we have a list of breakers, on the right we have what they actually control. To reset a breaker I need to go down the list on the left, find the one I want, count down on the right to find the right breaker, and switch it. How can we make this representation of what each breaker corresponds to better?

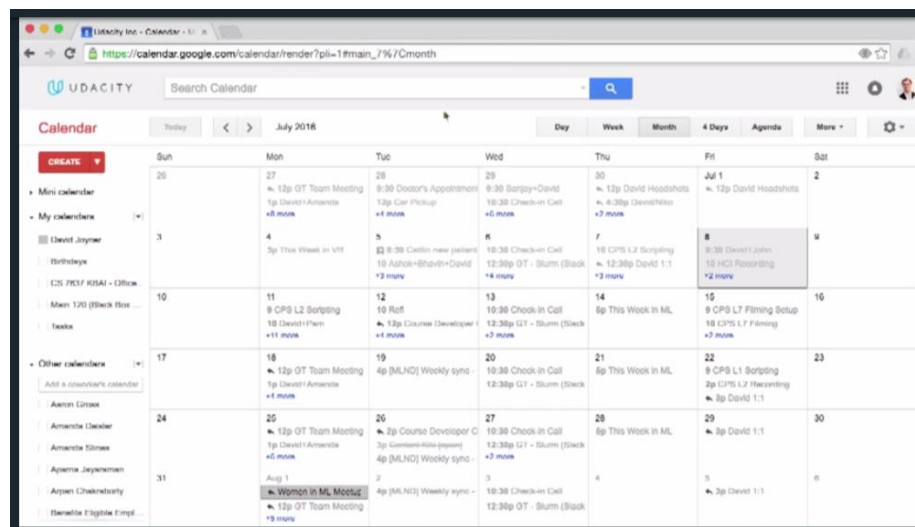
There are a number of things we can do here. The simplest change we could make would simply be to make the breakers themselves writable. Instead of writing a list on the left that we have to then map up to the breakers themselves on the right, we could just write on each breaker what it controls. That way we just have to look at the breakers themselves to find the breaker that we're interested in. But

then they still have to manually scan through all of them. We could further augment this by having a floor plan over here that actually gives the numbers on the floor plan for the breaker we want. So all I have to do is jump straight to the room that I'm interested in, find the number, go over the the list of breakers, and the label written on it would then confirm that I chose the right one. Now, if we wanted to get really crazy we could actually lay out the breakers themselves to correspond to the floor plan. We can have a floor plan and actually put the breakers on the floors that they control. Or we could even just put the breakers in the rooms themselves. So if the power goes out to a certain room, I just go find the breaker in that room. But there we're starting to run into some of the other constraints on the problems. So it's probably best to stick to what we can control, without requiring that the physical device be manufactured differently in the first place.

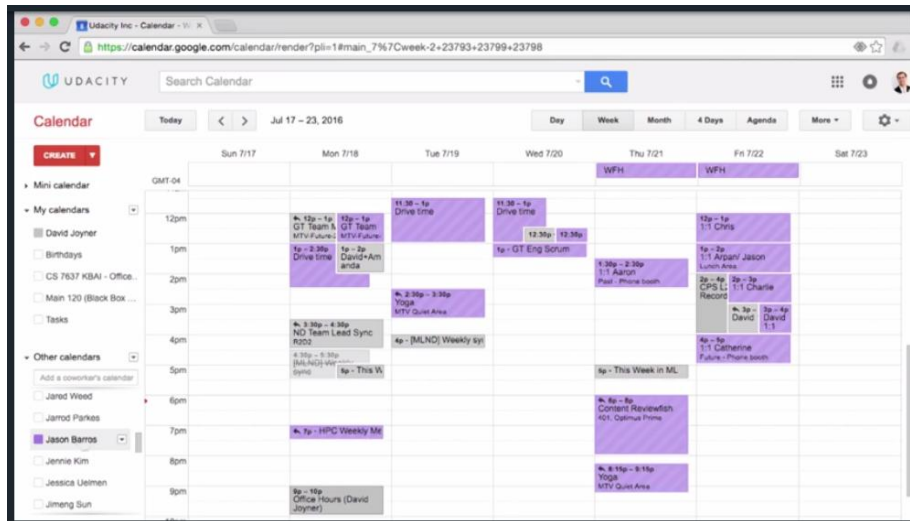
Representations in Interfaces



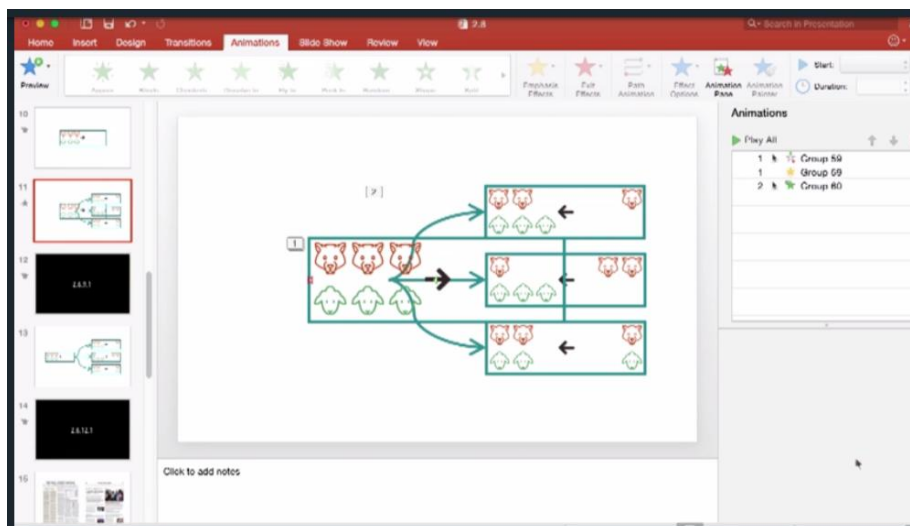
Representations are all around us in the real world, but they play a huge role in interfaces. Designing representations of the current state of a system is actually one of the most common tasks you might perform as an interface designer. So let's take a look at a few, here's Google Calendar which is a representation of my week. Notice how it actually uses space to represent blocks of time. It allows me to quickly feel how long different things are going to take.



An alternate visualization might show an entire month instead of a week, but it would lose those indicators that linked the individual appointments. So it doesn't really represent the structure and pace of my day, the way the weekly calendar does. This representation also allows me to very easily find conflicts in my schedule. So I know when I might need to reschedule something. On Friday I can see that I have a conflict for one of my meetings. And this interface also makes it easy to reschedule.



I can pull up the calendar for the other person I'm meeting with and identify places where we both have free in our schedule.

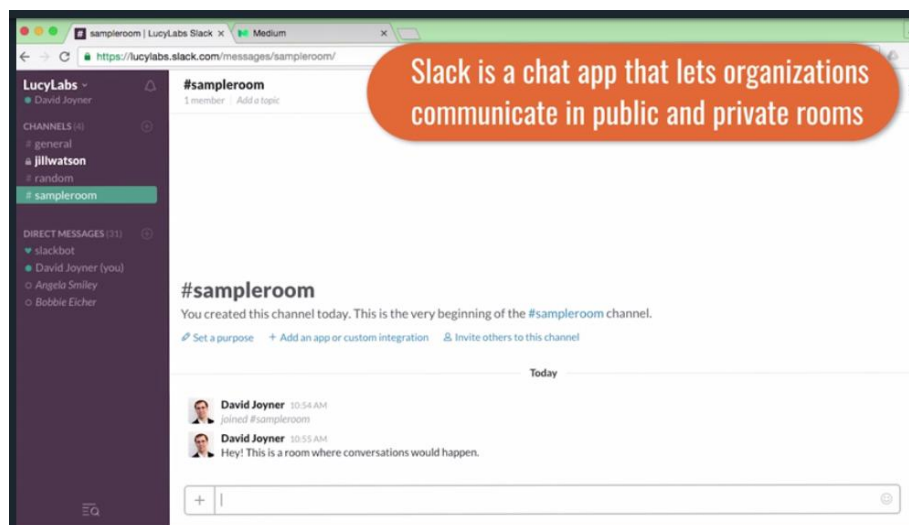


Another example of this is the PowerPoint animation pane. The numbers here represent when different animations happen concurrently. The middle icon represents what triggers the animation, and the right icon indicates the general nature of the animation. Whether it's a movement, a highlight or an appearance. The PC version of PowerPoint makes this even better by actually showing you a timeline of the different animations to the right. That lets you very easily visualize when two different things are going to happen at the same time. Or when something waits for something else to happen. These are just two of the many many representations you use whenever you use a computer. Scroll bars for example, are representations of your relative position in a document. Highlighting markers like that rectangle are representations of what you currently have selected. All these representations work together to help your mental model match the real state of the system. Representations when used correctly can make many tasks trivial, or even invisible. And we as interface designers have a lot of control over representations in our designs.

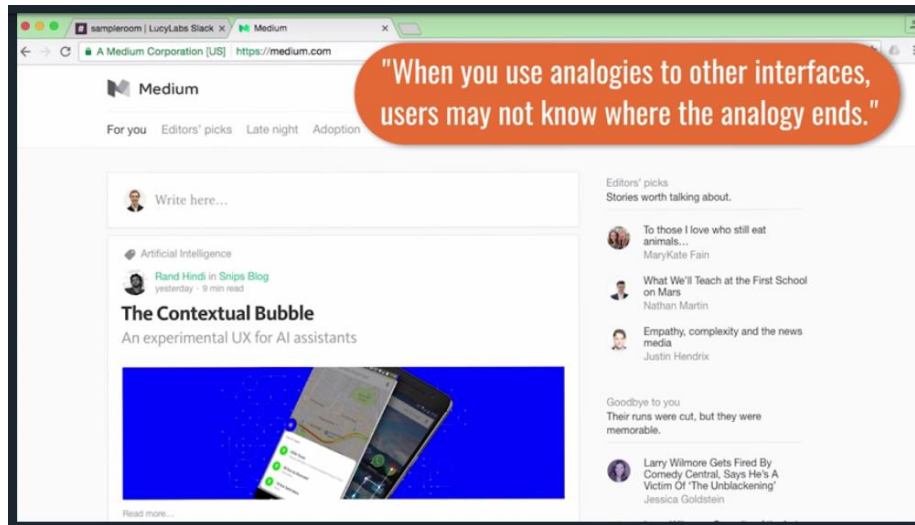
Metaphors and Analogies



Analogies and metaphors are powerful tools for helping users understand your interface. If you can ground your interface in something they already know, you can get a solid foundation in teaching them how to use your interface. For example, the Wall Street Journal's website heavily leverages an analogy to the Wall Street Journal print edition. The headlines, the grids, the text all appear pretty similarly, so someone familiar with the print edition could pretty easily understand the online edition.



If you've ever tried to explain a tool that you use to someone that's never seen it, you've probably encountered something like this. For example, both at Udacity and in the Georgia Tech OMSCS program, we use Slack for communicating with each other. If you've never actually seen Slack, it's a chat app for organizations to talk in different public and private rooms. But listen to what I just said, it's a chat app. In my description, I leveraged an analogy to something you've already seen. Now, Slack is a pretty easy example, because it is a chat app. It's barely even an analogy to say it's like a chat app, because it is a chat app. It's a very full-featured chat app with a lot of integrations and things like that, but it's fundamentally a chat app.

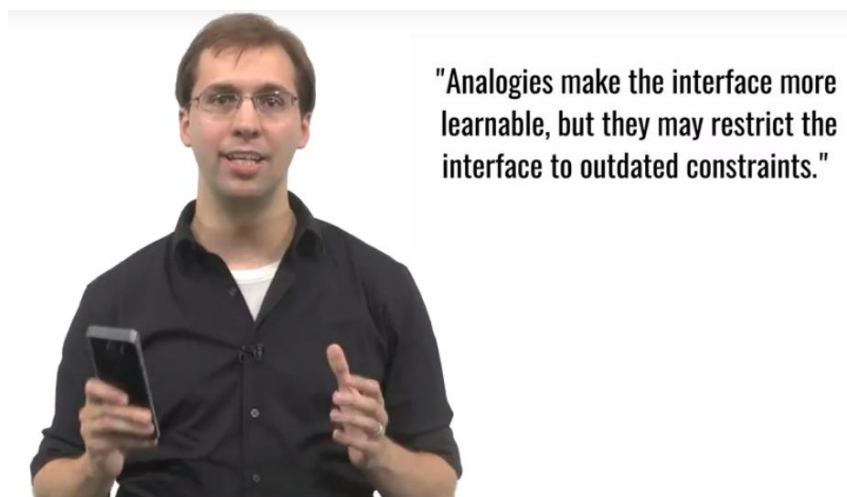


What about something harder? How about Medium? Medium is a writing platform that's kind of like a blogging service, but also kind of like a publishing service, but also kind of like a news feed. You write articles, kind of like WordPress blog posts, but you can publish them through organizations, which is kind of like traditional newspapers or news aggregators like the Huffington Post. My articles, for example, are published through Udacity. So it's not just like a blog, because it's not just my personal blog, there is a publisher element to it. But the actual content is very similar to a blog-like platform. Articles are then published to interested people more like a news feed. So if I scroll down, I'll see the articles that Medium thinks I would be interested in reading. And in that way, it's more like Facebook or Twitter. So notice that my entire explanation of Medium was based on analogies to other services like WordPress, Huffington Post, and Facebook. But analogies and metaphors have a downside. When you choose to use them, users don't know where the analogy ends. When I describe Medium's news feed as kind of like Facebook or kind of like Twitter, users might wonder where the retweet or share options are. Or when I describe it like a blog platform like WordPress, people might wonder where the comments are. And it doesn't really supply comments in the way that we're used to. So while analogies are powerful ways to help users understand our interfaces, we also need to pay special attention that what misconceptions they might introduce.

Exploring HCI: Metaphors and Analogies



One of the challenges encountered by every new technology is helping the user understand how to use it. Smartphones maybe pretty ubiquitous by now but we're still figuring out some elements of how to best use these things. Typing efficiency on a touch screen, for example, still hasn't caught up to efficiency with a full keyboard. But that's because typing on a phone was also a pretty straightforward transition from a regular keyboard, because the onscreen keyboard was designed just as an analogy to the real one. There are probably more efficient ways to enter text into a phone, but they wouldn't be as easily learnable as this straightforward analogy to a physical keyboard. This illustrates both the positive and negative sides of using analogies in designs.



Analogies make the interface more learnable but they also may restrict the interface to outdated requirements or constraints. Take a moment and think about how this applies to your chosen area of HCI. If you're looking at things like gestural interfaces or touch-based interfaces, what analogies can you draw to other interfaces to make your designs more learnable? At the same time, what do you risk by using those analogies?

Design Principles Revisited

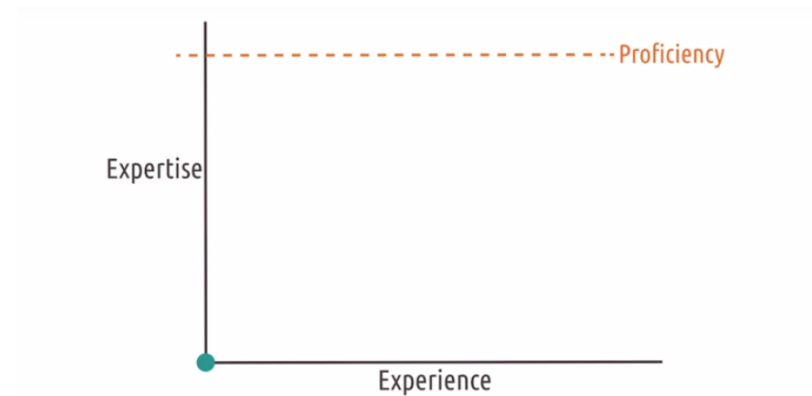
In our lesson on design principles, we touch on a number of principles that are relevant to these ideas of mental models, representations, and metaphors. First, the idea that people reason by analogy to pass interfaces, or by metaphors to the real world, is one of the reasons that the principle of consistency is so important. We want to be consistent with the analogies and metaphors that people use to make sense of our interfaces. Second, when we say that an interface should teach the user how the system works, we're echoing the idea of affordances. The way the system looks, should tell the user how it's used. Just by observing the system the, user should be learning how to interact with it. Third, representations are important because they map the interface, to the task at hand. A good representation is one that users can use predict the outcomes of certain actions. In other words, a good representation lets users predict the mapping between their actions in the interface, and the outcomes out in the world.

New Functionality Meets Old Interfaces

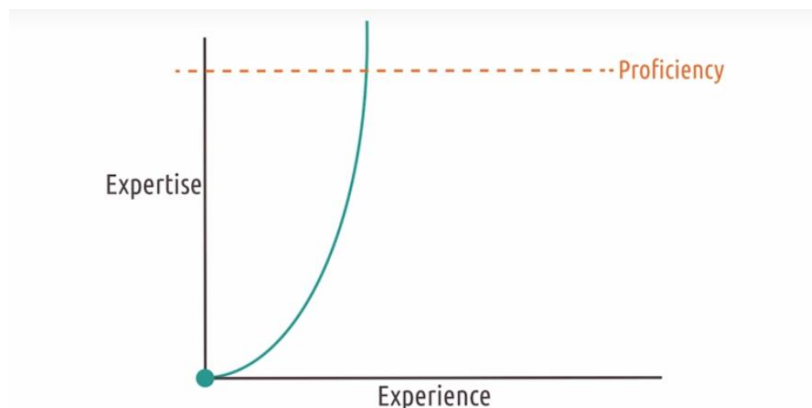


In designing interfaces, we want to leverage analogies to the real world, and principles from past interfaces whenever possible, to help the user learn the new interface as quickly as they can. But there's a challenge here. Why are we designing technology if we're not providing users anything new? It's one thing to take the technology they're already using, and make it more usable. But generally, we also want to enable people to do things they've never done before. That means there are no analogies, no expectations, no prior experiences for them to leverage. How do you tell someone that's used to control their own thermostat that they don't need to anymore. So, while we need to leverage analogy and prior experience wherever possible. We also need to be aware that eventually, we're going to do something interesting, and they're going to break down. Eventually, we're going to have to teach the user to use the unique elements of our interface.

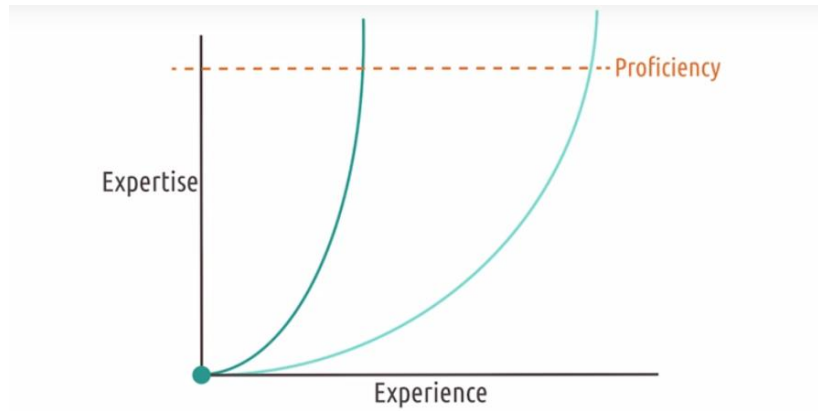
Learning Curves



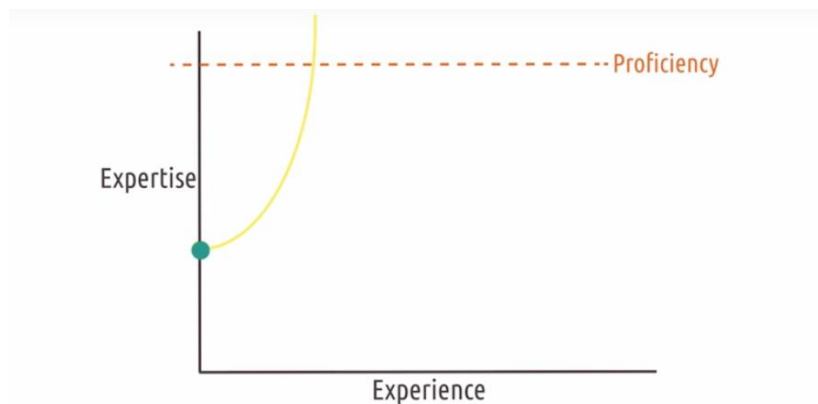
Every interface requires a user to do some learning to understand how to use it. Very often, we visualize this as a learning curve. A learning curve plots expertise against experience. Generally, as the user gains more experience, they also gain more expertise. Here, our user starts with no experience at all, and so they also have no expertise at all. Our goal is for them to end with an expertise above this line of proficiency. However, the shape and steepness of this curve can vary.



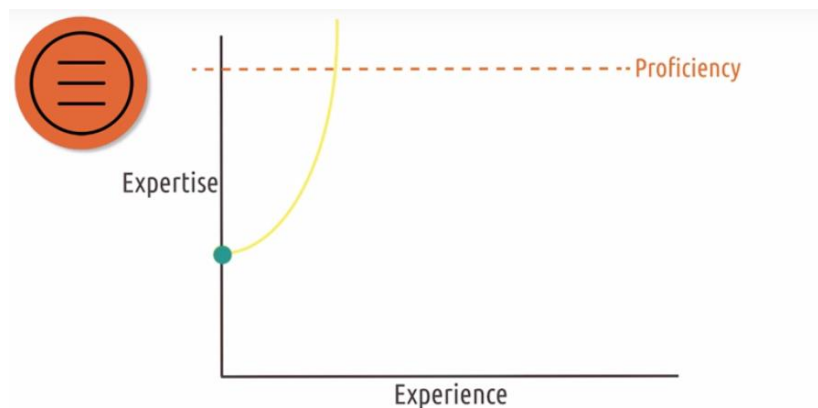
Ideally, we want a learning curve that grows quickly with a relatively little experience. This is actually what we call a steep learning curve, although usually when you hear steep learning curve, it means the exact opposite. Technically, steep is good because steep means we're increasing very quickly with relatively little experience. People often use steep to mean the opposite, and that's because steep calls to mind connotations of a high difficulty level, like climbing a steep mountain. So steep is actually a poor representation of this concept. So instead, let's for us call this a rapid learning curve, which means that expertise grows very quickly with relatively little experience. Rapid calls to mind probably the proper connotation that a rapid learning curve is rapid learning, which is probably something we want.



Interfaces that are more difficult to use would have slower learning curve. Here, the user needs a lot more experience to reach the same level of proficiency. So, how do we help our user reach proficiency faster?



For one, if we're consistent with existing conventions and use analogies that users understand, we can actually start them off with effectively some initial expertise.



For example, when you download a new smartphone app, you know that the three horizontal lines that often appear in the top-right, likely indicate a menu. That's a consistent convention used across multiple apps. And so using it means that when users open your app, they already have some

expertise. From there, we want to make the ascension as rapid as possible. One way we can do that is by using representations and affordances that help the user immediately understand how to use the interface. So good design is in part about helping users achieve proficiency as quickly as possible. Either through starting them off with some initial expertise or helping them grow in their expertise with as little experience as possible.

User Error: Slips and Mistakes

Slips

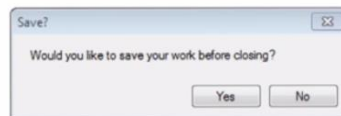


Mistakes



As we design interfaces, we will no doubt encounter instances where the user makes mistakes. Sometimes this might be because our users are stressed or distracted. But other times it might be because our users fundamentally don't understand our interfaces or don't understand their own goals. As the designers, though, we know that there's really no such thing as user error. Any user error is a failure of the user interface to properly guide the user to the right action. In designing interfaces, there are two kinds of user error that we're interested in avoiding. The first are called slips. Slips occur when the user has the right mental model but does the wrong thing anyway.

Slips



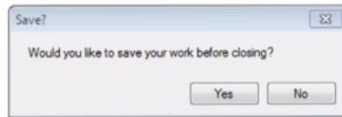
The user has the right mental model, but does the wrong thing anyway.

Mistakes



Take this box, for example, prompting a user closing a program on whether they'd like to save their work. In all likelihood, the user probably knows exactly what they'd like to do, and typically it's that they're going to want to save their work. If you ask them to explain what they should do, they would say "click yes." But imagine if the order of these buttons was flipped. If the "No" was on the left and the "Yes" was on the right. A user might click on the left just because they're used to seeing yes on the left, even though they know they really want to click yes. Or, imagine that no was selected by default, so that if the user just presses enter when this dialog comes up it automatically says no. In that case also, they knew that they wanted to save their work but what they did, didn't match the goal they wanted to accomplish.

Slips



The user has the right mental model, but does the wrong thing anyway.

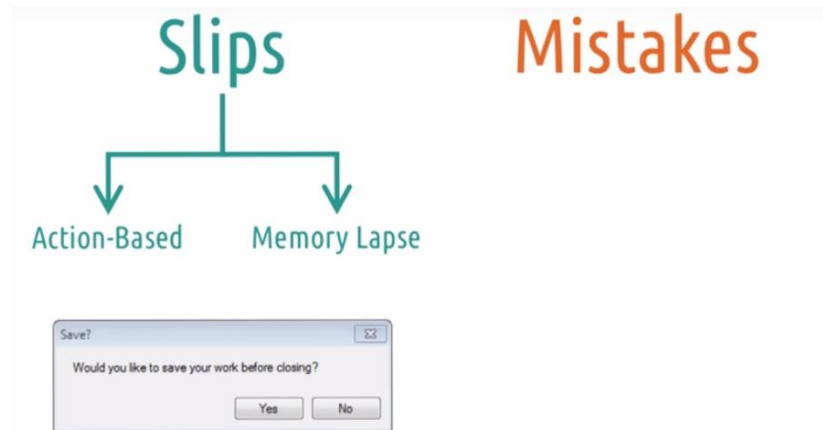
Mistakes



The user has the wrong mental model, and does the wrong thing as a result.

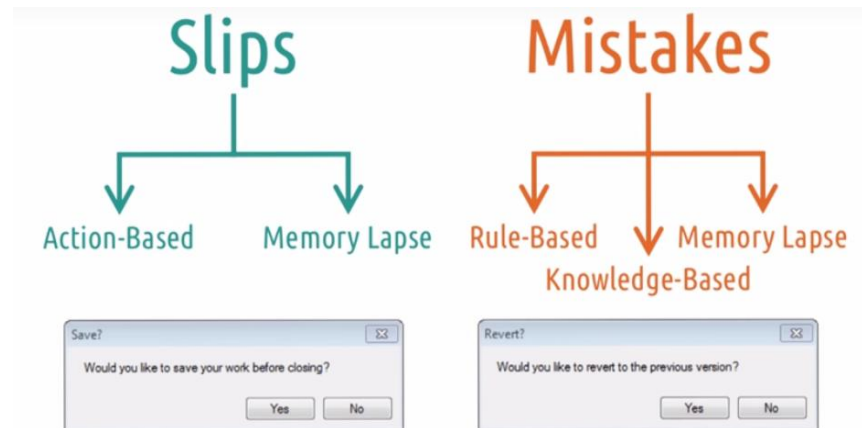
A mistake on the other hand, happens when the user has the wrong mental model and does the wrong thing as a result. Take this prompt for example. The user is asked to revert to the original file. That's really just a backwards way of asking if they want to save. But this is foreign terminology to many users. Their mental model of what saving is, doesn't tell them necessarily what to do in this instance. What's more, they don't really have a choice here. Without a cancel button, they're forced to choose, knowing that one option could mean losing their changes. Here the problem is a mismatch between their internal model and the way the system is working, or at least the way it describes itself. So a slip occurs when the user knows the right thing to do but does the wrong thing anyway. But a mistake occurs when the user doesn't even know the right thing to do.

Types of Slips



Don Norman further divides slips into two different categories. He describes action-based slips and memory lapse slips. Action-based slips are places where the user performs the wrong action, or performs a right action on the wrong object, even though they knew the correct action. They might click the wrong button, or right-click when they should left-click. A memory lapse slip occurs when the user forgets something they knew to do. For example, they might forget to start a timer on a microwave. They knew what to do, they just forgot about it. So action-based slips are doing the wrong thing, and memory lapse slips are forgetting to do the right thing. In this dialog, clicking No when you mean to click Yes would be an example of an action-based slip. The very existence of this dialog is meant to prevent a memory lapse slip, where a user would forget to save their work before closing.

Types of Mistakes



Norman also divides mistakes in the multiple categories, in this case, three categories. Rule based mistakes, knowledge base mistakes and memory lapse mistakes. Rule based mistakes occur where the user correctly assesses the state of the world but makes the wrong decision based on it. Knowledge based mistakes occur where the user incorrectly assesses the state of the world in the first place. Memory lapse mistakes are similar to memory lapse slips, but this focuses on forgetting to fully execute a plan not just forgetting to do something in the first place. If the user clicks the wrong button in this dialog, it could be due to multiple different kinds of mistakes. Maybe they correctly knew they wanted to save their changes but they didn't realize that clicking no is actually what would save, that would be a rule-based mistake. They knew they wanted to save, but they made the wrong decision based on that knowledge. Or perhaps they didn't even realize they wanted to save in the first place. Maybe they didn't think they made any changes, when in actuality they did. That would be a knowledge based mistake. They applied the right rule based on their knowledge but their knowledge was inaccurate. If they were to shut down their computer and never come back and answer this dialogue in the first place, that might be considered a memory lapse mistake. They didn't fully execute the plan of closing down the application. So in our designs, we want to do everything we can to prevent all these different kinds of errors. We want to help prevent routine errors by leveraging consistent practices like designing dialogues the way users are used to. We also want to let our interface off load some of the demands on working memory from the user to the computer to avoid memory lapse errors. And we want to leverage good representations to help users develop the right mental models to minimize these rule-based and knowledge-based errors. And while errors are inevitable, we should make sure to leverage the tolerance principle to make sure the repercussions can never be too bad.

Quiz: Exercise: Slips vs Mistakes



When you're looking to improve an interface, user errors are powerful places to start. They're indicative either of weaknesses in the user's mental model or places where the system isn't capturing the user's correct mental model. So let's try to address an error Morgan's encountering. Morgan usually texts with her boyfriend but she texts with some other people too. But she finds she's often sending the wrong messages to the wrong people. The app by default brings up the last open conversation and usually that's her boyfriend. But sometimes it's someone else and she accidentally messages them instead. First, is this a slip or is this a mistake?

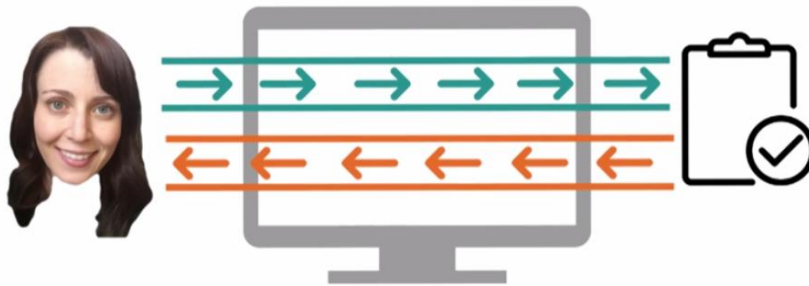


Morgan accidentally sends a text message to the wrong person. Is this a slip or a mistake?

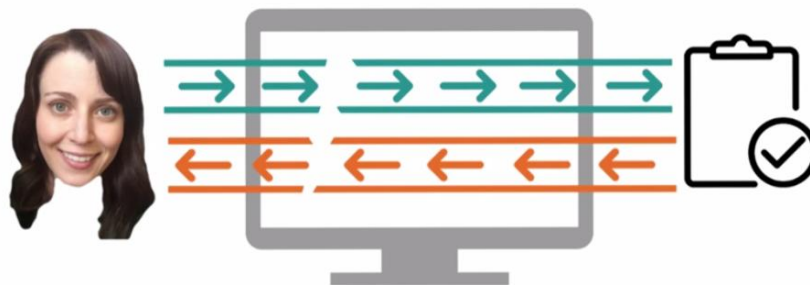
• Slip
• Mistake

I would argue this is a slip. Morgan knows who she means to message but the phone's behavior tricks her into sending things to the wrong people. What's more, this might be either an action based slip or memory lapse slip. Maybe Morgan is tapping the wrong person, or maybe she's forgetting to check who she's messaging. So take a second and brainstorm a design for this that can prevent this from happening in the future without over complicating the interaction too much. I would argue that the best way to do this is simply to show more pervasive reminders of who Morgan is currently texting. We could show the recipient's picture on the send button, for example. That way, the interaction is no more complex, but Morgan also has to directly acknowledge who she's messaging to send a message.

Learned Helplessness



The feedback cycle on HCI is reliant on a relationship between the user's input and the interface's output. The idea of this cycle is that the user learns from the output what they should have input. If they encounter an error, they receive feedback on how to avoid that next time. If they do something correctly, they see that the goal was accomplished. That's the principle of feedback.



But what happens when there is no discernible interaction between the input and the output? What happens when there's a break in this cycle? What happens when the user acts in the system over and over and over again, but never receives any output that actually helps? What if they never even receive output that indicates that the computer is understanding them or receiving input from them? That's when something called learned helplessness sets in.



The human working with the interface learns that they're helpless to actually use the system. They learn that there is no mapping between their input and the output that they receive. And as a result,

they believe that there's just nothing they can do to accomplish their goals. And no one wants to feel that way. No one wants to feel like no matter what they do they're doomed to failure. No one wants to feel like they're failing at something that everyone else seems to do very easily. And so it's very natural for people to develop this resistance to even trying to learn about this interface.

Learned Helplessness and Education



Just like mental models, learned helplessness is also a topic related as much to education as it is to HCI. If you've ever spent any time in a teaching role, you very likely encountered students that are very resistant to being taught. And the reason is they have learned that no matter what they do, they never succeed. They've learned to be helpless based on their past experiences. In all likelihood, there have actually been situations where you've been the one learning that you're helpless. In fact, if you're a parent, I can almost guarantee you've been in that situation. There are times when your child was crying and inconsolable and you had no clue why. We had one of those right before we filmed this video. Nothing you did helped. And you learned that you were helpless to figure out what your child wanted. So if you're a parent and you're dealing with learned helplessness as an interface designer, just imagine that you are the user and the interface is your screaming child. What feedback would you need from your child to figure out how you can help them? And how can you build that kind of feedback into your interface? [SOUND]

Quiz Expert Blind Spot

**"I am not
my user."**



Generally, when we're developing interfaces, we're going to be experts in those domains. It's rare that you design an interface to help people do something that you yourself don't know how to do. But as a result, there's risk for something called expert blind spot. When you're an expert in something, there are parts of the task that you do subconsciously without even really thinking about them. For example, a professional basketball player knows exactly where to place their hands on the ball when taking a shot. I know exactly what to do when I walk in the studio. Amanda knows exactly what to do when she gets behind the camera. And yet, if we were suddenly asked to train someone else, there are lots of things we'd forget to say or lots of things we would assume would just be obvious. That's exactly what you're doing when you're designing an interface. You're teaching the user how to use what you've designed. You're teaching them without the benefit of actually talking to them, explaining things to them, or demonstrating things for them. You're teaching them through the design of the interface. So, you have to make sure that you don't assume that they're an expert too. You have to overcome that expert blind spot because we are not our users. We are not the user. That can be the motto of all of HCI. I am not my user. Say it with me, I am not my user. One more time, I am not my user. Now type it.



Type, "I am not my user."



Now, write it on a Post-it note, and stick it to your monitor. If you wear glasses, write it on the inside of the lens. Record yourself saying it on your phone, and set that as your ringtone. Do whatever you have to do to remember. I am not my user.

Quiz: Reflections: Learned Helplessness and Expert Blindspot



In order for us to really sympathize with users suffering from the effects of learned helplessness and expert blind spot, it's important for us to understand what it's like to be in that position. We've all experienced these things at some point in life, although at the time, we might not have understood what was happening. So take a second and reflect on a time when you experienced learned helplessness and the effects of expert blind spot from someone trying to teach you something. It might have been in a class, it might be learning a new skill, or it might be doing something that everyone else seems to do just fine day to day, but for whatever reason, you've always struggled with.



The fact that I'm filming this in the kitchen probably tells you where I experience this. Anything related to cooking, I feel completely helpless. I've given myself food poisoning with undercooked meat lots of times, I once forgot to put the cheese on a grilled cheese sandwich. I accidentally made toast, and it wasn't even good toast. And I've always heard, it's just so easy, just follow the recipe, but no, it's not that easy, because many recipes are written for experts. So for example, here's a recipe from my wife's cookbook. It calls for a medium saucepan. Is this a medium saucepan? I have no idea. Calls for one egg

beaten with a splash of water. A splash, like a splash when you over fill a water bottle or a splash when your sibling soaks you at the pool? Pulse to combine. Cook until the edges are golden brown. What's golden brown? Give me a color code and I'll compare it, but otherwise, I don't know where on the spectrum from golden to brown, golden brown lies. These are examples of places where the directions are given in a way that assumes I already have some expertise, that I really don't have.

Conclusion to Mental Models



In this lesson, we talked about mental models. We discussed what mental models are and how the user uses them to make sense of a system.



We discussed how good representations can help users achieve strong models.



We then discussed how issues with interfaces can lead to two different kinds of user error. Slips and mistakes.



We then discussed learned helplessness which can come from giving poor feedback on user errors.

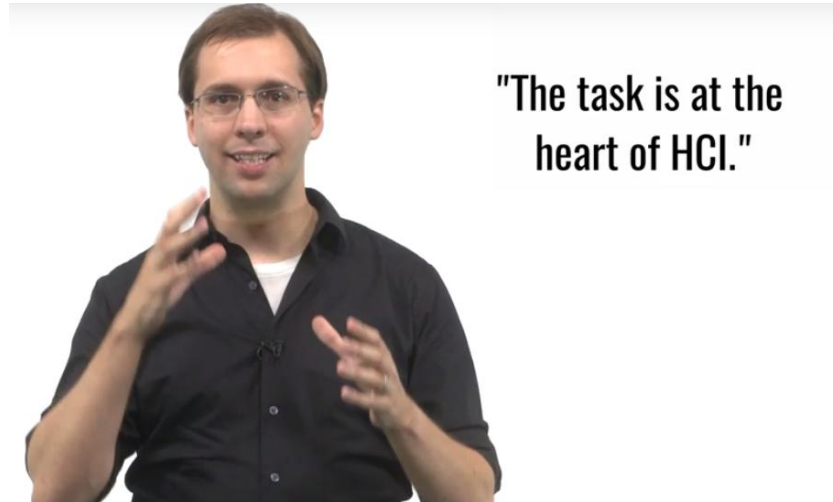


And finally, we discussed expert blindspot and the importance of understanding that you are not your own user.

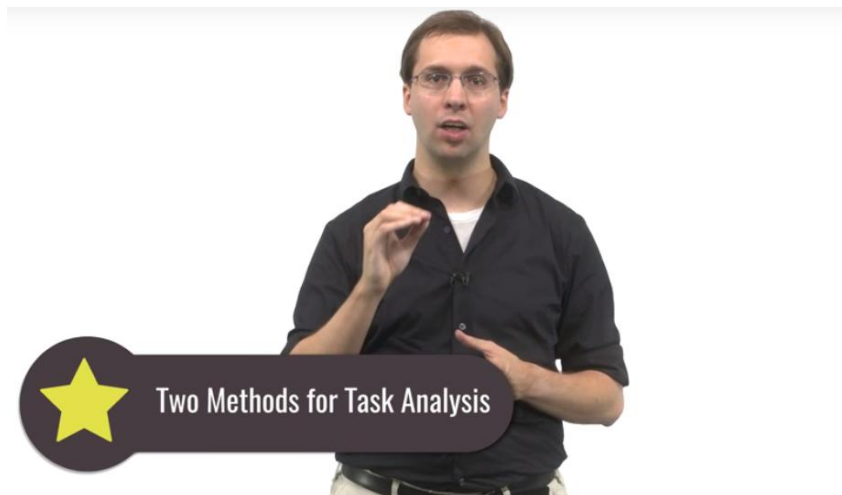
2.7 Task Analysis

Compiled by Shipra De, Summer 2017

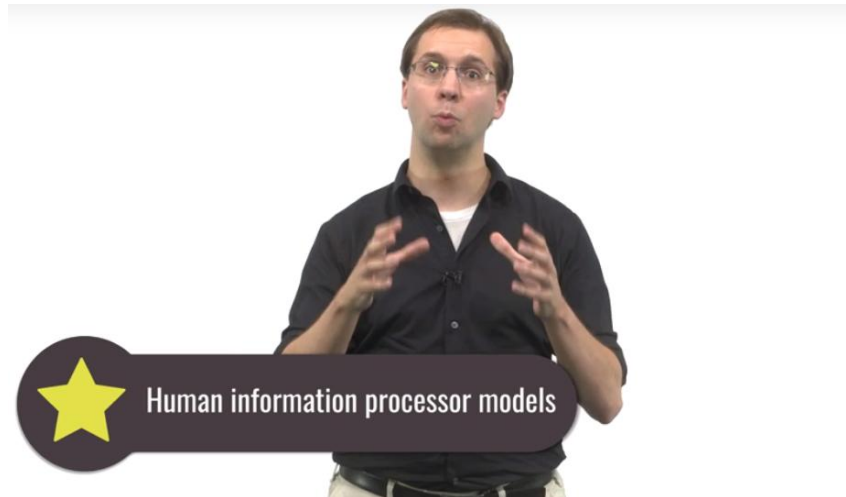
Introduction to Task Analysis



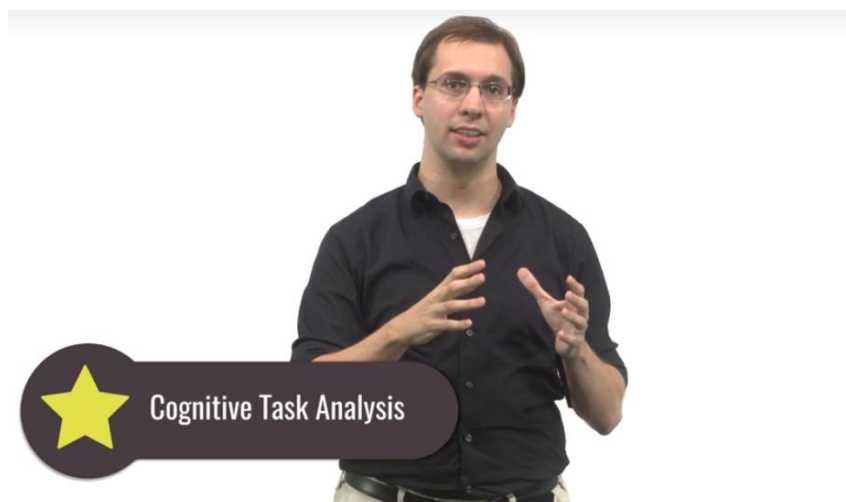
[MUSIC] When looking at human computer interaction, we're really looking at the tasks that users perform. We look at the tasks that they're performing now and we try to restructure those tasks to be more efficient using new interfaces. In all of this, the task is at the heart of the exercise. What task are they performing?



So today, we're going to talk about two methods for formally articulating the tasks that people are completing.



First, we'll discuss human information processor models, especially the GOMS Model which focuses on the input to the user and the output from the user. Note this is similar to the processor model of the user that we discuss elsewhere.

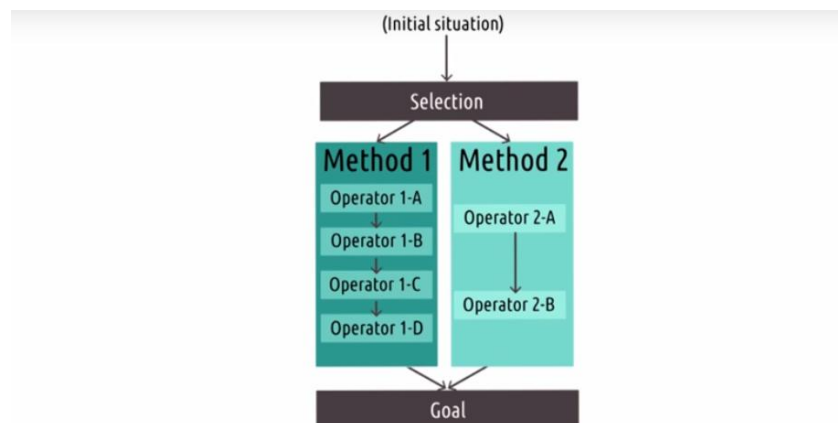


Second, we'll discuss cognitive task analysis. A way of trying to get inside the users head instead of focusing just on the input and the output. Note that that's similar to the predictor model of the user that we also discuss elsewhere.

The GOMS Model

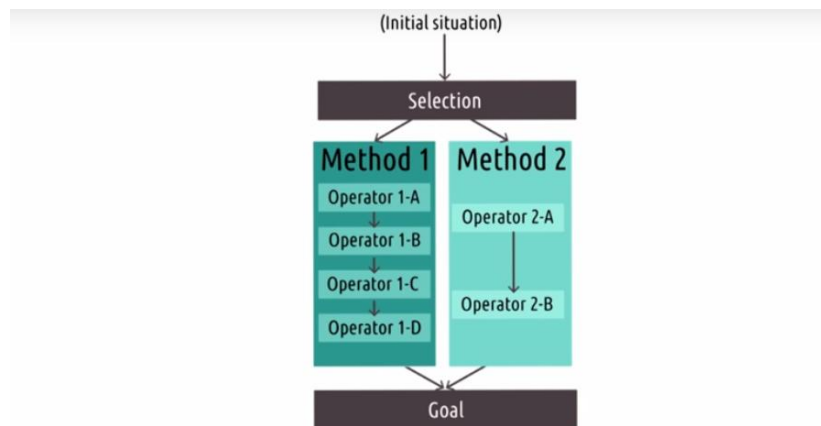
Goals Operators Methods Selection rules

The GOMS model is a human information processor model so it builds off the processor model of the human's role in a system. The GOMS model gets its name from the four sets of information it proposes gathering about a task. G, stands for the users Goals in the system. O, stands for the Operators the user can perform in the system. M stands for the Methods that the user can use to achieve those goals. And S stands for the Selection rules that the user uses to choose among different competing methods. So the GOMS Model proposes that every human interact with the system has a set of Goals that they want to accomplish. They have sent methods that they can choose from to accomplish those goals. Each of those methods is comprised of a series of Operators that carries out that method. And they have some Selection rules that help them decide what method to use and when.

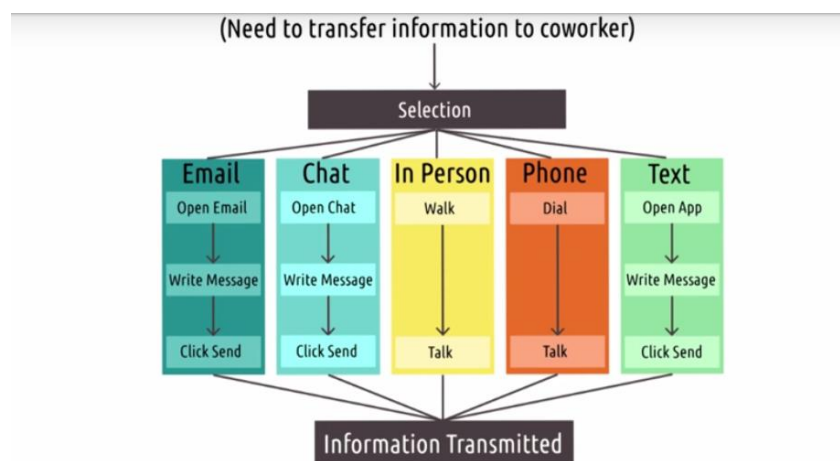


The GOMS model is often visualized like this. The user starts with some initial situation, and they have a goal in mind that they want to accomplish, so they apply their selection of rule to choose between different competing methods to accomplish that goal. Once they've chosen a method, they execute that series of operators and makes that goal a reality

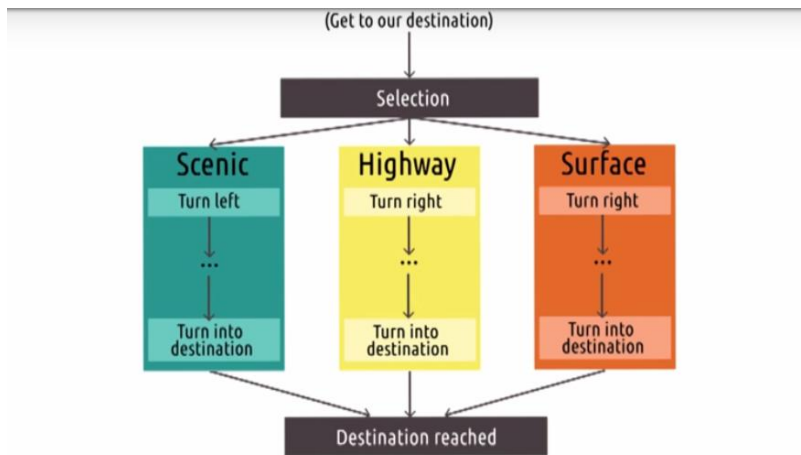
GOMS Model in Action



We can take the GOMS model and apply it to a number of different domains. So let's take the example of needing to communicate a message to a coworker.



We have an initial situation, which is the need to transfer information to a coworker. That carries with it the implicit goal of the information having been transferred. We might have a number of different methods in mind for how we could do that. We could email them, we could walk over and talk to them in person. And we also have some selection rules that dictate how we choose amongst these methods. If what we need to transfer is very time-sensitive, maybe we walk over and talk to them in person or call them on the phone. If the information we need to transfer is complex and detailed, maybe we write them an email. Or if it's more casual, maybe we chat with them or text them. No matter what method we choose, we then execute the series of operators that carries out that method, and the result is our goal is accomplished, the information has been transmitted.

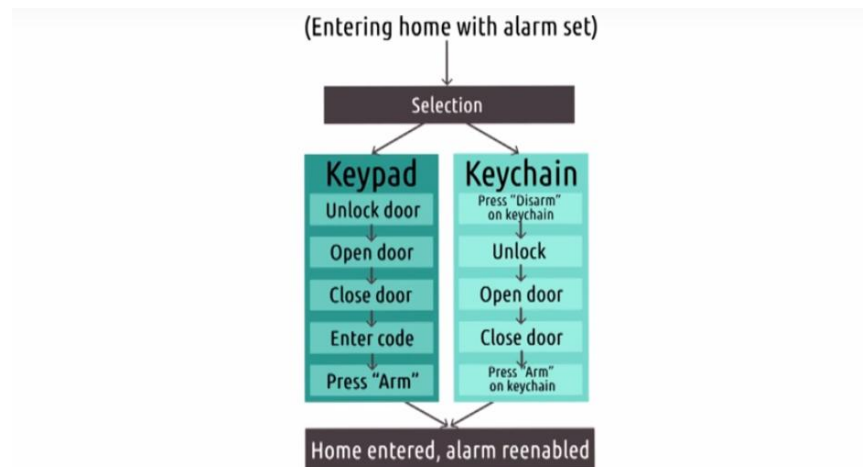


Or we could also take the problem of navigation. Our initial situation is the need to get to our destination, which carries with it the implicit goal of having reached our destination. We might have different methods, like take the scenic route, take the highway route, take the surface streets, and some selection rules that might say something like, when it's rush hour on the highway, take surface streets, or if it's not time sensitive, take the scenic route. After choosing, we execute those operators and reach our goal. So in this way, GOMS models capture our goals, our different methods for carrying out those goals, and the individual operators that we use to execute those methods.

Quiz: Design Challenge: Security System 1

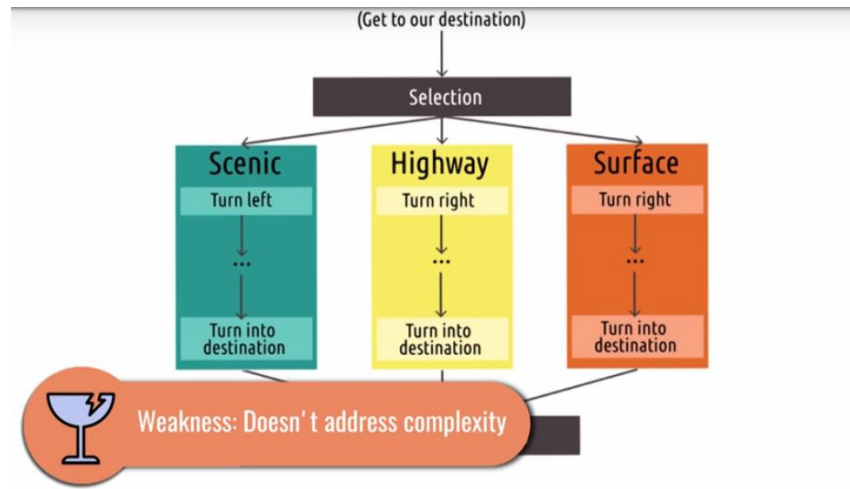


Let's try this out. We're going to watch Morgan enter the house and undo her security system two different ways. After you watch the video, try to outline Morgan's goals, outcomes, methods, and selection rules. [MUSIC] Now try to outline the goals, outcomes, methods and selection rules for these two methods of disabling the security system.

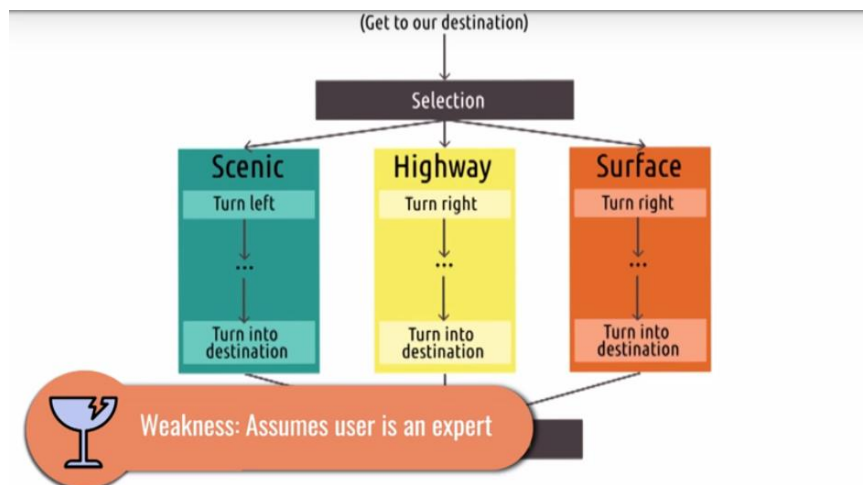


Here's one example of how you might design a GOMS model for disabling a security system. Our initial situation is that we're entering the home with the alarm set and we have two methods for disabling the alarm. We can use the keypad or we can use the keychain. Either way, our goal is that we've entered the home and reenabled the alarm. Our selection rules might be something like if we have our hands full, we're going to use the keypad so that we can get inside and put the stuff down. But if we don't have our hands full, we'll use the keychain. You might come up with other models for this that have either different methods, different operators, different selection rules. There are a lot of different ways we can capture a task with the GOMS model, depending on what you choose to focus on.

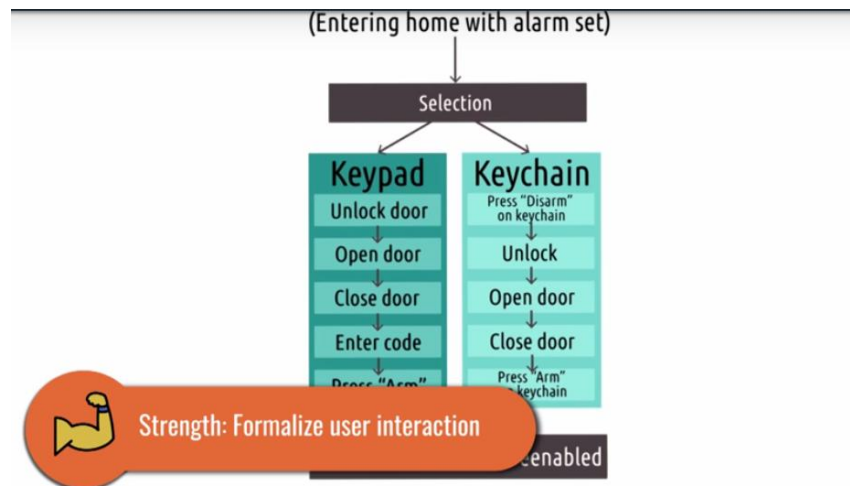
Strengths and Weaknesses of GOMS



There are strengths and weaknesses to the GOMS representation for tasks. One weakness is that it doesn't automatically address a lot of the complexity of these problems. For example, there are likely many different methods and sub-methods for addressing this goal. Before even getting to selection rules among what route to take, you might decide whether to take public transportation or whether to work from home that day. In parallel to that, even after deciding to drive, you might decide what car to take if your family has more than one car. The standard GOMS model leaves those kinds of things out, although there are augmented versions that have been created to deal with this kind of complexity, like CMN-GOMS and NGOMSL. We'll talk about those a bit more later.



A second weakness is that the GOMS model assumes the user already has these methods in mind. That means the user is already an expert in the area. GOMS does not do a good job of accounting for novices or accounting for user errors. For example, if you're driving in an unfamiliar location, you don't even know what the methods are, let alone how to choose among them.



The strength of GOMS, on the other hand, is its ability to formalize user interaction into steps that we can actually use to make predictions. We can measure how long each of these operators takes, and so we can predict the overall efficiency of using a certain interface. For example, in this GOMS model, if we had included the operator "pull keys out of the user's pocket" we might quickly identify that the relative efficiency of these two methods is very much dependent on how long that step takes. The keychain method may be a lot faster if the user can get their key chain out pretty quickly. But for other users, the fact that they need to pull something out of their pocket while holding bags or holding a baby makes the keypad a more efficient option. By performing that kind of reasoning, we can focus on areas that either method and the interface as a whole can be improved.

Paper Spotlight: "The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast"

The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast

BONNIE E. JOHN
Carnegie Mellon University
and
DAVID E. KIERAS
University of Michigan

Since the publication of *The Psychology of Human-Computer Interaction*, the GOMS model has

There are several varieties of GOMS models, these varieties share the commonality of goals operators, methods, and selection criteria, but they differ in what additional elements they provide. Bonnie John and David Kieras covered four popular variations in a paper from 1996. The first is the Vanilla GOMS we've talked about so far. And the other three, are KLM GOMS, CMN GOMS, and NGOMSL. Let's talk about what those acronyms actually mean.

2.1 The Keystroke-Level Model

The Keystroke-Level Model (KLM) is the simplest GOMS technique [Card et al. 1980a; 1983, Ch. 8]. To estimate execution time for a task, the analyst lists the sequence of operators and then totals the execution times for the individual operators. In particular, the analyst must specify the method used to accomplish each particular task instance. Other GOMS techniques discussed below predict the method given the task situation, but the KLM does not. Furthermore, the specified methods are limited to being in sequence form and containing only keystroke-level primitive operators. Given the task and the method, the KLM uses preestablished keystroke-level primitive operators to predict the time to execute the task.

The original KLM presentation included six types of operators: **K** to press a key or button, **P** to point with a mouse to a target on a display, **H** to home hands on the keyboard or other device, **D** to draw a line segment on a grid, **M** to mentally prepare to do an action or a closely related series of primitive actions, and **R** to represent the system response time during which the user has to wait for the system. Each of these operators has an estimate of execution time, either a single value, a parameterized estimate (e.g., **K** is dependent on typing speed and whether a key or mouse button

They start with the Keystroke-Level Model, which is the simplest technique. Here, the designer simply specifies the operators and execution times for an action, and sums them to find the complexity of an interaction. This method proposed six different types of operators, although for moderate interfaces, we would need some new ones to cover touch screens and other novel interfaces.

2.2 Card, Moran, and Newell GOMS (CMN-GOMS)

CMN-GOMS is the term we use to refer to the form of GOMS model presented in Card et al. [1983, Ch. 5] and Card et al. [1980b]. CMN-GOMS has a strict goal hierarchy. Methods are represented in an informal program form that can include submethods and conditionals. A CMN-GOMS model, given a particular task situation, can thus predict both operator sequence and execution time.

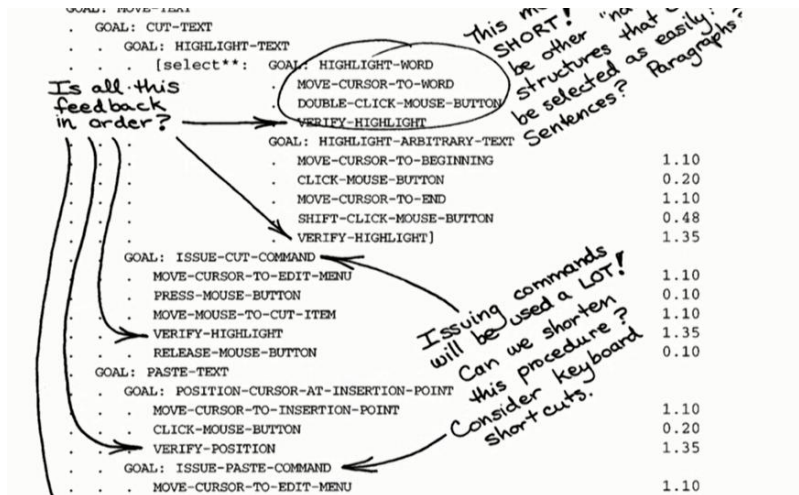
Card et al. [1983] do not describe the CMN-GOMS technique with an explicit “how to” guide, but their presentation of nine models at different levels of detail illustrates a breadth-first expansion of a goal hierarchy until the desired level of detail is attained. Card et al. report results in which such models predicted operator sequences and execution times for text-editing tasks, operating systems tasks, and the routine aspects of computer-aided VLSI layout. These examples are sufficiently detailed and

ACM Transactions on Computer-Human Interaction, Vol. 3, No. 4, December 1996.

A second variation, is CMN-GOMS. CMN-GOMS is an extension of GOMS that features sub methods and conditions in a strict GOAL hierarchy.



For example, here we see a hierarchy of GOALS, as well as the ability to choose between multiple GOALS in different areas. Notice also the level of granularity behind these GOMS models. The GOALS go all the way down to little GOALS, like moving text or deleting phrases. These are very, very low-level GOALS.



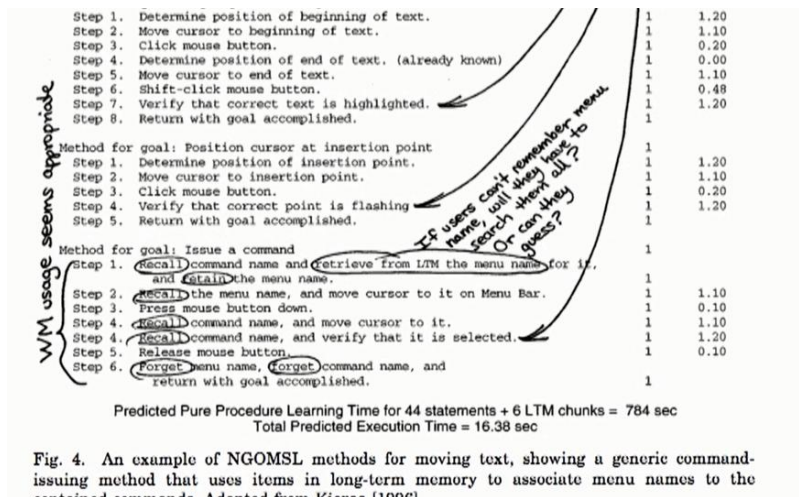
Notice also the way this model is being used. The authors are using it to find the places where there's a lot of complexity that can be cut out. They do this by modeling how long each individual action takes, as well as looking at the number of interactions required and seeing if they can be cut down a bit.

2.3 Natural GOMS Language (NGOMSL)

NGOMSL is a structured natural-language notation for representing GOMS models and a procedure for constructing them [Kieras 1988; 1996]. An NGOMSL model is in program form and provides predictions of operator sequence, execution time, and time to learn the methods. An analyst constructs an NGOMSL model by performing a top-down, breadth-first expansion of the user's top-level goals into methods, until the methods contain only primitive operators, typically keystroke-level operators. Like CMN-GOMS, NGOMSL models explicitly represent the goal structure, and so they can represent high-level goals such as collaboratively writing a research paper.

2.3.1 Architectural Basis and Constraints. The NGOMSL technique refines the basic GOMS concept by representing methods in terms of a cognitive architecture called *cognitive complexity theory* (CCT) [Bovair et al. 1990; Kieras and Polson 1985]. CCT assumes a simple serial-stage architecture in which working memory triggers production rules that apply at a fixed rate. These rules alter the contents of working memory or execute primitive external operators such as making a keystroke. GOMS methods

A third variation is called Natural GOMS Language. Natural GOMS language, or NGOMSL, is a natural language form of GOMS that lends itself to human interpretation. In all these cases, the important point of emphasis is the way that these models allow us to focus in on places where we might be asking too much of the user.



For example, in this model, the user is being asked to carry a lot of information in working memory. By making the assumptions and actions and operators this detailed, this model lets us target where working memory's being overly taxed in a way that we might miss when we're doing higher level designs.

5 Tips: Developing GOMS Models



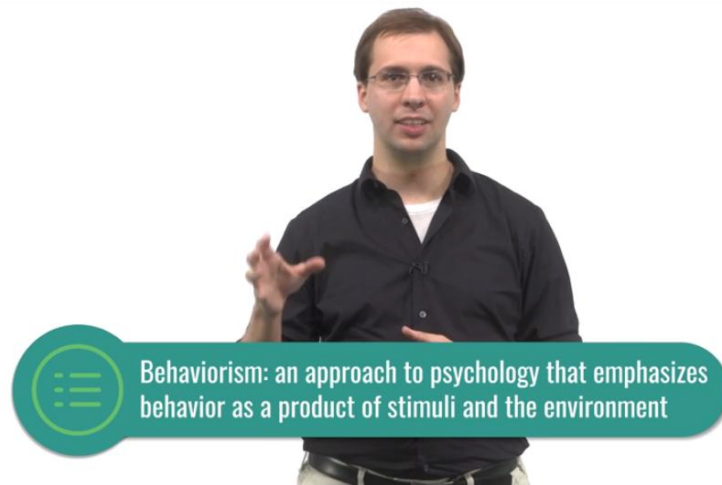
Here are five quick tips for developing GOMS models. Number 1, focus on small goals. We've used some pretty big examples, but GOMS was really designed to work in the context of very small goals, like navigating to the end of a document. You can abstract up from there, but start by identifying smaller, moment to moment goals. Number 2, nest goals instead of operators. It's possible to nest goals. For example, in our GOMS model of navigation, we could develop it further and break the overall task of navigating down to smaller goals like changing lanes or plotting routes. Operators, however, are the smallest atoms of a GOMS model. They don't break down any further and those must be the actual actions that are performed. Number 3, differentiate descriptive and prescriptive. Make sure to identify whether you're building a model of what people do or what you want them to do. You can build a goal model of what people should do with your interface. But you shouldn't trick yourself into thinking that's necessarily what they will do. Number 4, assign costs to operators. GOMS was designed to let us make predictions about how long certain methods will take. The only way we can do that is if we have some measurement of how long individual operations take. Usually this is time, but depending on the domain, we might be interested in phrasing the cost differently as well. Number 5, use GOMS to trim waste. One of the benefits of GOMS is it lets you visualize where an unnecessary number of operators are required to accomplish some task. That's bolstered by the costs we assign to those operators. So use GOMS to identify places where the number of operators required can be simplified by the interface.

GOMS to Cognitive Task Analysis



GOMS models are human information processor models. This method largely assumes the human is an input output machine, and it doesn't get too much into the internal reasoning of the human. Instead, it distills their reasoning into things that can be described explicitly like goals and methods. Some would argue, myself included, that human reasoning is actually too nuanced and complex to be so simplified. They, or we, advocate other models to get more into what goes on inside the user's head. That's where cognitive task analysis comes in. Cognitive task analysis is another way of examining tasks, but it puts a much higher emphasis on things like memory, attention, and cognitive load. Thus, cognitive task analysis adopts more of the predictor view of the human's role in the system.

Quiz: Reflections: Task Analysis



This conflict between more processor-oriented and more predictor-oriented models of the user actually gets at the core of an old battle in psychology between behaviorism and cognitivism. Behaviorism emphasized things that could be observed. We can see what input a person is receiving. We can see the output they're producing. And that might be all we need to understand the design of things.



Cognitivism, on the other hand, suggests we can and should get into the mind of what people are actually thinking and how systems like memory and learning and perception actually work. So take a moment and reflect on what you think about this. When designing interfaces, how much attention should you devote to observable goals, operators and methods? And how much do you devote to understanding internal thought processes, like cognition, learning, and memory?

You can probably guess my bias on this issue, given that I've already badmouthed the processor model and I also teach cognitive systems. So I'm generally going to prefer methods that focus on cognition. I

think it's important to note here though that both approaches have significant value. The GOMS model and its emphasis on identifying goals and operators is actually very useful in HCI. Because it forces us to very clearly and deliberately identify user goals and the sequence of actions that accomplish them. We can get so caught up in user experiences that we forget the user experience is born out of individual operators. So while I wouldn't advocate focusing solely on the user as some kind of input-output information processor, there's value in defining the user's operation as clearly and specifically as we define a computer's.

Cognitive Task Analysis



"Cognitive task analyses are concerned with the underlying thought process associated with performing a task."

Cognitive Task Analysis is not really a single method, but it's more of a type of method to approaching the evaluation of how people complete tasks. Performing a cognitive task analysis involves a number of different techniques and methods that we'll discuss more when we discuss the design life cycle. For right now, though, we're interested in what kinds of information we're trying to gather, not how we're gathering it. Cognitive task analyses are especially concerned with understanding the underlying thought process in performing a task. Not just what we can see, but specifically what we *can't* see. There are a lot of different methods for performing cognitive task analyses, but most methods follow a particular common sequence.



- ① Collecting preliminary knowledge
- ② Identify knowledge representations
- ③ Apply focused knowledge elicitation methods
- ④ Analyze and verify data acquired
- ⑤ Format results for intended application

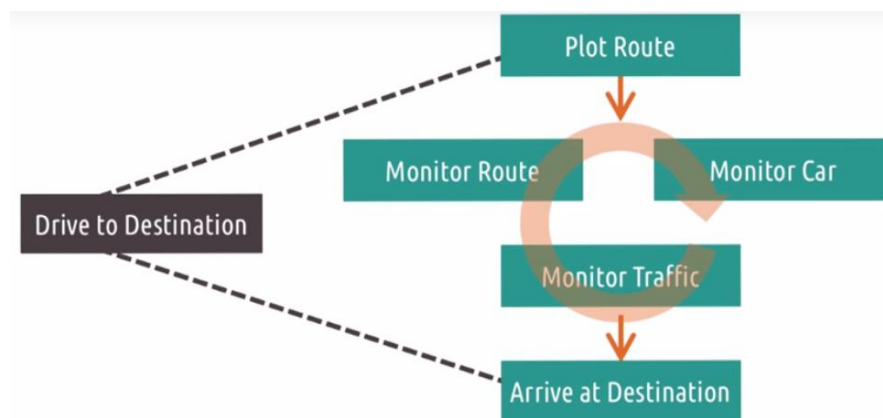
First, we want to collect some preliminary knowledge. While we as interface designers don't need to become experts in a field, we need a good bit of familiarity with it. So, we might observe people performing the task, for example. In navigation, for example, we might just watch someone driving and using a GPS.

Our second step is to identify knowledge representations. In other words, what kinds of things does the user need to know to complete their task? Note that we're not yet concerned with the actual knowledge they have, only the types or structures of the knowledge that they have. For example, we want to know: does this task involve a series of steps in a certain order? Does it involve a collection of tasks to check off in any order? Does it involve a web of knowledge to memorize? For navigation, for example, we would identify that the structure of the knowledge is a sequence of actions in order, as well as some knowledge of things to monitor as we go.

Then, in the third stage, we actually want to populate those knowledge representations. This is the stage where we start to recognize what the user actually knows. With navigation, for example, they know to start the GPS, to enter an address, and to obey the turns while monitoring traffic and speed and things like that. During this stage, we identify all the specific actions they take, the knowledge they must have in mind to take those actions, the interruptions that can change their thought processes, the equipment involved, and the sensory experience of the user. We do this by applying focused knowledge elicitation methods. In other words, we get users to tell us what's going on in their heads or what's going on in their environment. Or sometimes we do things that help us understand parts of the task that the user isn't even themselves aware of.

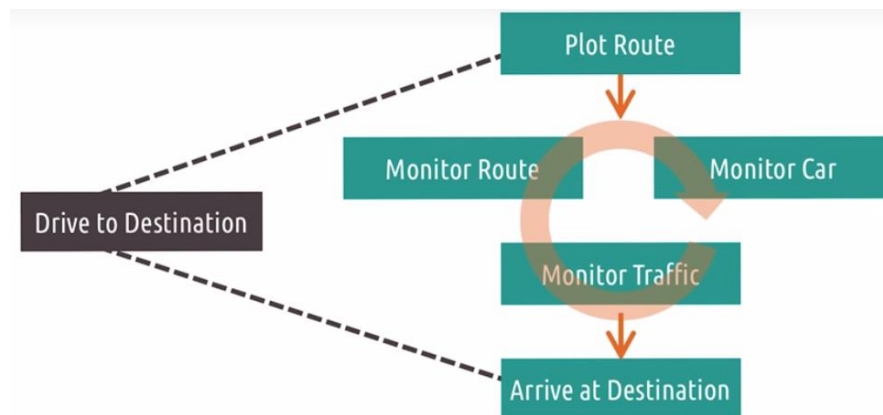
Then, we analyze and verify the data we acquired. Part of that is just confirming with the people we observed that our understanding is correct. We might watch them do something and infer it's for one reason when in reality it's for a very different reason. So we want to present to our users our results and make sure that they agree with our understanding of their task. Then, we attempt to formalize it into structures that can be compared and summarized across multiple data-gathering methods.

And finally, we format our results for the intended application. We need to take those results and format them in a way that's useful for interface design. We want to develop models that show what the user was thinking, feeling, and remembering at any given time and make those relationships explicit. The result might look something like this:

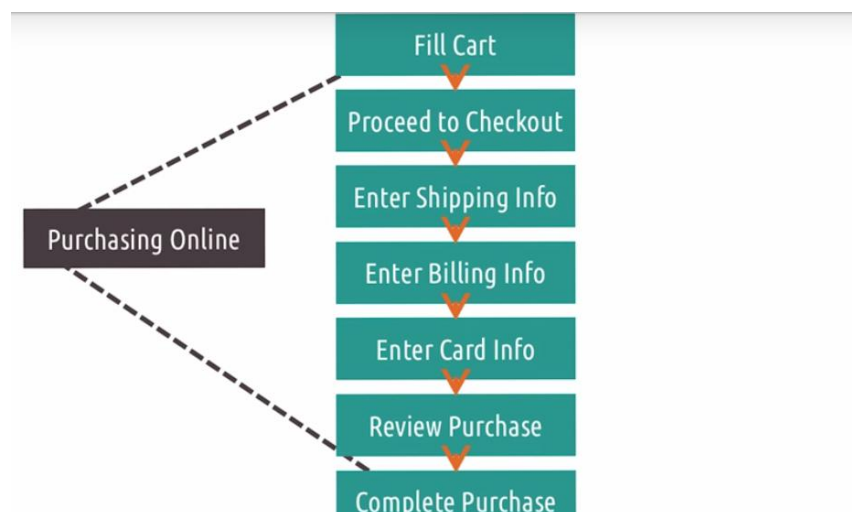


Here we see a very high-level model of the process of driving to a destination. What's interesting to note is that these tasks in the middle are highly cognitive rather than observable. If I had no knowledge about driving and sat in a passenger's seat watching the driver, I might never know that they're monitoring their route progress or keeping an eye on their dashboard for how much fuel they have left. If you have kids you may have experienced this personally, actually. To a kid sitting in the back seat, Mommy or Daddy are just sitting in the driver's seat just like they're sitting in the passenger's seat. They don't have a full understanding of the fact that you have a much higher cognitive load and you're doing a lot more things while you're driving than they are. That's because what you're doing is not observable. It's all in your head. So to get at these things I might have the user think out loud about what they're doing while they're going it. I might have them tell me what they're thinking while they're driving the car. That would give me some insights into these cognitive elements of the task.

Hierarchical Task Analysis

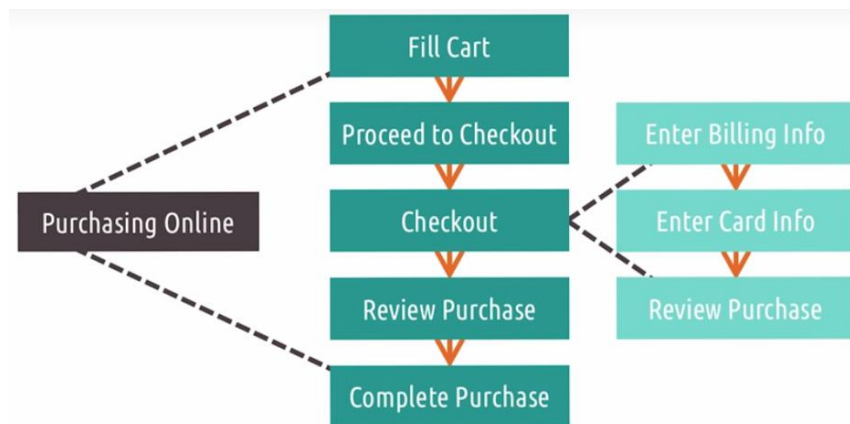


Cognitive task analysis advocates building models of human reasoning and decision-making in complex tasks. However, a challenge presented here is that very often, large tasks are actually composed of many multiple smaller tasks. We can see this plainly present in our cognitive model of driving. These tasks are so high-level that it's almost useless to describe driving in these terms. Each part could be broken down into various sub-tasks like iteratively checking all the cars around you or periodically checking how long it is until the next turn needs to be made. What's more, these smaller tasks could then be used in different contexts. Route-monitoring, for example, isn't only useful while driving a car -- it might be useful while running or biking or while riding as a passenger. Traffic monitoring might be something an autonomous vehicle might do, not just the human user. So, the analysis of a task in a particular context could be useful in designing interfaces for other contexts if we break the analysis down into subtasks. So let's take a simple example of this.



Here's a somewhat simple model of the act of buying something online. Notice that a lot of the tasks involved here are general to anyone shopping on any web site, and yet every web site needs to provide all of these functions. As a side note, notice also the interesting analogy going on with the top two.

Online, there is no cart or checkout station, but we borrowed those to help the user understand the shopping process online and how similar it is to shopping in a store.



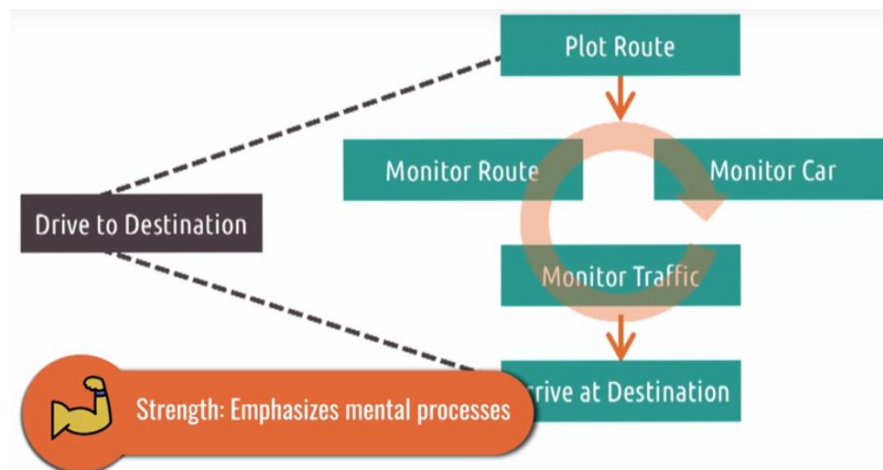
Anyhow, if we treat this cognitive task analysis more hierarchically we can start to see a well-defined subtask around this checkout process. Every online vendor I've ever encountered has these steps in its checkout process. Now because this is so well-defined, we could leverage existing tools, like existing payment widgets or something like PayPal. This hierarchical task analysis helps us understand what tools might already be available to accomplish certain portions of our task or how we might design certain things to transfer between different tasks and different contexts. Hierarchical task analysis also lets the designers of the site abstract over this part of the process and focus more on what might make their particular site unique. This type of task analysis is so common that you generally will find tasks and subtasks whenever you're looking at the results of a cognitive task analyses. So it's important to remember the strengths supplied by this hierarchy: abstracting out unnecessary details for a certain level or abstraction, modularizing designs or principles so they can be transferred between different tasks or different contexts, and organizing cognitive task analysis in a way that makes it easier to understand and reason over. Lastly, it's also important to note that the cognitive and hierarchical task analyses we've shown here are extremely simplistic, mostly, honestly, because of limited screen real estate. When you're creating real cognitive models, you'll likely have several levels of abstractions, several different states, and additional annotating information like what the user has to keep in mind or how they might be feeling at a certain stage of the analysis. We'll put some examples of some good, thorough models in the notes.

Quiz: Design Challenges: Security System 2

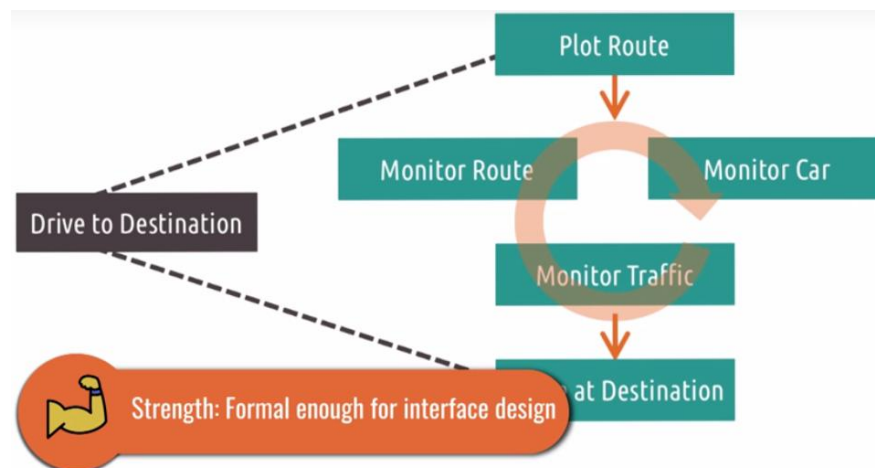
Let's watch the videos of Morgan disabling her security system again. This time, though, let's try to approach this from a more cognitive task analysis perspective. We won't be able to do that fully because doing a full cognitive task analysis means interviewing, asking the user to think out loud, and more, but we can at least try out this approach. Remember, in doing a cognitive task analysis for a task like this, your goal is to build a model of the sequence of thoughts going on inside the user's. Pay special attention to what she needs to remember at each step of the process.

What we saw here was that to get inside and disable the alarm, there was a sequence of actions that had to be completed, but some of them could be completed in different orders. If she used the keypad, she had to first unlock the door and then open the door. Then she could either disable the alarm on the keypad or close the door. And after closing the door, she could re-lock the door, though she could also do that before disarming the alarm. So there's some choices there. With the keychain, the sequence of tasks related to the door remain the same, but she had the option of disarming the alarm before even entering. However, that required remembering to do so. When using the keypad, she didn't have to remember because the alarm beeps at her until she turns it off. But she has to remember the key code. Performing these cognitive task analyses gives us the information necessary to evaluate different approaches and look for areas of improvement. For example, if she can disable the alarm just by pressing the keychain button, why does she need to press it at all? Why doesn't it just detect that she's coming in with a keychain in her pocket?

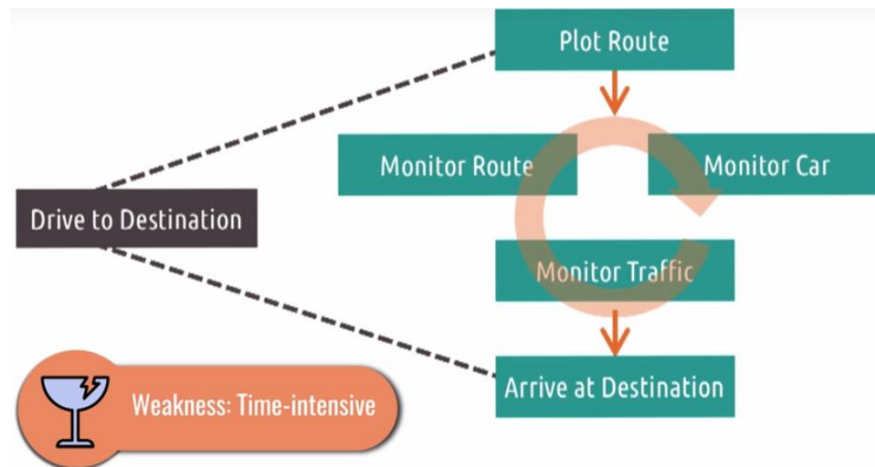
Cognitive Task Analysis Strengths and Weaknesses



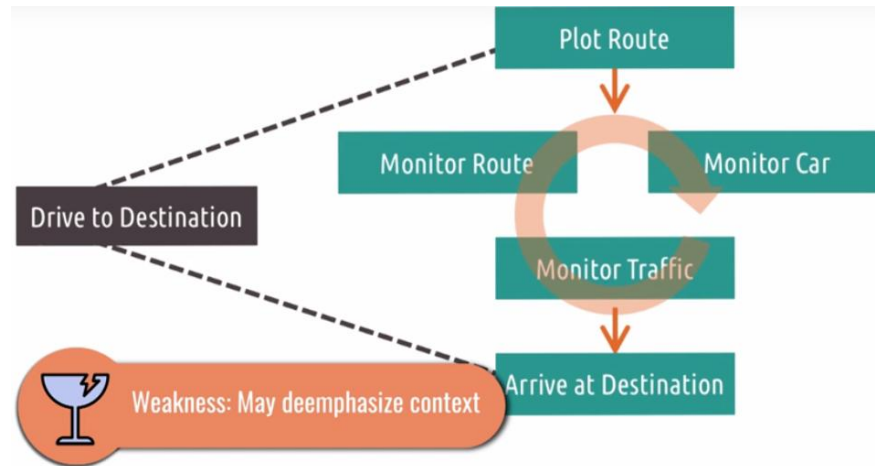
Just like GOMS models, cognitive task analysis also have some strength and some weaknesses. One strength is that they emphasize mental processes. Unlike the GOMS model, cognitive task analysis puts an emphasis on what goes on inside the users head. It's thus much better equipped to understand how experts think and work.



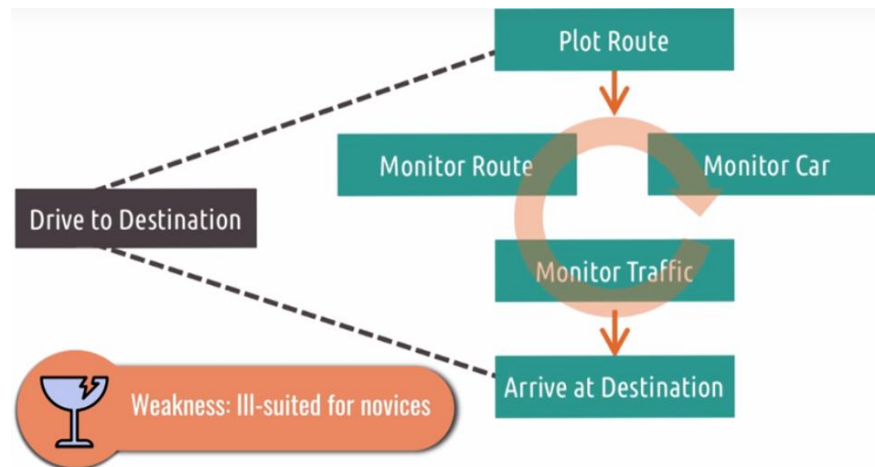
The information it generates is also formal enough to be used for interface design, for comparison in mode alternatives and more. There are disadvantages though.



One, cogni-task analysis are incredibly time-consuming to perform. They involve talking to multiple experts for extended period of time, then systematically analyzing the data.

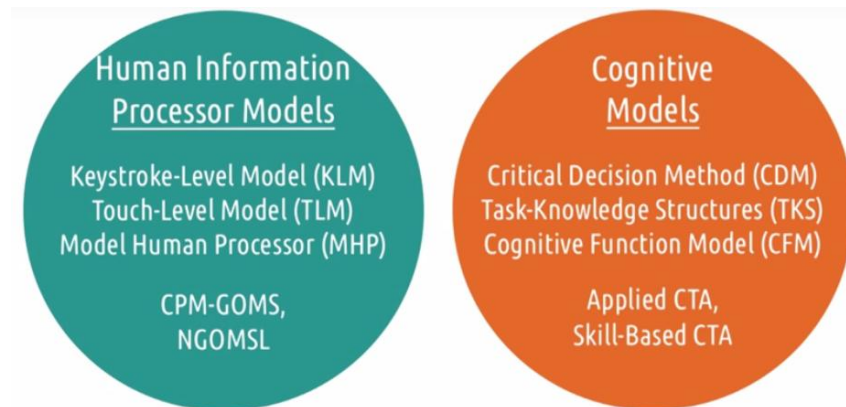


A second weakness is that cogni-task analysis risk deemphasizing context. In zooming in on the individual's own thought processes, cogni-task analysis risks deemphasizing details that are out in the world. Like the role of physical capabilities or interactions amongst different people, or different artifacts.



And third, like GOMS models, cogni-task analysis also isn't well suited to novices. It's well suited to expert users who have very strong models of the way they work and clearly understand their own mental thought processes. But they're not very well suited for novice users who are still trying to learn how to use an interface.

Other Task Analysis Frameworks

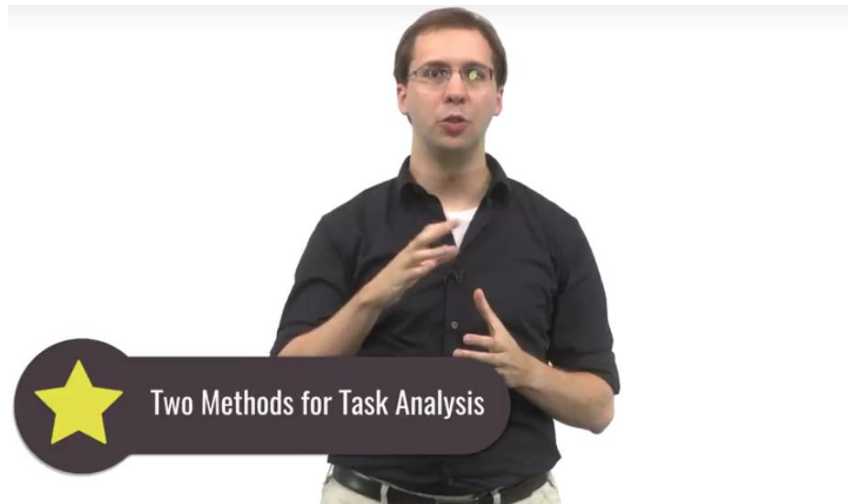


GOMS and cognitive tasks analysis are just two of the many alternatives to understanding how users approach tasks. More in line with the human information processor models, there exist models like KLM, TLM, and MHP, which capture even finer grain actions for estimating performance speed. There are other extensions to GOMS as well that add things like sub goals, or other ways of expressing content like CPM-GOMS and NGOMSL. CPM-GOMS focuses on parallel tasks, while NGOMSL provides a natural language interface for interacting with GOMS models. More on the lines of cognitive models, there exist other methods as well like CDM, TKS, CFM, Applied Cognitive Task Analyses, and Skill-Based Cognitive Task Analyses. CDM puts a focus on places where critical decisions occur. TKS focuses on the nature of humans' knowledge. CFM focuses on complexity. ACTA and Skill-Based CTA are two ways of gathering the information necessary to create a cognitive model. There also exist other frameworks more common in other disciplines, for example, production systems are common to an artificial intelligence. But they're intended to model cognitive systems the same way these cognitive models do. So we can apply production systems here as well and attempt to prescribe rules for users to follow.

Exploring HCI: Task Analysis

Every possible application of HCI involves users completing some sort of task. That task might be something within a domain. In educational technology, for example, that task might be learning how to do a certain kind of problem. If your area is more technological, the task might be something the user is doing through your application, like using virtual reality and gesture recognition to sculpt a virtual statue. Take a moment and try to think of the kinds of tasks you might be interested in exploring in your chosen application area. Do they lend themselves more to an information processor model like GOMS? Or to cognitive models like hierarchical task analysis? And how can you tell?

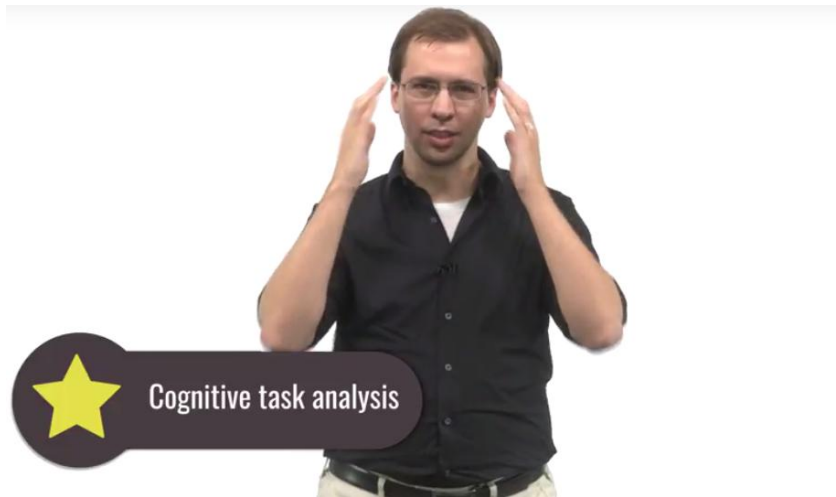
Conclusion to Task Analysis



Today we've talked at length about two general methods for approaching task analysis.



One, the GOMS family of approaches tries to distill tasks down to their goals, operators, methods and selection rules.



The other, cognitive task analyses, aim to get in the head of the user and understand what they're thinking, feeling and remembering at every stage of the task. When we discussed design life cycle we focus a bit on how to fill these models with information but our focus there is on the methods for gathering the information rather than the structure of the information itself. So it's important to keep these methods in mind when we talk about that.

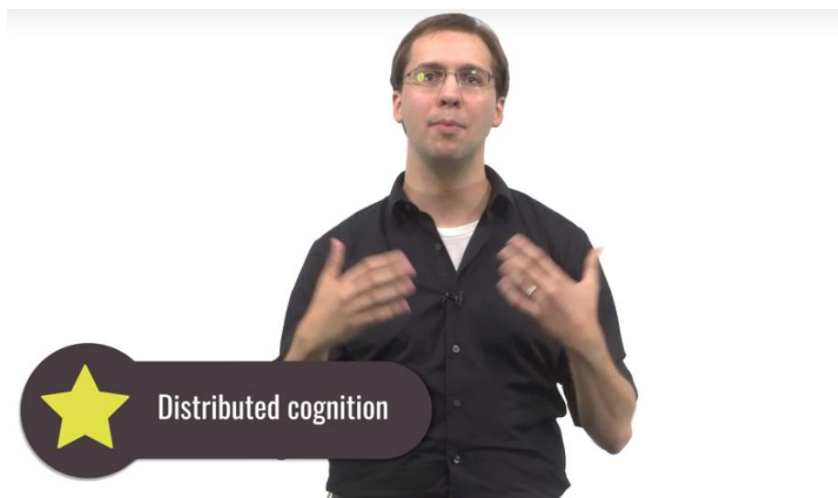
2.8 Distributed Cognition

Compiled by Shipra De, Summer 2017

Introduction to Distributed Cognition



[MUSIC] In discussing a human-computer interaction, there's often a tendency to look narrowly at the user interacting with the computer. Or slightly more broadly at the user interacting with the task through some computer. But many times we're interested in zooming even further out. We're interested, not only in the interaction between the human, the computer and the task, but also in the context in which that interaction takes place. So today we're going to look at four different models or theories, of the context surrounding HCI.

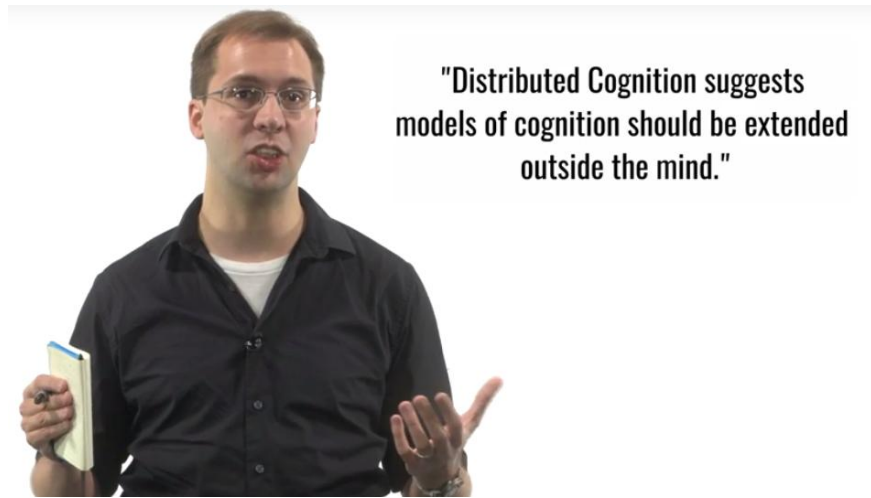


We'll focus primarily on distributed cognition, which is one of the dominant theories on the interplay between multiple agents, artifacts, and contexts.



We'll also touch on three other significant theories, social cognition, situated action, and activity theory.

Distributed Cognition



Cognition on its own is interested in thought processes and experiences and we naturally think of those as occurring inside the mind. But distributed cognition suggests that models of the mind should be extended outside the mind. This theory proposes expanding the unit we use to analyze intelligence from a single mind to a mind equipped with other minds and artifacts and their relationships among them.

So let's take an example or this. Amanda, give me a hard addition problem.

<Amanda: $1238 + 6437$ >

Okay can I do that in my head? No, I honestly can't even remember the numbers you just read to me. But I have a pen and paper here, and using those, I can pretty easily write down the numbers. So give those numbers to me again.

<Amanda: $1238 + 6437$ >

Okay, and using that I can now do the calculation by hand, and the answer is 7675.

Now, did I get smarter when I grabbed the pen and paper? Not really, not by the usual definition of "smarter" at least. But the system comprised of myself, the pen, the paper is a lot more than just my mind on its own. The cognition was distributed amongst these artifacts. Specifically, the paper took care of remembering the numbers for me and remembering and tracking my progress so. So, instead of adding 1238 plus 6437, I was really just adding $8+7$, $3+3$, $2+4$ and so on.

Paper Spotlight: "How a Cockpit Remembers Its Speeds"

COGNITIVE SCIENCE **19**, 265-288 (1995)

How a Cockpit Remembers Its Speeds

EDWIN HUTCHINS

University of California, San Diego

Cognitive science normally takes the individual agent as its unit of analysis. In many human endeavors, however, the outcomes of interest are not determined entirely by the information processing properties of individuals. Nor can they be inferred from the properties of the individual agents, alone, no matter how detailed the knowledge of the properties of these individuals may be. In some

One of the seminal works in distributed cognition research is a paper in the Journal of Cognitive Science from 1995 called "How a Cockpit Remembers its Speeds" by Edwin Hutchins. You might recognize Edwin Hutchins' name from our lesson on direct manipulation and invisible interfaces. He was one of the coauthors there along with Don Norman. It's one of my favorite papers, in part simply because of the very subtle change in emphasis in its title. We tend to think of 'remembering' as a uniquely human or biological behavior. We describe computers as having memory, but we don't usually describe computers as remembering things. Remembering is more of a human behavior. But the paper title twists that a little bit. It isn't the human, it's not the pilot remembering, it's the cockpit remembering. What is the cockpit? The cockpit is a collection of controls, sensors, and interfaces, as well as the pilots themselves. The paper title tells us that it's this entire system: the pilots, the sensors, the controls, and the interfaces among them that do the remembering. The system as a whole, the cockpit as a whole is remembering the speed, not just the human pilot or pilots inside the cockpit. No individual part in isolation remembers what the system as a whole can remember.

How a Cockpit Remembers Its Speed

COGNITIVE SCIENCE **19**, 265-288 (1995)

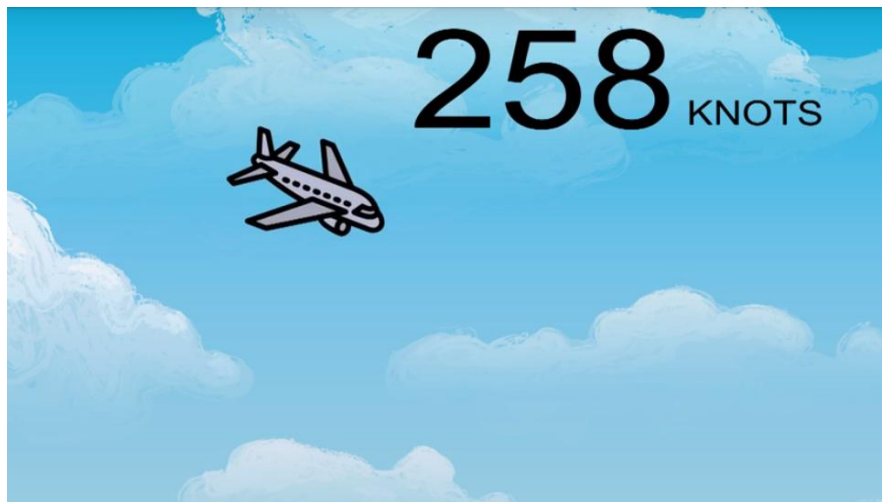
How a Cockpit Remembers Its Speeds

EDWIN HUTCHINS

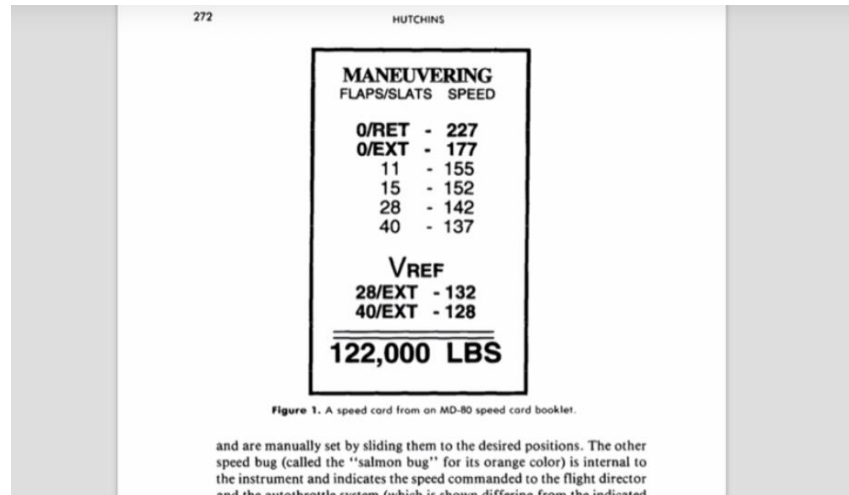
University of California, San Diego

Cognitive science normally takes the individual agent as its unit of analysis. In many human endeavors, however, the outcomes of interest are not determined entirely by the information processing properties of individuals. Nor can they be inferred from the properties of the individual agents, alone, no matter how detailed the knowledge of the properties of those individuals may be. In commercial aviation, for example, the successful completion of a flight is produced

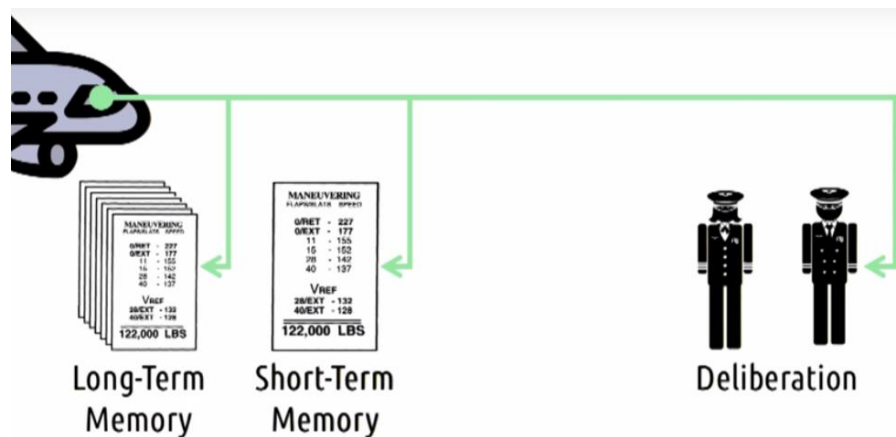
In order to understand the application of distributed cognition to the cockpit, it's important for us to first understand what challenge it's addressing. The technical reasons for this are a bit complex, and I strongly encourage reading the full paper to get the full picture. But to understand the simplified description I'll give here, here's what you need to know.



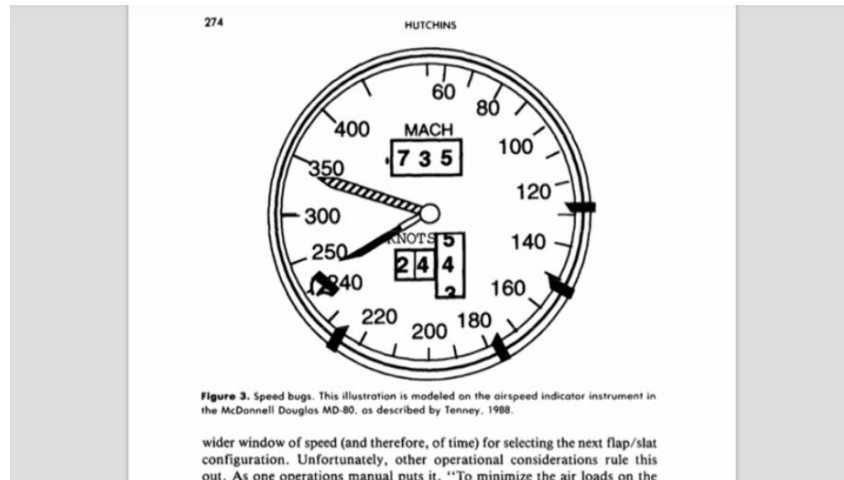
When a plane is descending for landing, there exists a number of different changes the pilots need to make to the wing configurations. These changes are made at different speeds during the descent. When the plane slows down to a certain speed, it demands a certain change to the wing configuration. The speeds at which these configuration changes must happen differ based on a number of different variables. So for every flight there's a unique set of speeds that must be remembered. That's why the title of this paper is, How a Cockpit Remembers Its Speeds, Speeds, plural. It isn't just remembering how fast it's going now, it's remembering a sequence of speeds at which multiple changes must be made. The configuration changes to the wings must be made during the descent at narrowly defined times. That creates a high cognitive load. Pilots must act quickly. And mistakes could mean the deaths of themselves and hundreds of others. So how do they do this?



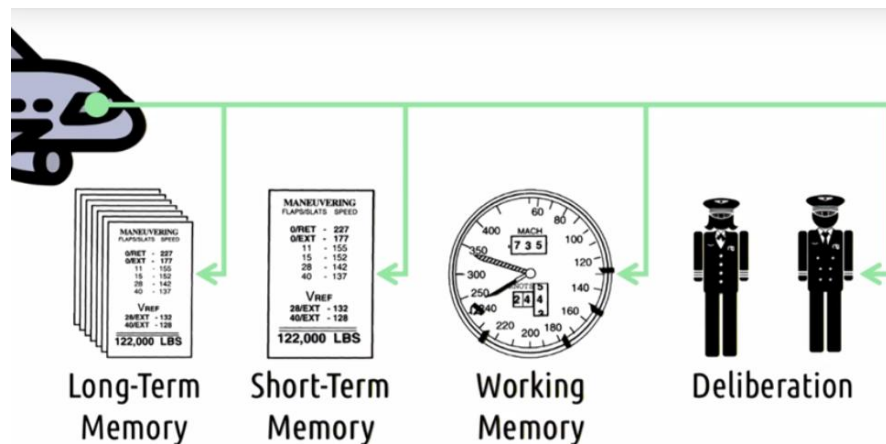
First, the pilots have pages that contain the speeds for their descent, based on different parameters. The cockpit itself has an entire booklet of pages like this.



So we know that the cockpit has its pilots who are responsible for actually reasoning over things. But that booklet forms that cockpit's long term memory of different speeds for different parameters. Then, prior to the descent, the pilots find the page from that booklet that corresponds to their current parameters. They pull it out and pin it up inside the cockpit. That way, the sheet is accessible to both pilots. And they're able to check one another's actions throughout. This becomes one form of the cockpit's short term memory, a temporary representation of the current speeds. At this point, we have to attribute knowledge of those speeds to the cockpit itself. If we were to isolate either pilot, they would be unable to say what the speeds are from memory, but without the pilots to interpret those speeds, the card itself is meaningless. So it's the system of the entire cockpit, including the pilots, the booklet and the current card that remembers the speeds.



Then as the pilots begin the descent, they mark the different speeds right on the speedometer with these little speed bugs. The speed bugs tell them which speeds to remember in a way that can just be visually compared. When they see the speedometer pass a speed bug, they know it's time to make a certain change.



This is like the working memory for the cockpit. The short-term memory stores the numbers in a way that the pilots can reason over, but the speed bugs on the speedometer store them in a way that they can very quickly just visually compare. They don't need to remember the numbers themselves, or do any math. All they have to do is visually compare the speed bugs to the current position of the speedometer. So what do we see from the system as a whole? Well, we see the long term memory in the book of cards. We see a short term memory in the card they selected. We see a working memory in the speed bugs on the speedometer. And we see decisions on when to make configuration changes distributed across the pilots and these artifacts. No single part of this cockpit, not the pilots, not the speed bugs, not the cards, could perform the action necessary to land a plane on their own. It's only the system as a whole that does so. That's the essence of distributive cognition. The cognition involved in landing this plane is distributed across the components of the system. This is a deeper notion than

just using interfaces to help us do tasks. The important thing here is that these different interfaces serve cognitive roles in the system.

Distributed Cognition and Cognitive Load



Distributed cognition is deeply related to the idea of cognitive load. Recall the cognitive load refers to your mind's ability to only deal with a certain amount of information at a time. Distributed cognition suggests that artifacts add additional cognitive resources. That means the same cognitive load is distributed across a greater number of resources. Artifacts are like plugging extra memory into your brain. Driving is a good example of this. Sometimes while driving, your cognitive load can be very, very high. You have to keep track of the other cars around you. You have to keep track of your own speed to monitor your route planning. You have to make predictions about traffic patterns. You have to pay attention to your own level of gasoline, or in my case, electric charge. You might be attending to something in your car as well, like talking to your passenger, or keeping an eye on your child in the back seat. It can be a big challenge. A GPS is a way of off-loading one of the tasks, navigation, on to another system. And thus, your cognition is now distributed between you and the GPS. Turn on cruise control and now it's distributed across the car, as well. Your off-loading the task of tracking your speed to the car. Every task you also de-artifacts, decreases your own personal cognitive load.

Quiz: Exercise: Distributed Cognition



Let's analyze a simple task from the perspective of distributed cognition. Here we see Morgan paying some bills the old fashioned way. For each bill she pulls it off the pile, reads it, writes a check and puts them together in a stack on the right. Where do we delineate this system? What are its parts?

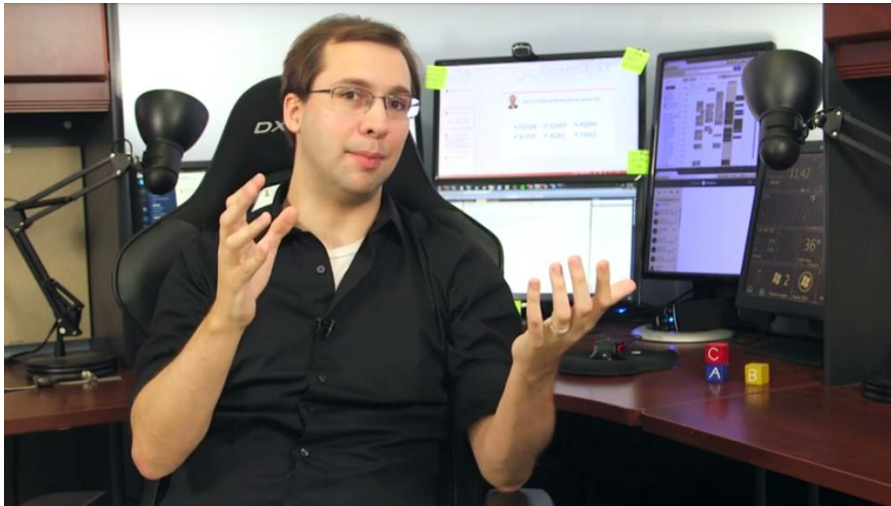


Which of the following are playing cognitive roles in the system that is paying Morgan's bills?

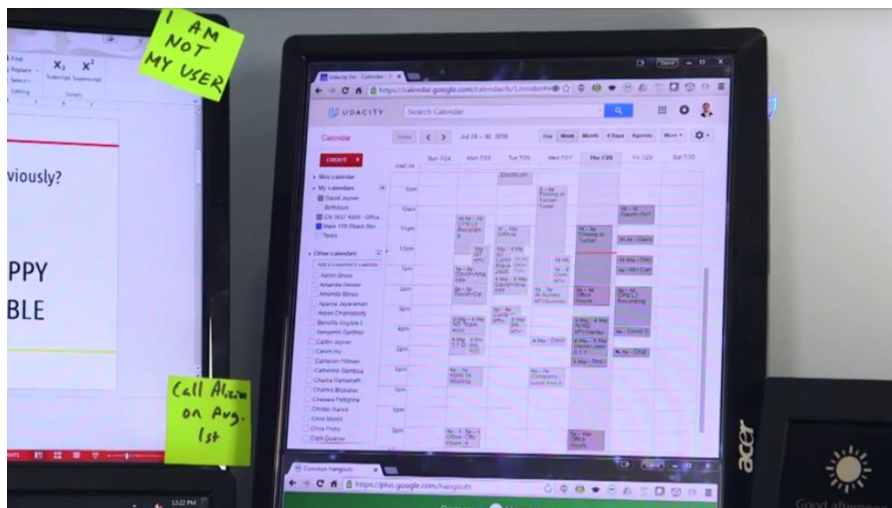
- | | |
|---|---|
| <input type="checkbox"/> Morgan herself | <input type="checkbox"/> The checkbook |
| <input type="checkbox"/> Morgan's chair | <input type="checkbox"/> The two piles of bills |
| <input type="checkbox"/> Morgan's table | <input type="checkbox"/> The bills themselves |
| <input type="checkbox"/> The pen | <input type="checkbox"/> The light overhead |

We're interested in any part of the system that performs some of the cognition for Morgan. While the chair, table, and light over head make this possible, they aren't serving any cognitive roles. Morgan herself, of course, is, and two piles of bills are too. They are an external memory of what bills Morgan has already paid, and what she still needs to pay. This way she doesn't have to mentally keep track of what bills she has left to do. The bills themselves remember a lot of the information for her as well like the amounts and the destinations they need to be sent to. What about the pen and checkbook? That's when things start to get a little bit more tricky. The checkbook itself is part of the system because it takes care of the record keeping task for Morgan. Checkbooks create carbon copies, which means Morgan doesn't have to think about tracking the checks manually. The pen is a means of communicating between these systems, which means it's part of our distributed cognition system as well.

Distributed Cognition as a Lens



Something important to note is that distributed cognition isn't really another design principle. Distributed cognition is more of a way of looking at interface design. It's a way of approaching the problem that puts your attention squarely on how to extend the mind across artifacts. We can actually view many of our design principles as examples of distributed cognition. So this is my computer, and when I set this up, I wasn't thinking about it in terms of distributed cognition. And yet we can use distributive cognition as a lens through which to view this design.



For example, I always have my calendar open on the right. That's a way of off loading having to keep track of my daily schedule in working memory. It bugs me if I have a teleconference to attend or somewhere I need to go. In fact I rely on this so much it gets me in trouble. It doesn't keep track of where I need to be for a given meeting and if I fail to keep track of that in working memory I might end up at home when I need to be at Georgia Tech. We can even view trivial things like a clock as an example of distributed cognition that prevents me from having to keep track of the passage of time

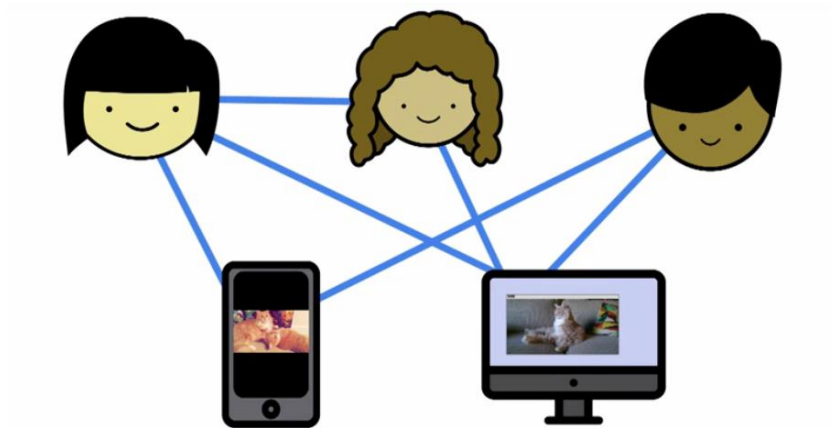
manually. The point is that distributed cognition is a way of looking at interfaces and interface design that focuses your attention on what systems as a whole can accomplish as opposed to individuals on their own.

Quiz: Reflections: Distributed Cognition

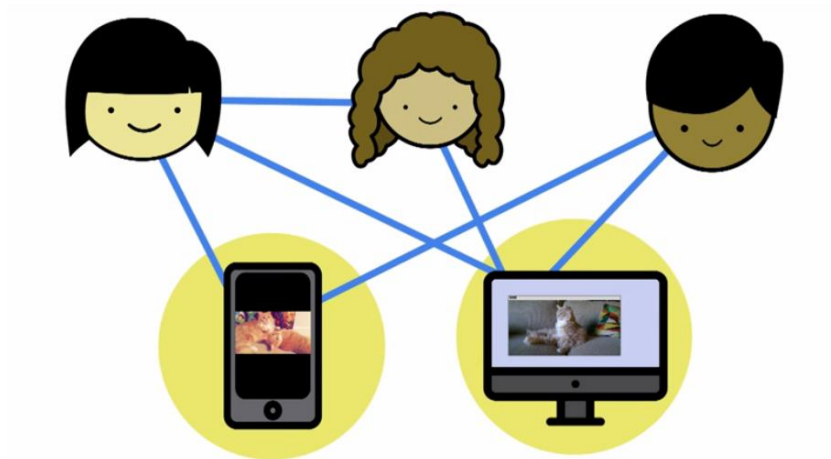
Distributed cognition is a fun one to reflect on because we can take it to some pretty silly extremes. We can go so far as to say that I don't heat up my dinner. The system comprised of myself and the microwave heats it up. And I offload the need to track the time to cook on to my microwave's timer. And that's a perfectly valid way of looking at things. But what we're interested in is places where interfaces don't just make our lives more convenient. We're interested in places where they systems comprised of us and interfaces are capable of doing more, specifically because those interfaces exhibit certain cognitive qualities. The systems might perceive, they might remember, they might learn, they might act on our behalf. In some way they're all floating a cognitive task from us. And as a result, the system comprised of us and the interface, is capable of doing more. So reflect on that a bit, what is the place where the system comprised of you and some number of interfaces is capable of doing more than you alone? Specifically, because of the cognitive qualities that the interfaces possess.

Almost any interface on the computer can be analyzed from the perspective of distributed cognition but right now, I'm most interested in my email. My email is an unbelievable extension of my long term memory because whenever I see anything in email, I know I don't actually need to commit it to my own long-term memory. It's there, it's safe forever, and if I ever need to find it again, I'll be able to find it. Now, finding it might take some work sometimes, but rarely as much work as manually remembering it. For me, I also mark messages as unread if I'm the one they're waiting on, or if I need to make sure I come back to them. And so, my email is an external memory of both all my communications via email, and tasks that are waiting on me to move forward.

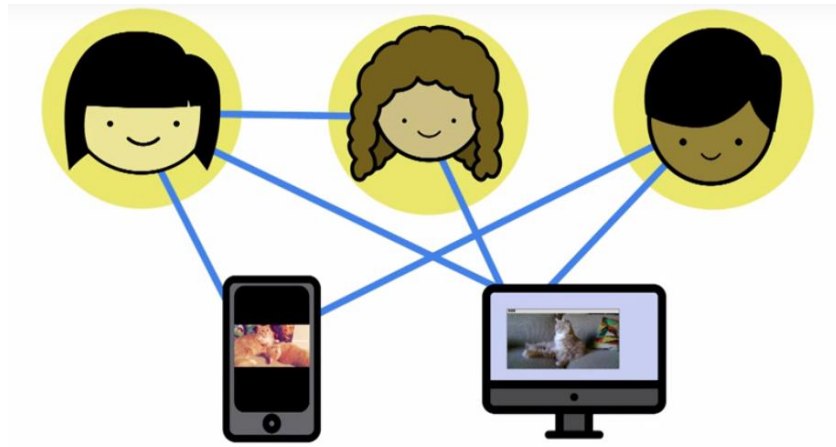
Distributed Cognition to Social Cognition



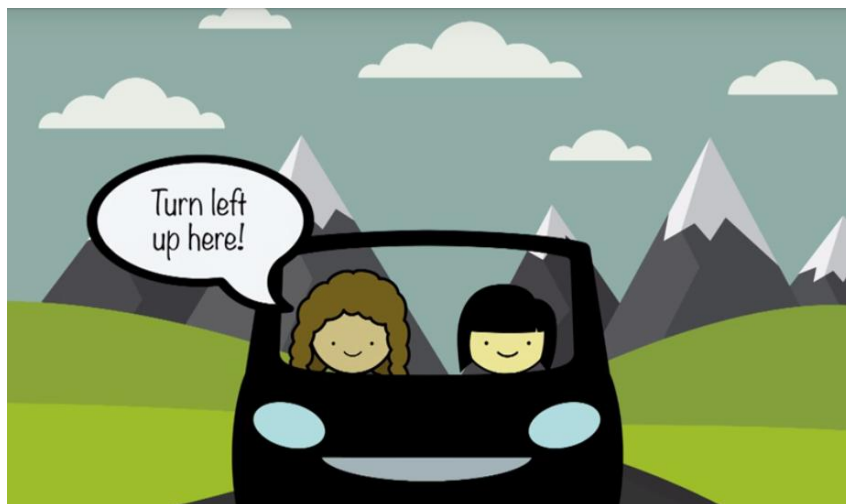
Distributed cognition is concerned with how the mind can be extended by relations with other artifacts and other individuals.



Because we're interface designers, we probably focus most of our time on the artifacts part of that. After all, even though we're designing tasks, the artifacts are what we're actually creating that's out in the world.



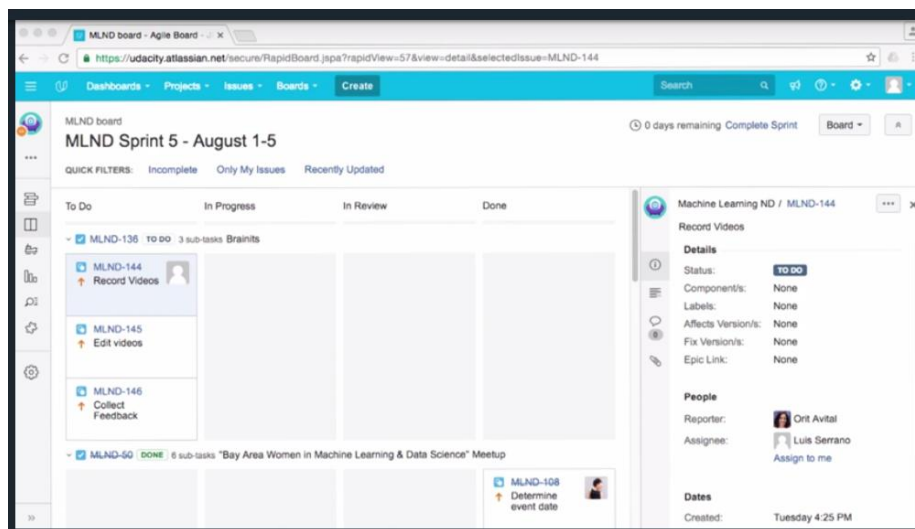
But the other part of distributed cognition, distributing across individuals presents a powerful opportunity as well. This used to be far more important, actually.



Before the days of GPS navigation, a different form of navigation assistance existed. It was your spouse or your friend sitting in the passenger seat, reading a map and calling out directions to you.



And while mobile devices and artificial intelligence may have replaced humans in some such systems, there are still lots of places where the role of distributing across humans is crucial. Here's an example of this in action today.



At Udacity, we use a tool for managing projects called JIRA. It breaks down projects into multiple pieces that can be moved through a series of steps and assigned to different responsible individuals. The entire value of JIRA is that it manages distributing tasks across members of a team. Thus, when a project is completed, it is completed by the system comprising the individual team members and JIRA itself.

Social Cognition



The social portion of distributed cognition is concerned with how social connections create systems that can, together, accomplish tasks. So for example, you and your friend sitting in the passenger seat, together form a system capable of navigating to a new destination without a GPS. But social cognition is not only concerned with how social relationships combine to accomplish tasks. It's also concerned with the cognitive underpinnings of social interactions themselves. It's interested in how perception, memory, and learning relate to social phenomena. As interface designers, though, why do we care? Well, in case you haven't noticed, one of the most common applications of interface design today involves social media. Everything is becoming social. Facebook tries to tell you when your friends are already nearby. Udacity tries to connect you with other students working on the same material as you. Video games are increasingly trying to convince you to share your achievements and highlights with your friends. And yet, often times, our interfaces are at odds with how we really think about social interaction. Designing for this well involves understanding the cognitive underpinnings of social relationships. My PlayStation, for example, has a feature for finding my real life friends, and then communicating to them my gaming habits. But really, I probably don't want them to know how much I might play video games. If I come unprepared for recording today, I certainly don't want Amanda to know it was because I was playing Skyrim for six hours last night. So if we're going to design interfaces that integrate with social interactions, we have to understand how social interactions actually work. So an understanding of social cognition is very important if that's the direction you want to take.

Quiz: Design Challenge: Social Cognition



Let's talk about challenge of designing for social relationships. I like to play video games. I'm friends with people from work. So it's natural that I might want to play games with people from work. But at the same time, my relationship with people from work isn't purely social. If they see I'm playing a game, maybe they say, hey, David's got some free time. I should ask him to help me out with something. Or if they see I spend a lot of time playing video games, maybe they more generally say hey, David's got plenty of time to take on a new tasks. How do we design a social video gaming system that nonetheless protects against these kinds of perceptions?

There are lot of creative ways we might tackle this problem. One might be a base social video game relationship around something like tinder. Tinder, if this is still around by the time you're watching this, is a dating app were you express interest in another's in are only connected if they also express interest in you. We can apply the same colonoscopy to video games. You can set it such that My Contacts can't just look up my game playing habits. But if they're also playing or interested in playing, they'll learn that I am playing as well. In terms of social cognition, that's kind of getting at the idea of an in-group. Your behaviors are only seen by those who share them and thus are in no position to judge them.

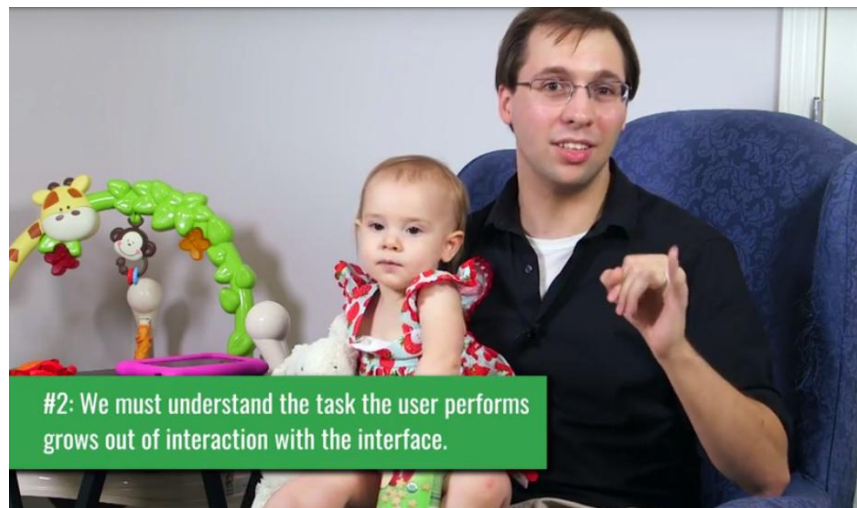
Situated Action



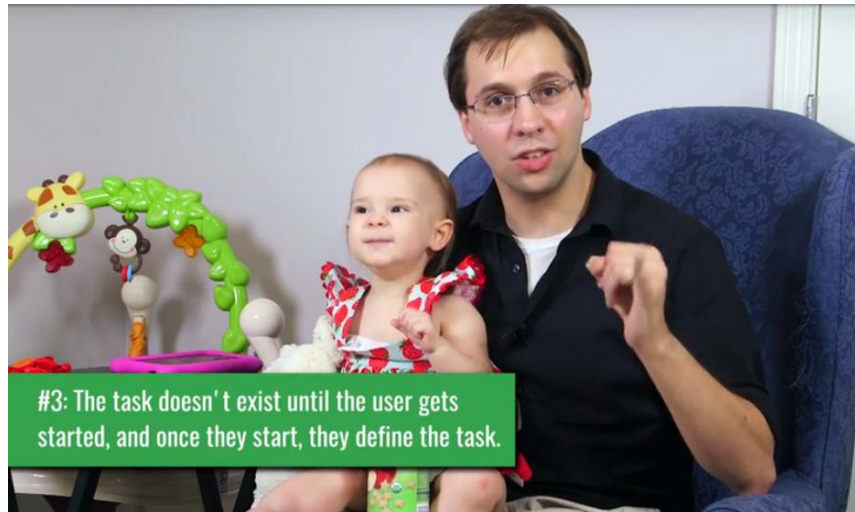
Like distributed cognition, situated action is strongly concerned with the context within which people interact. But unlike distributed cognition, situated action is not interested in the long-term and enduring or permanent interactions amongst these things. That's not to say the theory denies the existence of long-term memory, but it just has a different focus. Situated action focuses on humans as improvisers. It's interested not in the kinds of problems that people have solved before, but in the kinds of novel, situational problems that arise all the time. So, for an example of this, this is the first time I'm filming with my daughter on camera. I don't know how she'll act. I don't have contingency plans set up for how to react if she acts strangely or if she distracts me from my script. I'm just going to figure this out as I go along. This is the kind of interaction that situated action is interested in. And that's an important view for us to hold as interface designers. While we like to think we're in charge of structuring the task for our users, in reality the tasks that we perform is going to grow out of their interaction. We can try our best to guide it in certain directions, but until we actually get our hands on it, the task doesn't exist. The task of me filming with my daughter didn't exist until this moment. Once we've got our hands on it, the task is what we do, not what we designed. So as users, when we use an interface, when we actually do something, we're defining the task as we go along. So, there are three takeaways here.



One, we must examine the interfaces we design within the context in which they're used.



Two, we must understand that the task that users perform grows out of their interaction with our interfaces -- we don't define it.



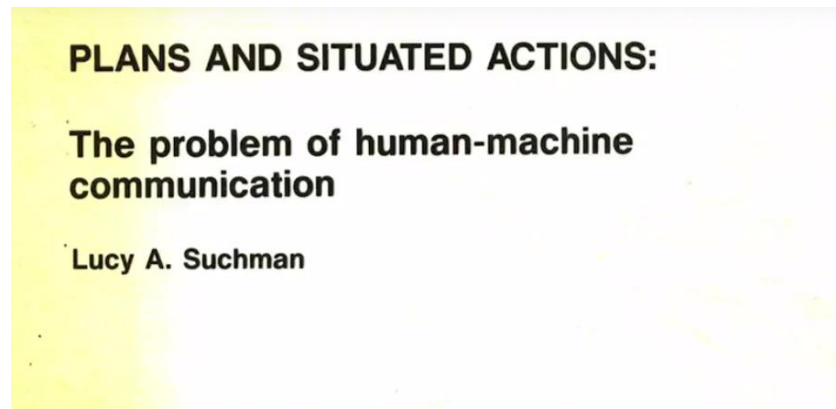
And three, we can try to structure it as much as we can, but until users get started, the task itself doesn't exist -- and once they get started, they play a significant role in defining the task.

Situated Action and Memory

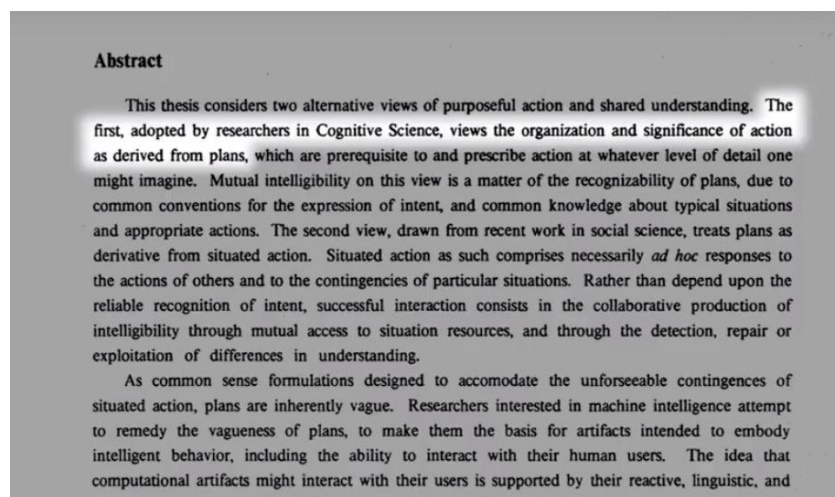


Situated action gives us a valuable lens to examine issues of memory. We mention in our lessons on memory and on design principles that recognition is easier than recall. People have an easier time recognizing the right answer, or option when they see it rather than recalling it from scratch. That's in part because memory is so context dependent. Recognition provides the necessary context to identify the right option. Relying on recall, means there's little context to cue the right answer to the users memory. Now I encountered an interesting example of the value of situated action a little while ago. My mother just had surgery. And so I would often go over to help her out with things. And every time I would go over, she'd have four, five favors to ask me. Inevitably I would forget a couple of those favors and have to be reminded, but she would always remember. Why was she so much better able to remember the favors than me? Does she just have a better memory? She didn't make a list. She didn't write them down or anything like that. So the distributed cognition perspective doesn't find an external memory being used or anything like that. My hypothesis from the perspective of situated action, is that she has the context behind the tasks. She knows why they need to be done. She knows what will happen if they aren't. For her, they're part of a broader narrative. For me, they're items on a list. I have no context for why they're there. Or what would happen if they're undone. For her, they're easy to remember because they're situated in a larger context. For me, they're difficult because they're isolated.

Paper Spotlight: "Plans and Situated Actions"



Lucy Suchman's 1985 book, *Plans and Situated Actions* is the seminal book on the philosophy of situated action. The book is a detailed comparison between two views of human action.



The first view, she writes, views the organization and significance of action as derived from plans. And this is a model we very often adopt when developing interfaces. Users make plans and users carry out those plans, but Suchman introduces a second view as well.

Abstract

This thesis considers two alternative views of purposeful action and shared understanding. The first, adopted by researchers in Cognitive Science, views the organization and significance of action as derived from plans, which are prerequisite to and prescribe action at whatever level of detail one might imagine. Mutual intelligibility on this view is a matter of the recognizability of plans, due to common conventions for the expression of intent, and common knowledge about typical situations and appropriate actions. The second view, drawn from recent work in social science, treats plans as derivative from situated action. Situated action as such comprises necessarily *ad hoc* responses to the actions of others and to the contingencies of particular situations. Rather than depend upon the reliable recognition of intent, successful interaction consists in the collaborative production of intelligibility through mutual access to situation resources, and through the detection, repair or exploitation of differences in understanding.

As common sense formulations designed to accommodate the unforeseeable contingencies of situated action, plans are inherently vague. Researchers interested in machine intelligence attempt to remedy the vagueness of plans, to make them the basis for artifacts intended to embody intelligent behavior, including the ability to interact with their human users. The idea that computational artifacts might interact with their users is supported by their reactive, linguistic, and

In this view, people simply act in the world, and plans are what we derive from those actions. Instead of plans dictating actions, plans are interpretations of actions. What this means for us as interface designers is that rather than assuming the user has a plan in mind that they're actively carrying out, we might consider viewing only their immediate interaction with the current screen instead. In other words, forget the history of actions that led the user to a certain screen and ask just once they're here how do they know what to do next?

7. Human-machine communication

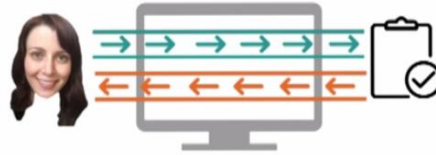
Interaction is always a *tentative* process, a process of continuously testing the conception one has of...the other. (Turner 1962, p. 23)

In Chapter 4, I outlined the view that the significance of actions, and their intelligibility, resides neither in what is strictly observable about behavior, nor in a prior mental state of the actor, but in an interactionally constructed relationship between observable behavior, circumstances and intent. Rather than enumerate an a priori system of shared rules for meaningful behavior, Chapter 5 described resources for constructing shared understanding, collaboratively and *in situ*. Face-to-face interaction was presented as the most powerful and highly developed system for accomplishing mutual intelligibility, exploiting a range of linguistic, observational and inferential resources.

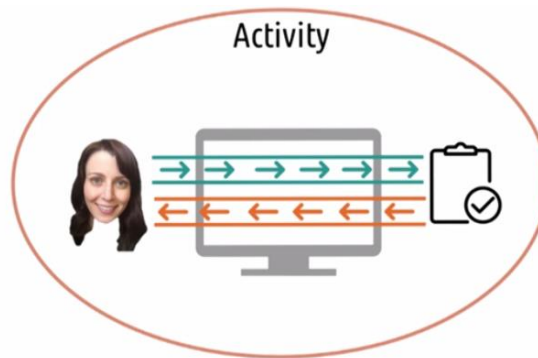
Given this view of the source of action's intelligibility, the situation of action can be defined as the full range of resources that the actor has available to convey the significance of his or her own actions, and to interpret the actions of others. Taking that preliminary definition of the situation as a point of departure, my interest in this chapter is to consider 'communication' between a person and a machine in terms of the nature of their respective situations. For purposes of the analysis, and without ascribing intent in any way, I will assume that the machine is behaving according to the resources of 'its' situation, the user according to the resources of hers. The aim of the analysis then is to view the organization of human-machine communication, including its troubles, in terms of

Later in the book, Lucy Suchman specifically touches on communication between humans and machines. There's a lot more depth here as well, the key take away for us is to focus on the resources available to the user at any given time. But I do recommend reading the book and this chapter for more insights.

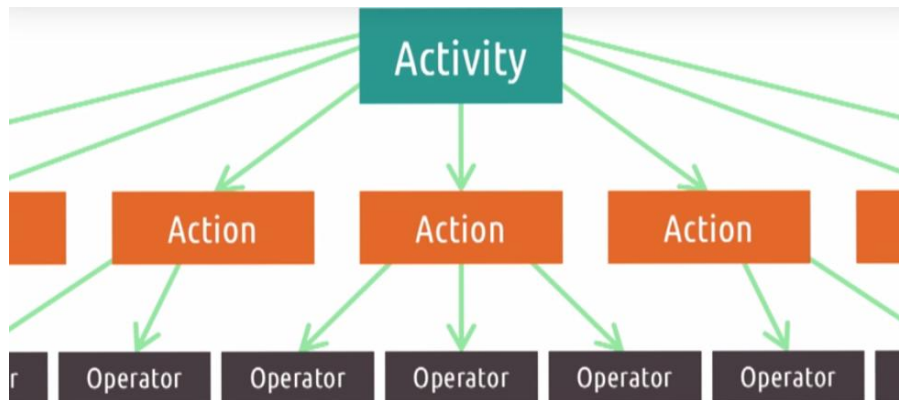
Activity Theory



Activity theory is a massive and well developed set of theories regarding interaction between various pieces of an activity. The theory as a whole is so complex that you could teach an entire class on it alone. It predates HCI. And in fact, activity theory is one of the first places the idea of interacting through an interface actually came from. In our conversations about HCI though, there are three main contributions of activity theory that I'd like you to come away with. First, when we discuss designing tasks and completing tasks through an interface, we risk missing a key component. Why? We could jump straight to designing the task, but why is the user completing the task in the first place? That can have significant implications for our design.



Activity theory generalizes our unit of analysis from the task to the activity. We're not just interested in what they're doing, but why they're doing it and what it means to them. Our designs will be different, for example, if users are using a system because they're required to or because they choose to. Notice how this is similar to our discussion of distributed cognition, as well. In distributed cognition, we were generalizing the unit of analysis from a person, to a system of people and artifacts. Here, we're generalizing the unit of analysis from a task to an activity surrounding a task. In both ways, we're zooming out on the task and the design space.



Second, activity theory puts an emphasis on the idea that we can create low level operations from higher level actions. We saw something similar to this with GOMS models, where methods were made up of operators. This has a special historical significance. Before activity theory and similar theories reached HCI in the 1980s, HCI was largely concerned with minute things, like how quickly a person can click a button or type in a command. Activity theory helped us zoom out from those low level interactions, those low level operators, to general user needs at the action or the activity levels. And third, activity theory points out that actions by the user can actually move up and down this hierarchy. A common example of this is driving a car. The first time you drove a car, shifting gears between park and drive was a very conscious action made up of operators like grabbing the gear shift and moving it in the right direction and letting go. You had to think about how to press the button, which way to push the stick, and when to release it. However, after driving a few times, shifting gears just becomes second nature. It becomes more like an operator. It shifted from being a conscious goal to an operator in your broader driving behavior. Notice a similarity here to our previous discussion on learning curves. How quickly an action moves from being a conscious action to a subconscious operator is also a function of how good the learning curve is on that design. Notice also, this is similar to the question of invisible interfaces. A good invisible interface helps users focus on their actions inside the task, rather than the operators they use to interact with the system.

Paper Spotlight: "Activity Theory and Human-Computer Interaction"

1

Activity Theory and Human-Computer Interaction

Bonnie A. Nardi

What is activity theory, and how will it benefit studies of human-computer interaction? This book addresses these questions. Many HCI researchers are eager to move beyond the confines of traditional cognitive science, but it is not clear exactly which direction to move in. This book explores one alternative for HCI research: activity theory, a research framework and set of perspectives originating in Soviet psychology in the 1920s. Just as HCI research is concerned with practical problems of design and evaluation, activity theorists from the outset have addressed practical needs, applying their research efforts to the problems of mentally and physically handicapped children, educational testing, ergonomics, and other areas. Following the lead of dialectical materialism, activity theory focuses on *practice*, which obviates the need to distinguish "applied" from "pure" science—understanding everyday practice in the real world is the very objective of scientific practice.

In 1996, Bonnie Nardi edited a prominent book on the study of context in human-computer interaction, titled *Context and Consciousness*. The entire book is worth reading, but two papers in particular stand out to me, both by Nardi herself. The first is a short paper that serves in some ways as an introduction to the book as a whole. It's not a long paper, only four pages, so I highly recommend reading it. It won't take you long. Here, Nardi outlines the general application of activity theory to HCI.

constructing consciousness (see Kaptein, chapter 5, Kuhl, this volume). Activity theorists argue that consciousness is not a set of discrete disembodied cognitive acts (decision making, classification, remembering), and certainly it is not the brain; rather, consciousness is located in everyday practice: you are what you do. And what you do is firmly and inextricably embedded in the social matrix of which every person is an organic part. This social matrix is composed of people and artifacts. Artifacts may be physical tools or sign systems such as human language. Understanding the interpenetration of the individual, other people, and artifacts in everyday activity is the challenge activity theory has set for itself.

Unlike anthropology, which is also preoccupied with everyday activity, activity theory is concerned with the development and function of individual consciousness. Activity theory was developed by psychologists, so this is not surprising, but it is a very different flavor of psychology from what the West has been accustomed to, as activity theory emphasizes naturalistic study, culture, and history.

The chapters in part I explain what activity theory is. They, along with the seminal article, "The Problem of Activity in Psychology" by the Russian psychologist Leont'ev (1974) (widely available in English in university libraries), form a primer of activity theory.

Activity theory offers a set of perspectives on human activity and a set of concepts for describing that activity. This, it seems to me, is exactly what HCI research needs as we struggle to understand and describe "context," "situation," "practice." We have recognized that technology use is not a mechanical input-output relation between a person and a machine; a much richer depiction of the user's situation is needed for design and evaluation. However, it is unclear how to formulate that depiction in a way that is not purely ad hoc. Here is where activity theory helps: by providing orienting concepts and perspectives. As Engeström (1993) has noted, activity theory does not offer "ready-made techniques and procedures" for research; rather, its conceptual tools must be "concretized according to the specific nature of the object under scrutiny."

As we expand our horizons to think not only about *usable* systems but now *useful* systems, it is imperative that we have ways of finding out what would be useful. How can we begin to understand the best ways to undertake major design projects, such as providing universal access to the Internet, effectively using computers in the classroom, supporting distributed work teams, and even promoting international understanding in ways both small (e.g., international video/e-mail pen-pals for schoolchildren) and large

She notes that activity theory offers a set of perspectives on human activity and a set of concepts for describing that activity. And this is exactly what HCI research needs as we struggle to understand and describe context, situation and practice. She particularly notes that the theory is uniquely suited to addressing some of the interesting issues facing HCI in 1996. And for that reason, it's also fascinating to view from a historical perspective. Today we understand the role that context has grown to play, especially with emerging technologies. It's fascinating to me to look back at how the community was constructing that debate 20 years ago.

Paper Spotlight: "Studying Context: A Comparison of Activity Theory, Situated Action Models, and Distributed Cognition"

4

Studying Context: A Comparison of Activity Theory, Situated Action Models, and Distributed Cognition

Bonnie A. Nardi

It has been recognized that system design will benefit from explicit study of the context in which users work. The unaided individual divorced from a social group and from supporting artifacts is no longer the model user. But with this realization about the importance of context come many difficult questions. What exactly is context? If the individual is no longer central, what is the correct unit of analysis? What are the relations between artifacts, individuals, and the social groups to which they belong? This chapter compares three approaches to the study of context: activity theory, situated action models, and distributed cognition. I consider the basic concepts each approach promulgates and evaluate the usefulness of each for the design of technology.¹

A broad range of work in psychology (Leont'ev 1978; Vygotsky 1978; Luria 1979; Scribner 1984; Newman, Griffin, and Cole 1989; Norman 1991; Salomon 1993), anthropology (Lave 1988; Suchman

In this lesson we covered three theories on studying context in human computer interaction. Distributed cognition, Situated action and Activity theory. If your having trouble keeping the three straight though, Nardi has a great paper for you. From her volume context in consciousness. Nardi wrote a comparison between the three philosophy is she titled, Studying Context: A Comparison of Activity Theory, Situated Action Models, and Distributed Cognition.

SITUATED ACTION MODELS

Situated action models emphasize the emergent, contingent nature of human activity, the way activity grows directly out of the particularities of a given situation.² The focus of study is situated activity or practice, as opposed to the study of the formal or cognitive properties of artifacts, or structured social relations, or enduring cultural knowledge and values. Situated action analysis do not deny that artifacts or social relations or knowledge or values are important, but they argue that the true locus of inquiry should be the "everyday activity of persons acting in [a] setting" (Lave 1988).³ That this inquiry is meant to take place at a very fine-grained level of minutely observed activities, inextricably embedded in a particular situation, is reflected in Suchman's (1987) statement that "the organization of situated action is an emergent property of moment-by-moment interactions between actors, and between actors and the environments of their action."

Lave (1988) identifies the basic unit of analysis for situated action as "the activity of persons-acting in setting." The unit of analysis is thus not the individual, not the environment, but a relation between the two. A *setting* is defined as "a relation between acting persons and the arenas in relation with which they act." An *arena* is a stable institutional framework. For example, a supermarket is an arena within which activity takes place. For the individual who shops in the supermarket, the supermarket is experienced as a setting because it is a "personally ordered, edited version" of the institution of the supermarket. In other words, each shopper shops only for certain items in certain aisles, depending on her needs and habits. She has thus "edited" the institution to match her personal preferences (Lave 1988).

An important aspect of the "activity of persons-acting in setting" as a unit of analysis is that it forces the analyst to pay attention to the flux of ongoing activity, to focus on the unfolding of real activity in a real setting. Situated action emphasizes responsiveness to the environment and the improvisatory nature of human activity (Lave 1988). By way of illustrating such improvisation, Lave's (1988) "cottage cheese" story has become something of a classic. A participant in the Weight Watchers program had the task of fixing a serving of cottage cheese that was to be three-quarters of the two-thirds cup of cottage cheese the program normally allotted.⁴ To find the correct amount of cottage cheese, the dieter, after nuzzling over the problem a bit, "filled a measuring cup two-thirds full of cheese, dumped it out on a

She starts by giving a great one page summary of each of these three views, which would be really good if you're having trouble understanding the finer points of these theories.

process (Flor and Hutchins 1991; Hutchins 1991a, 1991b; Nardi and Miller 1991). For example, Flor and Hutchins (1991) studied how two programmers performing a software maintenance task coordinated the task among themselves. Nardi and Miller (1991) studied the spreadsheet as a coordinating device facilitating the distribution and exchange of domain knowledge within an organization. In these analyses, shared goals and plans, and the particular characteristics of the artifacts in the system, are important determinants of the interactions and the quality of collaboration.

DIFFERENCES BETWEEN ACTIVITY THEORY, SITUATED ACTION MODELS, AND DISTRIBUTED COGNITION

All three frameworks for analyzing context that we have considered are valuable in underscoring the need to look at real activity in real situations and in squarely facing the conflux of multifaceted, shifting, intertwining processes that comprise human thought and behavior. The differences in the frameworks should also be considered as we try to find a set of concepts with which to confront the problem of context in HCI studies.

The Structuring of Activity

An important difference between activity theory and distributed cognition, on the one hand, and situated action, on the other hand, is the treatment of motive and goals. In activity theory, activity is shaped first and foremost by an object held by the subject; in fact, we are able to distinguish one activity from another only by virtue of their differing objects (Leont'ev 1974; Kozulin 1986; Kuutti 1991, this volume). Activity theory emphasizes motivation and purposefulness and is "optimistic concerning human self-determination" (Engeström 1990). A distributed cognition analysis begins with the positing of a *system goal*, which is similar to the activity theory notion of object, except that a system goal is an abstract systemic concept that does not involve individual consciousness.

Even more usefully, she goes on to give commentary on the difference between these three big theories.

Attention to the shaping force of goals in activity theory and distributed cognition, be they conscious human motives or systemic goals, contrasts with the contingent, responsive, improvisatory emphasis of situated action. In situated action, one activity cannot be distinguished from another by reference to an object (motive); in fact Lave (1988) argues that "goals [are not] a condition for action.... An analytic focus on direct experience in the lived-in world leads to ... the proposition that goals are constructed, often in verbal interpretation" (emphasis in original). In other words, goals are our musings out loud about why we did something *after* we have done it; goals are "retrospective and reflexive" (Lave 1988).

In a similar vein, Suchman (1987), following Garfinkel (1967), asserts that "a statement of intent generally says very little about the action that follows." If we appear to have plans to carry out our intent, it is because plans are "an artifact of our reasoning about action, not ... the generative mechanism of action." (emphasis in original). Suchman (1987) says that plans are "retrospective reconstructions."⁶ The position adopted by Lave (1988) and Suchman (1987) concerning goals and plans is that they are post hoc rationalizations for actions whose meaning can arise only within the immediacy of a given situation.

Lave (1988) asks the obvious question about this problematic view of intentionality: "If the meaning of activity is constructed in action ... from whence comes its intentional character, and indeed its meaningful basis?" Her answer, that "activity and its values are generated simultaneously," restates her position but does not explicate it. Winograd and Flores (1986) also subscribe to this radically situated view, using the colorful term "throwness" (after Heidegger) to argue that we are actively embedded, or "thrown into," in an ongoing situation that directs the flow of our actions much more than reflection or the use of durable mental representations.

In activity theory and distributed cognition, by contrast, an object-goal is the beginning point of analysis. An object precedes and motivates activity. As Leont'ev (1974) states, "Performing operations that do not realize any kind of goal-directed action [and recursively, a motive] on the subject's part is like the operation of a machine that has escaped human control."

First, she notes the activity theory and distributive cognition are driven by goals. Whereas situated action de-emphasizes goals for a focus on improvisation.

Attention to the shaping force of goals in activity theory and distributed cognition, be they conscious human motives or systemic goals, contrasts with the contingent, responsive, improvisatory emphasis of situated action. In situated action, one activity cannot be distinguished from another by reference to an object (motive); in fact Lave (1988) argues that "goals [are not] a condition for action.... An analytic focus on direct experience in the lived-in world leads to ... the proposition that goals are constructed, often in verbal interpretation" (emphasis in original). In other words, goals are our musings out loud about why we did something *after* we have done it; goals are "retrospective and reflexive" (Lave 1988).

In a similar vein, Suchman (1987), following Garfinkel (1967), asserts that "a statement of intent generally says very little about the action that follows." If we appear to have plans to carry out our intent, it is because plans are "an artifact of our reasoning about action, not ... the generative mechanism of action." (emphasis in original). Suchman (1987) says that plans are "retrospective reconstructions."⁶ The position adopted by Lave (1988) and Suchman (1987) concerning goals and plans is that they are post hoc rationalizations for actions whose meaning can arise only within the immediacy of a given situation.

Lave (1988) asks the obvious question about this problematic view of intentionality: "If the meaning of activity is constructed in action ... from whence comes its intentional character, and indeed its meaningful basis?" Her answer, that "activity and its values are generated simultaneously," restates her position but does not explicate it. Winograd and Flores (1986) also subscribe to this radically situated view, using the colorful term "throwness" (after Heidegger) to argue that we are actively embedded, or "thrown into," in an ongoing situation that directs the flow of our actions much more than reflection or the use of durable mental representations.

In activity theory and distributed cognition, by contrast, an object-goal is the beginning point of analysis. An object precedes and motivates activity. As Leont'ev (1974) states, "Performing operations that do not realize any kind of goal-directed action [and recursively, a motive] on the subject's part is like the operation of a machine that has escaped human control."

She goes on to summarize that situated actions says goals are constructed retroactively to interpret our past actions.

particular situation. In terms of identifying activity, activity theory provides the more satisfying option of taking a definition of an activity directly from a subjectively defined object rather than imposing a definition from the researcher's view.

These divergent notions of the structuring of activity, and the conceptual tools that identify one activity distinctly from another, are important for comparative work in studies of human-computer interaction. A framework that provides a clear way to demarcate one activity from another provides more comparative power than one that does not. Analyses that are entirely self-contained, in the way that a truly situated description of activity is, provide little scope for comparison. The level of analysis of situated action models—at the moment-by-moment level—would seem to be too low for comparative work. Brooks (1991) criticizes human-factors task analysis as being too low level in that all components in an analysis must "be specified as at atomic a level as possible." This leads to an ad hoc set of tasks relevant only to a particular domain and makes cross-task comparison difficult (Brooks 1991). A similar criticism applies to situated action models in which a focus on moment-by-moment actions leads to detailed descriptions of highly particularistic activities (such as pricing cheeses in a bin or measuring out cottage cheese) that are not likely to be replicated across contexts. Most crucially, no tools for pulling out a higher-level description from a set of observations are offered, as they are in activity theory.

Persistent Structures

An important question for the study of context is the role that persistent structures such as artifacts, institutions, and cultural values play in shaping activity. To what extent should we expend effort analyzing the durable structures that stretch across situations and activities that cannot be properly described as simply an aspect of a particular situation?

For both activity theory and distributed cognition, persistent structures are a central focus. Activity theory is concerned with the historical development of activity and the mediating role of artifacts. Leon'ev

41

Nardi also evaluates the role of permanent, persistent structures, noting their important for activity theory and distributed cognition. But present attention for situated action. So here we again see a similarity between activity theory and distributed cognition.

People and Things: Symmetrical or Asymmetrical?

Kapteinlin (chapter 5, this volume) points out that activity theory differs fundamentally from cognitive science in rejecting the idea that computers and people are equivalent. In cognitive science, a tight information processing loop with inputs and outputs on both sides models cognition. It is not important whether the agents in the model are humans or things produced by humans (such as computers). (See also Bødker, this volume, on the tool perspective.)

Activity theory, with its emphasis on the importance of motive and consciousness—which belong only to humans—sees artifacts and people as different. Artifacts are mediators of human thought and behavior; people and things are not equivalent. Bødker (this volume) defines artifacts as instruments in the service of activities. In activity theory, people and things are unambiguously asymmetrical.

Distributed cognition, by contrast, views people and things as conceptually equivalent; people and artifacts are "agents" in a system. This is similar to traditional cognitive science, except that the scope of the system has been widened to include a collaborating set of artifacts and people rather than the narrow "man-machine" dyad of cognitive science.

While treating each node in a system as an "agent" has a certain elegance, it leads to a problematic view of cognition. We find in distributed cognition the somewhat illogical notion that artifacts are cognizing entities. Flor and Hutchins (1991) speak of "the propagation of knowledge between different individuals and artifacts." But an artifact cannot know anything; it serves as a medium of knowledge for a human. A human may act on a piece of knowledge in unpredictable, self-initiated ways, according to socially or personally defined motives. A machine's use of information is always programmatic. Thus a theory that posits equivalence between human and machine damps out sources of systemic variation and contradiction (in the activity theory sense; see Kuutti, this volume) that may have important ramifications for a system. The activity theory notion of artifacts as mediators of cognition seems a more reasoned way to discuss relations between artifacts and people.

Activity theory instructs us to treat people as sentient, moral beings (Tikhomirov 1972), a stance not required in relation to a machine and often treated as optional with respect to people when they are viewed simply as nodes in a system. The activity theory position would seem to hold greater potential for leading to a more responsible technology design in which people are viewed as active beings in control of their tools for creative purposes rather than as automata whose operations are to be automated away, or

So what makes them different? Well, Nardi writes that the main difference between activity theory and distributed cognition is their evaluation of the symmetry between people and artifacts.

People and Things: Symmetrical or Asymmetrical?

Kapteinlin (chapter 5, this volume) points out that activity theory differs fundamentally from cognitive science in rejecting the idea that computers and people are equivalent. In cognitive science, a tight information processing loop with inputs and outputs on both sides models cognition. It is not important whether the agents in the model are humans or things produced by humans (such as computers). (See also Bødker, this volume, on the tool perspective.)

Activity theory, with its emphasis on the importance of motive and consciousness—which belong only to humans—sees artifacts and people as different. Artifacts are mediators of human thought and behavior; people and things are not equivalent. Bødker (this volume) defines artifacts as instruments in the service of activities. In activity theory, people and things are unambiguously asymmetrical.

Distributed cognition, by contrast, views people and things as conceptually equivalent; people and artifacts are "agents" in a system. This is similar to traditional cognitive science, except that the scope of the system has been widened to include a collaborating set of artifacts and people rather than the narrow "man-machine" dyad of cognitive science.

While treating each node in a system as an "agent" has a certain elegance, it leads to a problematic view of cognition. We find in distributed cognition the somewhat illogical notion that artifacts are cognizing entities. Flor and Hutchins (1991) speak of "the propagation of knowledge between different individuals and artifacts." But an artifact cannot know anything; it serves as a medium of knowledge for a human. A human may act on a piece of knowledge in unpredictable, self-initiated ways, according to socially or personally defined motives. A machine's use of information is always programmatic. Thus a theory that posits equivalence between human and machine damps out sources of systemic variation and contradiction (in the activity theory sense; see Kuutti, this volume) that may have important ramifications for a system. The activity theory notion of artifacts as mediators of cognition seems a more reasoned way to discuss relations between artifacts and people.

Activity theory instructs us to treat people as sentient, moral beings (Tikhomirov 1972), a stance not required in relation to a machine and often treated as optional with respect to people when they are viewed simply as nodes in a system. The activity theory position would seem to hold greater potential for leading to a more responsible technology design in which people are viewed as active beings in control of their tools for creative purposes rather than as automata whose operations are to be automated away, or

Activity theory regards in this fundamentally different, giving that humans have consciousness.

(c) 2017 by David Joyner, all rights reserved.

People and Things: Symmetrical or Asymmetrical?

Kapteinin (chapter 5, this volume) points out that activity theory differs fundamentally from cognitive science in rejecting the idea that computers and people are equivalent. In cognitive science, a tight information processing loop with inputs and outputs on both sides models cognition. It is not important whether the agents in the model are humans or things produced by humans (such as computers). (See also Bodker, this volume, on the tool perspective.)

Activity theory, with its emphasis on the importance of motive and consciousness—which belong only to humans—sees artifacts and people as different. Artifacts are mediators of human thought and behavior; people and things are not equivalent. Bodker (this volume) defines artifacts as instruments in the service of activities. In activity theory, people and things are unambiguously asymmetrical.

Distributed cognition, by contrast, views people and things as conceptually equivalent; people and artifacts are “agents” in a system. This is similar to traditional cognitive science, except that the scope of the system has been widened to include a collaborating set of artifacts and people rather than the narrow “man-machine” dyad of cognitive science.

While treating each node in a system as an “agent” has a certain elegance, it leads to a problematic view of cognition. We find in distributed cognition the somewhat illogical notion that artifacts are cognizing entities. Flor and Hutchins (1991) speak of “the propagation of knowledge between different individuals and artifacts.” But an artifact cannot know anything; it serves as a medium of knowledge for a human. A human may act on a piece of knowledge in unpredictable, self-initiated ways, according to socially or personally defined motives. A machine’s use of information is always programmatic. Thus a theory that posits equivalence between human and machine damps out sources of systemic variation and contradiction (in the activity theory sense; see Kuutti, this volume) that may have important ramifications for a system. The activity theory notion of artifacts as mediators of cognition seems a more reasoned way to discuss relations between artifacts and people.

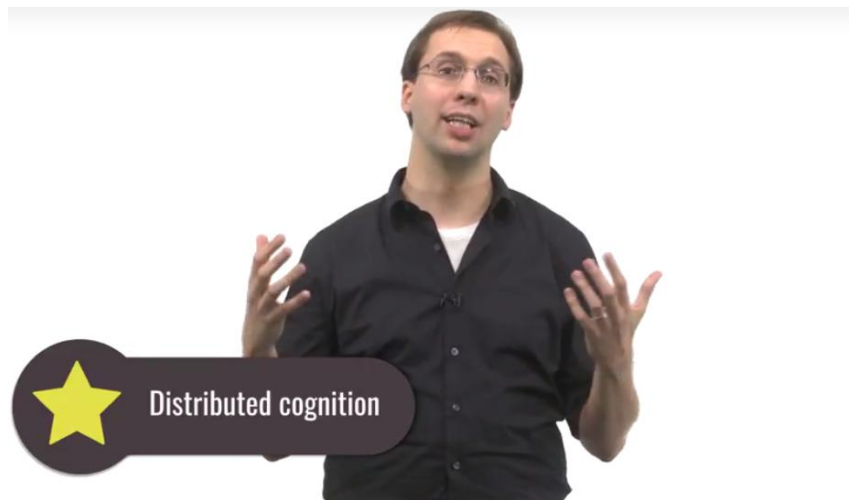
Activity theory instructs us to treat people as sentient, moral beings (Tikhomirov 1972), a stance not required in relation to a machine and often treated as optional with respect to people when they are viewed simply as nodes in a system. The activity theory position would seem to hold greater potential for leading to a more responsible technology design in which people are viewed as active beings in control of

Distributed cognition by contrast believes that artifacts can serve cognitive roles. And so those should be considered conceptually equivalent to humans. So that gives a high level overview of the difference between these three theories. These theories are big and complex of course and the complete paper goes into much more detail. But this should provide a decent glimpse at the distinctions, at least enough to get you started reading the paper for yourself.

Exploring HCI: Distributed Cognition

Distributed cognition is a perspective on analyzing systems that helps us emphasize the cognitive components of interfaces themselves. It helps us look at things we design as extensions of the user's own cognition. We can view anything from notes on a desktop to the entire Internet as an extension of the user's own memory. We can view things like Gmail's automatic email filtering as off-loading cognitive tasks from the user. In looking at things through this lens, we focus on the output not just of people or into interfaces, but on the combination of people and interfaces together. So, what are they able to do together that neither of them could do individually? As we close this lesson, think about this in terms of your chosen areas of HCI. What are the cognitive components of the areas with that you're dealing? How do augmented reality and wearable devices off-load some of the user's cognition onto the interface? And as occasional technology, or in HCI for health care, what are the tasks being accomplished by the system's comprised of users and interfaces?

Conclusion to Distributed Cognition



In this lesson, we've talked about distributed cognition and a couple related theories. The commonality of all these theories was their emphasis on context in integrated systems. Distributed cognition is interested in how cognition can be distributed among multiple individuals and artifacts, all working together. By taking a distributed view of interface design, we can think about what the combination of our users and our interfaces are able to accomplish.



Situating action and activity theory give additional perspectives of this, focusing respectively on the importance of ad hoc improvisation and the need to emphasize users' motives beyond just their goals. The common ground for all these theories is that our interfaces are not simply between user and their task, but they also exist in some kind of greater context.

2.9 Interfaces and Politics

Compiled by Shipra De, Summer 2017

Introduction to Interfaces and Politics



[MUSIC] In 1980, Langdon Winner published a highly influential essay in which he ask, do artifacts have politics? In other words, do technical devices have political qualities? And the answer is yes. All toasters are democrats.



Thermostats, as you might expect, are members of labor party.



And pretty surprisingly automobiles are actually green party members. Okay, I'm kidding. That's not what we mean when we ask if artifacts have politics. Here, when we say politics we mean whether artifacts can personify specific forms of authority or power, whether for good or bad. What we're referring to is the fact that artifacts are interfaces we design, change the world around us, just the way politicians or business interests do. Sometimes that's by design, we might design interfaces not for usability, or research, but to create change in the world. Other times that social change happens in ways we didn't anticipate, we design interfaces that are used and affect the world in ways we never anticipated.



So in this lesson, we're going to talk about two dimensions of this, designing for change and anticipating the change from our designs.



We'll also touch on a field that explores these issues more deeply called Value-Sensitive Design.

Change: A Third Motivation



Most commonly in HCI, we're interested in designing for usability. We want to make tasks easier through technology. So in a car, we might be interested in designing a GPS that can be used with the fewest number of taps. Or a dashboard that surfaces the most important information at the right time. Sometimes we're also interested in designing for research, though. We might design a dashboard that includes some kind of visualization of the speed to see if that changes the way the person perceives how fast that they're going. But a third motivation is to somehow change the user's behavior. Designing for change in response to some value that we have. Often times that may actually conflict with those other motivations. If we're trying to discourage an unhealthy habit, we might want to make the interface for that habit less usable. Cars actually have a lot of interfaces created with that motivation in mind. If I started driving without a seatbelt on, my car will beep at me. Some cars will cap your speed at a certain number. Those interfaces serve no usability goals but rather they serve the goal of user safety. Now, that's a simplistic example, but it shows what I call the three goals of HCI. Help a user do a task, understand how a user does a task, or change the way a user does a task due to some value that we hold, like safety or privacy.

Paper Spotlight: "Do Artifacts Have Politics?"

LANGDON WINNER

Do Artifacts Have Politics?

IN CONTROVERSIES ABOUT TECHNOLOGY AND SOCIETY, there is no idea more provocative than the notion that technical things have political qualities. At issue is the claim that the machines, structures, and systems of modern material culture can be accurately judged not only for their contributions of efficiency and productivity, not merely for their positive and negative environmental side effects, but also for the ways in which they can embody specific forms of power and authority. Since ideas of this kind have a persistent and troubling presence in discussions about the meaning of technology, they deserve explicit attention.¹

Writing in *Technology and Culture* almost two decades ago, Lewis Mumford gave classic statement to one version of the theme, arguing that "from late neolithic times in the Near East, right down to our own day, two technologies have recurrently existed side by side: one authoritarian, the other democratic, the

The most influential paper on the interplay between artifacts and politics, came from Langdon Winner in 1980. The paper describes numerous ways in which technologies, interfaces, and other artifacts, demonstrate political qualities, demonstrate political motivations.

lithic times in the Near East, right down to our own day, two technologies have recurrently existed side by side: one authoritarian, the other democratic, the first system-centered, immensely powerful, but inherently unstable, the other man-centered, relatively weak, but resourceful and durable."² This thesis stands at the heart of Mumford's studies of the city, architecture, and the history of technics, and mirrors concerns voiced earlier in the works of Peter Kropotkin, William Morris, and other nineteenth century critics of industrialism. More recently, antinuclear and prosolar energy movements in Europe and America have adopted a similar notion as a centerpiece in their arguments. Thus environmentalist Denis Hayes concludes, "The increased deployment of nuclear power facilities must lead society toward authoritarianism. Indeed, safe reliance upon nuclear power as the principal source of energy may be possible only in a totalitarian state." Echoing the views of many proponents of appropriate technology and the soft energy path, Hayes contends that "dispersed solar sources are more compatible than centralized technologies with social equity, freedom and cultural pluralism."³

An eagerness to interpret technical artifacts in political language is by no means the exclusive property of critics of large-scale high-technology systems. A long lineage of boosters have insisted that the "biggest and best" that science and industry made available were the best guarantees of democracy, freedom, and social justice. The factory system, automobile, telephone, radio, television, the space program, and of course nuclear power itself have all at one time or another been described as democratizing, liberating forces. David Lilienthal, in *T.V.A.: Democracy on the March*, for example, found this promise in the phos-

For example, he opens by noting the belief that nuclear power can only be used in a totalitarian society because of the inherent danger of the technology.

lithic times in the Near East, right down to our own day, two technologies have recurrently existed side by side: one authoritarian, the other democratic, the first system-centered, immensely powerful, but inherently unstable, the other man-centered, relatively weak, but resourceful and durable."² This thesis stands at the heart of Mumford's studies of the city, architecture, and the history of technics, and mirrors concerns voiced earlier in the works of Peter Kropotkin, William Morris, and other nineteenth century critics of industrialism. More recently, antinuclear and prosolar energy movements in Europe and America have adopted a similar notion as a centerpiece in their arguments. Thus environmentalist Denis Hayes concludes, "The increased deployment of nuclear power facilities must lead society toward authoritarianism. Indeed, safe reliance upon nuclear power as the principal source of energy may be possible only in a totalitarian state." Echoing the views of many proponents of appropriate technology and the soft energy path, Hayes contends that "dispersed solar sources are more compatible than centralized technologies with social equity, freedom and cultural pluralism."³

An eagerness to interpret technical artifacts in political language is by no means the exclusive property of critics of large-scale high-technology systems. A long lineage of boosters have insisted that the "biggest and best" that science and industry made available were the best guarantees of democracy, freedom, and social justice. The factory system, automobile, telephone, radio, television, the space program, and of course nuclear power itself have all at one time or another been described as democratizing, liberating forces. David Lilienthal, in *T.V.A.: Democracy on the March*, for example, found this promise in the phos-

Solar power on the other hand, pushes society towards a more distributed and a egalitarian structure. But of course, we understand that nuclear isn't on its own authoritarian. It has no consciousness, it can't take political power. Winner is proposing that the push for certain technologies carries with it certain necessary political adjustments. That's part of what it means to suggest that artifacts have politics. In the paper, Winner outlines two distinct ways in which artifacts can be political.

Inherently Political Technologies

None of the arguments and examples considered thus far address a stronger, more troubling claim often made in writings about technology and society—the belief that some technologies are by their very nature political in a specific way. According to this view, the adoption of a given technical system unavoidably brings with it conditions for human relationships that have a distinctive political cast—for example, centralized or decentralized, egalitarian or inegalitarian, repressive or liberating. This is ultimately what is at stake in assertions like those of Lewis Mumford that two traditions of technology, one authoritarian, the other democratic, exist side by side in Western history. In all the cases I cited above the technologies are relatively flexible in design and arrangement, and variable in their effects. Although one can recognize a particular result produced in a particular setting, one can also easily imagine how a roughly similar device or system might have been built or situated with very much different political consequences. The idea we must now examine and evaluate is that certain kinds of technology do not allow such flexibility, and that to choose them is to choose a particular form of political life.

A remarkably forceful statement of one version of this argument appears in Friedrich Engels's little essay "On Authority" written in 1872. Answering anarchists who believed that authority is an evil that ought to be abolished altogether, Engels launches into a panegyric for authoritarianism, maintaining, among other things, that strong authority is a necessary condition in modern industry. To advance his case in the strongest possible way, he asks his readers to imagine

One type is inherently political technologies. These are technologies that due to their very design, are only compatible with certain political structures. Certain technologies like nuclear power, whether due to complexity, or safety, or resources, require considerable top down organization. Those lend themselves to authoritarian power structures. Others like solar power, someone might argue, are only possible in a more distributed and egalitarian society. So these technologies, by their very nature, dictate the need for certain political structures.

prefer to speak of technologies, smaller or larger pieces of systems or hardware of a specific kind. My intention is not to settle any of the issues here once and for all, but to indicate their general dimensions and significance.

Technical Arrangements as Forms of Order

Anyone who has traveled the highways of America and has become used to the normal height of overpasses may well find something a little odd about some of the bridges over the parkways on Long Island, New York. Many of the overpasses are extraordinarily low, having as little as nine feet of clearance at the curb. Even those who happened to notice this structural peculiarity would not be inclined to attach any special meaning to it. In our accustomed way of looking at things like roads and bridges we see the details of form as innocuous, and seldom give them a second thought.

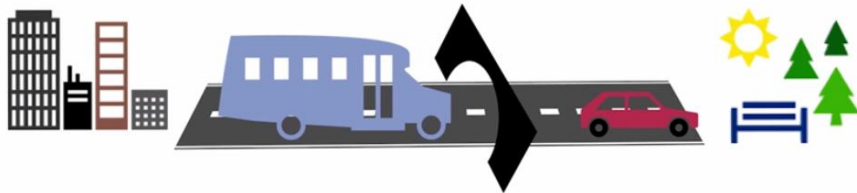
It turns out, however, that the two hundred or so low-hanging overpasses on Long Island were deliberately designed to achieve a particular social effect. Robert Moses, the master builder of roads, parks, bridges, and other public works from the 1920s to the 1970s in New York, had these overpasses built to specifications that would discourage the presence of buses on his parkways. According to evidence provided by Robert A. Caro in his biography of Moses, the reasons reflect Moses's social-class bias and racial prejudice. Automobile-

The other type he discusses are technical arrangements as forms of order. Technologies can be used to achieve changes to social order when used in the correct way. The technology itself has no inherent political leanings, like nuclear or solar power, but its use in a particular context, for a particular purpose, can nonetheless accomplish some political goals. Winner uses the example of a factory in Chicago in the 1880s. They replaced workers with automated machines that produced inferior goods as a way of busting up the Union. The new technology was actually inferior, but it was used to serve a political purpose. So according to Winner, artifacts may have two kinds of politics. They may be inherently political in that they're only compatible with certain forms of political order, or they may be used to achieve political motives even though they have no inherent politics on their own.

Negative Change by Design

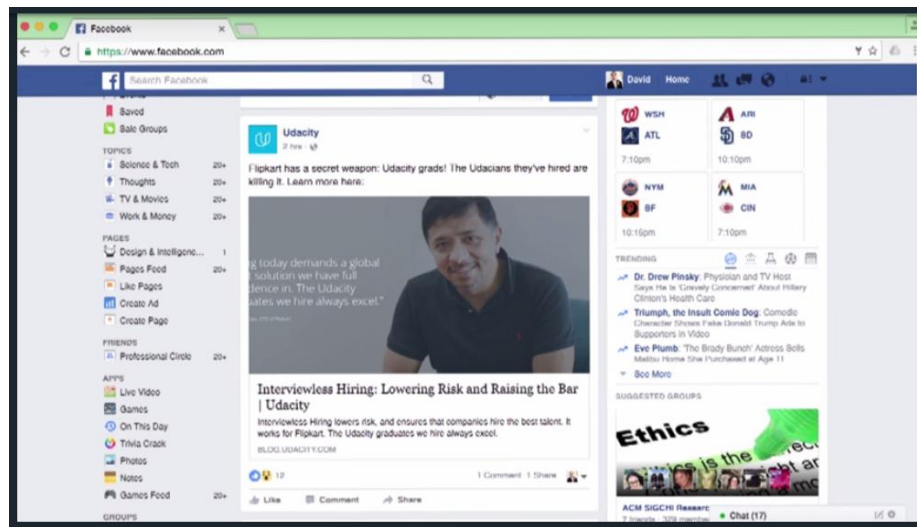


Let's start with the bad news. The ability of interfaces to change behavior can be abused. We're not just talking about places where people put explicit barriers up like blocking certain people from accessing their content. There are instances where people create seemingly normal designs with underlying political motivations. Winner describes one such instance in his essay *Do Artifacts Have Politics?* Robert Moses was an influential city planner working in New York City, in the early 1900s. As part of his role, he oversaw the construction of many beautiful parks on Long Island. He also oversaw the construction of parkways, roads to bring the people of New York to these parks. That's actually where the word parkway comes from. But something unfortunate happened.

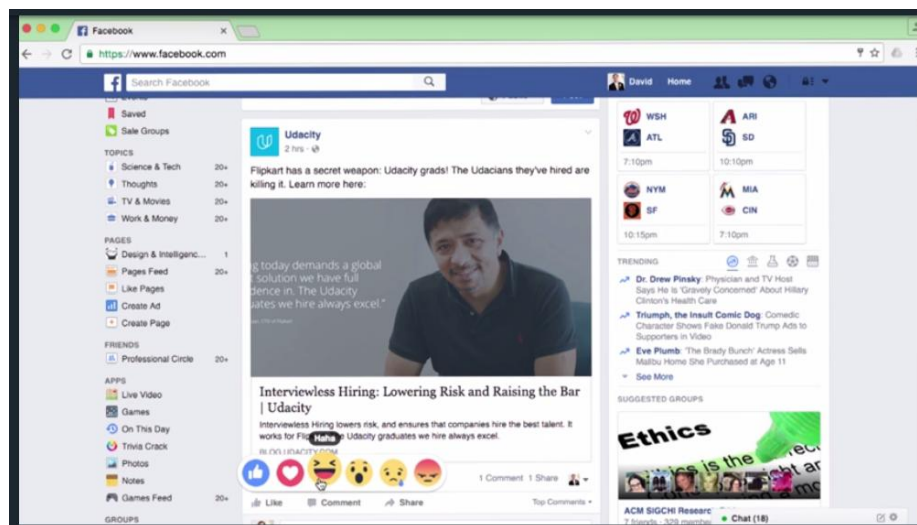


The bridges along these parkways were too low for buses to pass under them. As a result, public transportation couldn't really run easily to his parks. And as a result of that, only people wealthy enough to own cars were able to visit his parks. What an unfortunate coincidence, right? The evidence shows it's anything but coincidence. Moses intentionally constructed those bridges to be too low for buses to pass under. As a way of keeping poor people from visiting his parks. His political motivations directly informed the design of the infrastructure and the design of the infrastructure had profound social implications. This is an example of winners technology as a form of social order. The bridges could have been taller. There's nothing inherently political about those bridges. It was the way that they were used that accomplished this political motivation. As an interesting aside, I learned recently that the design of Central Park inside New York City was an example of the exact opposite dynamic. The designers were encouraged to put in places where only carriages could access so affluent people would have somewhere to go away from poor people. But the designers specifically made the entire park accessible to everyone. It's not too hard to imagine things kind of like that happening today either. One of the arguments from proponents of Net neutrality is that without it, companies can set up fast lanes that prioritize their own content or worse severely diminished content of their competitors or content critical of the company.

Positive Change by Design

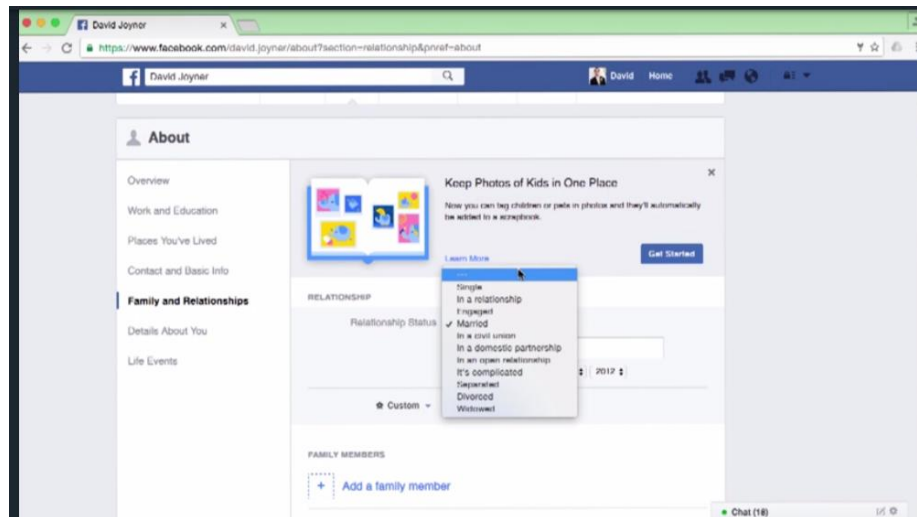


We can design for positive social changes well though. This goes beyond just encouraging people to be nice or banning bad behavior. Interfaces can be designed that'll lead to positive social change through natural interaction with the system. One example of this that I like is Facebook's ubiquitous Like button. For years, many people have argued for a Dislike button to compliment the Like button. Facebook has stuck with the Like button though, because by its design, it only supports positive interactions. It dodges cyberbullying, it dodges negativity. For usability purposes, it's a weakness because there are interactions I can't have naturally in this interface. But this specific part of the Like button wasn't designed with usability in mind.

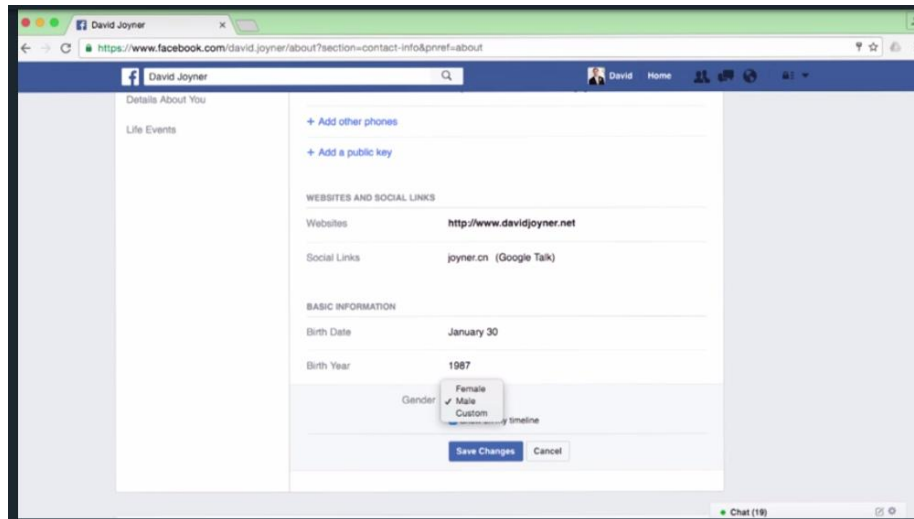


More recently, Facebook has added to the like button with five new emotions, love, haha, wow, sad and angry. Even with these five new emotions though, the overall connotation is still positive. For three of them, it's obvious why. Love, haha and wow are more positive emotions. Sad and angry are

negative emotions, but used in this context, they take on more of a sympathetic connotation. If someone is ranting about getting into a car accident, it seems weird to like that. But if you react with this angry emoticon, then you're basically saying you're angry on their behalf. It might be possible to use this for the more negative connotation like if someone said they like a political candidate and you react angrily, then you could be opposing their political view. But in the majority of cases, these are still going to be used in a way that fosters positive interaction. So it seems that this interface was designed to foster positive social interactions online. At the expense of usability, it would come with supporting all social interactions online.

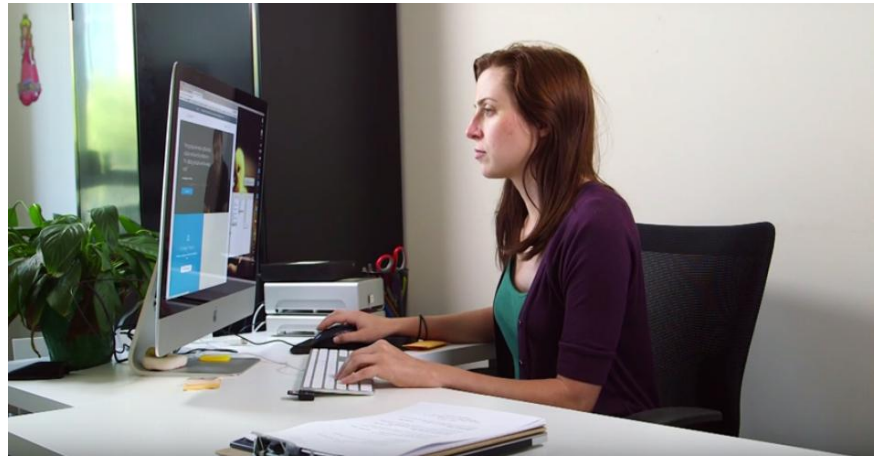


This also doesn't have to be strictly about dictating change, but it can also be about supporting change. For example, until a few years ago, Facebook had a more limited set of relationship options. They had married, engaged, in a relationship, single and it's complicated. As its target audience went from being college students to everyone, they also added separated, divorced and widowed. But it was a couple of years after that that they then added in a civil union and in a domestic partnership. Adding these concepts didn't magically create these social constructs, they existed legally before Facebook added them here. But adding them here supported an ongoing societal trend and gave them some validity. And made people for whom these were the accurate relationship labels feel like they really were part of Facebook's target audience, they were part of modern culture. That an accurate representation of their relationship status was available on this drop down meant they could accurately portray who they were on their Facebook profile.

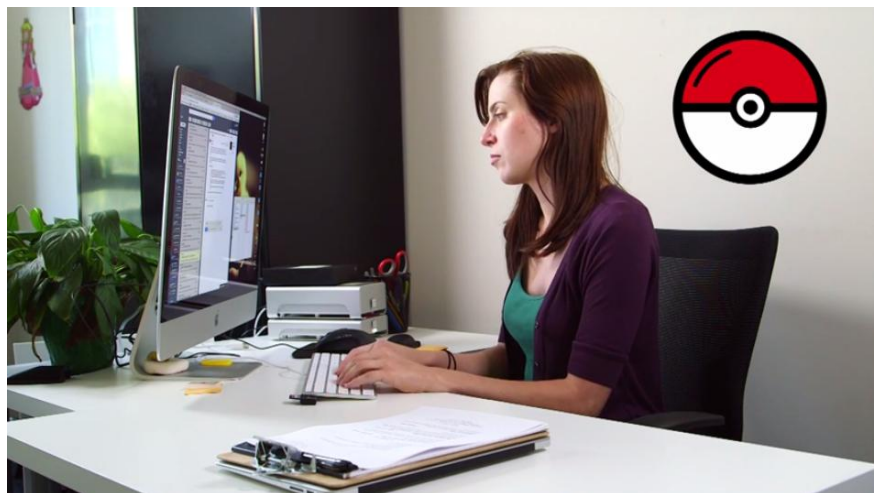


The same can be said for the more recent trend to expand Facebook's gender options to allow people to put in a custom gender. This supports a diverse group of people feeling as if the interface is designed with them in mind. Which in turn supports society's general movement towards acceptance.

Quiz: Design Challenge: Change by Design



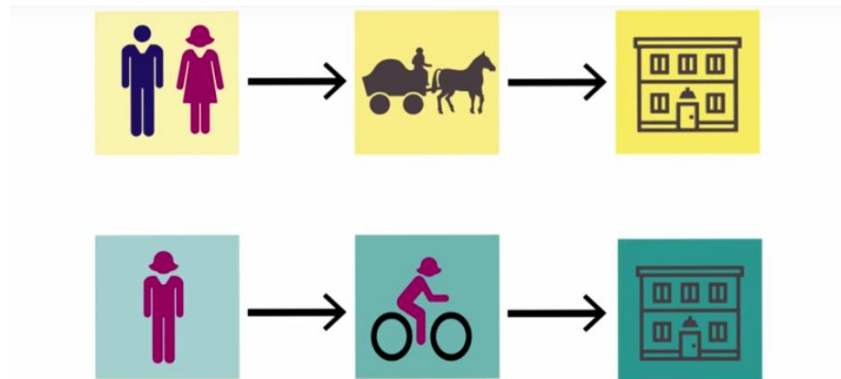
Let's tackle Change by Design by designing something for Morgan. So Morgan has a desk job. That means she spends a lot of her time sitting. However, for health reasons, it's best for her to get up once per hour and walk around just for a few minutes. There are a lot of ways we could tackle this by physically changing the design of her environment to a standing desk or by giving her an app that directly reminds her or rewards her for moving around. But let's try to do something a little bit more subtle. So let's design something for Morgan's smartphone that gets to move around for a couple minutes every hour without directly reminding her to walk around or rewarding her for doing so.



So here's one idea. Imagine a weather tracking app that crowdsourced weather monitoring. Every hour participants are buzzed to go outside and let their phone take some temperature readings, maybe take a picture of the sky. That design has nothing at all to do with moving around, but that's the side effect of it. Participation in this seemingly unrelated activity has the benefit of getting people moving. Pokemon GO is a great example of this in a different context. It doesn't spark the same kind of intermittent exercise but it gets people to exercise more generally, all without ever actually telling them to do so.

(c) 2017 by David Joyner, all rights reserved.

Positive Change by Happenstance



Positive change doesn't always have to happen by design though. In fact there are numerous examples of positive change happening more as a bi-product of technological advancement rather than as a goal of it. In Bijker's, of Bicycles, Bakelites and Bulbs, this is the bicycle example. The story looks at what women can do before and after the invention of the bicycle. Before the bicycle, women tend to be pretty reliant on men for transportation. People generally got around with carriages, which were pretty expensive, so only wealthy people would own them. And so, typically men would own them. So if a woman wanted to go to a restaurant or go to a show, she typically had to go either with her spouse or with her father. As a result, society rarely saw women acting individually. They were usually in the company of whoever the prominent male in their life was at the time. But then the bicycle came along. The bicycle was affordable enough and targeted at individuals, so now women could get around on their own. So now a woman could go to a show or go to a restaurant by herself, instead of relying on a man to take her. In the book though, what Bijker covers is not just the fact that this enabled more individual transportation, but rather that this enabled a profound social shift. This technological innovation allowed women to start acting independently. And it also demanded a wardrobe change, interestingly enough, because you couldn't wear a dress on a bicycle. So the invention of the bicycle simultaneously changed women's attire, and changed the level of independence they could show in modern society. And both these changes force society to challenge traditional gender roles. The bicycle's role in women's liberation was so significant that Susan B Anthony actually once said, I think bicycling has done more to emancipate women than anything else in the world. But when the bicycle was invented, it's doubtful that the inventor sat down and said surely this will be a great way to emancipate women and change our culture's gender roles. That's not what they had in mind. They were inventing a technological device. But as an unintended positive side effect, that technological device profoundly changed society.

Negative Change by Happenstance



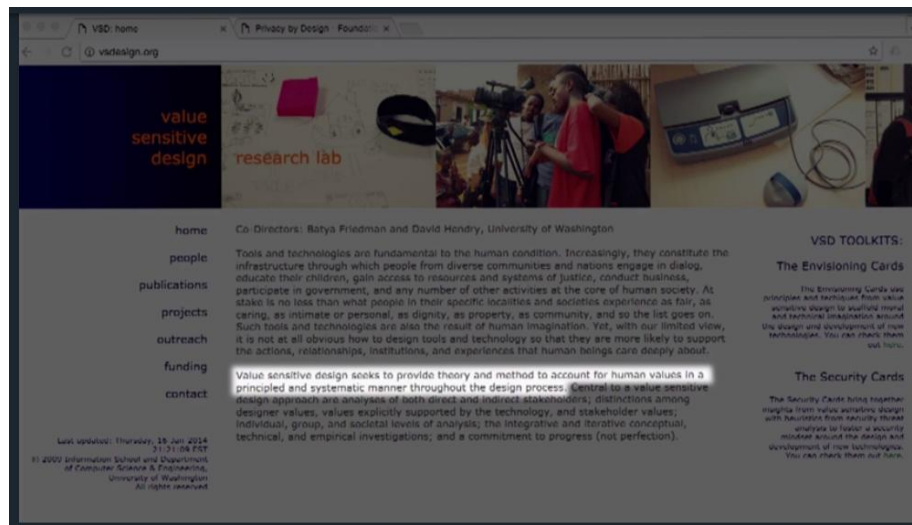
Just as we can create positive changes by accident, if we aren't careful, we can also inadvertently create negative changes as well, or further preserve existing negative dynamics. A good example of this is the proliferation of the Internet in the first place. When the Internet first came along, it piggybacked on existing phone lines. Then it started piggybacking on more expensive cable TV lines. And now it's following along with very expensive fiber optic lines. At every stage of the process, areas with more well developed infrastructure get the latest Internet speeds first. However, generally the areas with well developed infrastructure are the wealthier areas in the first place, either because wealthier citizens paid for the improved infrastructure. Or because people with the means to move wherever they want to move, will move somewhere with better infrastructure.



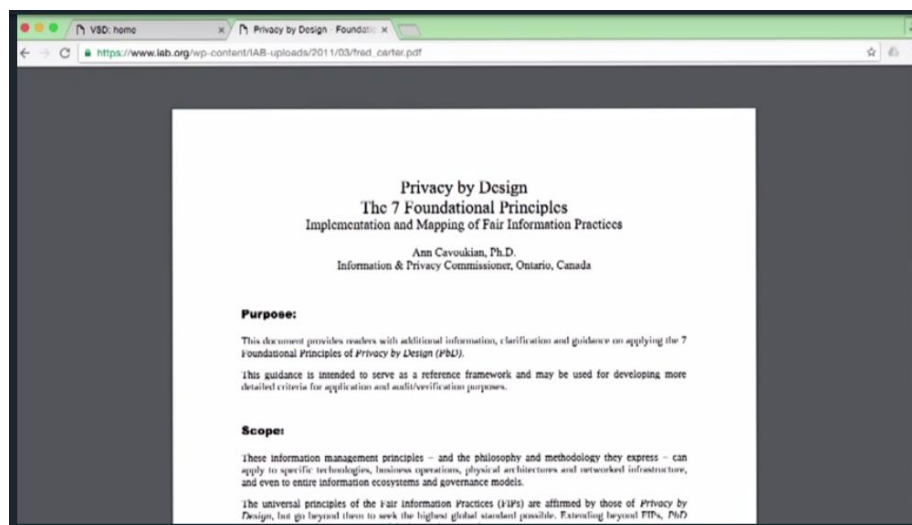
High speed internet access is a big economic boon. And yet areas that are already economically advantaged are generally the first ones to get higher speed internet access. Even today, in poorer parts of the United States the only available Internet connections are slow, unreliable satellite connections with strict data caps. And in the rest of the world this issue can be even more profound, where many

areas have no internet access whatsoever. And yet, this isn't intentional. Unlike the bridges on Long Island, no one is saying, let's withhold broadband access from poor people to keep them poor. Instead, it's natural to install better connections where there's already an existing infrastructure to build on. But that very natural plan has profoundly negative implications for equitable access to the Internet. So if we're not careful, completely innocent and completely logical design ideas can actually perpetuate negative effects in society.

Value-Sensitive Design



In HCI, we describe the idea of interfaces becoming invisible. Some of that is a usability principle, but it also applies more broadly to the way that interfaces integrate themselves into our everyday lives. And if our interfaces are going to integrate into people's lives, then they need to share the same values as those individuals as well. This connects to the field of value sensitive design. The Value Sensitive Design Lab at the University of Washington defines this idea by saying, value sensitive design seeks to provide theory and method to account for human values in a principled and systematic manner throughout the design process. In this way, value sensitive design is another dimension to consider when designing interfaces. Not only is an interface useful in accomplishing a task and not only is it usable by the user, but is it consistent with their values?



One of the most well-developed application areas of value sensitive design is privacy by design. Privacy is a value, and privacy by design has aimed to preserve that value in the design of systems. It's possible

to design useful usable interfaces that don't take privacy into account anywhere. That's what makes an examination of user's values an extra dimension of interface design.

Paper Spotlight: "Value Sensitive Design and Information Systems"

Value Sensitive Design and Information Systems

BATYA FRIEDMAN, PETER H. KAHN, JR., AND ALAN BORNING
University of Washington

Forthcoming in P. Zhang & D. Galletta (Eds.), *Human-Computer Interaction in Management Information Systems: Foundations*. M.E. Sharpe, Inc: NY.

Value Sensitive Design is a theoretically grounded approach to the design of technology that accounts for human values in a principled and comprehensive manner throughout the design process. It employs an integrative and iterative tripartite methodology, consisting of conceptual, empirical, and technical investigations. We explicate Value Sensitive Design by drawing on three case studies. The first study concerns information and control of web browser cookies, implicating the value of informed consent. The second study concerns using high-definition plasma displays in an office environment to provide a "window" to the outside world, implicating the values of physical and psychological well-being and privacy in public spaces. The third study concerns an integrated land use, transportation, and environmental simulation system to support public deliberation and debate on major land use and transportation decisions, implicating the values of fairness, accountability, and support for the democratic process, as well as a highly diverse range of values that might be held by different stakeholders, such as environmental sustainability, opportunities for business expansion, or walkable neighborhoods. We conclude with direct and practical suggestions for how to engage in Value Sensitive Design.

Batya Friedman is one of the co-directors of the Value Sensitive Design Research Lab at the University of Washington. And she co-authored one of the seminal papers on the topic, Value Sensitive Design and Information Systems. Friedman, Kahn, and Borning together provide this excellent paper on the philosophy. In it, they cover three investigations for approaching Value Sensitive Design.

3.1 Conceptual Investigations

Who are the direct and indirect stakeholders affected by the design at hand? How are both classes of stakeholders affected? What values are implicated? How should we engage in trade-offs among competing values in the design, implementation, and use of information systems (e.g., autonomy vs. security, or anonymity vs. trust)? Should moral values (e.g., a right to privacy) have greater weight than, or even trump, non-moral values (e.g., aesthetic preferences)? Value Sensitive Design takes up these questions under the rubric of conceptual investigations.

In addition, careful working conceptualizations of specific values clarify fundamental issues raised by the project at hand, and provide a basis for comparing results across research teams. For example, in their analysis of trust in online system design, Friedman, Kahn, and Howe [2000], drawing on Baier [1986], first offer a philosophically informed working conceptualization of trust. They propose that people trust when they are vulnerable to harm from others, yet believe those others would not harm them even though they could. In turn, trust depends on people's ability to make three types of assessments. One is about the harms they might incur. The second is about the good will others possess toward them that would keep those others from doing them harm. The third involves whether or not harms that do occur lie outside the parameters of the trust relationship. From such conceptualizations, Friedman et al. were able to define clearly what they meant by trust online. This definition is in some cases different from what other researchers have meant by the term – for example, the Computer Science and Telecommunications Board, in their thoughtful publication *Trust in Cyberspace* [Schneider 1999], adopted the terms "trust" and "trustworthy" to describe systems that perform as expected along the dimensions of correctness, security, reliability, safety, and survivability. Such a definition, which equates "trust" with expectations for machine performance, differs markedly from one that says trust is fundamentally a relationship between people (sometimes mediated by machines).

First they cover conceptual investigations. Conceptual investigations are like thought experiments where we explore the role values play in questions like, who are the direct and indirect stakeholders? And how are both classes of stakeholders affected?

3.2 Empirical Investigations

Conceptual investigations can only go so far. Depending on the questions at hand, many analyses will need to be informed by empirical investigations of the human context in which the technical artifact is situated. Empirical investigations are also often needed to evaluate the success of a particular design. Empirical investigations can be applied to any human activity that can be observed, measured, or documented. Thus, the entire range of quantitative and qualitative methods used in social science research is potentially applicable here, including observations, interviews, surveys, experimental manipulations, collection of relevant documents, and measurements of user behavior and human physiology.

Empirical investigations can focus, for example, on questions such as: How do stakeholders apprehend individual values in the interactive context? How do they prioritize competing values in design trade-offs? How do they prioritize individual values and usability considerations? Are there differences between espoused practice (what people say) compared with actual practice (what people do)? Moreover, because the development of new technologies affects groups as well as individuals, questions emerge of how organizations appropriate value considerations in the design process. For example, regarding value considerations, what are organizations' motivations, methods of training and dissemination, reward structures, and economic incentives?

3.3 Technical Investigations

As discussed in Section 2.3 (Value Sensitive Design's Constellation of Features), Value Sensitive Design adopts the position that technologies in general, and information and computer technologies in particular, provide value suitabilities that follow from properties of the technology. That is, a given technology is more suitable for certain activities and more readily supports certain values while rendering other activities and

Second, they cover empirical investigations. Empirical investigations go out and use real users, exploring how they make sense of interfaces and answering questions like, how do stakeholders apprehend individual values in the interactive context? And how do they prioritize individual values and usability considerations?

3.3 Technical Investigations

As discussed in Section 2.3 (Value Sensitive Design's Constellation of Features), Value Sensitive Design adopts the position that technologies in general, and information and computer technologies in particular, provide value suitabilities that follow from properties of the technology. That is, a given technology is more suitable for certain activities and more readily supports certain values while rendering other activities and values more difficult to realize.

In one form, technical investigations focus on how existing technological properties and underlying mechanisms support or hinder human values. For example, some video-based collaborative work systems provide blurred views of office settings, while other systems provide clear images that reveal detailed information about who is present and what they are doing. Thus the two designs differentially adjudicate the value trade-off between an individual's *privacy* and the group's *awareness* of individual members' presence and activities.

In the second form, technical investigations involve the proactive design of systems to support values identified in the conceptual investigation. For example, Fuchs [1999] developed a notification service for a collaborative work system in which the underlying technical mechanisms implement a value hierarchy whereby an individual's desire for privacy overrides other group members' desires for awareness.

At times, technical investigations – particularly of the first form – may seem similar to empirical investigations insofar as both involve technological and empirical activity. However, they differ markedly on their unit of analysis. Technical investigations focus on the technology itself. Empirical investigations focus on the individuals, groups, or larger social systems that configure, use, or are otherwise affected by the technology.

4. VALUE SENSITIVE DESIGN IN PRACTICE: THREE CASE STUDIES

Third, they cover technical investigations. Technical investigations are like empirical investigations that target the systems instead of the users. They ask the same kind of questions, but they are especially targeting whether or not the systems are compatible with the values of the users.

software (releasing the source code along with the executable), writing the code in as clear and understandable a fashion as possible, using a rigorous and extensive testing methodology, and complementing the Open Source software with an Open Process that makes the state of our development visible to anyone interested. For example, in our laboratory, a battery of tests is run whenever a new version of the software is committed to the source code repository. A traffic light (a real one) is activated by the testing regime – green means that the system has passed all tests, yellow means testing is under way, and red means that a test has failed. There is also a virtual traffic light, mirroring the physical one, visible on the web (www.urbansim.org/fireman). Similarly, the bug reports, feature requests, and plans are all on the UrbanSim project website as well. Details of this Open Process approach may be found in Freeman-Benson and Borning [2003].

Thus, in summary, Borning et al. are using Value Sensitive Design to investigate how a technology – an integrated land use, transportation, and environmental computer simulation – affects human values on both the individual and organizational levels; and how human values can continue to drive the technical investigations, including refining the simulation, data, and interaction model. Finally, employing Value Sensitive Design in a project of this scope serves to validate its use for complex, large-scale systems.

5. VALUE SENSITIVE DESIGN'S CONSTELLATION OF FEATURES

Value Sensitive Design shares and adopts many interests and techniques from related approaches to values and system design – computer ethics, social informatics, CSCW, and Participatory Design – as discussed in Section 2.2. However, Value Sensitive Design itself brings forward a unique constellation of eight features.

First, Value Sensitive Design seeks to be proactive: to influence the design of technology early in and throughout the design process.

The paper also proposes some of the fundamental features of value sensitive design. For example, value sensitive design should be proactive.

Second, Value Sensitive Design enlarges the arena in which values arise to include not only the work place (as traditionally in the field of CSCW), but also education, the home, commerce, online communities, and public life.

Third, Value Sensitive Design contributes a unique methodology that employs conceptual, empirical, and technical investigations, applied iteratively and integratively (see Section 3).

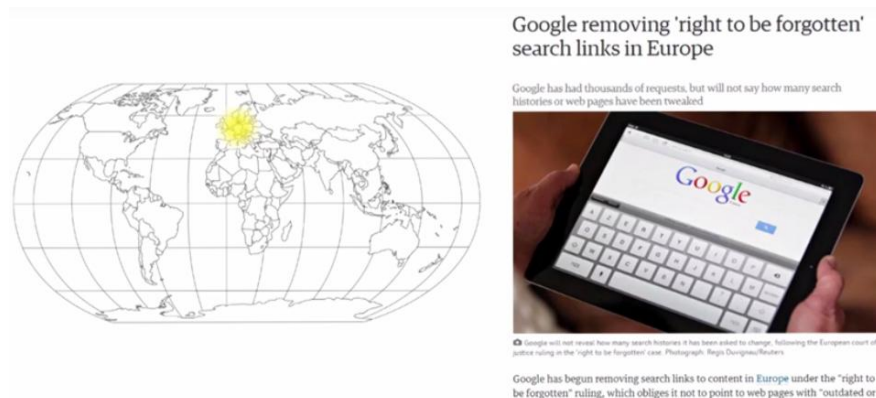
Fourth, Value Sensitive Design enlarges the scope of human values beyond those of cooperation (CSCW) and participation and democracy (Participatory Design) to include all values, especially those with moral import. By moral, we refer to issues that pertain to fairness, justice, human welfare and virtue, encompassing within moral philosophical theory deontology [Dworkin 1978; Gewirth 1978; Kant 1785/1964; Rawls 1971], consequentialism ([Smart and Williams 1973]; see Scheffler [1982] for an analysis), and virtue [Foot 1978; MacIntyre 1984; Campbell and Christopher 1996]. Value Sensitive Design also accounts for conventions (e.g., standardization of protocols) and personal values (e.g., color preferences within a graphical user interface).

Fifth, Value Sensitive Design distinguishes between usability and human values with ethical import. Usability refers to characteristics of a system that make it work in a functional sense, including that it is easy to use, easy to learn, consistent, and recovers easily from errors [Adler and Winograd 1992; Norman 1988; Nielsen 1993]. However, not all highly usable systems support ethical values. Nielsen [1993], for example, asks us to imagine a computer system that checks for fraudulent applications of people who are applying for unemployment benefits by asking applicants numerous personal questions, and then checking for inconsistencies in their responses. Nielsen's point is that even if the system receives high usability scores some people may not find the system socially acceptable, based on the moral value of privacy.

Sixth, Value Sensitive Design identifies and takes seriously two classes of stakeholders: direct and indirect. Direct stakeholders refer to parties – individuals or

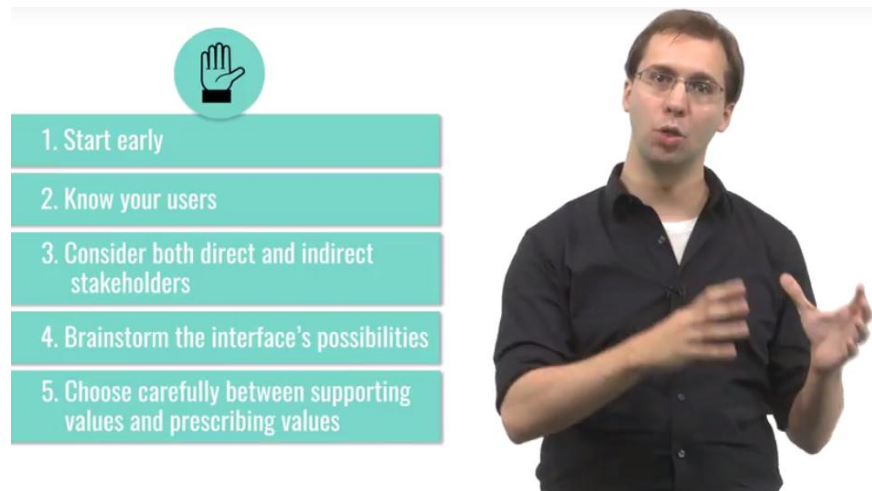
And value sensitive design distinguishes between usability and human values. If you're planning to work in an area where human values play a significant role, and I would argue that that's probably most areas of HCI, I highly recommend reading through this paper. It can have a profound impact, not only on the way you design interfaces, but on the way you approach user research.

Value-Sensitive Design Across Cultures



One of the challenges with value sensitive design is that values can differ across cultures. The internet makes it technologically possible to design single interfaces that are used by people in nearly every country, but just because it's technologically possible doesn't mean it's practically possible. And one reason for that is different countries and cultures may have vastly different values. A relatively recent news worthy example of this occurred with the rights to be forgotten. The right to be forgotten is a law in the European union, that allows individuals some control over what information is available about them online. That's a value held by the European Union. However, technologies like Google were not generally developed with that value in mind. So there's actually been an extraordinary effort to try to technologically support that right to be forgotten, while still providing search capabilities. Making this even more complicated is the fact that the value isn't universally shared. Many people argue that the law could actually effectively become internet censorship. So now we start to see some conflict in the values between different cultures. One cultures value of privacy, might run awry of another cultures value of free speech. If we're to design interfaces that can reach multiple cultures, we need to understand the values of those cultures. Especially if it might force us to design different systems for different people in order to match their local values.

5 Tips: Value-Sensitive Design



1. Start early
2. Know your users
3. Consider both direct and indirect stakeholders
4. Brainstorm the interface's possibilities
5. Choose carefully between supporting values and prescribing values

Here are five tips for incorporating value sensitive design into your interfaces. Number 1, start early. Identify the values you want to account for early in the design process, and check on the throughout the design process. The nature of value sensitive design is that it might have significant connections, not just to the design of the interface, but to the very core of the task you're trying to support. Number 2, know your users. I know I say this a lot but in order to design with values in mind, you need to know your users values. Certain values are incompatible with one another or at least present challenges for one another. Privacy, as a value, is in some ways in conflict with the value of record keeping. To know what to design, you need to know your users values. Number 3, consider both direct, and indirect stakeholders. We usually think about direct stakeholders. Those are the people that actually use the system that we create. Value sensitive design encourages us to think about indirect stakeholders as well. Those are people who do not use the system, but who are nonetheless affected by it. When you're designing the internal system for use by a bank for example it's used by bank employees but bank customers are likely to be impacted by the design. Number 4, brainstorm the interface's possibilities. Think not only about how you're designing the system to be used, but how it could be used. If you wanted to make a system that made it easier for employees to track their hours, for example, consider whether it could be used by employers to find unjust cause for termination. Number 5, choose carefully between supporting values and prescribing values. Designing for change is about prescribing changes in values, but that doesn't mean we should try to prescribe values for everyone. At the same time, there are certain values held in the world that we would like to change with our interfaces if possible with regard to issues like gender equality or economic justice. Be careful, and be deliberate about when you choose to support existing values, and when you choose to try to change them with your interfaces.

Exploring HCI: Interfaces and Politics

The idea of artifacts or interfaces having political clout, brings up two challenges for us as interface designers. First, we need to think about places where we can use interface design to invoke positive social change. And second, we also need to think about the possible negative ramifications of our interfaces. What undesirable stereotypes are we preserving or what new negative dynamics might we create? Now obviously, I work in online education and I have been struck by both sides of this. On the positive side, I've been amazed by the power of online education to diminish significance of superficial obstacles to people's success. I've spoken with people who have had difficulty succeeding in traditional college settings due to social anxiety disorders or other disabilities. Things that had no real connection to how well they understood the material, but they made it difficult to interact with other people or to attend the physical classes. But by putting everything in forums and emails and texts and videos, they've been able to overcome those obstacles, but there's also the risk that online education will only benefit people who already have advantages. The early data suggests that the majority of consumers of online education are middle class, American white males. There's little data to suggest that it's reaching minorities, reaching women, reaching international students or reaching economically disadvantaged students. And while I believe that's a problem that can be solved, it's certainly something we need to address. Otherwise, we risk online education being a luxury more than an equalizer. So, that's how these principles relate to online education. Take a moment and reflect on how they apply to the area of HCI that you chose to explore. What role can your technology play in creating positive societal change and what risks are there if your technology catches on?

Reversing the Relationship



We've talked a good bit about how technology and interfaces can affect politics and culture and society, but we wouldn't be telling the whole story if we didn't close by noting the alternate relationship as well. Political relationships and motivations can often have an enormous impact on the design of technology. From Bijker's book *Of Bicycles, Bakelites, and Bulbs*, the bulbs part refers to the battle of the design of the first fluorescent lightbulb in 1938. General Electric created a new kind of light that was far more energy efficient. The power companies were afraid that this would reduce power consumption and cut into their profits. After a long drawn out battle involving the Anti Trust Division of the US government and the US Department of War, the fluorescent bulbs that were ultimately sold were not as good as they technologically could be in order to preserve others' business interests. That issue is more prevalent today than ever. More and more, we see compatibility between devices and usage policies for technologies determined not by what's technologically possible but by what satisfies political or business needs. So here's an example. To keep up with everything that I like to watch on TV I have five different subscriptions. I have cable TV, I have Hulu, I have Amazon Prime, I have Netflix and I have an HBO subscription on top of my cable subscription. And that's not to mention things that I watch for free on their own apps like Conan or anything on YouTube. And you might think wouldn't it be awesome to just have one experience that could navigate among everything I want to watch. And it would be awesome, and there's no technological reason against it. But there's a complicated web of ownership and licensing and intellectual property agreements that determine the way that technology works. Technology changes society but society changes technology too.

Quiz: Reflections: Interfaces and Politics



You have almost certainly experienced political or business motivations changing the way in which a technology of yours works. Similar to the fluorescent light bulb, often times these motivations are to preserve the power or profit of an influential organization in the face of radical change. Sometimes they might be the products of a relationship or an agreement between vendors or organizations to emphasize one another's content. Generally, these are instances where technology either performs sub-optimally or has certain features because someone besides the user benefits. So reflect for a second and see if you can think of an instance where some technology you use was designed with this kind of political motivation in mind.

This question can have some pretty loaded answers and I encourage you to give those answers. But I'm going to give a slightly more innocuous one, exclusivity agreements in video games. Imagine I'm a video game developer and Amanda is Nintendo. And I'll say hey, Nintendo I'll agree to release my game only on your console, if you agree to promote my game in your console advertisements. I benefit from free advertising, Nintendo benefits from getting a selling point for its console. There's probably no technological reason my game can't run on other consoles. But there's this business relationship determining the way that the technology works.

Conclusion to Interfaces & Politics



In this lesson, we've discussed the different ways in which interfaces interact with existing power structures or political motivations. We looked at how interfaces can have negative repercussions, either by design or by happenstance.



We looked more optimistically at how interfaces can be powerful tools for equality and justice in the world, whether intentionally or accidentally.



We also looked at how it's important to keep in mind different cultures values while designing interfaces. Now notice how all these perspectives hearken back to the idea that user experience exists not only in individuals and groups, but in societies.

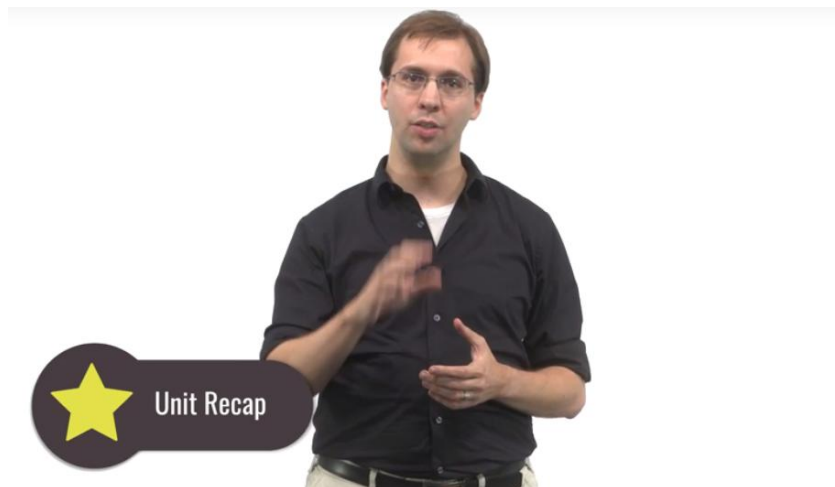
2.10 Conclusions to Principles

Compiled by Shipra De, Summer 2017

Introduction to Conclusions to Principles



[MUSIC] In this unit we've talked about the various different design principles that have been uncovered after years of research and work in HCI. And while they are presented in many ways as individual sets of guidelines and principles, there is a lot of overlap among them.

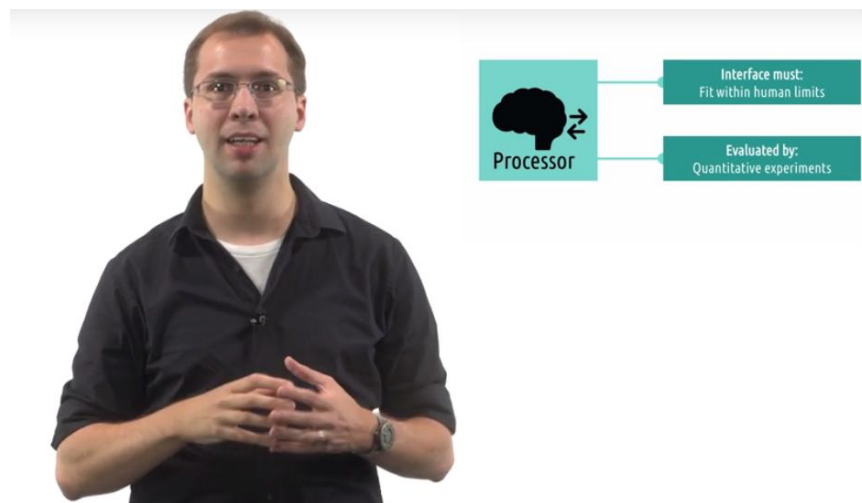


So in this recap of the unit, we'll try to tie all these seemingly different threads together. We'll also ask you to reflect on how you might apply some of these concepts as a whole to certain design problems.

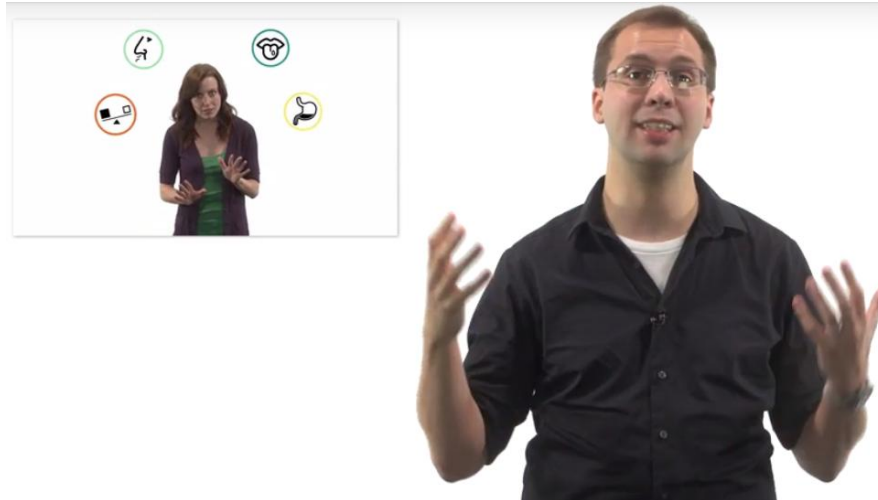
Zooming Out: Human as Processor



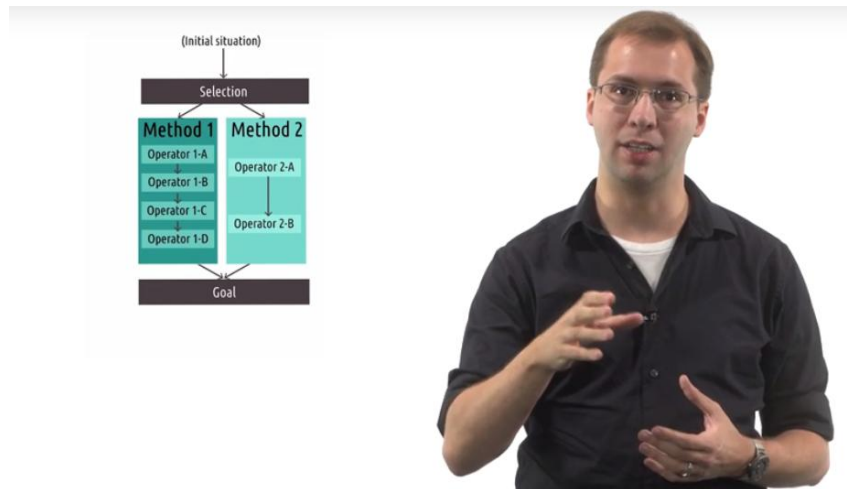
One way of knitting together the different ideas of HCI is to start very close and zoom out. At the narrowest level, we might view HCI as the interaction between a person and an interface.



This is the processor model of the role to human knit system. This too looks almost like an interaction between two computers, one just happens to be a human.



But humans' actions are approached almost computationally. If you're going to take this model, we need to understand a lot about what the human can sense, remember, and physically do.

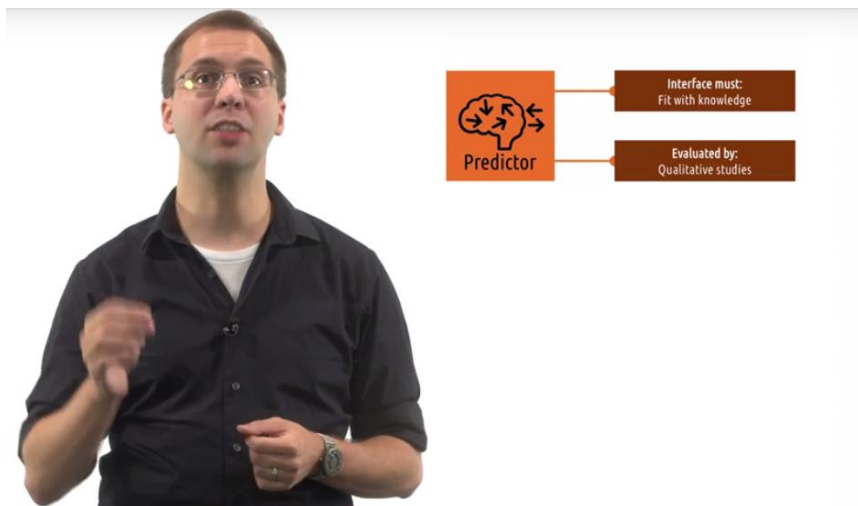


The GOMS model approaches HCI in this manner as well. It distills the human's role into goals, operators, methods and selection of rules, all of which can be externalized. But this is a pretty narrow view of HCI.

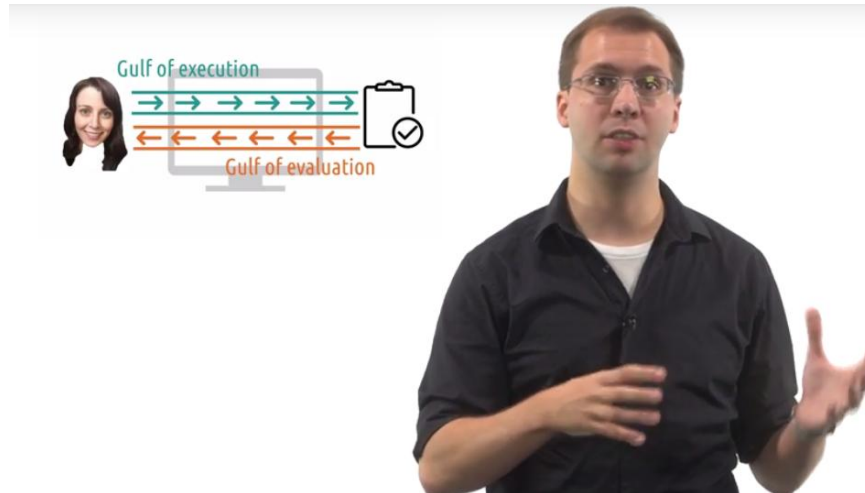
Zooming Out: Human as Predictor



For the most part, we're interested in something more task-focused. In fact, this is where we'll likely spend the majority of our time. This is the user interacting through some interface to accomplish some task.



That's what we mean by the predictor model of the user. The user is actively involved in looking at the task, making predictions about what to do, and making predictions about what will happen.



This is where we looked at the gulf of execution and the gulf of evaluation. How hard is it for the user to interact with a task? And how hard is it for them to get feedback on their interaction?



Here we also look at how the interface can ideally disappear from this interaction, making the user feel like they're working directly with the task, not interacting with an interface.



Now, many of our design principles are constructed specifically to help with this.

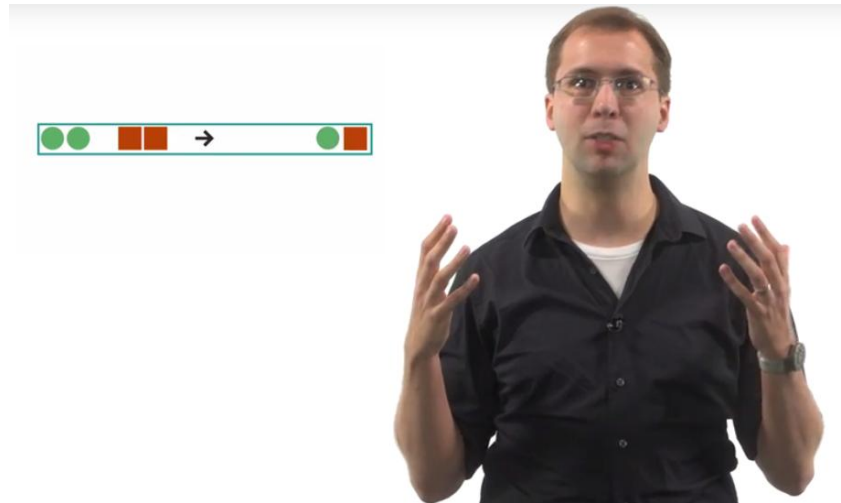
	Don Norman	Jakob Nielsen	Constantine & Lockwood	Universal Design
Discoverability	✓	✓	✓	✓
Simplicity	✓	✓	✓	✓
Affordances	✓	✓	✓	✓
Mapping	✓	✓	✓	✓
Perceptibility	✓	✓	✓	✓
Consistency	✓	✓	✓	✓
Flexibility	✓	✓	✓	✓
Equity	✓	✓	✓	✓
Ease	✓	✓	✓	✓
Comfort	✓	✓	✓	✓
Structure	✓	✓	✓	✓
Constraints	✓	✓	✓	✓
Tolerance	✓	✓	✓	✓
Feedback	✓	✓	✓	✓
Documentation	✓	✓	✓	✓



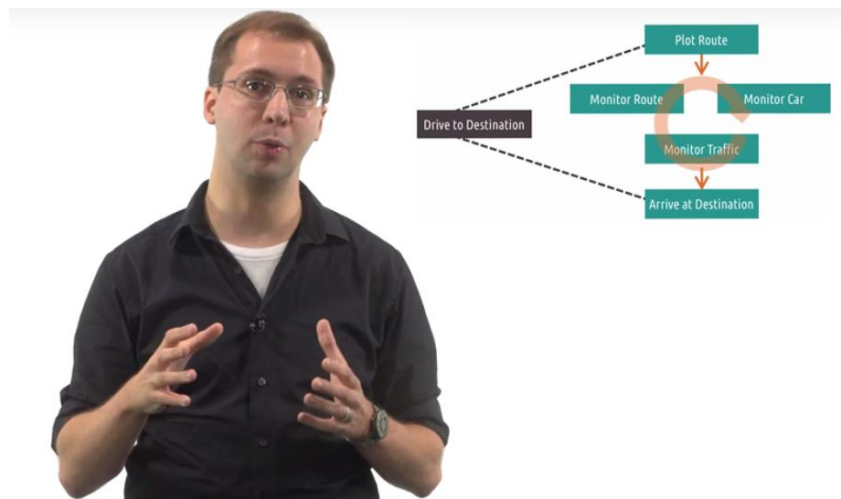
To help users more quickly make sense of the interface, and understand the underlying task.



But in order to design this interaction effectively, we have to understand the way the user thinks about the task they're performing.

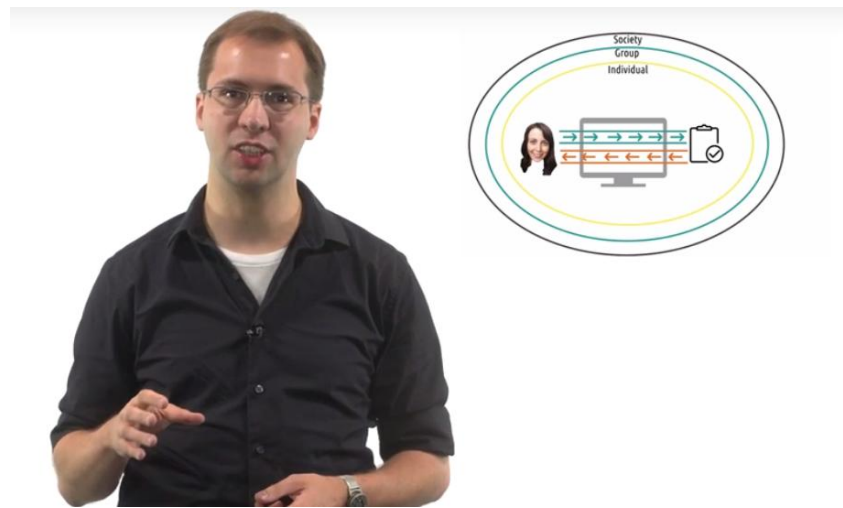


We have to understand their mental models and in turn, we have to help make sure their mental models match the actual task. Here we have to get into questions like understanding the user's errors and understanding the mapping between representations and the understanding tasks. We also have to address questions like expert blind spot and learned helplessness.



Now, fortunately, we have a tool to help us with this. Cognitive task analysis and its related hierarchical task analysis. So much of what we deal with in HCI occurs within this process of a human completing a task through some interface.

Zooming Out: Human as Participant



However, that's not all we're interested in. We're also interested in how this interaction occurs beyond just the individual and the interface and the task.



That's what was meant by the participant model of the user. The user is not merely interacting with an interface or interacting with a task through an interface. They're interacting with other interfaces, other individuals and society as a whole. They are active participants in the world around them.



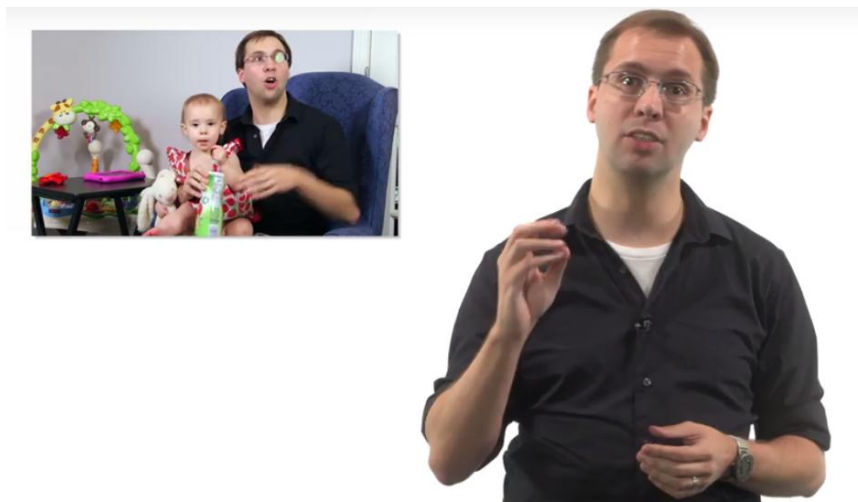
For example, sometimes we're interested not only in the tasks of the users performing, but also in their motivations and reasons for performing it.



That's what activity theory advocates. Treating the unit of analysis not as a task, but as an activity including some elements of the context surrounding the task.



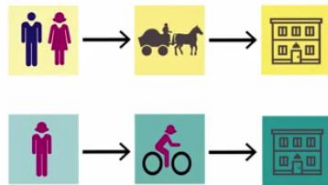
Other times we're interested in how artifacts and minds combine to help accomplish the task. That's what distributing cognition advocates.



Or other times we are interested in deeply understanding the situated context in which a person is acting, that's where situated action comes in.



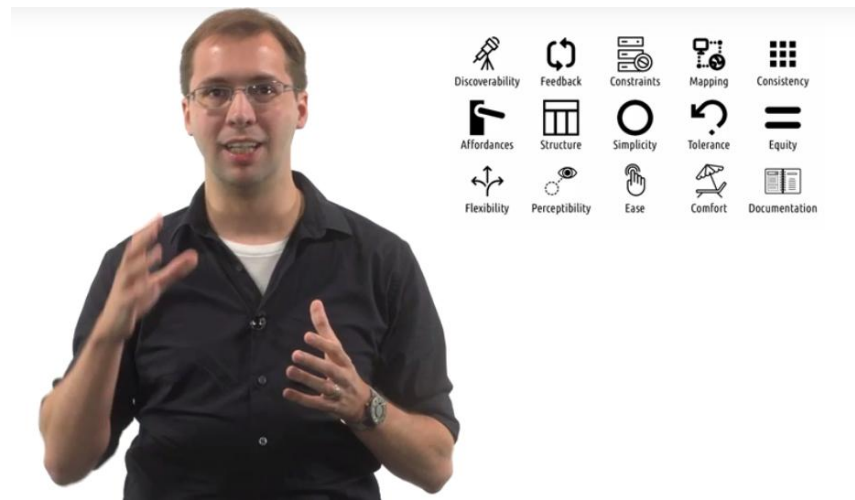
And other times, we're interested in how this all integrates with existing social norms and social relationships. That's what social cognition tries to examine.



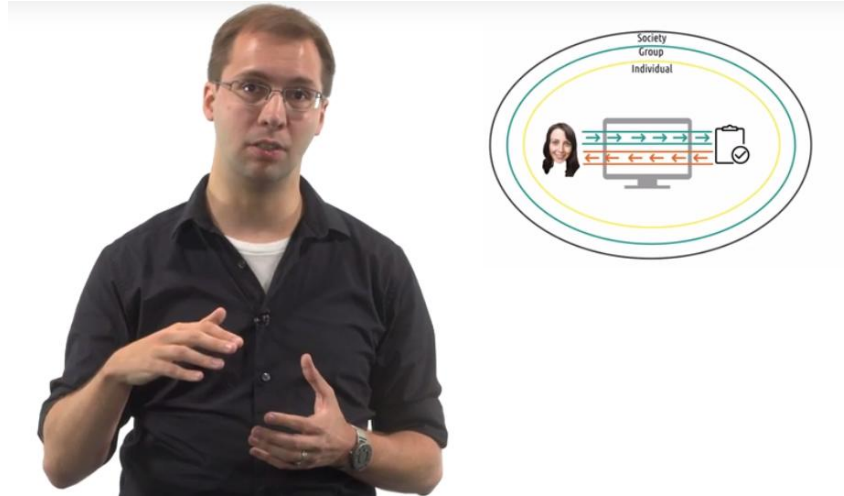
Other times we're interested in dynamics even broader than this. Sometimes, we're interested in how the interfaces we design can create positive social change.



Or sometimes we're interested in how the interfaces we design might risk perpetuating existing negative relationships in society.



That's exactly the goal of some of our design guidelines as well. To use interfaces to create a more equal society for all people.



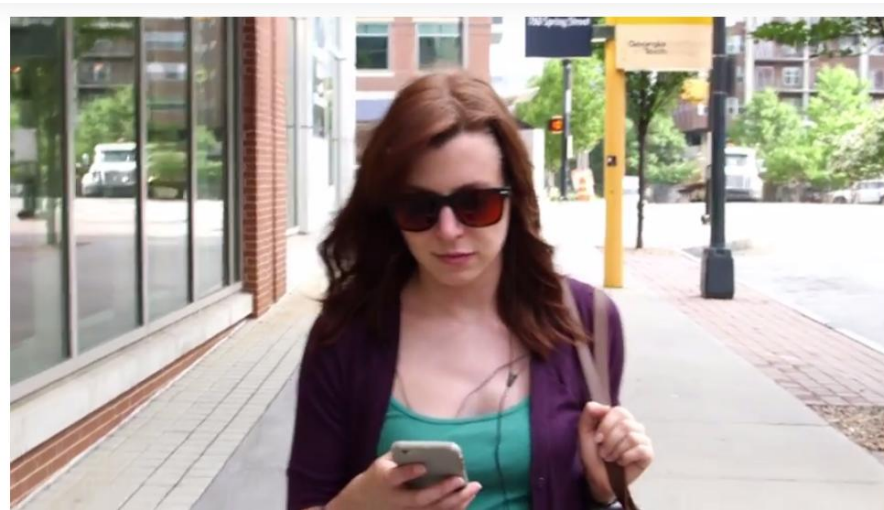
So this diagram describes from a very low level to a very high level, what we're interested in throughout this portion of HCI.

Quiz: Reflections: HCI Principles

When we started that conversations I commented that when you do HCI right, users won't actually know you've done anything at all. Good HCI disappears between the user and the tasks that they're completing. As a result, people can underestimate how complex good HCI can be to leverage. In this unit, one of my goals has been to help pull back the curtain on all the theories that go on behind the scenes of the designs of some of the interfaces that you use everyday. So as we close this unit, take a second to reflect on the interfaces you use that have disappeared between you and the task. Focus especially on interfaces that were never visible in the first place, not interfaces that became invisible by learning. Which of the principles that we've discussed do you now see at play in the design of some of those interfaces?

For me, I'm actually using an example of one of these interfaces right now. I have a teleprompter in front of me. I didn't always use a teleprompter, but as soon as I tried one, I was hooked. And part of that is because of how well designed the interface is. The very first time I used it, it made things immediately easier, instead of introducing a new learning curve. First, it uses very simple interactions, quick presses that accomplish anything I could need during the actual recording process. Second, it builds on a good mental model of the task that I'm performing. It understands, that while recording, the only things I need to do regularly are pause, play, scroll back, and scroll forward. There are a lot of other options that it has but it keeps those out of the way during the actual recording process, because they're not necessary at the time. Personally though, I think the teleprompter is great to analyze from the prospective of distributed cognition. I struggle when recording with talking too fast. Without the teleprompter, I have to remind myself to slow down, while also remembering what to say. That's a lot to keep in memory at the same time. The teleprompter lowers the cognitive load involved in remembering what to say, but it also controls my speed because I can't read what hasn't yet appeared. So the teleprompter takes care of two cognitive processes. Remembering what I have to say, and monitoring the speed at which I am presenting. So the system comprised of the teleprompter and me is better at recording than I am alone.

Quiz: Design Challenge: Designing Audiobook



Let's go back to our original design challenge for Morgan from the very beginning of this unit. We talked about how Morgan wants to be able to listen to audio books on the go which includes things like leaving bookmarks and taking notes. Using everything we've talked about so far, revisit this problem. Start by thinking narrowly about the physical interactions between Morgan and the interface and then zoom out to her interactions with the task as a whole, then zoom out even further to how the interaction between Morgan and the interface or relates to other things going on in the world around here. And last, think about how interfaces like this have the potential to effect society itself.

There are a lot of designs that you could propose, but the question here isn't what you design, the question is, how will you develop it? How will you evaluate it? How do you know which ideas are good and which are bad? Now we've given some heuristics and principles for doing this, but that doesn't automatically get you to a good interface. That just kind of establishes a baseline. That's just what the principles portion of this course covers. To fully develop interfaces using these principles, we need the methods of HCI as well.

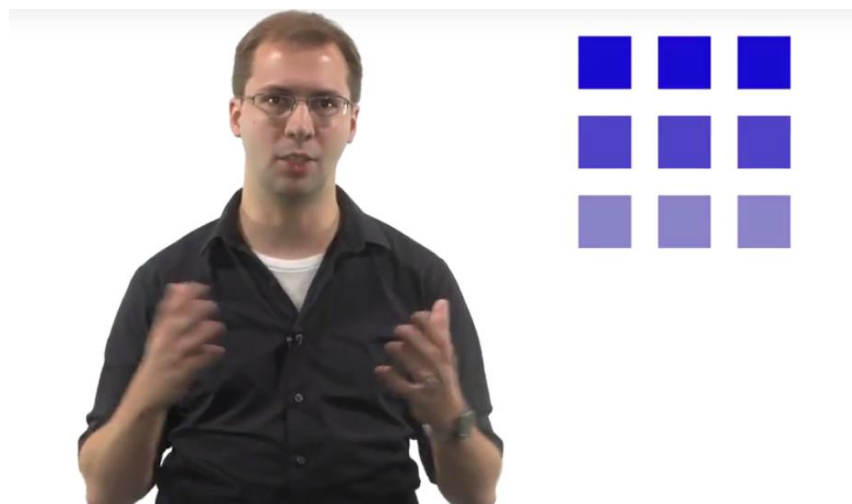
Explore HCI: HCI Principles

Throughout this unit, I have repeatedly asked you to revisit the area of HCI you chose to keep in mind throughout our conversations. Now, take a second and try to pull all those things together. You thought about how you chose an area applies to each of the models of the human's role. How it applies to the various different design guidelines and how it interacts with society and culture as whole. How does moving through those different levels change the kinds of designs you have in mind? Are you building it from low level interactions to high level effects? Are you starting at the top with the desired outcome and working your way down to the individual operations? There are no right or wrong answers here. The important thing is reflecting on your own reasoning process.

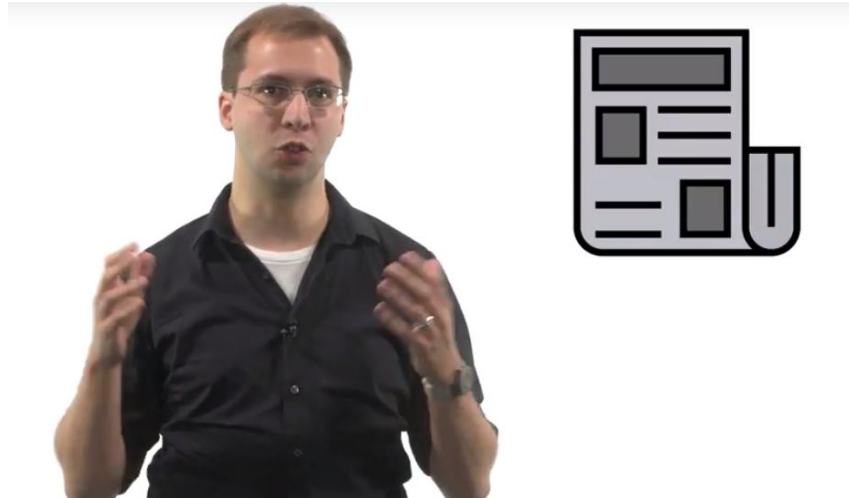
5 Tips: On-Screen UI Design



I mentioned in one of our first conversations that this course is distinct from user interface design course. We're interested in general principles and general methods for designing interactions between humans and various different kinds of computational devices. Designing for traditional screens has its own set of specific principles. However it's quite likely that you'll be designing for this kind of traditional device. Here are five quick tips for designing effective on screen user interfaces. Number one, use a grid.



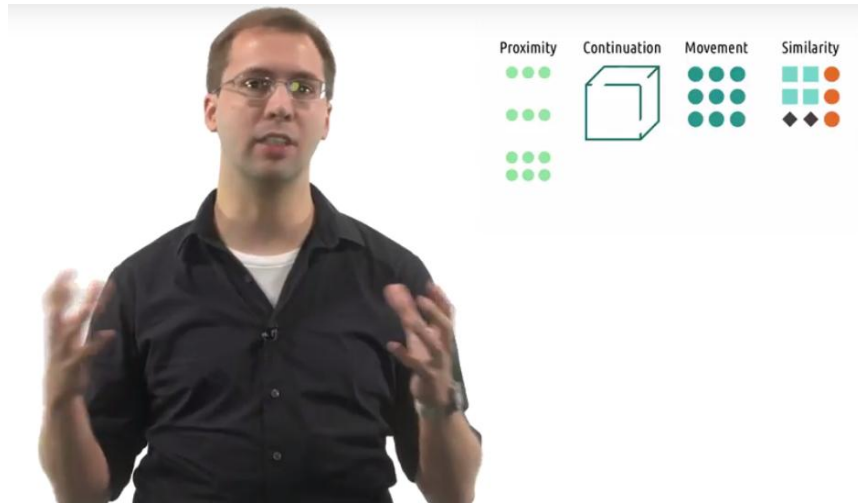
Grids are powerful ways of guiding user site around your interface, highlighting important content, grouping together content and so on.



There's a reason why newspapers and magazines have used grid based layouts for decades.



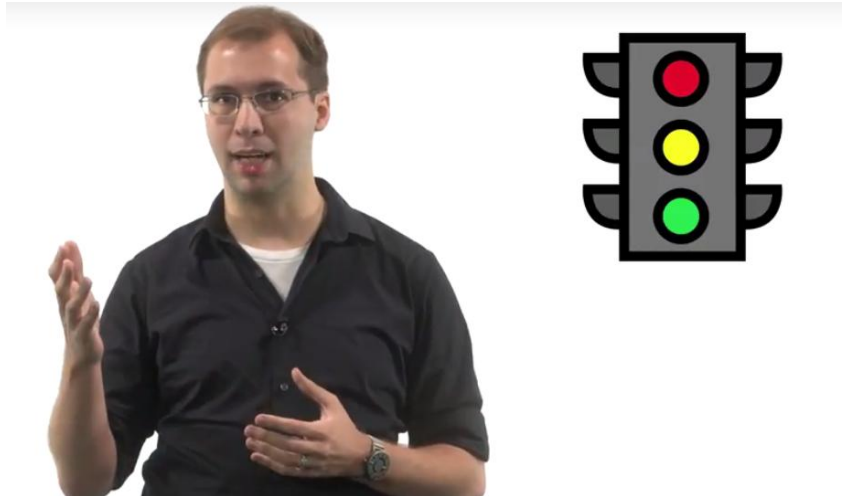
Number two, use whitespace. Users are very good at consuming small chunks of information at a time. Notice how news articles often use very short paragraphs and highway signs have lots of whitespace around the text. Whitespace works with grids to provide context and guide the users visual perception of the interface. Number three, know your Gestalt principles. Gestalt principles in UI designer refer to how users perceive groups of objects.



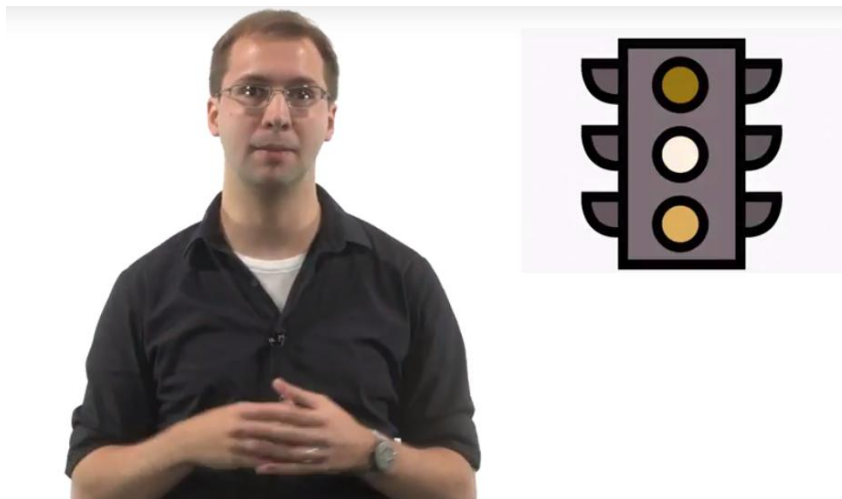
Users group the objects together when they're spatially close together, when they're visually similar, when they're moving together. So use that.



Number four, reduce clutter. The eye has difficulty processing cluttered information, so reduce clutter wherever possible. Grids, whitespacing, Gestalt principles can help with this, because they invisibly communicate content that might otherwise need to be communicated visually. Instead of drawing a box to group controls together, you can surround them with whitespace. Instead of using headers and text to label different portions of some content, you can separate them with a grid, and so on. Number five, design in grayscale. Color can be a powerful tool but it also runs array a universal design. There are enough color blind individuals in the world, that rely on that color too much, it's really problematic. Color can emphasize a content and structure of your interface but it shouldn't be necessary to understand it.



Take a stop light, for example. Red means stop and green means go.

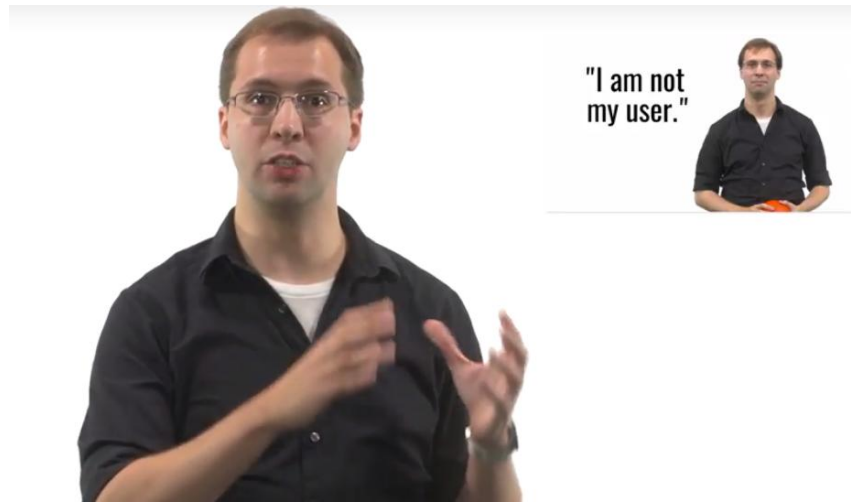


Which is a problem if you are deuteranopic or red, green color blind. But the red light is always at the top and the green light is always at the bottom. Even if you are deuteranopic, you can still understand what the light is saying. Color emphasizes the content. But the content doesn't rely on the color. If you're going to be doing a lot screen based UI design, I do highly recommend taking a class on the topic. It will cover these principles in a lot more depth, give stronger rules for designing good grids and using whitespace, and cover one of my favorite topics. Typography.



Amanda can testify that more thought went into selecting the fonts to use in this course than nearly any other element of its design. If you're curious, it's Ubuntu Condensed for the diagrams and Oswald for the loops, well, except that one.

Only Half of the Picture



There's one final thing you must understand about the guidelines and heuristics and principles we've talked about. They're only half the picture. They're necessary for good design, but they aren't sufficient. You can't just grab these guidelines off the shelf, throw them at a new task, and expect to have a perfect interface the first time. These principles give you a solid foundation. But every domain, every task, every audience has its own unique requirements and criteria. To design usable interfaces, you have to understand your specific user. And remember, that isn't you. What that means is that you have to go out and find what your users need. You have to get inside their head and understand the task. You have to prepare multiple different ideas for them to try out. You have to evaluate those ideas with real users. And you have to take that experience and use it to improve your design and start the process all over again. These guidelines and principles are useful in making that process as efficient as possible. But they aren't sufficient on their own. These principles are only half the picture, the other half are the methods.

3.1 Introduction to Methods

Compiled by Shipra De, Summer 2017

Introduction to Research Methods



[MUSIC] When you start a design new interface, your goal is to design a interface that will meet the needs of the users better than the existing design. It's very rare that we design for tasks that users have never even perform before or almost always developing new ways to accomplish old tasks. Facebook, for example, was a new way to socialize and communicate with others, but the fundamental activities of socializing and communicating weren't new. Facebook just met those needs better. Or at least met certain needs better depending on who you ask. In order to design interactions that are better than existing designs, it's important to take into consideration the users' needs at every stage of the design process. That's what this unit of this course will cover.



We don't generally want to build creative novel interfaces just for the sake of creativity or novelty. We want novelty to have a purpose and to achieve that purpose it must be used with a strong understanding of the users task.



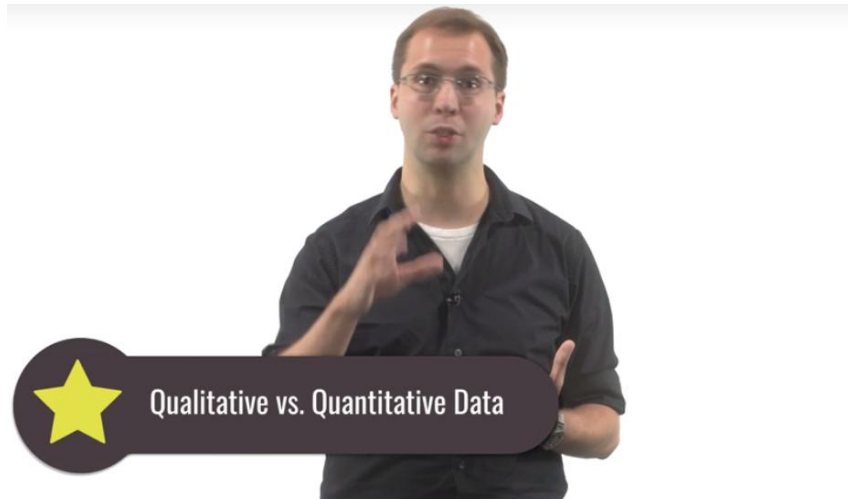
So in this unit we'll cover the Design Life Cycle as well as methods for gathering feedback and information from users at every stage of the cycle. However, before we get started on that, we need to set up some basic concepts we used throughout our discussions.



We'll start at discussing User-centered design.



And then we'll introduce the four stages design life cycle. We'll discuss a few general methods for pursuing the design life cycle.

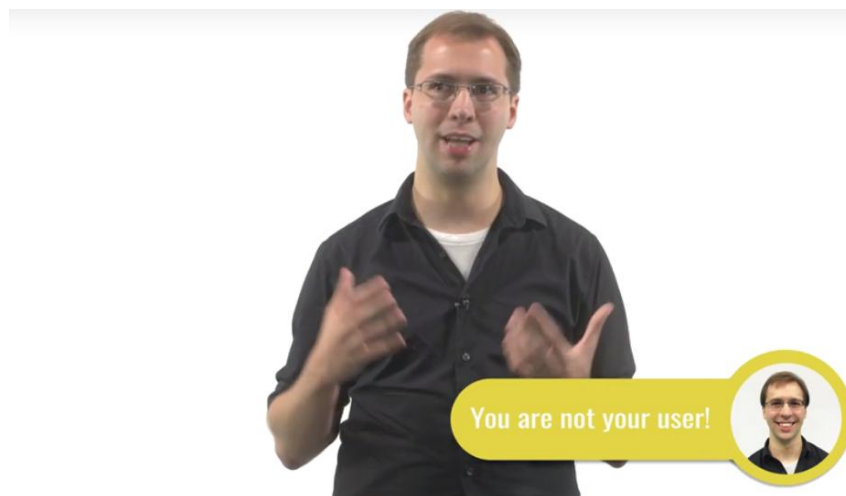


And then, finally, we'll discuss the two kinds of information or data that we gather, Qualitative and Quantitative.

User-Centered Design



User-centered design is design that considers the needs of the user throughout the entire design process. As far as we're concerned, that's pretty much just good design but oftentimes that isn't the way design is done. Design is often done to meet some functional specifications of what the tool must technically be able to accomplish, instead of considering the real needs of the user.



Or sometimes, people will go through an entire design process believing they understand the needs of the user without ever really checking. User-centered design is about prioritizing the user's needs while also recognizing that we don't know the user's needs. So we need to involve them at every stage of the process. Before we start, we need to examine the users needs in depth, both by observing them and by asking direct questions. After we start designing, we need to present our design alternatives and prototypes to the user, to get feedback. And when we near a design that we like, we need to evaluate the quality of the design with real users. Having a good working knowledge of HCI principles helps us go through this more quickly.



But we can't design great interfaces just by applying guidelines and heuristics alone. We have to interact with our users, understand their needs and involve them in the evaluation.

Principles of User-Centered Design

1. The design is based upon an explicit understanding of users, tasks and environments.



The International Standards Organization, has outlined six principles to follow when pursuing user-centered design. Number one, the design is based upon an explicit understanding of users, tasks and environments. That means that we must gather information about the users, the tasks they perform, and where they perform those tasks, and we need to leverage that knowledge throughout the design process.

2. Users are involved throughout design and development.



Number two, users are involved throughout design and development. Involvement can take on many forms, from regularly participating with interviews and surveys about designs and prototypes to actually working on the design team alongside the designers themselves.

3. The design is driven and refined by user-centered evaluation.



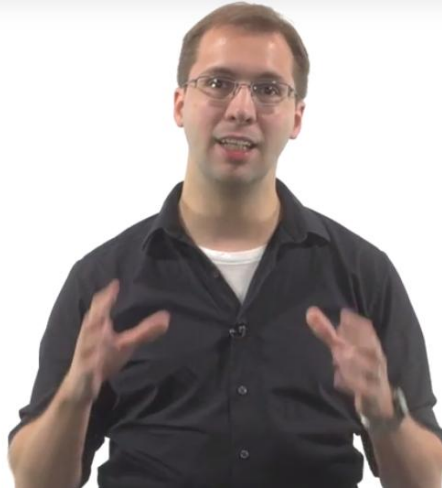
Number three, the design is driven and refined by user-centered evaluation. We absolutely must have real users evaluating the prototypes and interfaces that we assemble.

4. The process is iterative.



Number 4, the process is iterative. No tool is developed once, released, and then abandoned. Designs undergo constant iteration and improvement, even after being released.

5. The design addresses the whole user experience.



Number 5, the design addresses the whole user experience. Many designers are tempted to delineate a certain portion of the experience as their primary interest, but we must address the entire user experience.

6. The design team includes multidisciplinary skills and perspectives.

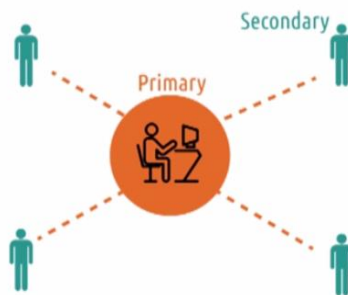


Number six, the design team includes multidisciplinary skills and perspectives. Good teams for pursuing user-centered design include people with a number of different backgrounds, like psychologists, designers, computer scientists, domain experts, and more. So keep these principles in mind when you're doing user experience design.

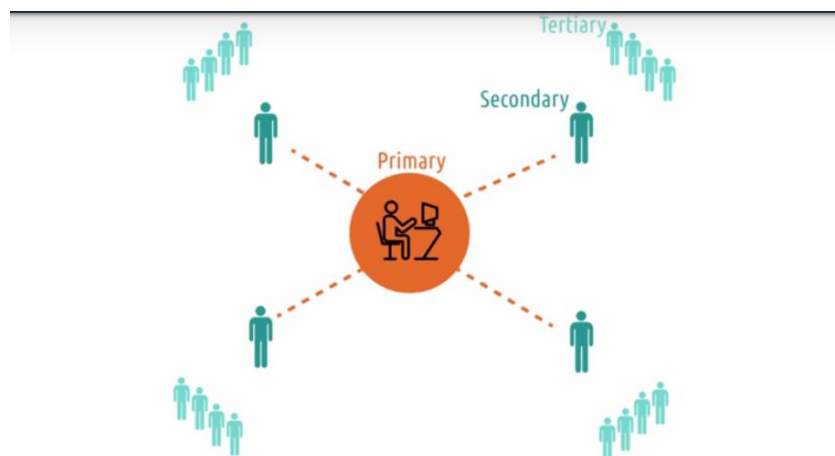
Stakeholders



When we talk about user-centered design, we throw around the word user as if it's pretty obvious what it means. The user is the person who uses the interface that we create, right? However, that's not the only person in whom we're interested. There are multiple stakeholders in this design, and we want to explore how our design is going to affect all of them. The user themselves is what we call the primary stakeholder. They're the person who uses our tool directly.



Secondary stakeholders are people who don't use our system directly, but who might interact with the output of it in some way.



Tertiary stakeholders are people who never interact with the tool or even interact with its output, but who are nonetheless impacted by the existence of the tool. So let's take a couple of examples of this. Imagine we're designing a new grade book tool that makes it easier for teachers to send progress reports to parents. Teachers would interact with the tool, inputting grades and feedback. And so

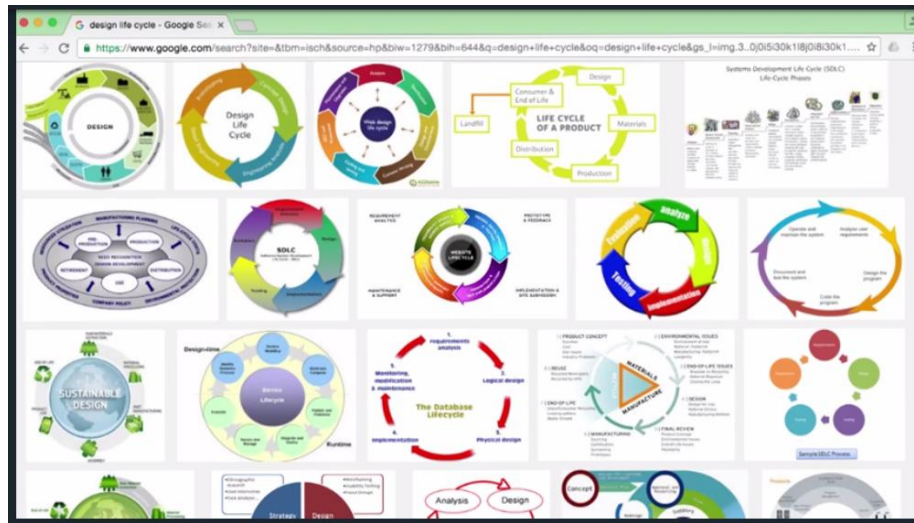
teachers would be our primary stakeholders. Parents receive the output of that tool. Parents receive the progress reports. And so they're secondary stakeholders. They interact with the output of the system, but not with the system itself. Students don't use the system at all, and maybe they don't even see the progress reports unless parents decide to share them. But they're nonetheless affected by the existence of this system. So they're tertiary stakeholders. School administrators might be another stakeholder, but where they fall in this would differ based on how the school sets up the relationship. If they can use the tool to directly monitor and intervene in student progress, they might be primary stakeholders. If they just see aggregated progress reports so they can monitor things, they might be secondary stakeholders. If they never interact with the system in any way, they're nonetheless likely affected by the fact the system is there. And so they'd be tertiary stakeholders. In designing this tool, we need to keep all three kinds of stakeholders in mind. For example, how does parents having more consistent access to great information affect the students? It might foster increased involvement by parents, but it might also facilitate helicopter parenting, where parents are too controlling over their kid's school work and prevent them from developing the necessary meta-cognitive skills and self-discipline that they need to succeed later in life. User-centered design isn't just about catering to the user in the center of this diagram, but it's also about looking at the impact of our designs on all the stakeholders.

Quiz: Reflections: HCI Methods

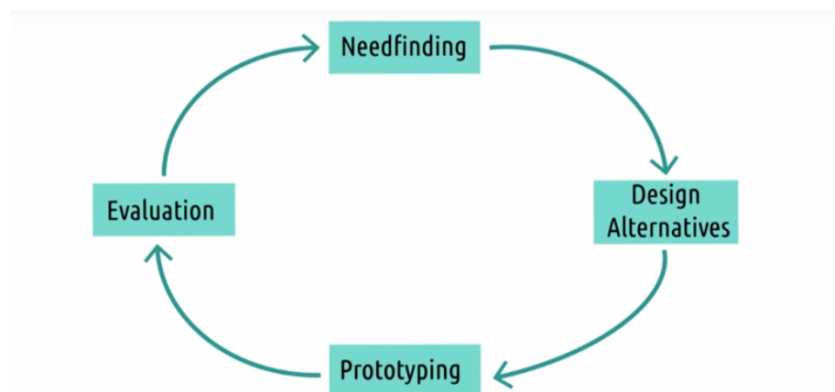
You might actually come from a software engineering background, and so while user-centric designs sounds obvious to some people, you might have experienced the other side of the coin. In many industries and domains, software engineers are still left to design user interfaces themselves. There's a fantastic book about this called *The Inmates Are Running the Asylum*, by Alan Cooper. Where he compares technology to a dancing bear at a circus. He notes that people marvel at the dancing bear, not because it's good at dancing, but because it dances at all. The same way, people marvel at certain pieces of technology not because they work well, but because they work at all. The book was released in 2004 and since then the user has become more and more a focal point of design. And yet there are still places where individuals with little HCI background are designing user-facing interfaces for one reason or another. Since it's a stronger chance you've worked in software engineering, reflect on that a bit. Have you seen places where software engineers, data scientists, or even non-technical people were put in charge of designing user interfaces? How did it go?

I encountered this in my first job, actually. Somewhere between my freshman and sophomore years at Georgia Tech, I had a job as a user interface designer for a local broadcast company. I designed an interface, then I handed it over to a team of engineers for implementation. Late in the process, the requirements were changed a bit and new configuration screen was needed that the engineers just went ahead and looked up. We got the final tool and it all worked beautifully and perfectly, except for this configuration screen. It was a list of over 50 different settings, each with 3 to 5 radio buttons to the side. Each setting was a different length. Each radio button label was a different length. There was kind of placed all over the canvas. There was no grid. It was illegible. It was unusable, but it was technically functional. It met the requirements described in terms of what the user must be able to do, just not how usable it was. Fortunately, there's a greater appreciation of the value of user-centered design now, than there was then. So, many spaces have become so crowded that the user experience is what can really set a company apart. I've actually been noticing a trend lately toward new user experiences around really old tasks. I use an app called stash to buy and sell small amounts of mutual funds. Buying mutual funds has been around forever and E-Trade has even been doing that online for a long time. What differentiates stash is the new user experience. Automated investing, simple tracking, simplified summaries. User experience design really has become a major differentiator between success and failure.

The Design Life Cycle



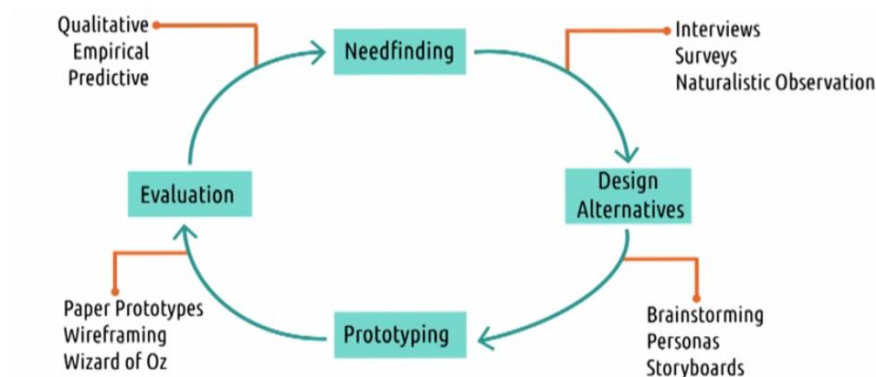
User centered design is about integrating the user into every phase of the design life cycle. So, we need to know two things. What the design life cycle is and how to integrate the user into each phase. Now if you look up design life cycles you'll find a lot of different ideas. We're going to discuss in terms of a four phase design life cycle that's pretty general and probably subsumes many of the other ones you'll find.



The first part of this is Needfinding. In Needfinding, we gather a comprehensive understanding of the task the users are trying to perform. That includes who the user is, what the context of the task is, why they're doing the task, and any other information related to what we're designing. Second, we develop multiple design alternatives. These are very early ideas on the different ways to approach the task. It's important to develop multiple alternatives to avoid getting stuck in one idea too soon. The third step is prototyping. We take the ideas with the most potential and we build them into prototypes that we can then actually put in front of a user. Early on we might do this in very low fidelity ways like with a paper and pencil, or even just verbally describing our ideas. But as we go on we refine and improve. Fourth, and most importantly, we perform user evaluation. We take our ideas that we prototyped and put them in front of actual users. We get their feedback, what they like and what they don't like, what

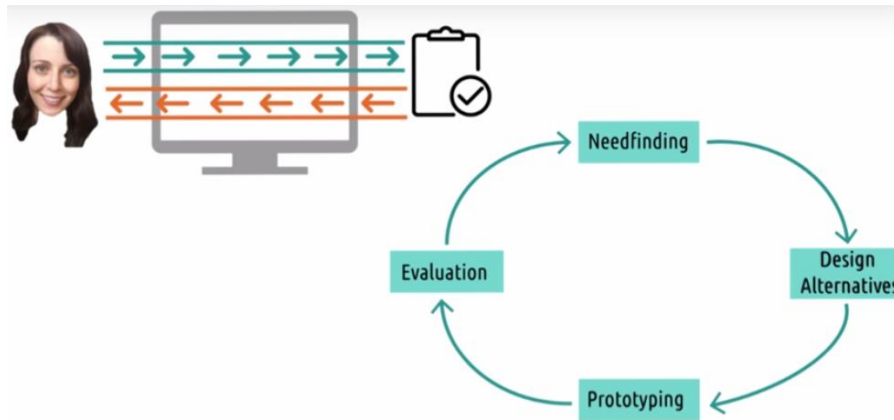
works and what doesn't work, and then the cycle begins anew. The feedback we gain from the users, as well as our experience with these prototypes and design alternatives, improves our understanding of the problem. We now know new areas of the problem we might need to explore with some more need finding. Now we might have some new ideas for design alternatives, or some new ways of expanding the designs that we already have. We then take those things and use them to improve our prototypes. Either prototyping new ideas that we didn't have before, or making our prototypes more rigorous and more polished, so that we can get even better evaluation. And then we put them in front of users again. Each time we go through this cycle our understanding improves, our designs improve, and our prototypes improve. Eventually our prototypes develop to the point of being designs ready for launch, but the cycle doesn't end there. We keep iterating, now with live users doing the evaluation.

Methods for the Design Life Cycle



At every stage of this design life cycle, we're interested in gathering information from the user to better inform our designs. To do that, we need a number of methods to actually obtain that information. Fortunately, there are a number of methods we can employ to try to gather the information we need. And in fact, the majority of this unit will go through all these different methods. These will become the tools in your toolbox. Things you can call upon to grab the information you need when you need it. Not the many of this method are so well-developed that you can cover them in the entire units, or even entire courses. For example, we'll spend about three or four minutes talking about naturalistic observation. And yet, there are entire textbooks and courses on how to do naturalistic observation really well. The goal of this is to give you enough information to get started and enough to know what you need to explore next. Remember, one of the original goals of this class was not just to understand more HCI but also to understand how big the field of HCI actually is.

Design Life Cycles meets Feedback Cycles

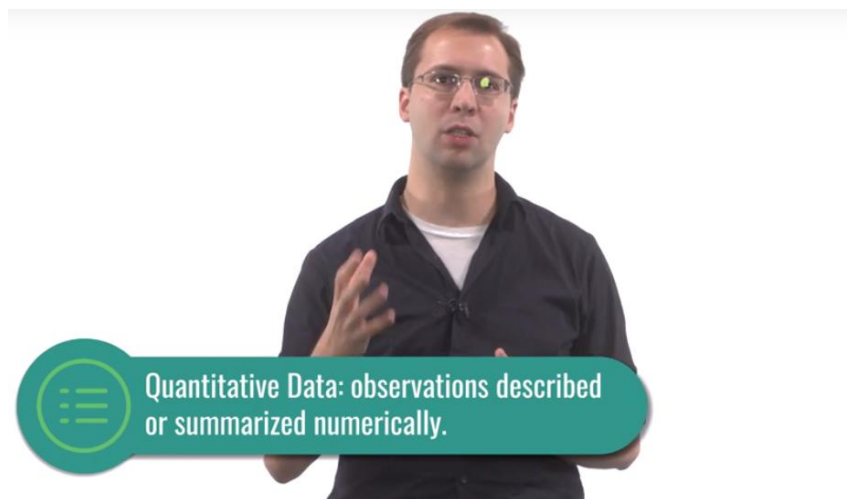


When we talk about feedback cycles, we talk about how they're ubiquitous across nearly every field. And each CI itself, isn't any different. In a feedback cycle, the user does something in the interface to accomplish some goal. And then judges based on the output from the interface, whether the goal was accomplished, then they repeat and continue. In HCI, we're brainstorming and designing interfaces to accomplish goals. And then based on the output of our evaluations, we judge whether or not the goals of the interface were actually accomplished, then we repeat and continue. In many ways, we're doing the same things that our users are doing, trying to understand how to accomplish a task in an interface. Only in our case, our interface is the tools to build and evaluate interfaces and our goal is to help them accomplish their goal.

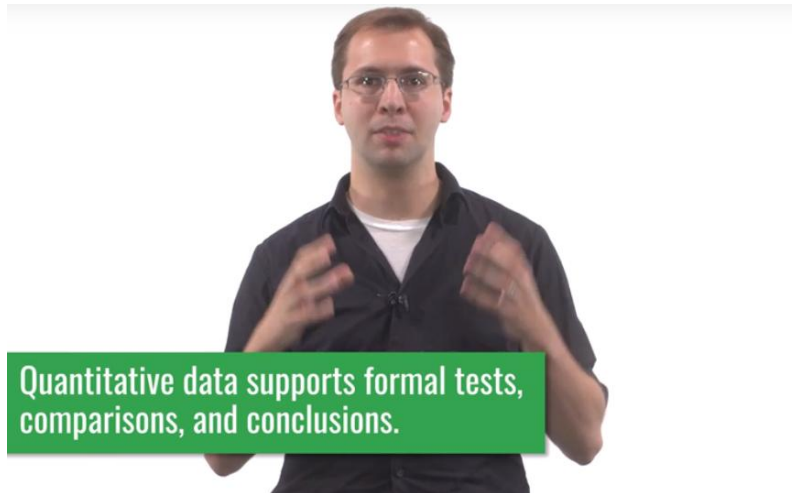
Qualitative vs. Quantitative Data



There is one final distinction we need to understand going forward because it's going to come up at every stage of the design lifecycle, qualitative versus quantitative data. At every stage of the design lifecycle, we're interested in gathering data from users. Early on that might be descriptions of what they do when they're interacting with a task. Or it might be measures of how long certain tasks take to complete, or how many people judge a task to be difficult. Later on, though, it might be whether or not users prefer our new interfaces or how much better they perform on certain tasks.



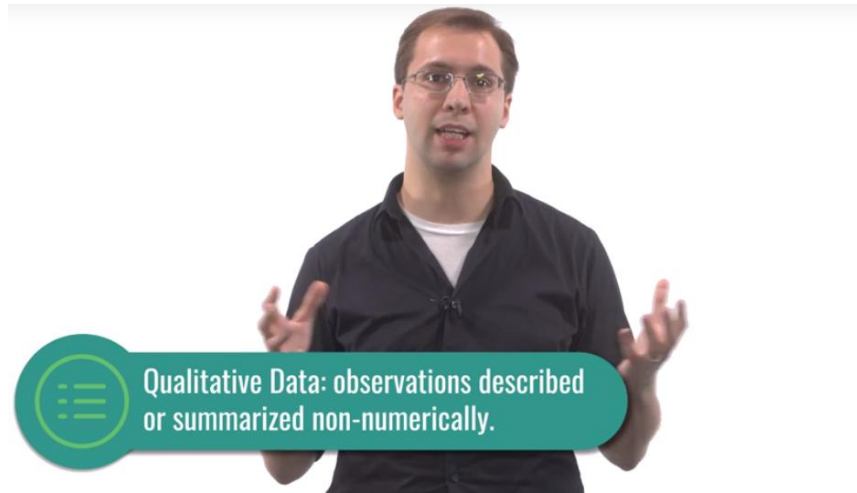
Now data will always fall into one of two categories, qualitative and quantitative. Quantitative is probably the easier one to describe, quantitative data describes anything numeric.



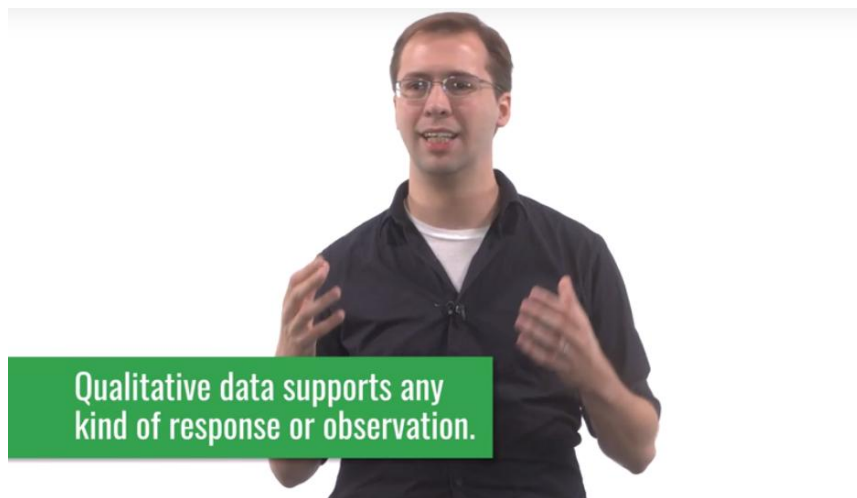
When data is summarized numerically, we can perform statistical tests and summaries on it, draw formal conclusions, and make objective comparisons. Now there are a lot of strengths of quantitative data, but those strengths come in large part because quantitative data only captures a narrow view of what we might be interested in examining.



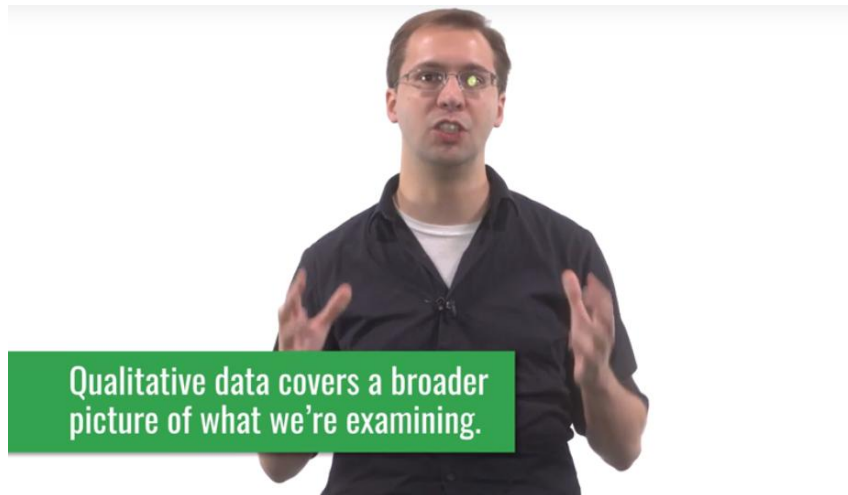
It's very strong for a very small class of things.



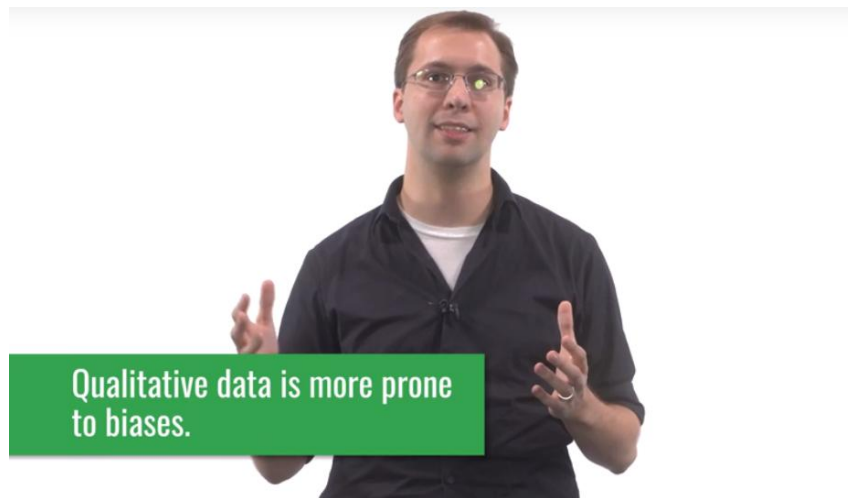
Qualitative data covers pretty much everything else.



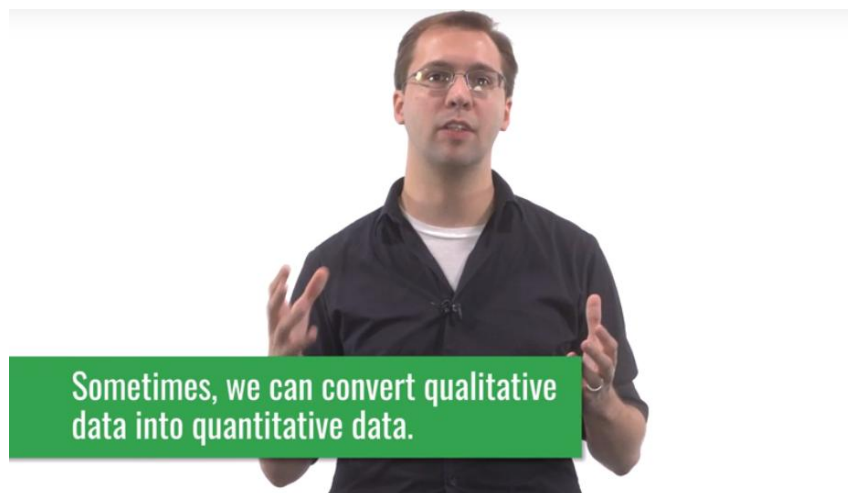
Qualitative Data covers descriptions, accounts, observations, it's often in natural language. It could be open ended survey responses, or interview transcripts, or bug reports or just your personal observations.



Because of its flexibility, qualitative data gives us a much broader and more general picture of what we're examining. But the cost is that it's hard to generate formal conclusions based on qualitative data.



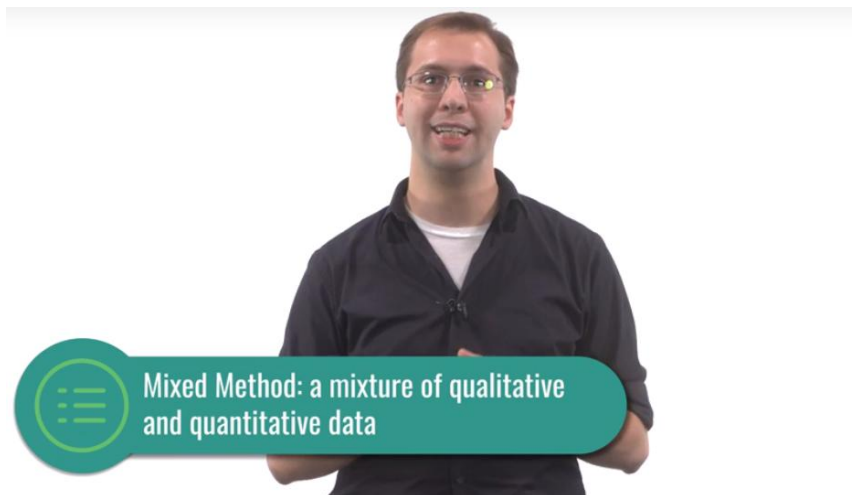
Qualitative data may be more prone to biases.



Now in some circumstances, we can convert qualitative data into quantitative data. For example, we could count the number of survey respondents to an end of course survey who mentioned course difficulty in their free response questions. Now the free response question would be qualitative data but numerically summarizing it generates quantitative data. Generally speaking, though, quantitative data and qualitative data serve different purposes in a design life cycle.



I've heard it described quantitative data provides the what, the qualitative data provides the how or the why. When performing need finding or when doing some initial prototype evaluations, we're likely interested in users' qualitative descriptions of their tasks or their experiences with the interface. It's generally only after a few iterations that we start to be interested in quantitative analysis, to find numeric improvements or changes.



We can also use these in conjunction with one another, collecting both quantitative and qualitative data from the same participants. That's referred to as a mixed method approach, it's a mix of qualitative and quantitative data to paint a more complete picture of the results.

Quiz: Exercise: Quantitative vs Qualitative



Mark which of these data are quantitative.

- ☒ Responses to: 'On a scale of 1 to 5, rate this course's difficulty.'
- ☒ Responses to: 'How much time did you spend per week on this course?'
- ☒ Responses to: 'What did you like about this course?'
- ☒ Count of students mentioning office hours to the above question.
- ☒ Percentage of students that completed the survey.
- ☒ Responses to a forum topic requesting non-anonymous course reviews.
- ☒ The number of participants in a late-semester office hours session.
- ☒ The transcript of the conversation of that late-semester office hours session.

Let's do a quick exercise on quantitative versus qualitative data. Let's imagine we're doing end of course evaluations for some class. For each of the following types of data, mark whether it would be considered quantitative or qualitative. You can skip ahead, if you don't want to listen to me read all these out. We have responses to on a scale of 1 to 5, rate this course's difficulty, responses to how much time did you spend per week on this course, responses to what did you like about this course, count of students mentioning office hours to the above questions, percentage of students that completed the survey, responses to a forum topic requesting non-anonymous course reviews, the number of participants in a late-semester office hours session, and the transcript of the conversation of that late-semester office hours session.



Mark which of these data are quantitative.

- ☒ Responses to: 'On a scale of 1 to 5, rate this course's difficulty.'
- ☒ Responses to: 'How much time did you spend per week on this course?'
- ☐ Responses to: 'What did you like about this course?'
- ☒ Count of students mentioning office hours to the above question.
- ☒ Percentage of students that completed the survey.
- ☐ Responses to a forum topic requesting non-anonymous course reviews.
- ☒ The number of participants in a late-semester office hours session.
- ☐ The transcript of the conversation of that late-semester office hours session.

Quantitative data is numeric. Any of these things that can be measured numerically, really qualify as quantitative data. Many of these things are things that we can just count. We count the number of students mentioning office hours. We count the number of participants. Here we basically count the number of students that completed the survey and divided by all the students in the class. And here we have them count how many hours per week they think they spend in the course. The first option can be

a little bit tricky. On a scale of one to five in a numeric scale, but because students have to choose one, two, three, four or five, it's not a continuous numeric scale. For example, we have no way of guaranteeing the student sees the difference between a three and a four as the same as the difference between a four and a five. The types of analysis we can do on the first kind of data, are more limited. But nonetheless it's still is measured numerically, so it still is quantitative data.

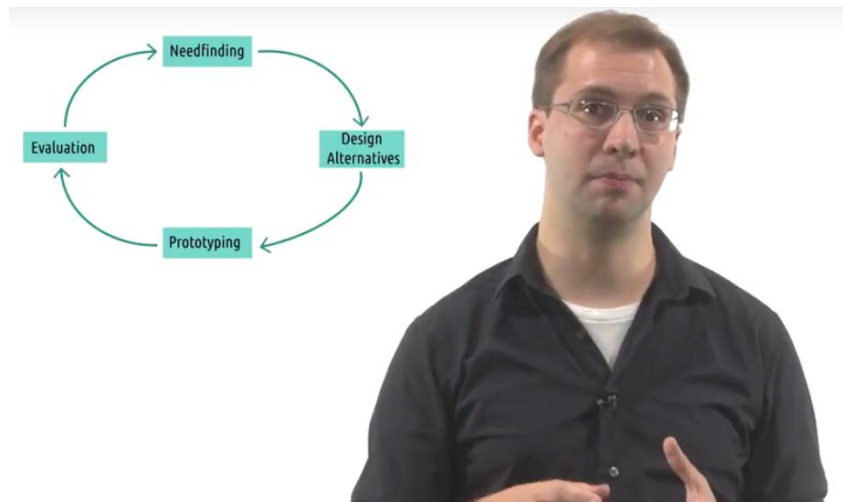
Design Challenge: Recording MOOCs

As we go through the unit in this course on methods, we're going to take a running example as a design challenge to explore the different HCI research methods. I'm going to choose a challenge quite close to home for me. Improving the MOOC recording process.

Exploring HCI: HCI Methods

As we go through the lessons in this unit, we're going to talk about the design life cycle. The process of finding user needs, brainstorming design alternatives, prototyping interfaces and evaluating those prototypes with users. Now depending on how you're taking this material, you might do this on your own with projects or assignments. Some of the most interesting applications of this material, though, are the emerging areas which might be outside the scopes of the assignments that you're doing. So as you're going through lessons in this unit, try to brainstorm a conceptual plan for how you might pursue the design life cycle in your area of interest. There are interesting issues that arise, unique different domains or application areas. In educational technology, for example, you need to take into consideration lots of stakeholders, students, parents, teachers, administrators and more. In virtual reality or wearable devices, you need to think a lot more about the technical constraints in finding creative ways to get around technological limitations to interaction. In computer supportive cooperative work, you might want to explore places where the phenomenon already exists like Wikipedia or geographically distributed companies. Throughout this unit, we'll revisit the stages of the design life cycle and explore how it might apply to your area of interest.

Conclusion to Introduction to Methods



In this lesson we introduce the concept of the design life cycle. There are a lot of versions of the design life cycle out there, but we're going to focus on the most general four-step cycle.



The cycle starts with needfinding, and then goes to constructing design alternatives, then to prototyping, and then to user evaluation, and then the cycle repeats.



The goal of this cycle is to realize user-centered design. Design that takes into consideration the user's needs at every step. In the rest of this unit, we're going to focus on filling up your design life cycle toolbox with tools for gathering the right kind of data at the right time, and using it in the right way.

3.2 Ethics and Human Research

Compiled by Shipra De, Summer 2017

Introduction to Research Ethics



[MUSIC] Before we start working with real users there are a few ethical considerations we have to make. If you are doing a research as part of a university these are part of your contract with the university to do research on their behalf. Even if you are doing research independently for an industry there are still ethical obligations to follow. These considerations are important not only to preserve the rights of our users, but also to ensure the value of the data that we gather.



In this lesson, we'll talk a little bit about where these kinds of ethical considerations came from.

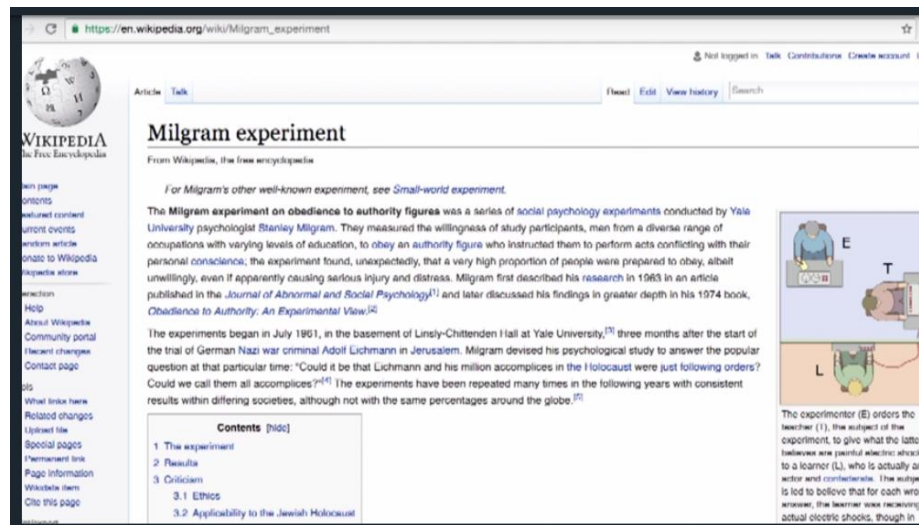


Then we'll talk about some of the basic ethical considerations that we need to make.

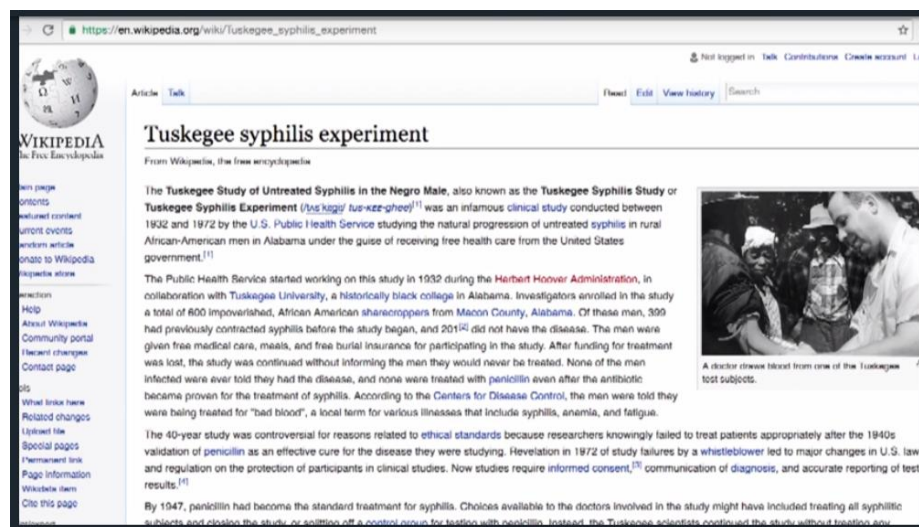


We'll also talk about Institutional Review Board, or IRB, the university organization that governs human subjects research.

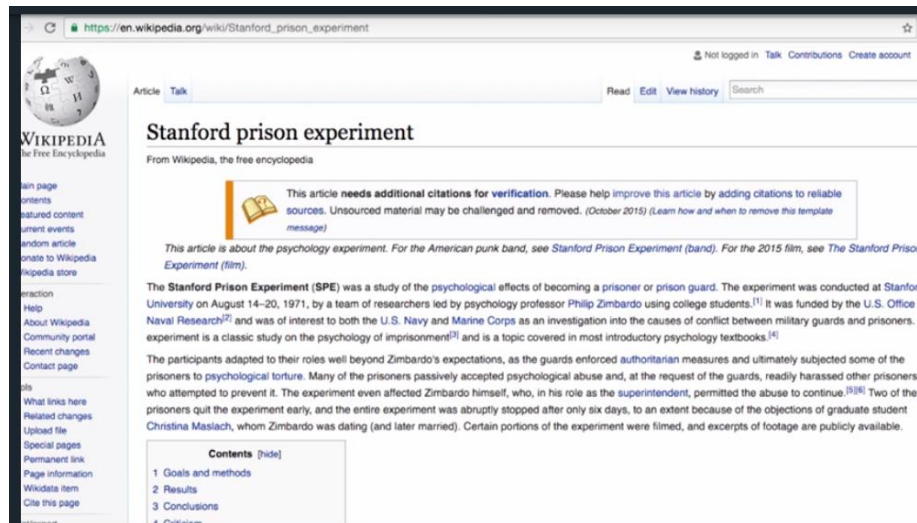
Origin of Institutional Review Board



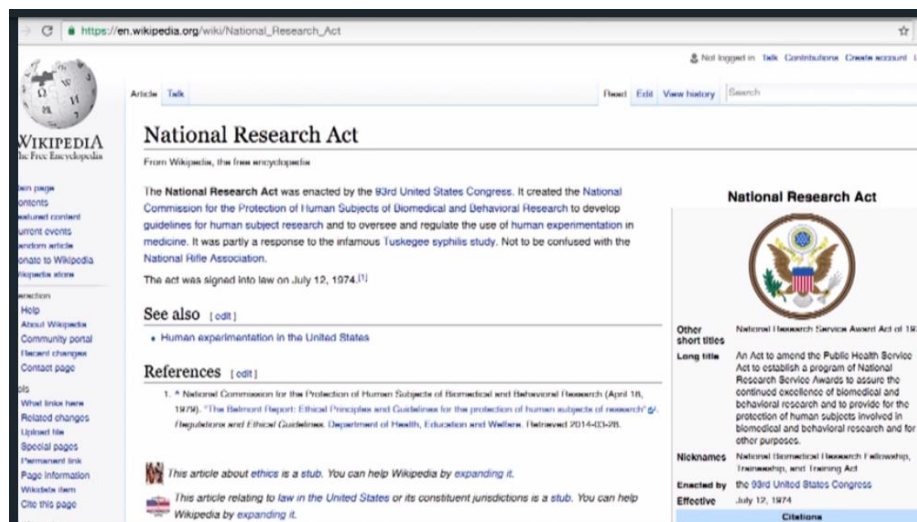
In the first half of the 20th century, a number of pretty clearly unethical human subjects experiments took place. Many of them were conducted by scientists working for the Axis powers during World War II. But famously, many were also conducted right here in our own backyard, here in the United States. Among them were Milgram's obedience experiment where participants were tricked into thinking that they had administered lethal shocks to other participants to see how obedient they would be.



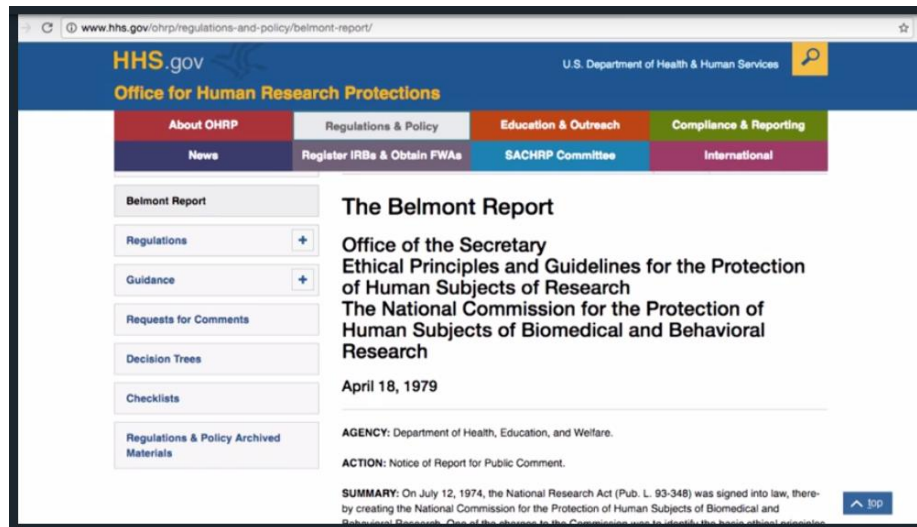
There was the Tuskegee syphilis study where rural African American men were intentionally injected with syphilis to study its progression.



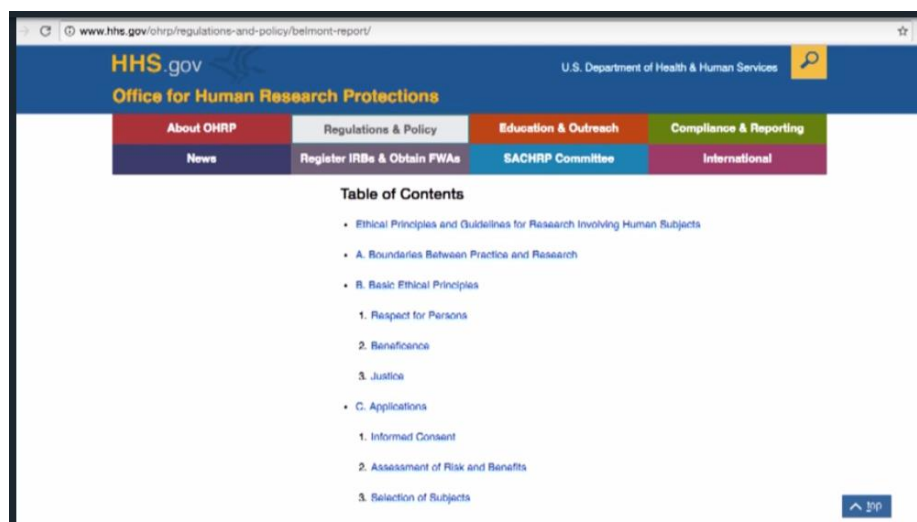
And there was the Stanford prison experiment where participants were psychologically abused to test their limits or test how they would act under different circumstances.



In response to all this, the National Research Act of 1974 was passed, which led to the creation of institutional review boards to oversee research at universities.



The Belmont Report further summarizes basic ethical principles that research must follow in order to receive government support.

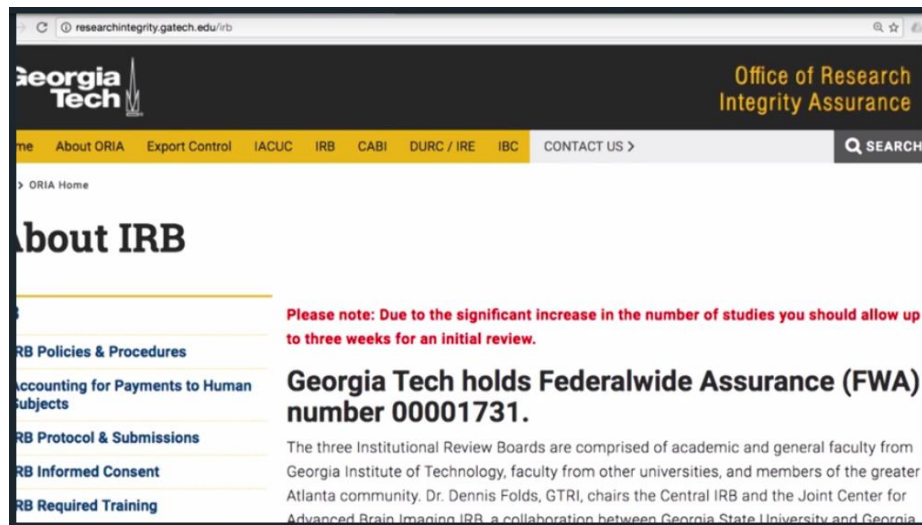


Among other things, the law dictated that the benefits to society must outweigh the risks to the subjects in the case of these experiments. It also dictated that subjects must be selected fairly, which was a direct response to the Tuskegee syphilis study. And perhaps most importantly, it demanded a rigorous informed consent procedures. So, the participants know what they're getting into and can back out at any time. These efforts all attempt to make sure that the positive results of research outweigh the negatives and that participant rights are always preserved.

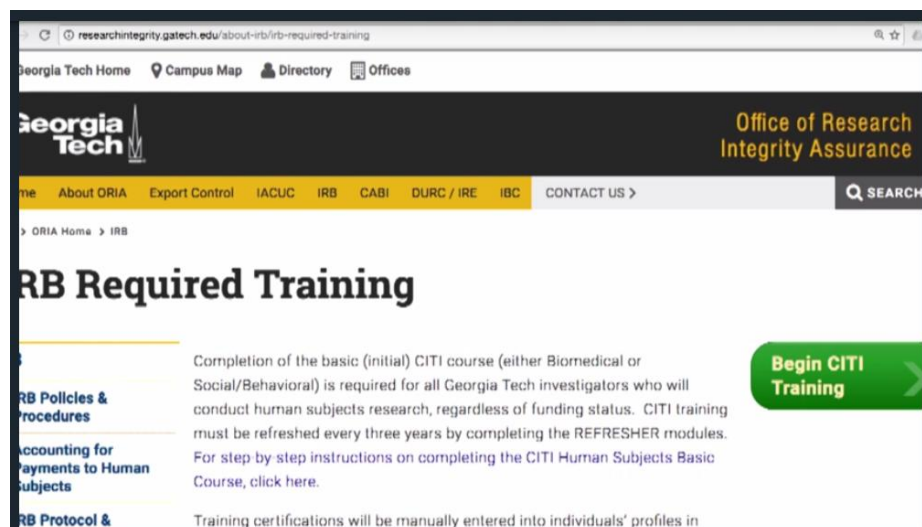
The Value of Research Ethics

In this lesson, we're largely focusing on the practical steps we go through to get approval for human subject's research. But before we get into that, I want to highlight that this isn't just a bunch of bureaucratic steps necessary to make sure the people are treated ethically at all stages of research, IRBs main task is to make sure the potential benefits of the study are worth the potential risks. So as part of that, part of the role is to make sure the potential benefits are significant. A lot of the steps of the process are going to ensure that the data that we gather is useful. So for example, the IRB is sensitive about the perception of coercion. When participants feel coerced to participate in research, the data they actually supply may be skewed by that negative perception which impacts our results. Similarly, we might design studies that have some inherent biases or issues to them. We might demand too much from participants or ask questions that are known to affect our results. Much of the initial training to be certified to perform research is similarly not just about doing ethical research but also about doing good research. By recording who is certified, IRB helps ensure that research personnel all understand the basics of human subjects research. IRB is also there to monitor for these things as well and many of the steps of this process ensure that the research we perform is sound and useful. After all, if the research we perform is not useful, then even the smallest risks will outweigh the nonexistent benefits.

Getting Started: CITI Training



If you're going to be doing a research just part of a University project or University class, you need IRB approval. Different universities have different processes and policies for getting started with IRB. We're going to discuss the Georgia Tech policies and sets where these class is based but you should check with your university if you're from somewhere else to make sure you're following the right policies for your school. To get started, we need to complete the required training. So here I'm showing the IRB website, which is researchintegrity.gatech.edu/irb. And to get started, you need to complete your required training. So click required training over on the left.



This'll take you to a page that overviews the IRB required training and gives you a link of the left to begin your CITI training.

Georgia Tech Login Service

Enter your GT Account and Password
Login requested by: idp.gatech.edu

GT Account:
djoyner3

Password:

☐ Warn me before logging me into other sites.

LOGIN clear

ATTENTION: When you are finished using all of your authenticated applications, please log out of this system and exit your browser to ensure you do not leave any of your applications (such as your e-mail) open to other users of this machine.

TERMS OF USE
This computer system is the property of Georgia Tech and is available for authorized use only, in accordance with the Computer & Network Usage and Security Policy (CNUSP). Users should have no expectation of privacy, as any and all files on this system may be intercepted, monitored, recorded, copied, audited, inspected, and disclosed to authorized site(s) and/or law enforcement personnel in order to meet administrative and/or legal obligations. By using this system, I acknowledge and consent to these terms.

[I don't know my GT Account](#)
[I don't know my password](#)
[My correct username and password aren't working](#)

For assistance, please contact the CIT Technology Support Center at 404-894-7173 (Mon-Fri 8am-6:00pm ET).

Additional documentation includes how to integrate your application with GT I login.

So click that, then login to your Georgia Tech account.

English Text size: A A David Joyner ID: 2432809 | Log Out | Help

CITI PROGRAM Collaborative Institutional Training Initiative Search Knowledge Base

Main Menu | My Profiles | My CEUs | My Reports | Support

Main Menu

Georgia Institute of Technology Courses

Course	Status	Completion Report	Survey
Group 2 Social / Behavioral Research Investigators and Key Personnel	Passed 07/22/2015	View/Print Share	Post-course evaluation

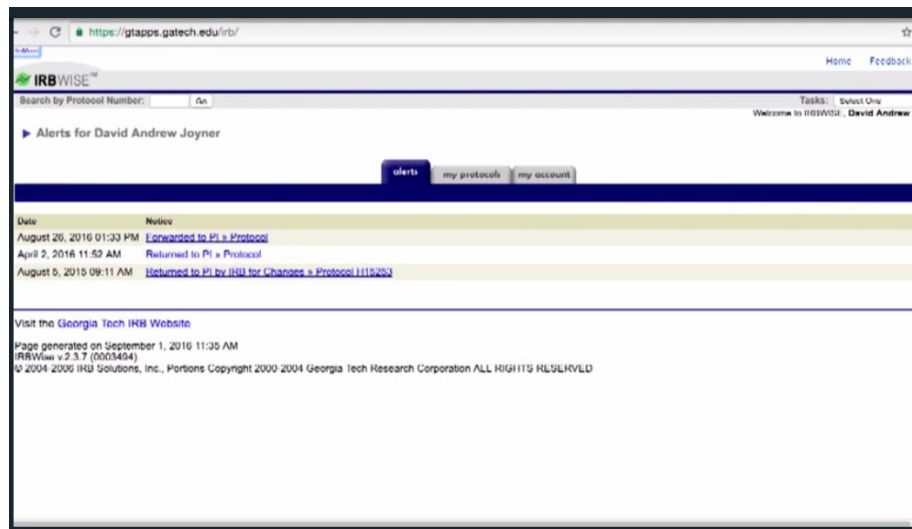
My Learner Tools for Georgia Institute of Technology

- Add a Course
- Remove a Course
- View Previously Completed Coursework
- Update Institution Profile
- View Instructions page
- Remove Affiliation

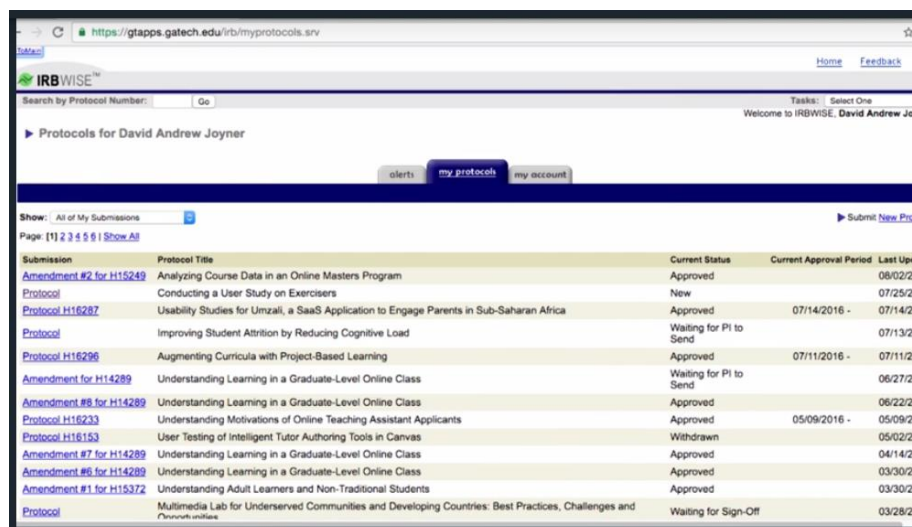
[Click here to affiliate with another institution](#)
[Affiliate as an independent learner](#)

And you're going to want to complete Group 2, Social and Behavioral Research Investigators and Key Personnel. I can't show you exactly what it looks like to sign up fresh because I've already completed it. But you should be able to add a course and add that as your course. After you've completed CITI training you'll receive your completion report and then you'll be ready to get started with IRB.

Getting Started: IRB

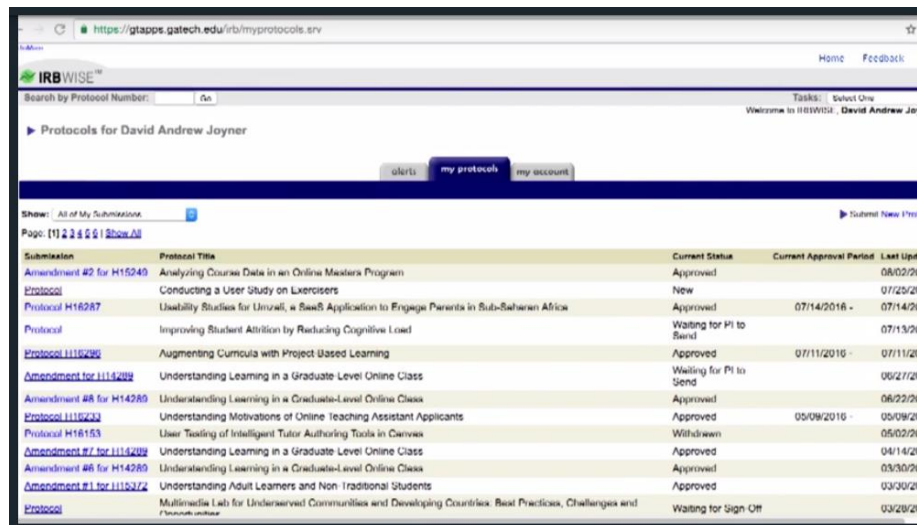


After you've completed any necessary training you can access the IRB application for your own university. We're doing this in terms of Georgia Tech, so here's the tool we used called IRBWISE.



Here under my protocols, you'll see a list of all of the protocols to which you're added. A protocol is a description of a particular research project. It outlines the procedures that the IRB has approved regarding consent, recruitment, experimentation and more. Here we see approved protocols. Protocols that are new and haven't been submitted yet. Protocols that are waiting for the IRB to act on them. And amendments to protocols. Amendments are how we change earlier protocols to add new researchers or change what we're approved to do. After a protocol is approved, any changes must be submitted to the IRB as an amendment to be accepted separately.

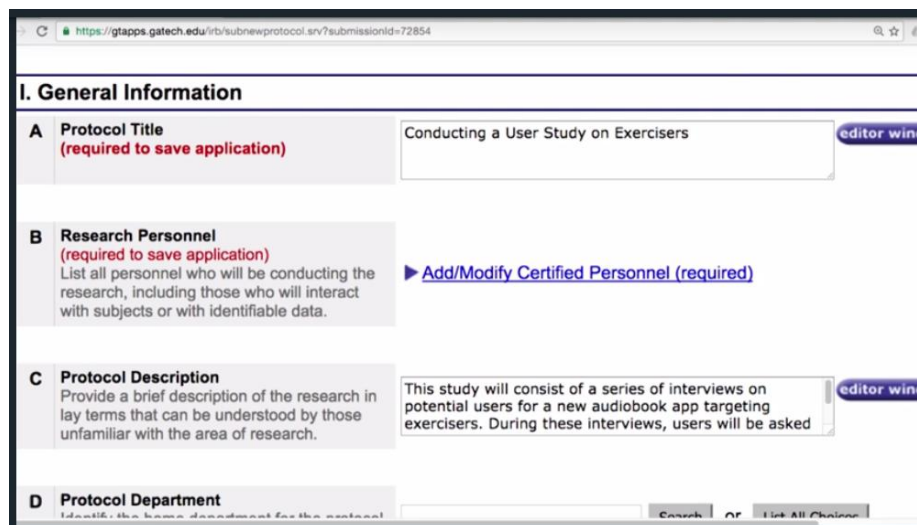
IRB Protocols: Basics



The screenshot shows the IRBWISE web application interface. At the top, there's a search bar and a welcome message for David Andrew Joyner. Below this, there are tabs for 'start', 'my protocols', and 'my account'. The 'my protocols' tab is selected, displaying a table of protocols. The table has columns for Submission, Protocol Title, Current Status, Current Approval Period, and Last Update. The protocols listed include amendments and protocols for various studies, such as 'Analyzing Course Data in an Online Masters Program' and 'Conducting a User Study on Exercisers'.

Submission	Protocol Title	Current Status	Current Approval Period	Last Update
Amendment #2 for H15240	Analyzing Course Data in an Online Masters Program	Approved		08/02/20
Protocol	Conducting a User Study on Exercisers	New		07/25/20
Protocol H16287	Usability Studies for Unizail, a SaaS Application to Engage Parents in Sub-Saharan Africa	Approved	07/14/2016 -	07/14/20
Protocol	Improving Student Attrition by Reducing Cognitive Load	Waiting for PI to Send		07/13/20
Protocol I116296	Augmenting Curricula with Project Based Learning	Approved	07/11/2016 -	07/11/20
Amendment for I114209	Understanding Learning in a Graduate-Level Online Class	Waiting for PI to Send		06/22/20
Amendment #8 for H14285	Understanding Learning in a Graduate-Level Online Class	Approved		06/22/20
Protocol I116233	Understanding Motivations of Online Teaching Assistant Applicants	Approved	05/09/2016 -	05/09/20
Protocol H16153	User Testing of Intelligent Tutor Authoring Tools in Genex	Withdrawn		05/02/20
Amendment #7 for I114209	Understanding Learning in a Graduate-Level Online Class	Approved		04/14/20
Amendment #6 for H14285	Understanding Learning in a Graduate-Level Online Class	Approved		03/30/20
Amendment #1 for I116272	Understanding Adult Learners and Non-Traditional Students	Approved		03/30/20
Protocol	Multimedia Labs for Underserved Communities and Developing Countries: Best Practices, Challenges and	Waiting for Sign Off		03/28/20

Generally speaking, you might not ever be asked to complete a protocol yourself. You might instead just be added to an existing protocol. Still, you need to make sure to understand the procedures outlined by any protocol to which you're added, because they still govern what you do. So we'll run through the process of creating a protocol, but this will also cover the details to know about any protocol to which you are added. So for this, I have a draft protocol covering our user study on people who exercise.



The screenshot shows the 'General Information' section of a protocol form in the IRBWISE application. It includes fields for Protocol Title, Research Personnel, Protocol Description, and Protocol Department. The 'Protocol Title' field is filled with 'Conducting a User Study on Exercisers'. The 'Research Personnel' field has a link to 'Add/Modify Certified Personnel (required)'. The 'Protocol Description' field is filled with a brief description of the study. The 'Protocol Department' field is empty.

I. General Information

A Protocol Title
(required to save application)
Conducting a User Study on Exercisers

B Research Personnel
(required to save application)
List all personnel who will be conducting the research, including those who will interact with subjects or with identifiable data.
[Add/Modify Certified Personnel \(required\)](#)

C Protocol Description
Provide a brief description of the research in lay terms that can be understood by those unfamiliar with the area of research.
This study will consist of a series of interviews on potential users for a new audiobook app targeting exercisers. During these interviews, users will be asked

D Protocol Department
Identify the home department for the protocol.
Search or List All Choices

Every protocol starts with the title of the protocol, and some certified personnel. These are required just to save the protocol.

[Skip To Main](#)

IRBWISE™

► **Associate Study Personnel**

Select Person (by Last Name):

please start typing [View](#)

[their certifications](#)

note: The search list above contains all current Georgia Tech students & employees. If you need to list someone on this protocol who is not in this list and is not affiliated with Georgia Tech, please send the following information to the [Office of Research Compliance](#):

- The person's name
- Organization/Company
- Phone #
- E-mail Address

We add approved research personnel on the study personnel page.

send the following information to the [Office of Research Compliance](#):

- The person's name
- Organization/Company
- Phone #
- E-mail Address
- Role on this protocol
- Proof of completion of Human Subject Training

Select Role:

Proof of Experience & Certifications:
Upload your current CV or resume. Include any license & certification such as medical license.

Attach Files: No file chosen
 No file chosen [Attach More..](#)

directions: This list contains all active students, faculty, and staff at Georgia Tech.

We would enter their name, select their role, and if their certification isn't already in the system, we would attach it here.

Certifications:
Upload your current CV or resume.
Include any license & certification
such as medical license.

Attach Files: Choose File No file chosen
Choose File No file chosen [Attach More..](#)

Add This Person Continue with Application

directions: This list contains all active students, faculty, and staff at Georgia Tech.

Study Personnel Listed:

Select	Name	Role	Certification	Documents
<input type="radio"/>	David Andrew Joyner	PI	<ul style="list-style-type: none"> Georgia Tech CITI Human Subjects Training Certification (Approved): July 22, 2015 - July 21, 2018 	

Modify Selected Delete Selected

TOP

After adding them, they'll appear down here. The primary investigator, PI, must always be added first and it must be a faculty member.

I. General Information

A Protocol Title
(required to save application)
Conducting a User Study on Exercisers [editor window](#)

B Research Personnel
(required to save application)
List all personnel who will be conducting the research, including those who will interact with subjects or with identifiable data. [Add/Modify Certified Personnel \(required\)](#)

C Protocol Description
Provide a brief description of the research in lay terms that can be understood by those unfamiliar with the area of research.
This study will consist of a series of interviews on potential users for a new audiobook app targeting exercisers. During these interviews, users will be asked [editor window](#)

D Protocol Department
Select the department that best describes the research [Search](#) [or](#) [List All Choices](#)

The protocol description covers the study at a high level. This should briefly touch on what will be done, what the goal of the research is and what subjects will be asked to do. It doesn't cover all the details, but it covers enough for someone to understand generally what we're going for with this study.

https://gtapps.gatech.edu/irb/subnewprotocol.srv?submissionId=72854

II. The Protocol: Research Design and Methodology

A Describe the research design, including the proposed research methodology. For research directly involving human subjects, describe in chronological order the procedures that will occur. If subjects will be assigned to various conditions, describe how and why assignments will be made. (Examples of studies not directly involving human subjects, but still needing IRB approval, include prospective record reviews, observational behavior without manipulation, and use of anonymized data).

This research is a qualitative study on the experiences of potential users engaging in exercise while also listening to audiobooks. Participants will be recruited from local gyms and parks via a collection of fliers. editor window

B State the duration of subject participation. How many hours, days, weeks or months? Specify number of sessions and, to the extent possible, state total amount of time for subject participation.

Participants will be asked to attend one one-hour session. Subjects will also be asked if they would be willing to be contacted about follow-up sessions in the future. editor window

C Describe study assessments and other data collection methods. Upload all instruments, including raw data.

Under the research design and methodology section, we describe our research. First, we describe the research design in the methodology. With human subjects research, this focuses on what users will experience and in what order. It also covers some experimental details like how subjects might be assigned different experimental conditions. Then we describe the duration of subject participation to make sure subjects aren't being asked to do too much.

https://gtapps.gatech.edu/irb/subnewprotocol.srv?submissionId=72854

C Describe study assessments and other data collection methods. Upload all instruments, including raw data, scales, questionnaires, surveys, focus group and interview guides, and so on at the end of this online application in the ATTACH DOCUMENTS section.

(NOTE: The IRB recognizes that such specificity may not be possible in ethnographic or anthropological studies. In such cases, provide sufficient detail for the board to understand the study methodology).

The primary study assessments will be a survey and a semi-structured interview. Participants will first complete a demographic survey. Then, during the study, participants will answer a series of interview questions. editor window

D Fully describe any potential benefits of this study. All ethical studies pose some benefit-- whether to individual research subjects, to the greater community, as a building block for further development of treatment, and so on. (If subjects will not benefit from participating, this should be disclosed in the benefits section of the consent document).

This study will contribute to a greater understanding of the challenges of multitasking during exercise, and inform the design of devices that address those challenges. editor window

Depending on what we're doing, we may need to provide data collection methods. This includes things like surveys, pre-tests and post-tests interview scripts, anything pre-prepared to elicit data for the participant. Then we also need to fully describe the potential benefits of the study. Remember, IRB is about making sure that the benefits outweigh the risks. If there are no benefits, then the benefits can't outweigh the risks. Benefits don't need to be to the individual participants though, but they could be to the greater community as a whole.

The primary study assessments will be a survey and a semi-structured interview. Participants will first complete a demographic survey. Then, during the study, participants will answer a series of interview questions

D Fully describe any potential benefits of this study. All ethical studies pose some benefit-- whether to individual research subjects, to the greater community, as a building block for further development of treatment, and so on. (If subjects will not benefit from participating, this should be disclosed in the benefits section of the consent document).

This study will contribute to a greater understanding of the challenges of multitasking during exercise, and inform the design of devices that address those challenges.

E Fully describe any known risks to subjects participating in this study and, to the best of your knowledge, indicate the likelihood of such risks occurring. Also state any measures to be taken to minimize or manage risks or to manage unpreventable risks.

There are no known risks associated with this study beyond those associated with routine exercising.

Similarly, we also needed to describe the risks associated with the study. For usability studies, very often our risks are not going to be very significant. Our risks might be those associated with normal work at a computer. But we still need to address this to acknowledge that we thought about what risks might arise to participants.

F Describe the statistical analysis plan, its design, and the rationale for the plan.

G What are the anticipated start and end dates for the proposed research? Include the expected number of years that data analysis will continue.

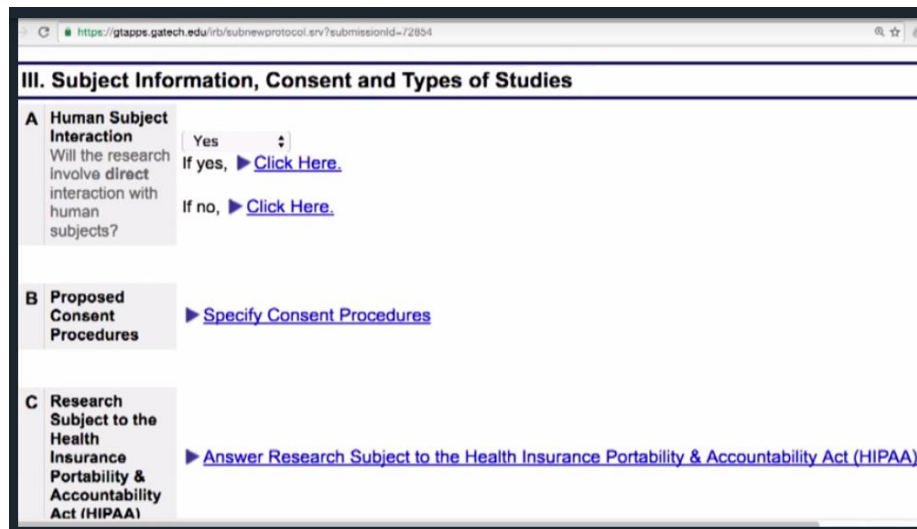
Federal regulations currently require that IRB approval remain active during data analysis (if subject not de-identified) even though subject enrollment and interaction may be complete. Be sure to include period of data analysis when calculating the end date if you will maintain subject identifiers.

Data collection will continue for 30 days, starting with the acceptance of this protocol; research on the collected data will continue for one year after that.

H Upload a fully annotated bibliography or reference section, including the results of the literature search in support of this proposed study.

Then we describe the plan for the statistical analysis if we have one. Qualitative research might not have a statistical analysis plan, so that's why I've left this blank. Finally, we need to describe start and end dates of the research. Very often, this will break the research into a data collection phase and a research phase, where we actually analyze the data that we collected. Now we won't generally need to worry about many of the remaining options on this form, because we're not doing clinical studies and we generally don't have external funding unless you're working on a professor's research project. So now let's move on to subject details.

IRB Protocols: Human Subject Interaction



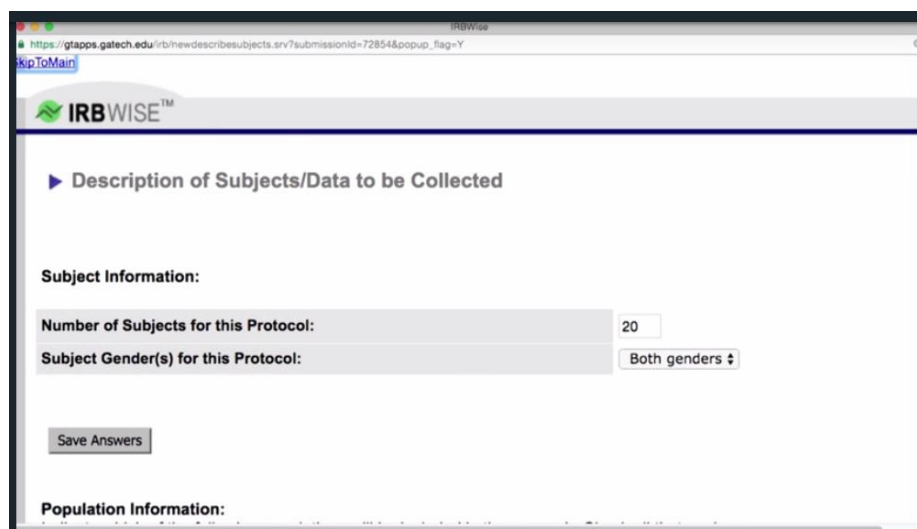
III. Subject Information, Consent and Types of Studies

A Human Subject Interaction
Will the research involve direct interaction with human subjects?
Yes ☐ If yes, [Click Here.](#)
If no, [Click Here.](#)

B Proposed Consent Procedures
[Specify Consent Procedures](#)

C Research Subject to the Health Insurance Portability & Accountability Act (HIPAA)
[Answer Research Subject to the Health Insurance Portability & Accountability Act \(HIPAA\)](#)

Because we're interested in human-computer interaction, we almost certainly will have human-subject interaction. So when it asks, will the research involve direct interaction with human subjects, we would click, yes.



IRBwise

[Skip To Main](#)

Description of Subjects/Data to be Collected

Subject Information:

Number of Subjects for this Protocol: 20

Subject Gender(s) for this Protocol: Both genders

Population Information:

That will bring us to a screen where we describe our subjects and the data we plan to collect from them. First, they want to know how many subjects we have and what genders. They want to make sure that we're not wasting participants' time if we're not going to actually analyze all their data. And that we're being fair to the different genders. A common problem in early research was over-representing male populations.

Population Information:
Indicate which of the following populations will be included in the research. Check all that apply.

Population	Description	Included?	Questions
Children (The state of Georgia defines children as persons younger than 18 years of age.)		<input type="checkbox"/>	(none)
Economically disadvantaged		<input type="checkbox"/>	(none)
Educationally disadvantaged		<input type="checkbox"/>	(none)
Employees or Subordinates of Investigators		<input type="checkbox"/>	(none)
Individuals that do not have the capacity to consent for themselves	(e.g. those with Alzheimer diseases)	<input type="checkbox"/>	(none)
Neonates		<input type="checkbox"/>	(none)
Non-Native English Speakers		<input type="checkbox"/>	(none)
Patients		<input type="checkbox"/>	(none)

Second, they'll want to know if we're targeting any vulnerable populations. People that might not have the capacity to give true informed consent. If we do, we need to make special accommodations to make sure they're fully aware of their rights as participants.

Subjects, Inclusion, and Exclusion Criteria

A Provide the scientific justification for the number of subjects to be enrolled in the study.

For clinical protocols, it is important to **STATISTICALLY** justify the number of participants needed and to state a precise number to be enrolled.

For non-clinical and minimal risk studies, participant numbers may be stated as a range, (i.e.: 100-500). We will mail surveys to 500 addresses and hope to have responses from 100 participants). If responses are received from more than 100 participants, over-enrollment will not have occurred. Web-enabled recruitment may result in far more responses than anticipated or needed. Researchers should be prepared to shut down a web recruitment site immediately if responses exceed the number of approved participants. Over-enrollment must be reported to the IRB as a protocol violation or deviation, and it may be unethical to accept responses from participants whose data are not needed and will not be utilized.

20 subjects will provide the adequate range of ages, genders, and experience levels, while also remaining feasible with the resources allocated to this study. [editor window](#)

B State the study population INCLUSION criteria. Inclusion criteria should be designed so that the

Third, they want the scientific justification for the number of subjects to enroll in our study. Like I said, they want to make sure that we're not going to waste the time of a bunch of participants and then just throw their data out. That wouldn't be very respectful for our participants' time. If we're doing statistical tests this might be the number of participants necessary to find a certain effect size, which we'll talk about when we talk about quantitative research. If we're doing qualitative research, this is usually a smaller number and is more based on how many different perspectives we need to get a good picture of what we're trying to analyze. Alternatively, we might have external limits on our number of participants. For example, if you're doing classroom research, your maximum number of students is the maximum number of students in the class.

B State the study population INCLUSION criteria. Inclusion criteria should be designed so that the study population has the attributes necessary for the purpose of the research to be accomplished.

Inclusion (and exclusion) criteria may include age, race, sex, ethnicity, type and stage of disease, medical history, certain behaviors, occupation, and so on. By explicitly defining these criteria, researchers increase the likelihood of obtaining reliable and useful data.

The inclusion criteria for the study are adults over the age of 18 who engage in exercise at least once per week. [editor window](#)

C State the study population EXCLUSION criteria. Exclusion criteria are those that disqualify potential subjects from participating.

Exclusion (and inclusion) criteria may include age, race, sex, ethnicity, type and stage of disease, medical history, certain behaviors, occupation, and so on. By explicitly defining these criteria, researchers increase the likelihood of obtaining reliable and useful data.

There are no exclusion criteria beyond participants not explicitly included in the inclusion criteria. [editor window](#)

Next, we state the inclusion and exclusion criterion. The inclusion criterion are, who we specifically including, who's our targeting audience? The exclusion criterion are those that we're specifically excluding. Often times, one of these will just be the inverse of the other, but there may be times when we need to be more specific. For example, if we were doing research on undergraduate computer science education, our inclusion criteria might be undergraduate students. But our exclusion criteria would be undergraduate students that have previously taken a computer science class.

F Provide steps to be taken to ensure additional protection of the rights and welfare of vulnerable populations.

[editor window](#)

G Indicate subject age range(s) below. Check all that apply.

- ☐ 0 (Birth) - 5 years
- ☐ 06 years - 11 years
- ☐ 12 years - 17 years
- ☒ 18 years - 69 years
- ☒ 70 years and older

As before, we can skip the questions, for our purposes at least, that are more based on clinical research. But at the bottom, we need to provide the steps necessary to ensure additional protection of the rights and welfare of vulnerable populations. For example, if we're working with 10 year olds, how do we make sure 10 year olds really understand that they really do have the right to opt out of this research? We need to provide a plan for that here if we're working with a vulnerable population.

Recruitment & Compensation

A Describe in detail the recruitment plan. Specify where and how potential subjects will be identified. By recruitment ads, word of mouth, email? If using flyers, email, advertisements, screen shots from websites, or other documents, upload copies in the ATTACHED DOCUMENT SECTION.

If recruitment will be by word of mouth, provide a brief script. The IRB does not expect the script to be followed verbatim; however, the recruitment language must be reviewed.

Flyers will be posted at local gyms and parks. The flyer to be used for recruitment is attached to this protocol. editor window

B Is a Georgia Tech Student Subject Pool being used? **NOTE:** Only the School of Psychology and the College of Management have formal Student Subject Pools. In order to recruit from among either group, advance arrangements must be made with the manager of that pool.

☒ No
☐ Yes

Finally, we also need to describe our recruitment procedures. How are we going to find our subjects? First, we'll note what we'll say and how we'll communicate with them here. If we're using the Georgia Tech subject pool, which is a system for finding research subjects within the Georgia Tech student body, we'll indicate so here.

C Describe the compensation plan for subject participation. If compensation will be class credit, state how much credit will be granted for student participation. Include plans for prorating compensation if subject does not complete study.

NOTE:

If a lottery or raffle is proposed as compensation: the State of Georgia requires a license for a true lottery. A license is not generally required if non-participants may enter the lottery without being in the study.

U.S. Tax Law requires a mandatory withholding of 30% for nonresident alien payments of any type.

Consult the IRB Policies and Procedures at www.researchintegrity.gatech.edu for additional guidance on both points.

Participants are not compensated for this study. editor window

And last, we'll note the kind of compensation we plan to provide the participants. Many times we won't compensate our participants at all. But if we're doing a bigger research study that actually has some external funding, it's pretty normal to give them some sort of monetary compensation for participation. If we're using the Georgia Tech subject pool, our compensation will often be extra credit in a certain class, very often a psychology class. Note that the recruitment procedures are very important, especially if you're doing something like classroom research, where there can be a very significant perception of coercion to get people to participate. If I, as an instructor, am recruiting students in my own class to participate in my research project, I have to be very clear that it won't come back to haunt them if they choose not to participate.

IRB Protocols: Consent Procedures

III. Subject Information, Consent and Types of Studies

A Human Subject Interaction
Will the research involve direct interaction with human subjects?
Yes
If yes, [Click Here.](#)
If no, [Click Here.](#)

B Proposed Consent Procedures
[Specify Consent Procedures](#)

C Research Subject to the Health Insurance Portability & Accountability Act (HIPAA)
[Answer Research Subject to the Health Insurance Portability & Accountability Act \(HIPAA\)](#)

One of the most important elements of IRB approval is consent, that was one of the things created by the Belmont Report. If we're doing any interaction with our human subjects, we definitely need to think about consent procedures.

► Subject Consent Information

Consent Procedures:

directions: Check all proposed consent procedures.

Name	Description
<input checked="" type="checkbox"/> Written Consent Required	Signed consent will be sought from the subject or from the subject's legally authorized representative. Per 45CFR46.116 (d) an IRB may approve a consent procedure which does not include, or which alters, some or all of the elements of informed consent set forth in this section, or waive the requirements to obtain informed consent provided the IRB finds and documents that:
<input type="checkbox"/> Waiver of Consent	(1) the research involves no more than minimal risk to the subjects; (2) the waiver or alteration will not adversely affect the rights and welfare of the subjects; (3) the research could not practically be carried out without the waiver or alteration; and

On the consent information page, first, we need to indicate what kind of consent we'll receive. Most commonly, this will be written consent required. In this case, participants will sign or digitally sign, a consent form to start the study, but in some cases a waiver may be obtained.

https://gtapps.gatech.edu/irb/assentconsent.srv?submissionid=72854&popup_flag=Y

☐ Waiver of Consent

(1) the research involves no more than minimal risk to the subjects;

(2) the waiver or alteration will not adversely affect the rights and welfare of the subjects;

(3) the research could not practicably be carried out without the waiver or alteration; and

(4) whenever appropriate, the subjects will be provided with additional pertinent information after participation.

Per 45CFR46.117(c) an IRB may waive the requirement for the investigator to obtain a signed consent form for some or all subjects if it finds either:

(1) That the only record linking the subject and the research would be the consent document and the principal risk would be potential harm resulting from a breach of confidentiality. Each subject will be asked whether the subject wants documentation linking the subject with the research, and the subject's wishes will govern; or

(2) That the research presents no more than minimal risk of harm to subjects and involves no procedures for which written consent is normally required outside of the research context

☐ Waiver of Documentation of Consent

****Please note that this option requires a consent document without the signature section: In cases where the requirement of documentation is waived (e.g., use of an anonymous survey is proposed, telephone survey, or web-based survey), a consent document in

First, a waiver of consent can be obtained under certain, pretty narrow circumstances, this generally means we don't need to receive the subject's consent at all. Most of the time this only applies when subjects will not be directly affected by the research. So for example, if we wanted to study educational or health data that's already been generated and is already anonymized, we might receive a waiver of consent. Because those subjects won't be impacted by the fact we're now researching their data. Similarly, if we were to go sit in a coffee house and just take notes on the order taking process in a way that didn't identify anyone. We might receive a waiver of consent, because our observation is not affecting those people.

https://gtapps.gatech.edu/irb/assentconsent.srv?submissionid=72854&popup_flag=Y

subject will be asked whether the subject wants documentation linking the subject with the research, and the subject's wishes will govern; or

(2) That the research presents no more than minimal risk of harm to subjects and involves no procedures for which written consent is normally required outside of the research context

☐ Waiver of Documentation of Consent

****Please note that this option requires a consent document without the signature section: In cases where the requirement of documentation is waived (e.g., use of an anonymous survey is proposed, telephone survey, or web-based survey), a consent document in Georgia Institute of Technology IRB-required format must still be used. However, the document is written in letter format (Dear Subject) and, rather than requiring subject signature to verify consent, the following text is used to end the letter:

If you _____ (e.g., complete the attached survey, answer these few questions etc.), it means that you have read -- or have had read to you -- the information contained in this letter and would like to be a volunteer in this research study. Thank you,
(Signatures of Investigators)

Informed Consent

The IRB has standard template forms located on the GT [ORIA](#) website. Please consult these templates

We might also receive a waiver of documentation of consent. This occurs for low risk research, where the written consent itself, would be the only record of the participants identity. This applies to a lot of survey research or unrecorded interviews, where participants can be informed of their rights at the start, and their own continued participation constitutes continued implied consent. There's no reason to have them sign a consent form, because that consent form is the only reason we'd ever be able to identify them after the study.

https://gtapps.gatech.edu/irb/assocconsent.srv?submissionid=72854&popup_flag=Y

Informed Consent

The IRB has standard template forms located on the GT [ORIA](#) website. Please consult these templates when writing these consent forms.

A If a waiver is selected above, provide a justification.

No waiver is requested. [editor window](#)

B Summarize the plan for obtaining informed consent. State when consent will be obtained, by whom, and whether it will be written or oral. Explain how researchers will determine whether subjects have been provided sufficient information to make an informed decision, that they comprehend what they are being asked to do, and that their participation is truly voluntary.

Consent will be obtained at the beginning of the study. Participants will complete the consent form, which will inform them that they may exit the study at any time. [editor window](#)

After selecting our option, we need to provide a justification, if we requested a waiver. If we didn't, then no justification is necessary. We'll then describe the plan for obtaining informed consent. Generally, this will be to provide the consent form to participants at the start of a study, and make it very clear that they can withdraw from the study at any time.

https://gtapps.gatech.edu/irb/assocconsent.srv?submissionid=72854&popup_flag=Y

C If subjects are unable to give consent (e.g., children or mentally incompetent), describe how and by whom permission will be granted.

If children will be enrolled, parental permission will be required in almost all cases. For mentally incompetent participants or wards of the state, a Legally Authorized Representative (LAR) may give permission.

[editor window](#)

D If a waiver of parental permission is sought, provide justification here.

If Georgia Tech students are being enrolled, a waiver of parental permission may be appropriate for the (occasional) minor aged college student.

Consult the guidance at www.researchintegrity.gatech.edu. Click on INSTITUTIONAL REVIEW BOARD, then INFORMED CONSENT.

[editor window](#)

If we're involving children, non English speakers or other at risk populations in our study, there may be some additional boxes to complete.

https://gtapps.gatech.edu/irb/assococonsent.srv?submissionid=72854&popup_flag=Y

the person giving permission has the authority to provide permission for the children to participate?

[editor window](#)

G If applicable, how will researchers assess whether subjects have continuing capacity to provide informed consent? Describe how informed consent will be confirmed and documented throughout the study, not just at the initial consent.

Participants will be clearly informed at the beginning of the study that they may withdraw at any time; therefore, continued participation constitutes continued consent.

[editor window](#)

H If decisionally impaired adults are to be enrolled, describe the provision for obtaining surrogate consent from a legally authorized representative (LAR). State how researchers will ensure that the person giving consent for a potential subject has that authority.

[editor window](#)

It's also important for us to assess whether participants are continuing to consent to the study. Often times, we do this by making it very clear at the start of the study that they can withdraw at any time. So that their continued participation constitutes implied continued consent.

https://gtapps.gatech.edu/irb/assococonsent.srv?submissionid=72854&popup_flag=Y

J Is deception or concealment proposed?

A study proposing the use of adequately justified deception or concealment may qualify for a waiver of consent. Deception in a study occurs when subjects are intentionally told something untrue about the study, such as its real purpose. Concealment occurs when the researcher intentionally withholds some of the research details from subjects. Deception is not authorized in FDA-regulated studies. On the other hand, the HHS regulations at 45 CFR 46.116(d), allow deception or concealment only when a waiver of informed consent is justified.

If the study involves DECEPTION, the following language must appear in the procedures section of the consent documents: During the study, you may be led to believe some things that are not true. When the study is over, we will tell you everything. At that time you can decide whether to let us use your information. You have the right to then require that your information be destroyed and not be used in the study.

For studies proposing CONCEALMENT, the following language should appear in the procedures section of the consent documents: We will not tell you everything about the study in advance. When the study is over, we will tell you everything. At that time you can decide whether to let us use your information. You have the right to then require that your information be destroyed.

No

[editor window](#)

Finally, it's also possible to have protocols where deception or concealment is proposed. In HCI, for example, we might want to tell participants that an interface is functioning even if someone behind the scenes is actually just making it look functional, so that we get good research data out of those participants. For example, if we were testing something like a new version of Siri, we might tell participants that it's functioning, when in reality someone is writing the responses by hand. If we're using deception or concealment like that, we'll indicate so here.

<https://gtapps.gatech.edu/irb/assentconsent.srv?submissionid=72854&popup=1&page=Y>

if the study involves DECEPTION, the following language must appear in the procedures section of the consent documents: During the study, you may be led to believe some things that are not true. When the study is over, we will tell you everything. At that time you can decide whether to let us use your information. You have the right to then require that your information be destroyed and not be used in the study.

For studies proposing CONCEALMENT, the following language should appear in the procedures section of the consent documents: We will not tell you everything about the study in advance. When the study is over, we will tell you everything. At that time you can decide whether to let us use your information. You have the right to then require that your information be destroyed.

No

[editor window](#)

[Upload documents](#)

[Save & Stay Here](#) [Save & Continue with Application](#) [Cancel](#)

Then finally, we need to upload our consent forms.

researchintegrity.gatech.edu/about-irb/irb-informed-consent

IRB Informed Consent

IRB

- IRB Policies & Procedures
- Accounting for Payments to Human Subjects
- IRB Protocol & Submissions
- IRB Informed Consent
- IRB Required Training
- Phlebotomy Research Protocol for Student Research Participants
- Phlebotomy Protocol for Research Participants Who Are Not Ga Tech Students
- Policies for Reporting Violations

Please note: Due to the significant increase in the number of studies you should allow to three weeks for an initial review.

- Adult Consent Template (Word: revised June 2012)
- Assent Template for use with children under 18 years of age (Word: revised June 2012)
- CABI IRB Template (Word: revised June 2012)
- FDA Consent Guidance (Word: revised Feb 2012)
- Parental Permission Template (Word: revised Feb 2012)
- Waiver of Consent (Word)
- Waiver of Documentation of Consent (Word)
- Waiver of Parental Permission (Word)

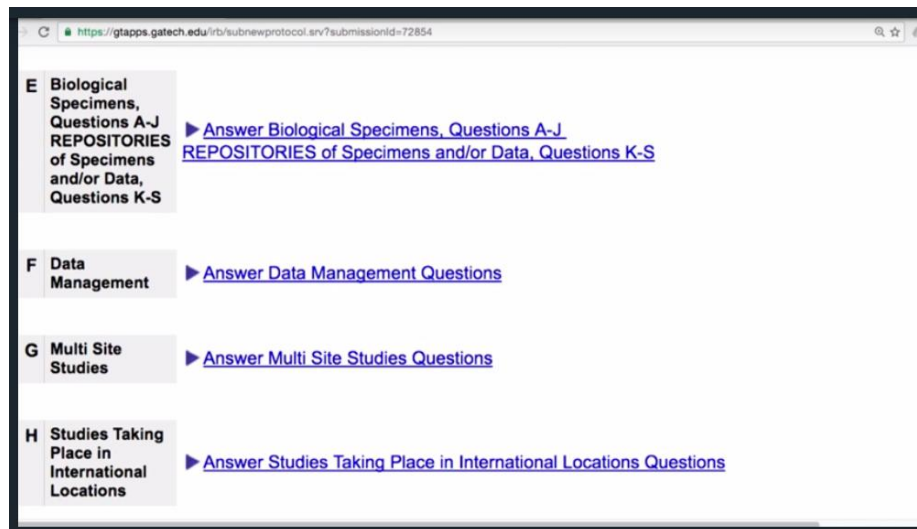
Consent and the Consent Process

General Issues in Informed Consent

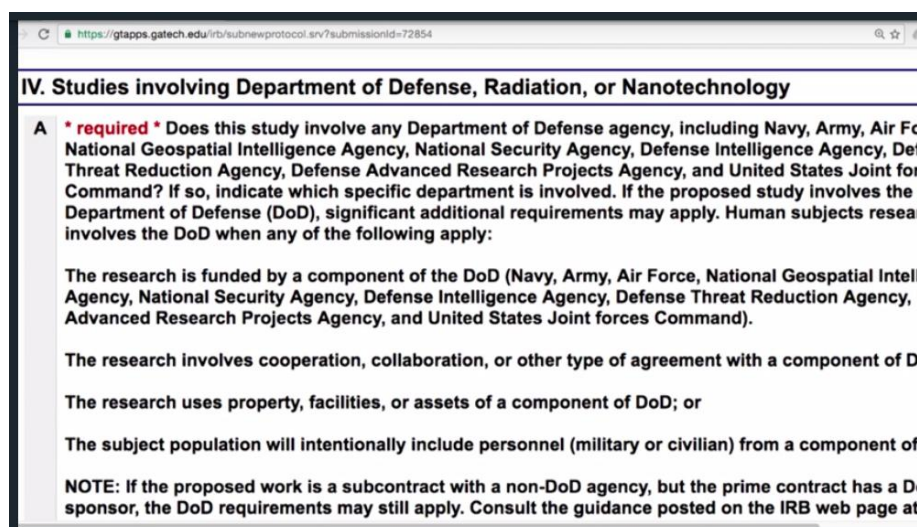
Control IRB Committee Members

At Georgia Tech the Office of Research Integrity Assurance provides consent form templates that we can tweak to match the specific needs of our study. The templates provide in depth directions on what to supply. Generally, this is where we disclose to participants the details of the rest of the protocol. What we're researching, why, how, and why they were invited to participate?

IRB Protocols: Wrapping Up

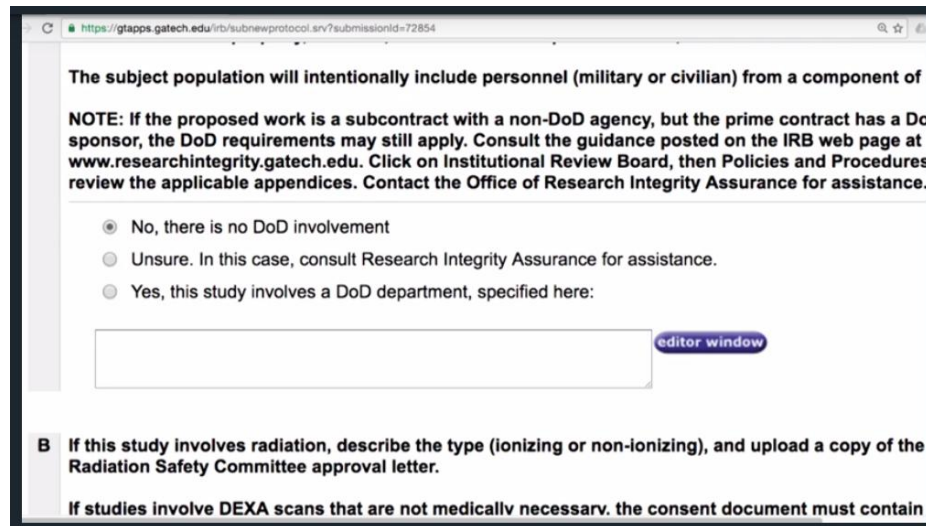


Of the remaining fields the only one we're likely interested in is the data management questions. The majority of the others cover either clinical research or biological research or other things that we hopefully won't touch on very much in human computer interaction. Unless the field has changed a lot by the time you're listening to this. Nonetheless though you should actually peek into the items to make sure that they don't apply to you if you're filling out a protocol like this. Under the Data Management section, we'll want to describe how we keep participants data safe. That'll include descriptions of the way we store it and define information about participants. And how we'll safeguard the data itself through password protection or encryption or anything like that.



Finally, there are always some questions that we answer for all studies even though they generally won't apply to us. Generally our studies won't involve the Department of Defense, generally they

shouldn't involve Radiation. And one day I really kind of hope they involve Nanotechnology but we're probably not quite there yet.



The subject population will intentionally include personnel (military or civilian) from a component of

NOTE: If the proposed work is a subcontract with a non-DoD agency, but the prime contract has a DoD sponsor, the DoD requirements may still apply. Consult the guidance posted on the IRB web page at www.researchintegrity.gatech.edu. Click on Institutional Review Board, then Policies and Procedures review the applicable appendices. Contact the Office of Research Integrity Assurance for assistance.

☒ No, there is no DoD involvement

☐ Unsure. In this case, consult Research Integrity Assurance for assistance.

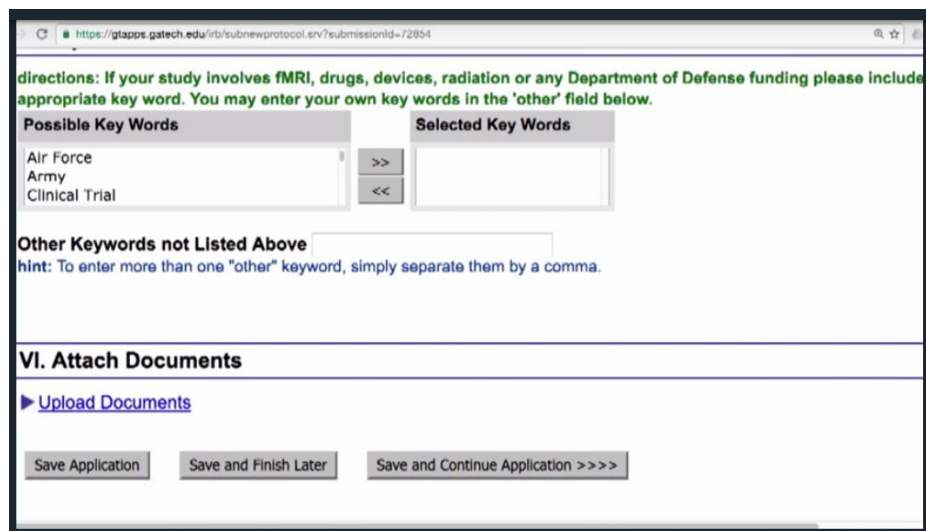
☐ Yes, this study involves a DoD department, specified here:

[editor window](#)

B If this study involves radiation, describe the type (ionizing or non-ionizing), and upload a copy of the Radiation Safety Committee approval letter.

If studies involve DEXA scans that are not medically necessary, the consent document must contain

So we'd mark no that there is no DoD involvement.



directions: If your study involves fMRI, drugs, devices, radiation or any Department of Defense funding please include appropriate key word. You may enter your own key words in the 'other' field below.

Possible Key Words

Air Force
Army
Clinical Trial

Selected Key Words

>>
<<

Other Keywords not Listed Above

hint: To enter more than one "other" keyword, simply separate them by a comma.

VI. Attach Documents

[Upload Documents](#)

[Save Application](#) [Save and Finish Later](#) [Save and Continue Application >>>>](#)

Finally, at the bottom, there's a place to upload some additional documents. This is where we would supply things like an interview script, a survey, a recruitment script and other documents that the IRB would want to see and approve. When we're done, we can click Save and Continue Application.

https://gtapps.gatech.edu/irb/newreviewpage.srv?submissionid=72854

Conflict of Interest

Conflict of Interest

A Have you (PRINCIPAL INVESTIGATOR), or will you, your spouse, domestic partner, or minor dependents:

Receive compensation from a company/entity including salary consulting fees or honoraria related to this research (do not include salary, grant support, and other payments for services from Georgia Tech)?

Receive royalty or licensing payments from a company/entity related to this research?

Have any intellectual property rights or royalties from such rights whose value may be affected by the outcome of this research, including royalties under any royalty-sharing agreements involving the University?

Receive gifts/benefits, including reimbursed or sponsored travel, from a company/entity related to this research?

Have equity or ownership interest (includes stock options) in a public or private company/entity related to this research?

On the next page, we can preview everything on one flat screen, and then check off at the end that we have no conflicts of interest, or report them, if we do.

https://gtapps.gatech.edu/irb/newreviewpage.srv?submissionid=72854

☐ Yes

editor window

File Uploaded:

upload file

<< Edit Application Save & Stay Here Save & Continue >>

TOP

Visit the [Georgia Tech IRB Website](#)

Page generated on September 1, 2016 12:08 PM
 RBWise v.2.3.7 (0003494)
 © 2004-2006 IRB Solutions, Inc., Portions Copyright 2000-2004 Georgia Tech Research Corporation ALL RIGHTS RESERVED

Then we'll click Save & Continue again.

https://gtapps.gatech.edu/irb/newprotocolsubmit.srv?submissionid=72854

Submit Protocol

● **ATTENTION PRINCIPAL INVESTIGATORS:** You must forward your application for sign off. Please select the name of your Department Head/Chair from the drop down list and add them as a recipient and then choose "Submit Protocol" at the bottom of the screen. IRB Wise will send an email to that person requesting their sign off. The Department Head/Chair will then be responsible for forwarding your application on the IRB. Please do not submit your application directly to the IRB. (mandatory unless you are the department head)

Recipient	Order
-----------	-------

Add Recipient:

or

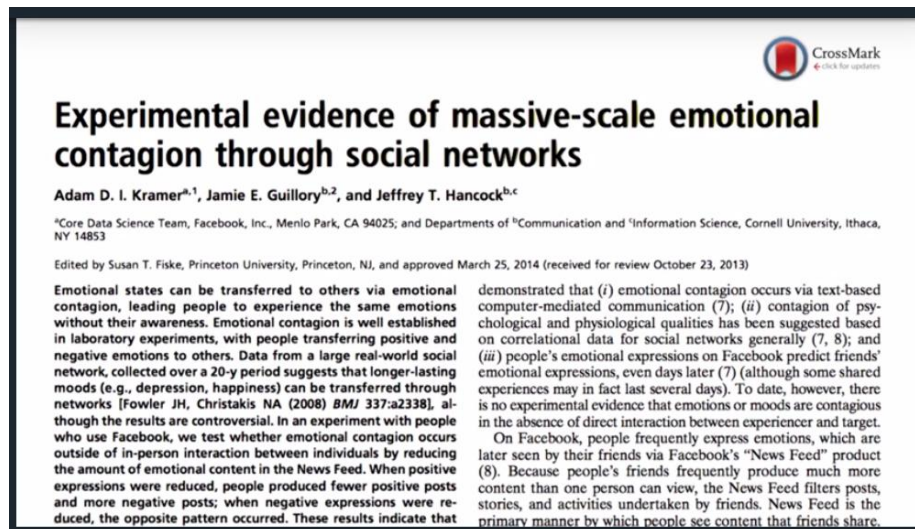
<search results> 1

<< Add Recipient

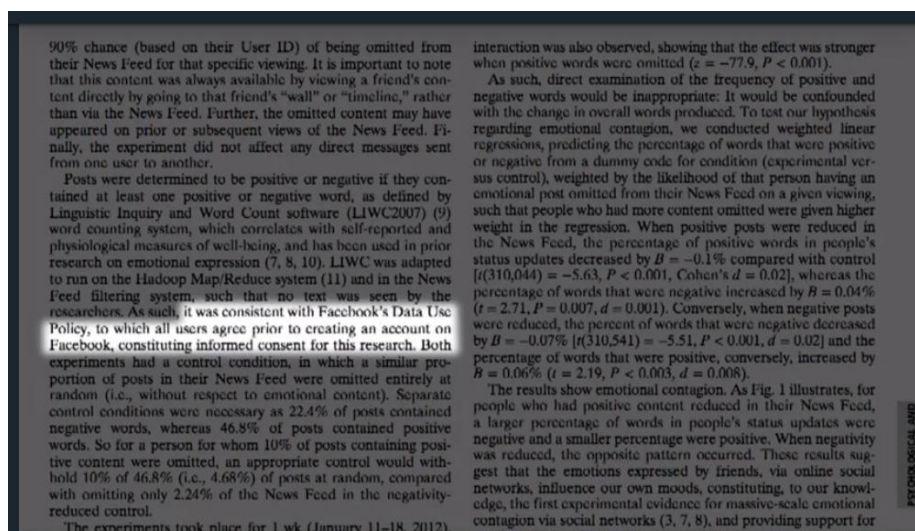
● **Submit the protocol directly to the IRB (department heads only)**

And for y'all, you would then submit it to your primary investigator. I am the primary investigator so I see something a little bit different here than what you would see. After submission, we'll generally hear back from IRB in about three weeks about whether the study was accepted and what changes need to be made if not.

Research Ethics and Industry

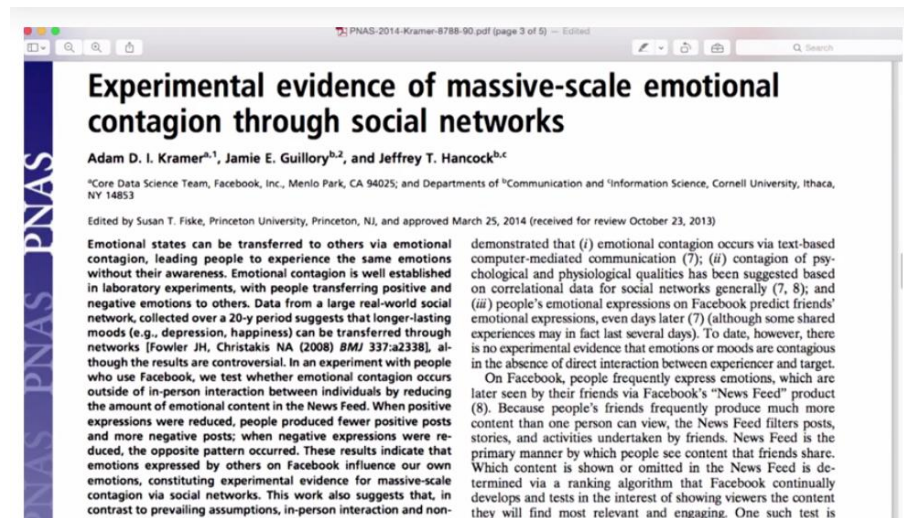


Institutional review boards govern any research institutions that receive support from the federal government. But what about research that doesn't receive any federal support? Very often, companies will do research on their users. This is especially common in HCI. Lots of companies are constantly doing very interesting testing on their users with a lot of rapid AB experiments. There's a lot of potential knowledge there, but at the same time much of what they do likely would not pass IRB if it were university research. This actually came up recently with Facebook in a paper they published titled experimental evidence of massive-scale emotional contagion through social networks. Basically, Facebook wanted to see if they could predict what would make users happy or sad. And as a result they tweaked the news feed for some users to test out their ideas. In other words, they tried to manipulate their user's mood for experimental purposes.




Now, Facebook argues that this was consistent with their own data use policy, which permits them to perform experiments like this. Some social scientists however would argue that this does not constitute informed consent. Informed consent they say is specific to a certain experiment, temporary for a known period of time and given without coercion. Some would argue that if you don't agree you can't use Facebook qualifies as coercion. These are some difficult issues and if you end up working in HCI industry, you'll likely find yourself wrestling with some of them.

Quiz: Exercise: Research Ethics and Industry



People are still discussing whether or not Facebook's study on its impact on users' moods was ethical. Facebook maintains that the study was consistent with its own data use policy, which constitutes informed consent. Opponents argue that it doesn't. What do you think? If you think that this was ethical, why do you think it was ethical? If you think that it was unethical, what could've made it ethical?



Was Facebook's experiment ethical?

☒ Yes, because... the terms of service covered it.

☐ No, because...

If you said yes, there are several reasons you might have stated. You might agree that because the terms of service covered it, it was technically ethical research. The users did agree to things like this.



Was Facebook's experiment ethical?

- ☒ Yes, because...
- ☐ No, because...

You may have actually read the article or read other publications about it and noted that Facebook actually has an internal IRB that reviews things like this.



Was Facebook's experiment ethical?

- ☒ Yes, because...
- ☐ No, because...

And in this case, an external IRB did review the study.



Was Facebook's experiment ethical?

- ☐ Yes, because...
- ☒ No, because...

If you said no, the reason you gave may have been that we know users are not aware of what's in terms of use. We have plenty of studies that indicate that users really don't spend any time reading what they're agreeing to. And while technically, it's true that they're still agreeing to it, what we're interested in here are participants' rights. If we know that users aren't reading what they're agreeing to, don't we have an ethical obligation to make sure they're aware before we go ahead with it.



Was Facebook's experiment ethical?

☐ Yes, because...

☒ No, because...

users cannot opt out.

We also might say no because users couldn't opt out of this study. Part of that is because opting out of the study alone means deactivating your entire Facebook account or just stopping using the tool. But part of it is that users also weren't aware that a study was now going on. They couldn't opt out of the study specifically, nor could they even opt out of it by closing down their entire Facebook account because they didn't know when the study had started.



Was Facebook's experiment ethical?

☐ Yes, because...

☒ No, because...

users weren't notified.

That ties into the other issue. Users weren't notified that they were participants in an experiment. So even though they technically agreed to it when they agreed to Facebook's terms of service, one could argue the fact they weren't notified when the study was beginning and ending means that it wasn't ethical research. I'm not going to give you a right or wrong answer to this. There's a very interesting conversation to have about this. But what's most important here are the interesting questions that it brings up. Especially in regard to companies doing human subjects research that doesn't have any oversight from the federal government. If you agreed with these reasons why it wasn't ethical, what could they have done to fix it? Maybe they could have separated out the consent process for research studies from the rest of Facebook as a whole. Maybe they could have specifically requested that individual users opt-in, and alert them when the study was done, but not tell them what's actually being manipulated. And even if the original study was ethical, there were likely things that could have reduced the backlash. At the same time, those things might have affected the results. These are the tradeoffs that we deal with.

Paper Spotlight: "Evolving the IRB: Building Robust Review for Industry Research"

Evolving the IRB: Building Robust Review for Industry Research

Molly Jackman and Lauri Kanerva*

Abstract

Increasingly, companies are conducting research so that they can make informed decisions about what products to build and what features to change. These data-driven insights enable companies to make responsible decisions that will improve peoples' experiences with their products. Importantly, companies must also be responsible in how they conduct research. Existing ethical guidelines for research do not always robustly address the considerations that industry researchers face. For this reason, companies should develop principles and practices around research that are appropriate to the environments in which they operate, taking into account the values set out in law and ethics. This paper describes the research review process designed and implemented at Facebook, including the training employees receive, and the steps involved in evaluating proposed

In a recent paper in the Washington and Lee Law Review, Molly Jackman and Lauri Kanerva, two Facebook employees, explore exactly this issue.

Abstract

Increasingly, companies are conducting research so that they can make informed decisions about what products to build and what features to change. These data-driven insights enable companies to make responsible decisions that will improve peoples' experiences with their products. Importantly, companies must also be responsible in how they conduct research. Existing ethical guidelines for research do not always robustly address the considerations that industry researchers face. For this reason, companies should develop principles and practices around research that are appropriate to the environments in which they operate, taking into account the values set out in law and ethics. This paper describes the research review process designed and implemented at Facebook, including the training employees receive, and the steps involved in evaluating proposed research. We emphasize that there is no one-size-fits-all model of research review that can be applied across companies, and that processes should be designed to fit the contexts in which the research is taking place. However, we hope that general principles can be extracted from Facebook's process that will inform other companies as they develop frameworks for research review that serve their needs.

Jackman and Kanerva specifically note that the ethical guidelines developed in the context of academia do not always address some of the considerations of industry.

Abstract

Increasingly, companies are conducting research so that they can make informed decisions about what products to build and what features to change. These data-driven insights enable companies to make responsible decisions that will improve peoples' experiences with their products. Importantly, companies must also be responsible in how they conduct research. Existing ethical guidelines for research do not always robustly address the considerations that industry researchers face. For this reason, companies should develop principles and practices around research that are appropriate to the environments in which they operate, taking into account the values set out in law and ethics. This paper describes the research review process designed and implemented at Facebook, including the training employees receive, and the steps involved in evaluating proposed research. We emphasize that there is no one-size-fits-all model of research review that can be applied across companies, and that processes should be designed to fit the contexts in which the research is taking place. However, we hope that general principles can be extracted from Facebook's process that will inform other companies as they develop frameworks for research review that serve their needs.

In response, the authors directly advocate for setting up a set of principles and practices for industry environments. In other words, rather than just ignoring the parts that aren't relevant for industry, the authors advise creating a new set of standards specifically for industry.

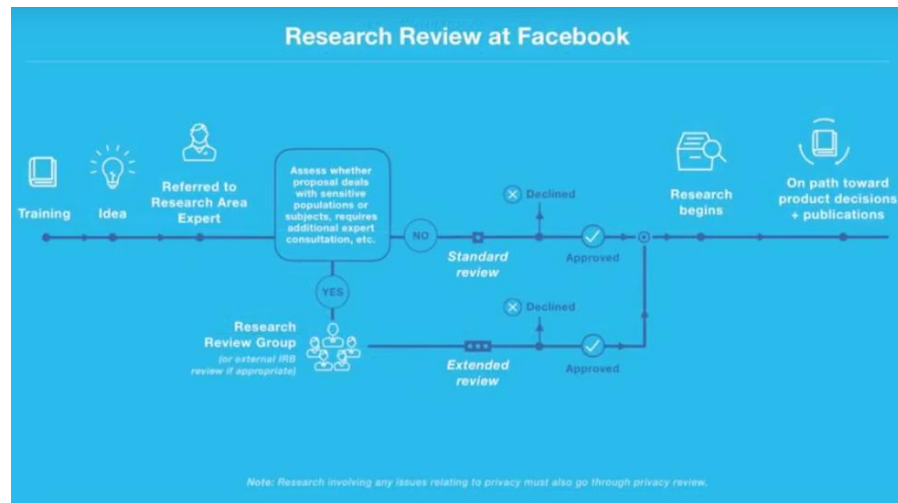
IV. Designing a Process

We designed our process to leverage the structure that already exists at Facebook, creating multiple training opportunities and research review checkpoints in the organizational flow. Figure 1 summarizes this process.

Figure 1: Research Review at Facebook



To do so, Facebook designed its own internal review process. In this case, Facebook's process is heavily reliant on deferring to a research area expert. And I don't say defer in a bad way. A key part of IRB is experiments are reviewed by people with no incentive to permit the study if it isn't ethical.



Facebook tries to replicate this by referring studies to an external reviewer who, in turn, decides whether or not an additional review and even external IRB is necessary. The other thing that's important to note though, is that Facebook isn't under any strong obligation to do this. Universities that receive federal funding are governed by the Belmont Report. But companies are not yet governed by any similar law. So we rely on companies to govern themselves. In Facebook's case, it seems to be going pretty well. But you might find yourself at a company that doesn't have such a program, and you'll have to apply these standards yourself.

Conclusion to Research Ethics



In this lesson, we've talked about research ethics. Research ethics guide the human subject's research we do to make sure we're respecting the rights of our participants. But it also make sure the data we're gathering is good and useful. At every stage of our design life cycle, we want to keep respect for our participants at the forefront of our thought. That means being wary of experimenting in ways that might negatively affect users. That also means only asking users to dedicate their time to evaluating interfaces that are well thought out. And that means respecting users' viewpoints and position in the design process.

3.3 Needfinding and Requirements Gathering

Compiled by Shipra De, Summer 2017

Introduction to Needfinding



[MUSIC] The first stage of the design life cycle is needfinding, or requirements gathering. This is the stage where you go and try to find out what the user really needs. The biggest mistake that the designer can make is jumping to the design process before understanding the user or the task. We want to develop a deep understanding of the tasks they're trying to accomplish and why. As we do this, it's important to try to come in with as few preconceived notions as possible.

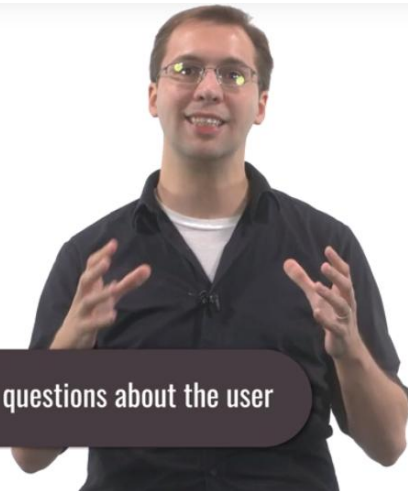
If all you have is a hammer,
everything looks like a nail.



There's an old adage that says when all you have is a hammer, everything looks like a nail. This is similar. If you come in having already decided what approach you want to take it's tempting to only see the problem in terms of the approach you've chosen. So we're going to go through a process that attempts to avoid as many preconceived notions as possible.



Defining questions about the user



We're going to start by defining some general questions we want to answer throughout the data gathering process about who the user is, what they're doing, and what they need.

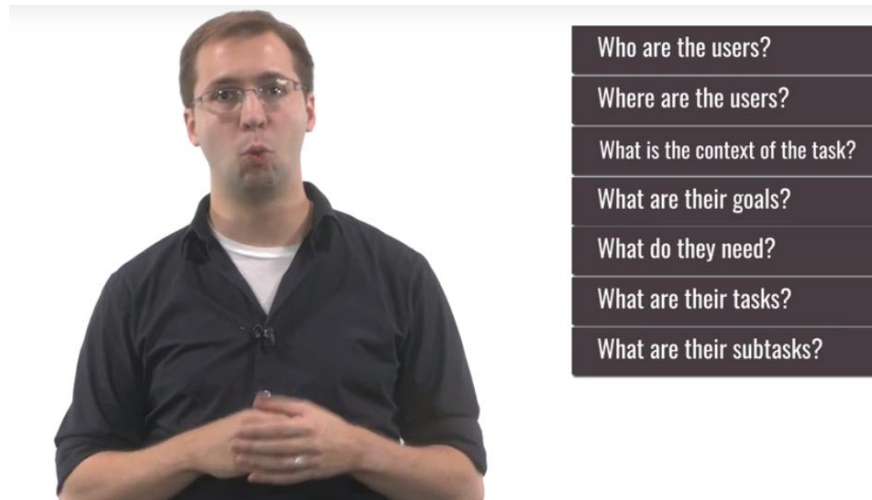


Then we'll go through several methods of generating answers to those questions to gain a better understanding of the user.



Then we'll talk about how to formalize the data we gather into a shareable model of the task and a list of requirements for our ultimate interface. Note that each of these tools could get a lesson on its own on how to do it, so we'll try to provide some additional resources to read further on the tools you choose to use.

Data Inventory



Before we start our need-finding exercises, we also want to enter with some understanding of the data we want to gather. These are the questions we ultimately want to answer. That's not to say we should be answering them every step of the way, but rather, we want to gather the data necessary to come to a conclusion at the end. Now, there are lots of inventories of the types of data you could gather, but here's one useful list. One, who are the users? What are their ages, genders, levels of expertise? Two, where are the users? What is their environment? Number three, what is the context of the task? What else is competing for users' attention? Four, what are their goals? What are they trying to accomplish? Five, right now, what do they need? What are the physical objects? What information do they need? What collaborators do they need? Six, what are their tasks? What are they doing physically, cognitively, socially? And seven, what are the subtasks? How do they accomplish those subtasks? When you're designing your need finding methods, each thing you do should match up with one or more of these questions.

The Problem Space



In order to do some real need finding, the first thing we need to do is identify the problem space. Where is the task occurring, what else is going on, what are the user's explicit and implicit needs? We'll talk about some of the methods for doing that in this lesson, but before we get into those methods, we want to understand the scope of the space we're looking at.





So consider the difference between these two actions. [MUSIC] Notice that in each of these, I'm doing the same task, turning off the alarm. But in the first scene we're focusing very narrowly on the interaction between the user and the interface. In the latter, we're taking into consideration a broader view of the problem space. We could zoom out even further if we wanted to and ask questions about Where and why people need alarm systems in the first place. That might lead us to designing things like security systems for dorm rooms or checking systems for office buildings. As we're going about need finding, we want to make sure we're taking the broad approach. Understanding the entire problem space in which we're interested, not just focusing narrowly on the user's interaction with a particular interface. So in our exploration of methods for need finding, we're going to start with the most authentic types of general observation, then move through progressively more targeted types of need finding.

User Types

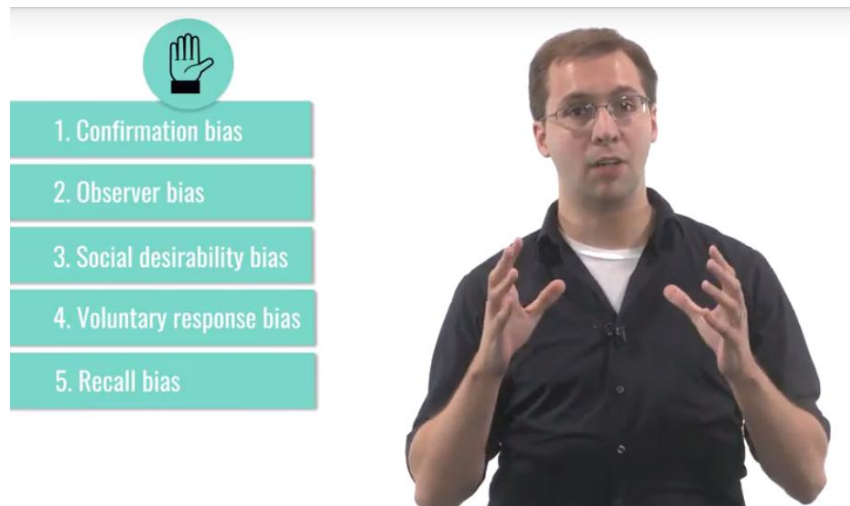


Just as we want to get an idea of the physical space of the problem, we also want to get an idea of the space of the user. In other words, we want to understand who we're designing for. That comes up a lot when doing design alternatives and prototyping, but we also want to make sure to gather information about the full range of users for whom we're designing. So let's take the example of designing an audiobook app for people that exercise. Am I interested in audiobooks just for kids or for adults too? Am I interested in experts who are exercising or novices at it? Am I interested in experts at listening to audiobooks or am I interested in novices at that as well? Those are pretty key questions. They differentiate whether I'm designing for business people who want to be able to exercise while reading, or exercisers who want to be able to do something else while exercising. The task is similar for both but the audience, their motivations and their needs are different. So, I need to identify these different types of users and to read funny exercises on all of them. One of the most successful products of all time succeeded because of the attention to user types. The Sony Walkman became such a dramatic success because they identified different needs for different types of people, design their product in a way that it met all those needs, but then they market it specifically to those different types of individuals.



You can read more about that in a book called *Doing Cultural Studies* by Hugh Mackay and Linda Janes.

5 Tips: Avoiding Bias in Needfinding



During needfinding, there are some significant considerations that need to be made to avoid biasing your results. Let's go through five of these possible biases. Number one, confirmation bias, confirmation bias is the phenomenon where we see what we want to see. We enter with some preconceived ideas of what we'll see, and we only notice the things that confirm our prior beliefs. Try to avoid this by specifically looking for signs that you're wrong, by testing your beliefs empirically, and by involving multiple individuals in the need finding process. Number two, observer bias, when we're interacting directly with users, we may subconsciously bias them. It might be more helpful, for example, with users using the interface that we designed compared to the ones that other people designed. On surveys, we might accidentally phrase questions in a way that elicits the answers that we want to hear. Try to avoid this by separating experimenters with motives from the participants. By heavily scripting interactions with users, and by having someone else review your interview scripts and your surveys for leading questions. Number three, social desirability bias, people tend to be nice. People want to help. If you're testing an interface and the participants know that you're the designer of the interface, they'll want to say something nice about it to make you happy. But that gets in the way of getting good data. Try to avoid this by hiding what the socially desirable response is by conducting more naturalistic observations and by recording objective data. Number four, voluntary response bias, studies have shown that people with stronger opinions are more likely to respond to optional surveys. You can see this often in online store reviews. The most common responses are often fives and ones. For us, that means if we perform quantitative analysis on surveys, we risk over sampling the more extreme views. Avoid this by limiting how much of the survey content is shown to users before they begin the survey and by confirming any conclusions with other methods. Number five, recall bias, studies have also shown that people aren't always very good at recalling what they did, what they thought, or how they felt during an activity they completed in the past. That can lead to misleading and incorrect data. Try to avoid this by setting tasks in contexts by having users think out loud during activities or conducting interviews during the activity itself. Now these biases can get a larger control also by making sure to engage in multiple forms of need finding.

Naturalistic Observation



For certain tasks, a great way for us to understand the users need is to simply watch. A great way for me to start understanding what it's like to need an audiobook app for exercising is to come somewhere where people are exercising and just watch them exercise. This is called naturalistic observation, observing people in their natural context. So I'm fortunate that I actually live across the street from a park, so I can sit here in my rocking chair on my porch and just watch people exercising. Now, I want to start with very specific observations and then generalize out to more abstract tasks. That way I'll observe something called confirmation bias which is basically when you see what you want to see, so what do I notice? Well, I notice that there's a lot of different types of exercisers. There are walkers, joggers, runners I see some rollerbladers, I see some people doing yoga. I see a lot of people riding bikes but the bikers seem to be broken into two different kinds of groups. I see a lot of people biking kind of leisurely but I also see some bikers who are a little bit more strenuous about it. I'm also noticing that while joggers might be able to stop and start pretty quickly, that's harder for someone riding a bike. So I might want to avoid designs that force the user to pull out their phone a lot because that's going to be dangerous and awkward for people riding bikes. Now I also see people exercising in groups and also people exercising individually. For those people exercising in groups, I don't actually know if they'd be interested in this. Listening to something might kind of defeat the purpose of exercising together. So I'm going to have to note that down as a question I want to ask people later. I also see that many people tend to stretch before and after exercising and I'm wondering if we can use that. Then we can have some kind of starting and ending sequence for this, so that a single session is kind of book ending by both stretching, and interacting with our app. Note that by just watching people engage in the task of exercising, I'm gathering an enormous amount of information that might affect my design. But note also, that while naturalistic observation is great, I'm limited ethically in what I can do. I can't interact with users directly and I can't capture identifying information like videos and photographs that's why I can't show you what I'm seeing out here. I'm also limited in that I don't know anything about what those users are thinking. I don't know if the people working out in groups would want to be able to listen to audiobooks while they're doing yoga. I don't know if bluetooth headsets

would be problematic for people riding bike, I need to do a lot more before I get to the design phase. But this has been very informative in my understanding of the problem space and giving me things I can ask people later on.

5 Tips: Naturalistic Observation



Here are five quick tips for doing naturalistic observation. Number one, take notes. Don't just sit around watching for a while. Be prepared to get a targeted information and observations about what you see. Number two, start specific and then abstract. Write down the individual little actions you see people doing before trying to interpret or summarize them. If you jump to summarizing too soon, you risk tunnel vision. Number three, spread out your sessions. Rather than sitting somewhere for two hours one day and then moving on, try to observe in shorter 10 to 15 minute sessions, several times. You may find interesting different information, and your growing understanding and reflection on past exercises will inform your future sessions. Number four, find a partner. Observe together with someone else. Take your own notes and then compare them later so you can see if you all interpreted the same scenarios or actions in the same way. Number five, look for questions. Naturalistic observations should inform the questions you decide to ask participants in more targeted need-finding exercises. You don't need to have all the answers based on observation alone. What you need is questions to investigate further.

Participant Observation



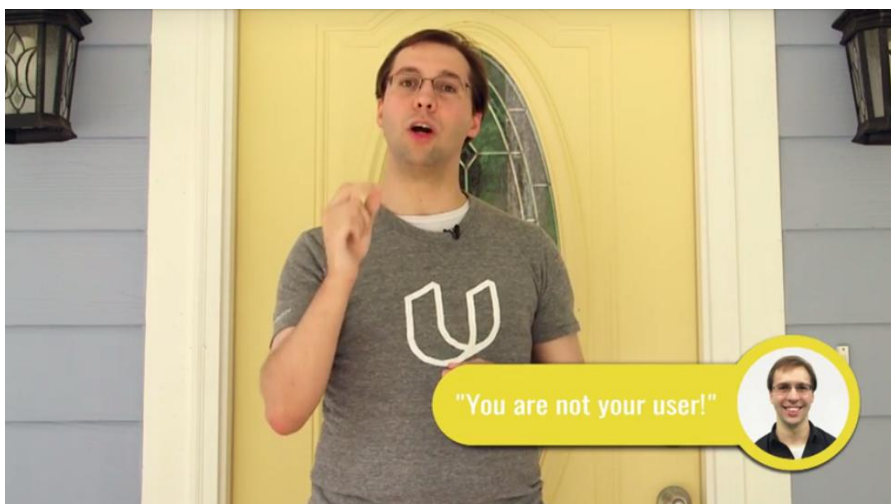
Sometimes it's not just enough to watch people engaging in a task. Sometimes we want to experience a task for ourselves. So that's what I'm going to do. I listen to audiobooks a lot. I don't really exercise. I should, but I don't. But I'm going to try this out. So I've got my audiobook queued up, I've got my mic on so I can take notes as I run. So I'm going to go on a jog and see what I discover.



So what did I learn?



I learned that I'm out of shape for one thing. I already knew that but I learned it again. I also learned that this app would be very useful for anyone doing participant observation on exercisers. Because I kept having to stop to record notes for myself, which I could have done with this app that I'm trying to implement. But aside from that, I noticed that unexpected things happen pretty often that made me wish that I could easily go back in my book. Or sometimes there are just things I just wanted to hear again, but there was no easy way to do that. I also notice that there's definitely the need there for me. I already planned to listen to everything again now that I'm home because there were notes I wanted to take that I couldn't take easily. I also noticed that while sometimes I wanted to take notes, sometimes I also just want to leave a bookmark.



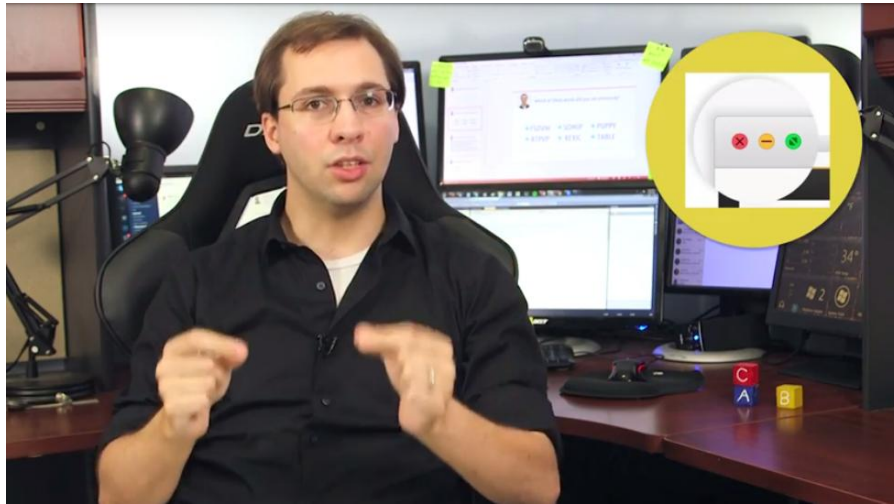
Now we do have to be careful here though. Remember you are not your user. When you're working as a participant observer, you can avail useful insights, but you shouldn't over represent your own experiences. You should use this experience as a participant observer to inform what you ask users going forward

Hacks and Workarounds



Let's zoom in a little bit more on what the user actually does or we can do naturalistic and participant observation without having to directly interact much with our users. We need to get inside users heads a little more to understand what they're thinking and doing. If you're trying to design interfaces to make existing tasks easier, one way to research that is to look at the hacks that users presently employ. How do they use interfaces in non-intended ways to accomplish tasks or how do they break out of the interface to accomplish a task that could have been accomplished with an interface? If you're designing a task meant to be performed at a desk like this, looking at the person's workspace can be a great way of accomplishing this. So for example, I have six monitors around. And yet, you still see Post-It notes on my computer. How could I possibly need more screen real estate? Well, Post-It notes can't be covered up. They don't take away from the existing screen real estate. They're visible even when the computer is off. So, this implicit notes here is the way to hack around the limitations of the computer interface. Now when you're looking at hacks, it's important to not just look at what the user does and assume you understand why. Look at their work around and ask them why they're using them. Find out why they don't just use the interface that's currently in place. You might find they just don't know about them, which presents a different kind of design challenge. Now hacks are related to another method we can use to uncover user needs as well, which are called errors. Whereas hacks are ways users get around the interface to accomplish their tasks, errors are slips or mistakes that users frequently make while performing the task within the interface.

Errors



When we're trying to make iterative improvements, one of the best places we can look is at the errors users make with the tools they currently have available. We can fix those errors, but we can also use those errors to understand a bit more about the user's mental model. So, here's a common example of an error, for me, which is a slip. I keep my email open on this window over here. I'll frequently forget that it's my active window while trying to type into a window over here. As a result, I'll hit a bunch of hotkeys in my email interface. I'll tag random emails, delete random emails. It's just kind of a mess. This is a slip because there's nothing wrong with my mental model of how this works. I understand that there's an active window and it's not selected. The problem is that I can easily forget which window is active. Mistakes on the other hand, are places where my mental model is weak and for me a place where that happens when I'm using my Mac. I'm used to a PC, where the maximize button always makes a window take up the entire screen. I've actually never fully understood the maximize button on a Mac. Sometimes it seems to work like a PC maximize button. Sometimes it just expands the window a bit, but not to the entire screen. Sometimes it enters even like a full screen mode, hiding the top task bar. I make mistakes there because I don't have a strong mental model of how it works. So, if you were watching me, you could see me making these errors, and you could ask me why I'm making them. Why did I choose to do that if that was my goal. That works for both discovering hacks and discovering errors: watch people performing their tasks, and ask them about why certain things happen the way that they do.



Discovering hacks and errors involves a little bit more user interaction than just watching people out in the wild. So how might we do that if we're doing something like creating an app that people are going to use in public? Well maybe we actually go up to people we see exercising out in public. We can actually get approval to do that. But that's going to be a little bit awkward, and the data we get might not be great. So at this point, we might be better off recruiting people to come in and describe their experiences. People experience hacks and errors pretty consciously, so our best bet would likely be to target exercise groups or local areas where exercisers frequent and recruit people to come in for a short study. Or, maybe we could recruit people to participate in a study during their normal exercise routine, taking notes on their experience or talking us through their thought process. We could actually take that to an extreme and actually adopt something like an apprenticeship approach, where we train to become users.

Apprenticeship and Ethnography



If we're designing interfaces for particularly complex tasks, we might quickly find out that just talking to our participants or observing them really isn't enough to get the understanding we need to design those interfaces. For particularly complex tasks, we might need to become experts ourselves in order to design those programs. This is informed by the domain of ethnography, which recommends researching a community or a job or anything like that, by becoming a participant in it. It goes beyond just participant observation though, it's really about integrating oneself into that area and becoming an expert in it and learning about it as you go. So we bring in our expertise and design in HCI and use that combined with the expertise that we develop to create new interfaces for those people. So for example, our video editors here at Udacity have an incredible incredibly complex workflow involving multiple programs, multiple workflows, lots of different people and lots of moving parts. There's no possible way I could ever sit down with someone for just an hour and get a good enough picture of what they do, to design a new interface that will help them out, I really need to train under them. I really need to become an expert at video editing and recording myself, in order to help them out. It's kind of like an apprenticeship approach. They would apprentice me in their field and I would use the knowledge that I gain to design new interfaces to help them out. So ethnography and apprenticeship are huge fields of research both on their own and as they apply to HCI. So if you're interested in using that approach take a look at some of the resources that we're providing.

Interviews with Focus Groups



A most targeted way of gather information from users though is just to talk to them. One way of doing that might be to bring them in for an interview. So I'm sitting here with Morgan, who's one of the potential users for our audio book app targeted at exercisers. And we're especially interested in the kinds of task you perform while exercising and listening to audio books at the same time. So to start, what kind of challenges do you run into doing these two things at once?

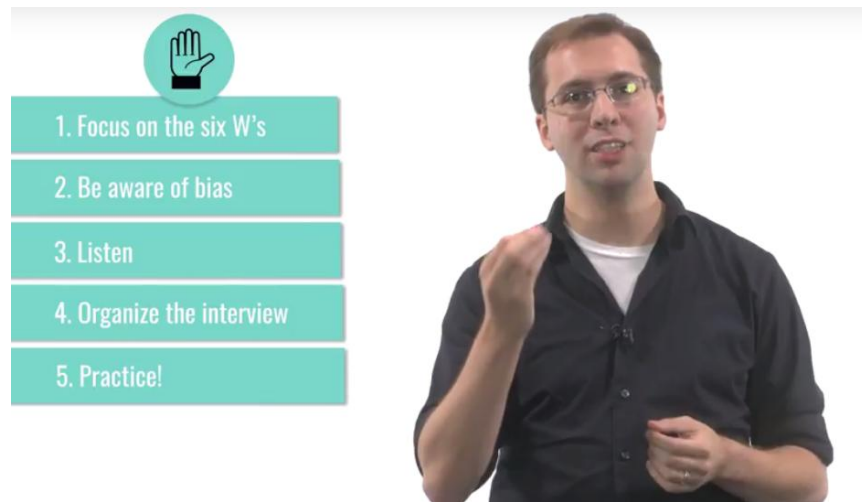
>> I think the biggest challenge is that it's hard to control it. I have headphones that have a button on them that can pause it and play. But if I want to do anything else I have to stop, pull up my phone and unlock it just to rewind.

Yeah, that makes sense. Thank you. Interviews are useful ways to get at with the user is thinking when they're engaging in a task. You can do interviews one on one like this or you can even do interviews in a group with multiple users at the same time.



Those tend to take on the form of focus groups, where a number of people are all talking together about some topic, and you can use them to tease out different kinds of information. Focus groups can elicit some information we don't get from this kind of an interview, but they also present the risk of overly convergent thinking. People tend to kind of agree with other instead of bringing in new ideas. So they should really be used in conjunction with interviews, as well as other need finding techniques.

5 Tips: Interviews



Here are five quick tips for conducting effective interviews. Now we recommend reading more about this before you actually start interviewing people. But these should get you started. Number one. Focus on the six W's when you're writing your questions. Who. What. Where. When. Why and how. Try to avoid questions that lend themselves to one world or only yes or no answers, those are better gathered via surveys. Use your interview questions to ask open-ended, semi-structured questions. Number two. Be aware of bias. Look at how you're phrasing your questions and interactions, and make sure you're not predisposing the participant to certain views. If you only smile when they say what you want them to say, for example you risk biasing them to agree with you. Number three. Listen. Many novice interviewers get caught up in having a conversation with a participant rather than gathering data from the participant. Make sure the participant is doing the vast majority of the talking. And don't reveal anything that might predispose them to agree with you. Number four. Organize the interview. Make sure to have an introduction phase. Some lighter questions to build trust and a summary at the end so the user understands the purpose of the questions. Be ready to push the interview forward or pull it back on track. Number five. Practice. Practice your questions on friends, family, or research partners in advance. Rehearse the entire interview, gathering subjects is tough so when you actually have them you want to make sure to get the most out of them.

Quiz: Exercise: interviews



Which of these would make good interview questions?

- ☒ Do you exercise?
- ☒ How often do you exercise?
- ☒ Do you exercise for health or for pleasure?
- ☒ What, if anything, do you listen to while exercising?
- ☒ What device do you use to listen to something while exercising?
- ☒ We're developing an app for listening to audiobooks while exercising. Would that be interesting to you?

Interviews are likely to be one of the most common ways you gather data. So let's run through some good and bad interview questions real quick. So here are six questions. Which of these would make good interview questions? Mark the ones that would be good. For the ones that would be bad, briefly brainstorm a way to rewrite the question to make it better. You can go ahead and skip forward to the exercise if you don't want to listen to me read them out. Number one, do you exercise? Number two, how often do you exercise? Number three, do you exercise for health or for pleasure? Number four, what, if anything do you listen to while exercising? Number five, what device do you use to listen to something while exercising? Number six, we're developing an app for listening to audio books while exercising. Would that be interesting to you?



Which of these would make good interview questions?

- ☐ Do you exercise?
- ☒ How often do you exercise?
- ☐ Do you exercise for health or for pleasure?
- ☒ What, if anything, do you listen to while exercising?
- ☒ What device do you use to listen to something while exercising?
- ☐ We're developing an app for listening to audiobooks while exercising. Would that be interesting to you?

Personally, I think three of these are good questions. Do you exercise, is not a great question, because it's kind of a yes or no question. How often do you exercise, is actually the better way of asking the same question. It subsumes all the answers to do you exercise, but leaves more room for elaboration or more room for detail. Do you exercise for health or for pleasure, is not a great question, because it

presents to the user a dichotomy. It might not be the way they actually think about the problem. Maybe there's some other reason they exercise. Maybe they do it to be social, for example. We want to leave open all the possibilities a user might have. So instead of asking, do you exercise for health or for pleasure, we probably want to ask, why do you exercise? The next two questions work pretty well, because they leave plenty of room for the participant to have a wide range of answers, and they're not leading them towards any particular answer. We're not asking, for example, what smartphone do you use to listen to something, because maybe they don't use a smartphone. This sixth one is interesting. We're developing an app for listening to audiobooks while exercising. Would that be interesting to you? What's wrong with that question? When we say, we're developing an app, we introduce something called social desirability bias. Because we're the ones developing the app, the user is going to feel some pressure to agree with us, to support our ideas. People like to support one another. And so even if they wouldn't be interested, they'll likely say that they would, because that's the supportive thing to say. No one wants to say, hey, great idea, David, but I would never use it. So what we want to make sure to do is create no incentive for a user to not give us the complete, honest answer. Worrying about hurting our feelings is one reason why they wouldn't be totally honest. So we might reword this question just to say, would you be interested in an app for listening to audiobooks while exercising? Now granted, the fact that we're the ones asking still probably will tip off the user that we're probably thinking about moving in that direction, but at least it's going to be a little more collaborative. We're not tipping them off that we're already planning to do this, we're telling them that we might be thinking about doing it. And so if they don't think it's a good idea, they kind of feel like they should tell us right now, to save us time down the road. So by rephrasing the question that way, we hopefully, avoid biasing the participant to just agree with us to be nice.

Think-Aloud

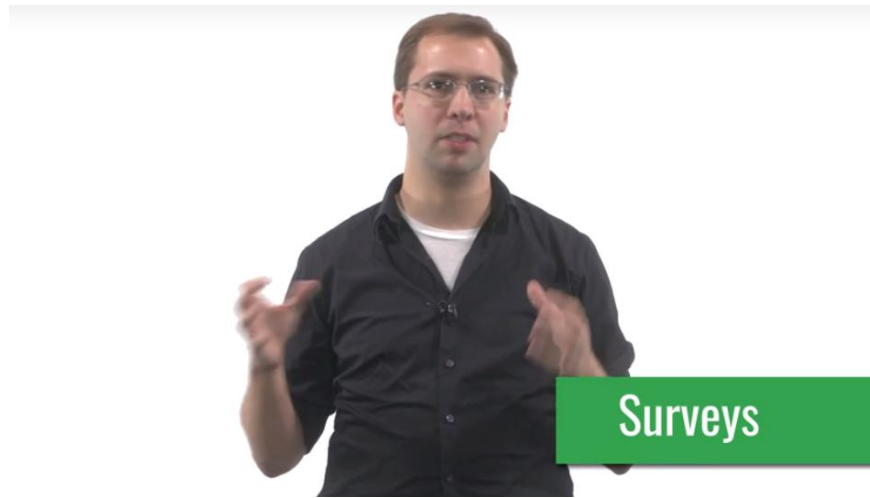


Think-aloud protocols are similar to interviews in that we're asking users to talk about their perceptions of the task. But with think-aloud, we're asking them to actually do so in the context of the task. So instead of bringing Morgan in to answer some questions about listening to audiobooks while exercising, I'll ask her to actually think out loud while listening to audiobooks and exercising. If this was a different task like something on a computer, I could have her just come into my lab and work on it. But since this is out in the world, what I might just do is give her a voice recorder to record her thoughts while she's out running and listening. Now think aloud is very useful, because it can help us get at users thoughts that they forget when they are no longer engaged in the task. But it's also a bit dangerous by asking people to think aloud about their task, we encourage them to think about it more deliberately and that can change the way they actually act. So while it's useful to get an understanding of what they are thinking, we should check to see if there are places where what they do differs when thinking out loud about it.



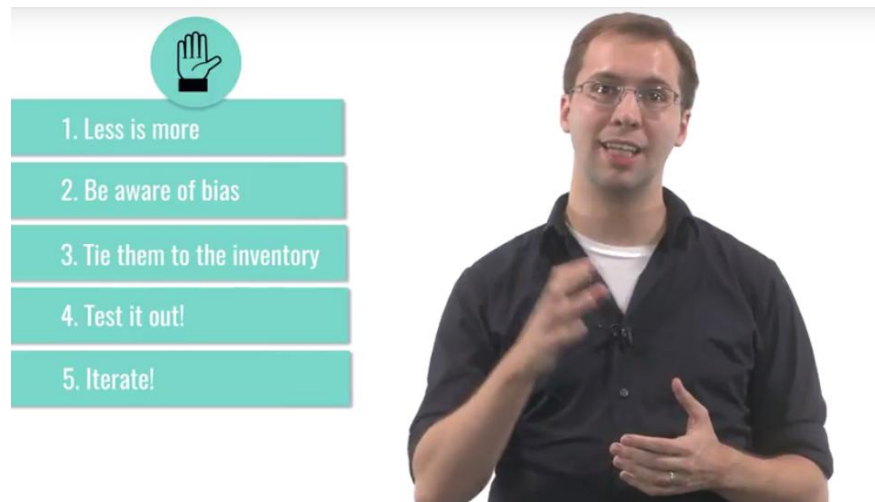
We can do that with what's called a post-event protocol, which is largely the same, except we wait to get the user's thoughts until immediately after the activity. That way, the activity is still fresh in their minds, but the act of thinking about it shouldn't affect their performance quite as much.

Surveys



Most of the other methods for need finding, like observation, interviewing, apprenticeship, require a significant amount of effort for what is often relatively little data. Or it's data from a small number of users. We might spend an entire hour interviewing a single possible user or an hour observing a small number of users in the world. The data we get from those interactions is deep and thorough, but sometimes, we also want broader data. Sometimes, we just want to know how many people encounter a certain difficulty or engage in a certain task. If we're designing an audio book app for exercisers, for example. Maybe we just want to know how often those people exercises or maybe we want to know what kind of books they listen to. At that point, a survey might be our more appropriate means of need finding. Surveys let us get a much larger number of responses very quickly and the questions can be phrased objectively, allowing for quicker interpretation. And plus, with the Internet, they can be administered asynchronously for at a pretty low cost. A few weeks ago, for example, I came up with the idea for a study on Friday morning. And with the great cooperation from our local IRB office, I was able to send out the survey to potential participants less than 24 hours later and receive 150 responses within a week. Now of course, the data I receive from that isn't nearly as thorough as what I would receive from interviewing some of those participants. But it's a powerful way of getting a larger amount of data. And it can be especially useful to decide what to ask participants during interviews or during focus groups.

5 Tips: Surveys



Survey design is a well documented art form. And in fact, designing surveys is very similar to designing interfaces themselves. So many of the lessons we learn in our conversations apply here are well. Here are five quick tips for designing and administering effective surveys. Number one, less is more. The biggest mistake that I see novice survey designers make is to ask way too much. That affects the response rate and the reliability of the data. Ask the minimum number of questions necessary to get the data that you need and only ask questions that you know that you'll use. Number two. Be aware of bias. Look at how you're phrasing the questions. Are there positive or negative connotations? Are participants implicitly pressured to answer one way or the other? Number three. Tie them to the inventory. Make sure every question on your survey connects to some of the data that you want to gather. Start with the goals of the survey and then write the questions from there. Number four. Test it out. Before sending it to real participants, have your co-workers or colleagues test out your survey. Pretend they're real users and see if you would get the data you need from their responses. Number five, iterate. Survey design is like interface design, test out your survey, see what works and what doesn't and revise it accordingly. Give participants a chance to give feedback on the survey itself, so that you can improve it for future iterations.

Quiz: Exercise: Surveys



What is wrong with this survey?

- On a scale of 1 to 4 with 1 meaning 'a lot' and 4 meaning 'not at all', how much do you enjoy exercising?
- Why do you like to exercise?
- On a scale of 1 to 6 with 1 meaning 'not at all' and 6 meaning 'a lot', how much do you like audiobooks?
- Have you listened to an audiobook this year?

Writing survey questions is an art, as well as a science. So let's take a look at an intentionally poorly designed survey, and see everything we can find that's wrong with it. So on the left is a survey. It's kind of short, mostly because of screen real estate. Write down in the box on the right everything that is wrong with this survey. Feel free to skip forward if you don't want to listen to me read out the questions. On a scale of 1 to 4 with 1 meaning a lot and 4 meaning not at all, how much do you enjoy exercising? Why do you like to exercise? On a scale of 1 to 6 with 1 meaning not at all and 6 meaning a lot, how much do you like audiobooks? Have you listened to an audiobook this year?



What is wrong with this survey?

- On a scale of 1 to 4 with 1 meaning 'a lot' and 4 meaning 'not at all', how much do you enjoy exercising?
- Why do you like to exercise?
- On a scale of 1 to 6 with 1 meaning 'not at all' and 6 meaning 'a lot', how much do you like audiobooks?
- Have you listened to an audiobook this year?

Ambiguous scale
Changing number of options
Reversing scale
Leading questions
Yes/no questions

Here are a few of the problems that I intentionally put into this survey. Some of them are kind of obvious, but hopefully a couple others were a little bit more subtle and a little bit more interesting. First when I say on a scale of one to four with one meaning a lot and four meaning not at all, what do two and three mean exactly? It's not a very clear scale to just say the endpoint. Just giving the endpoints doesn't give a very clear scale. We usually also want to provide an odd number of options, so that users have kind of a neutral central option. Sometimes we'll want to force our participants to take

one side or the other, but generally we want to give them that middle neutral option. Either way though, we definitely don't want to change the number of options between those two questions. Having one be 1 to 4 and the other be 1 to 6 is just confusing. And even worse, notice that we're reversing the scale between these two. In the first question, the low number means a lot. In the second question, the high number means a lot. That's just terrible design. We want to be consistent across our entire survey, both with the direction of our scale and the number of options unless there's a compelling reason not to. The second question is also guilty of being quite a leading question. Why do you like to exercise assumes the participant likes to exercise. What are they supposed to say if they don't? And finally, the last question is a yes or no question. Have you listened to an audiobook this year? Yes or no. No is kind of an interesting answer, but yes, I don't know if you listened to one audiobook this year or a 100 audiobooks this year. I don't know if you listened every single day or if you just listened once because you had a gift certificate. So we want to reword this question to be a little more open-ended and support a wider range of participant answers.

Other Data Gathering Methods



So far we've discussed some of the more common approaches that need finding. Depending on your domain though, there might be some other things you can do. First, if you're designing for a task for which interfaces already exist, you might start by critiquing the interfaces that already exist using some of the evaluation methods that we'll cover later in the evaluation lesson. For example, if you're wanting to design a new system for ordering takeout food, you might evaluate the interfaces of calling in an order, ordering via mobile phone or ordering via a website.



Second and similarly, [SOUND] if you're trying to develop a tool to address a problem that people that are already addressing, you might go look at user reviews and see what people already like and dislike about existing products. For example, there are dozens of alarm clock apps out there, and thousands of reviews. If you want to design a new one, you could start there to find out what people need or what their common complaints are.



Third, if you're working on a task that already involves a lot of automatic logging like web surfing, you could try to get some logs of user interaction that have already been generated. For example, say you wanted to build a browser that's better at anticipating what the user will want to open next. You could grab data logs and look for trends both within and across users. You can creative with your data gathering methods. The goal is to use a variety of methods to paint a complete picture of the user's task.

Quiz: Exercise: Needfinding Pros and Cons



Match the advantage to the needfinding method.

	Naturalistic Observation	Participant Observation	Errors and Hacks	Interviews	Surveys	Focus Groups	Apprenticeship	Think-Aloud
Analyzes data that already exists								
Requires no recruitment								
Requires no synchronous participation								
Investigates participant's thoughts								
Occurs within the task context								
Cheaply gathers lots of users' data								

In this lesson we've covered a wide variety of different methods for need finding. Each method has its own disadvantages and advantages. So let's start to wrap up the lesson by exploring this with an exercise. Here are the methods we've covered, and here are the potential advantages. For each row, for each advantage, mark which need-finding method actually has that advantage. Note that these might be somewhat relative, so your answer may differ from ours. Go ahead and skip to the exercise if you don't want to listen to me read these out. The columns from left to right are Naturalistic Observation, Participant Observation, Errors and Hacks, Interviews, Surveys, Focus Groups, Apprenticeship, and Think-Aloud. The potential advantages are Analyzes data that already exists, Requires no recruitment, Requires no synchronous participation, Investigates the participant's thoughts, occurs within the task context, and cheaply gathers lots of users' data.



Match the advantage to the needfinding method.

	Naturalistic Observation	Participant Observation	Errors and Hacks	Interviews	Surveys	Focus Groups	Apprenticeship	Think-Aloud
Analyzes data that already exists								
Requires no recruitment								
Requires no synchronous participation								
Investigates participant's thoughts								
Occurs within the task context								
Cheaply gathers lots of users' data								

Here's my answer to this very complicated exercise. Two methods that analyze data that already exists are Naturalistic Observation and Errors and Hacks. Naturalistic Observation doesn't necessarily analyze

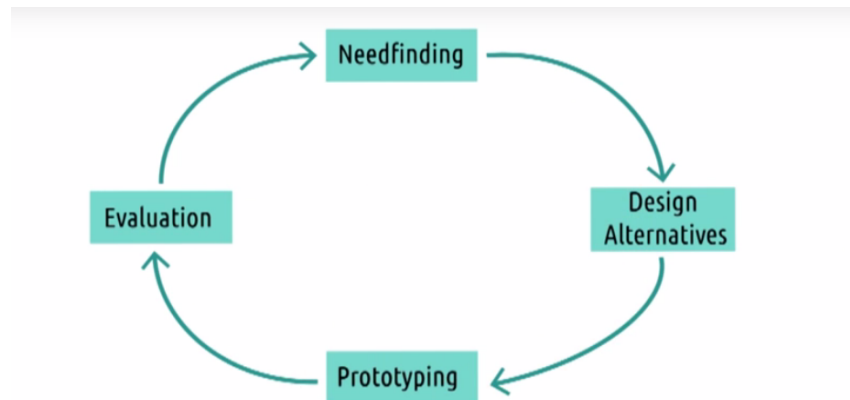
data that already exists, but it analyzes data that's being produced already on its own without observing it, so we don't have to go out and create an opportunity for data to happen. We just have to observe it and capture it where it's already taking place. Errors and Hacks, look at the way users already use interfaces to see what errors they regularly make or when they have to work around the interface. The two methods that require no recruitment are Naturalistic Observation and Participant Observation. In both cases, we don't need other human participants to come do anything differently based on the fact that we're doing some research. With interviews, surveys, Focus Groups, apprenticeship and Think-Aloud, we're always asking users to do something to accommodate us or to give us some data. And with Errors and Hacks, even if we can view that data on our own, we still need the user to give us permission to view their workspace or watch them do whatever they're doing. There might be some times when you can look for Errors and Hacks with Naturalistic Observation, but generally you need to get enough into the users head to understand why something's an error or why they need to use a certain hack. For the most part, all of these are going to need some synchronous participation. There might be some exceptions. For example, we could do a retrospective analysis of Errors and Hacks, or we can have someone do a Think-Aloud protocol where they actually write down their thoughts after doing a task. But generally speaking, the way most of these are usually done, they require synchronous participation. Surveys are the exception. Surveys we usually send out to someone, wait some period of time, and get back the results. So we never have to be interacting live with any of our participants. That's one of the reasons why surveys can get a lot more data than other methods. Adding more participants doesn't necessarily require more of our time, at least not to gather the data in the first place. Analyzing it might require more time at the end, but that's not synchronous either. As far as investigating participant thoughts is concerned, almost all these methods can investigate this when used correctly. We could do a survey does not actually investigate participants thoughts, but a well designed survey is going to trying get a heart of the users thinks about things. The only exception is Naturalistic Observation where by definition, we're just watching people we're not interacting with them or we're not asking them what they are thinking. It's always extremely valuable for us to be able to do some needfinding that occurs within the task context itself. And unfortunately interviews and surveys, which are some of our most common data gathering methods, very often don't occur within the task context. Naturalistic Observation and Participant Observation obviously do, but since they don't involved getting inside the real users head, their contributions are a little bit more limited. Apprenticeship and Think-Aloud really capture the benefits of occurring within the task context, because either way we get the user's thoughts while they're engaging with the task, or immediately thereafter. It is possible to do interviews and Focus Groups within the task contexts as well, it just isn't quite as common. Errors and Hacks are certainly debatable as well, because the Errors and Hacks themselves definitely occur within the task context, but our analysis of them usually doesn't. And finally, as we talk about when we discuss cognitive task analysis, one of the challenges with needfinding is that most of our approaches are extremely expensive. If we want to gather a lot of data cheaply, then we probably need to rely on surveys. Everything else is either going to incur a pretty significant cost or it just isn't capable of gathering a lot of data. For example, we could cheaply be participant observations for weeks on end, but we're only ever going to gather data from one person and that's never ideal.

Quiz: Design Challenge: Needfinding for Book Reading

The needfinding exercises that we've gone through so far focus on the needs of the exercisers. What can they do with their hands, what is the environment around them like while exercising, and so on? However, that's only half the picture for this particular design. Our ultimate goal is to bring the experience of consuming books to people that exercise, which means we also need to understand the task of book-reading on its own. Now a problem space is still around exercisers, so we wouldn't go through the entire design life cycle for book reading on its own. We don't need to design or prototype anything for them. But if we're going to bring the full book reading experience to people while exercising, we need to understand what that is. So take a moment and design an approach to needfinding for people who are reading on their own.

We could apply pretty much every single need-finding method that we've discussed to this task. We could, for example, go to the library and just watch people reading and see how they're taking notes. We've all likely done it ourselves. We can reflect on what we do while reading, although again, we need to be careful not to over-value our own priorities and approaches. Reading is common enough, that we can easily find participants for interviews, surveys, think alouds. The challenge here will be deciding who our users really are. Books are ubiquitous. Are we trying to cater to everyone who reads deliberately? If so, we need to sample a wide range of users or initially, we could choose a subset. We might cater to students who are studying or busy business people, or people that specifically walk or bike to work. We might start with one of those groups and then abstract out over time. We might eventually abstract all the way to just anyone who's unable to read and take notes the traditional way like people driving cars or people with visual impairments but that's further down the road. The more important thing is that we define who our user is, define the task in which we're interested, and deliberately design for that user and that task throughout the design life cycle.

Iterative Needfinding

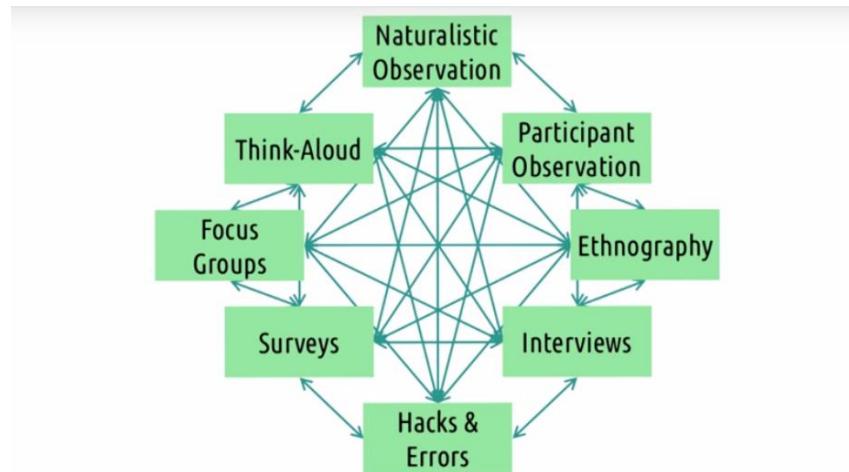


We've noted that design is a life cycle from needfinding to brainstorming design alternatives to prototyping to evaluation. And then, back to needfinding to continue the cycle again.

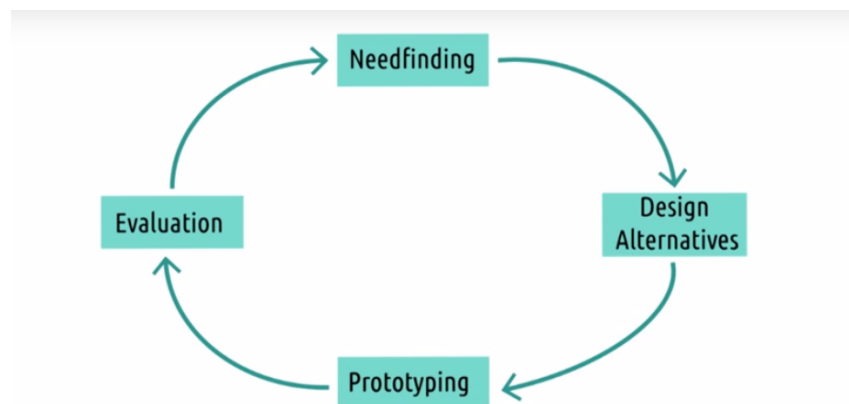


Needfinding on its own though can be a cycle by itself. For example, we might use the results of our naturalistic observation to inform the questions we asked during our interviews. For example, imagine that we noticed that very many joggers, jog with only one earphone in. That's a naturalistic observation, and then in an interview, we might ask, why do some of you jog with only one earphone in? And we might get the answer from the interview that it's to listen for cars or listen for someone trying to get their attention because they exercise in a busy area. Now that we understand why they have that behavior, maybe we develop a survey to try and see how widespread that behavior is, and ask, how many of you need to worry about what's around you when you're listening while driving? If we notice in those surveys a significant split in the number of people who were concerned about that, that might inform our next round of naturalistic observation. We might go out and look and see in what

environments are people only wearing one headphone and in what environments are they wearing both.

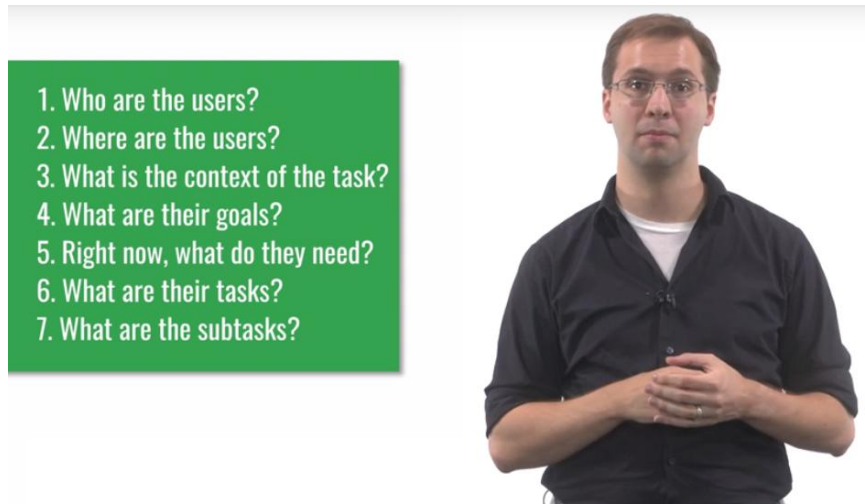


So in that way all of the different kinds of need finding that we do can inform our next round of other kinds of need finding. We can go through entire cycles just of need finding without ever going on to our design alternatives or prototyping stages. However, the prototyping and evaluation that we do will then become another input into this. During our evaluation we might discover things that will then inform what we need to do next as far as need finding is concerned. Creating prototypes and evaluating them gives us data on what works and what doesn't. And that might inform what we want to observe to better understand the task going forward.



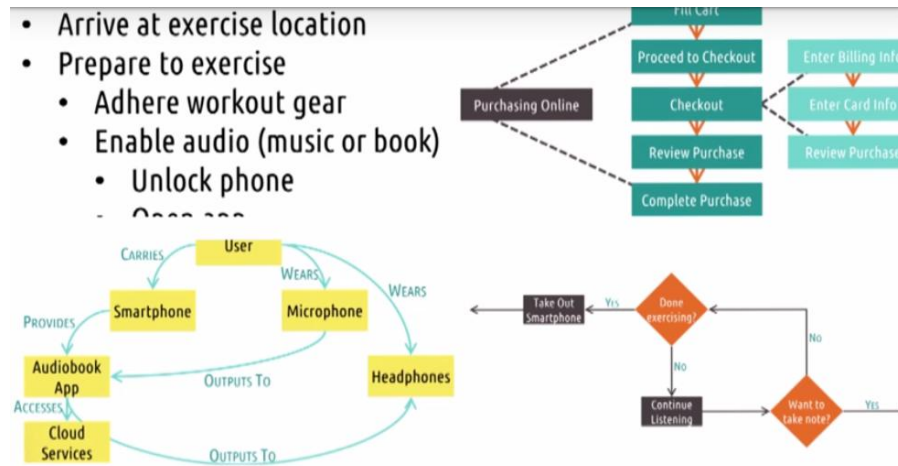
That's the reason why the output of evaluation is more needfinding. It would be a mistake to do one initial needfinding stage, and then jump in to a back and forth cycle of prototyping and evaluation.

Revisiting the Inventory



During these need-finding exercises, you'll have gathered an enormous amount of information about your users. Ideally, you've combined different sets of these approaches. You've observed people performing the tasks, you've asked them about their thought process, and you tried it some yourself. Pay special attention to some of the places where the data seem to conflict. Are these cases where you as the designer understand some elements of the task that the users don't? Or are these cases where your expertise hasn't quite developed to the point of understanding the task? Once you've gone through the data gathering process, it's time to revisit that inventory of things we wanted to gather initially. One, who are the users? Two, where are the users? Three, what is the context of the task? Four, what are their goals? Five, right now, what do they need? Six, what are their tasks? And seven, what are the subtasks? Revisit these, with the results of your data gathering in mind.

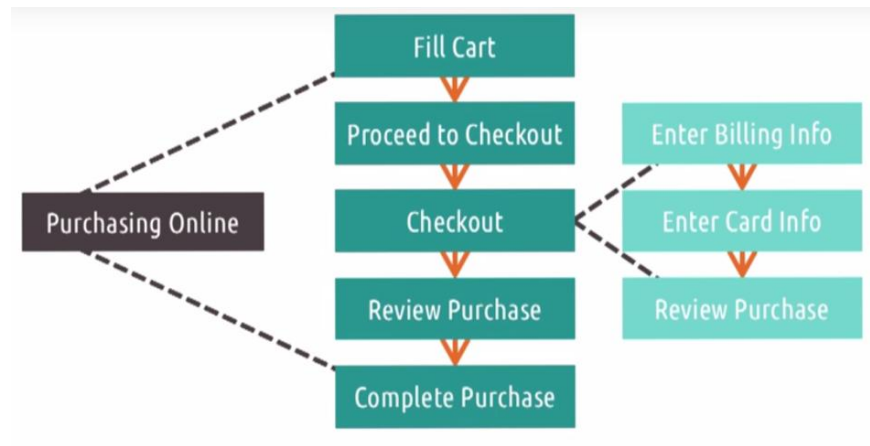
Representing the Need



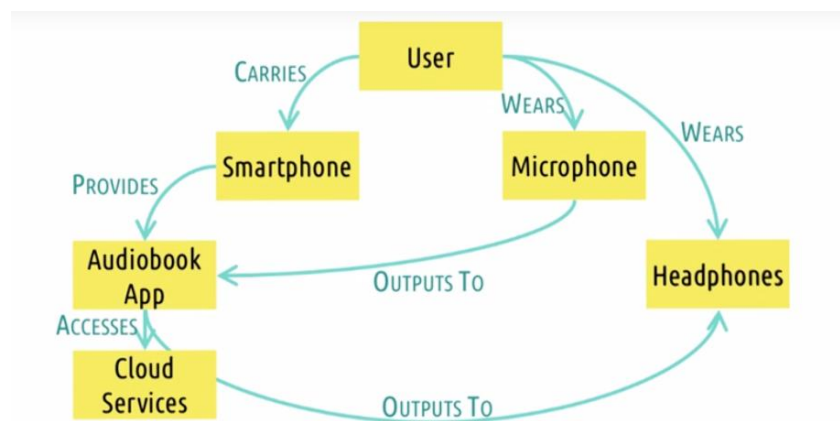
Now that you have some understanding of the user's needs, it's time to try to formalize that into something we can use in design. There are a number of different ways we can do this.

- Arrive at exercise location
- Prepare to exercise
 - Adhere workout gear
 - Enable audio (music or book)
 - Unlock phone
 - Open app
 - Select desired item
 - Press play
 - Insert headphones
- Stretch
- ...

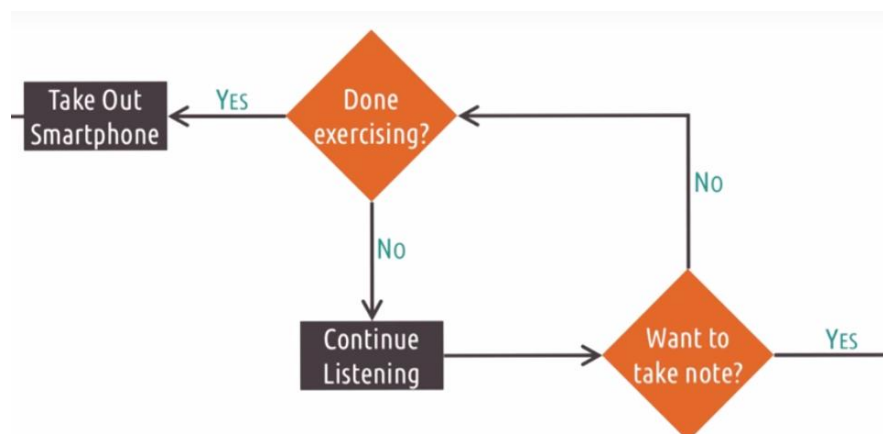
For example, maybe we create a step-by-step task outline of the user engaging in some task. We can break those tasks down into sub-tasks as well, all the way down to the operator level.



We can further develop this kind of task outline into a hierarchical network, like we talked about before. This might involve more complexity than simply a linear series of actions.



We might further augment this with a diagram of the structural relationships amongst the components in the system and how they interact. This might give us some information about how we get feedback back to the user or how they interact with our interface in the first place.



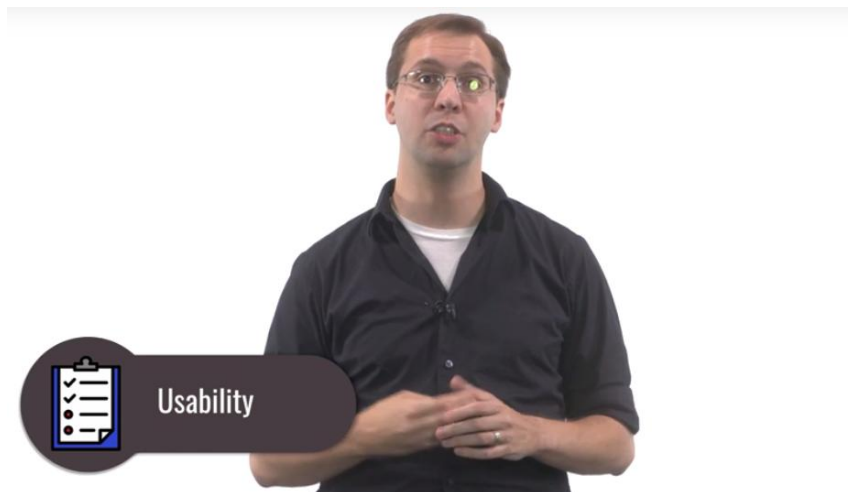
From there, we might develop this even more into a flow-chart equipped with decision-making points or points of interruptions. Notice how these representations are very similar to the outcomes of the task analyses we talk about in the principles unit of our conversations. We can similarly use the data gathered from here to summarize a more comprehensive task analysis that will be useful in designing and prototyping our designs.

Defining the Requirements

Finally, the final step for need-finding is to define our requirements. These are the requirements that our final interface must meet. They should be specific and evaluable, and they can include some components that are outside of users tasks, as well, as defined by the project requirements.



In terms of user tasks, we might have requirements for guarding functionality, what the interface can actually do.



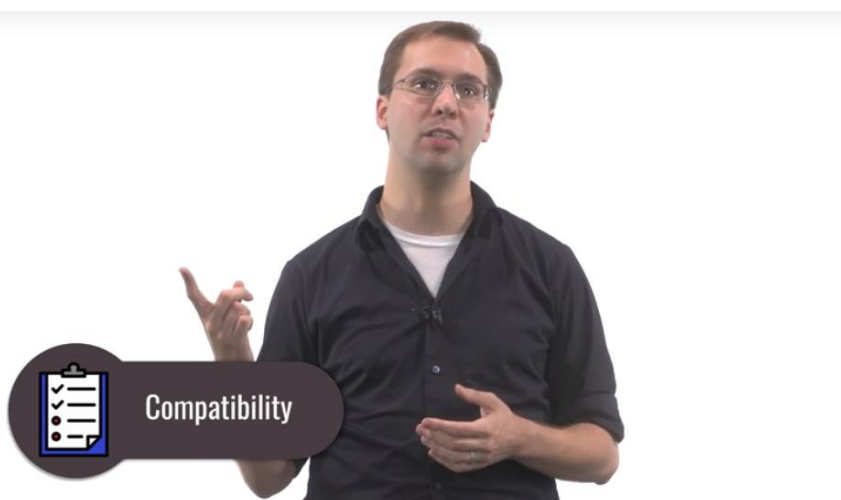
Usability, how certain user interactions must work.



Learnability, how fast the user can start to use the interface.



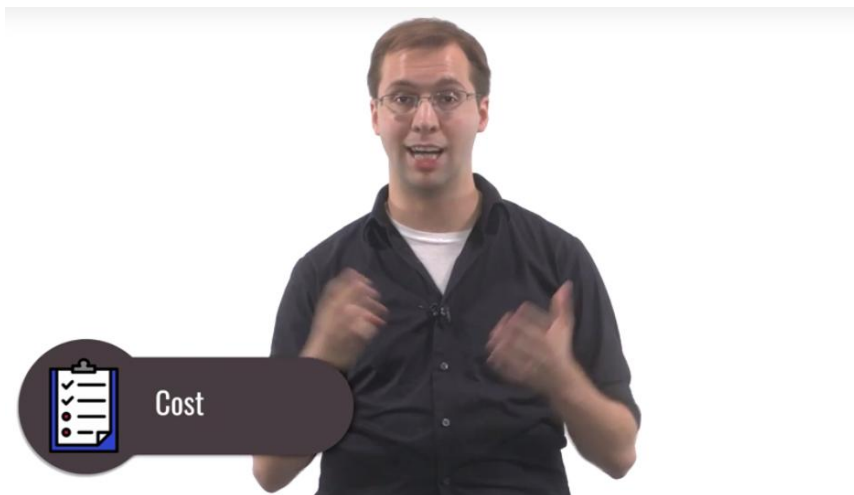
And accessibility, who can use the interface.



We might also have some that are generated by external project requirements, like compatibility, what devices the interface can run on.



Compliance, how the interface protects user privacy.



Cost, how much the final tool can actually cost, and so on. We'll use these to evaluate the interfaces we develop, going forward.

Exploring HCI: Needfinding

How might need finding work in your chosen area of HCI? If you're looking at designing for some technological innovation like augmented or virtual interactions, the initial phase might not actually be that different. Your goal is to understand how people perform tasks right now without your interface. So initially, you want to observe them in their naturalistic environment. Later though, you'll need to start thinking about bringing participants to you to experience the devices first hand. If you're interested in something like HCI for healthcare or education, you have a wealth of naturalistic observation available to you. You might even have existing interfaces doing what you want to do. And you can try to leverage those as part of your need finding exercises. Remember, no matter your area of application, you want to start with real users. That might be observing them in the wild, talking to them directly, or looking at data they've already generated.

Conclusion



Today, we've talked about need finding. Need finding is how you develop your understanding of the needs of your user. What tasks are they completing? What are the context of those tasks? What else is going on? What are they thinking during the task and what do they have to hold in working memory? All these things feed into your understanding of our users needs.



Now we've discussed a number of different techniques to approach this, ranging from low intervention to high intervention.



On the low side, we can just observe our users in the wild or we can become users ourselves and participate in the task.



Working up we can try to look more closely at users areas to find errors or hacks, or peruse the data that they're already generating.



We might interact with them directly through surveys, interviews or focus groups.



Or we might choose to work alongside them, not just participating in the task independently, but learning from them and developing expertise itself. Once you've gained a sufficient understanding, it's time to move on to the next step, brainstorming design alternatives.

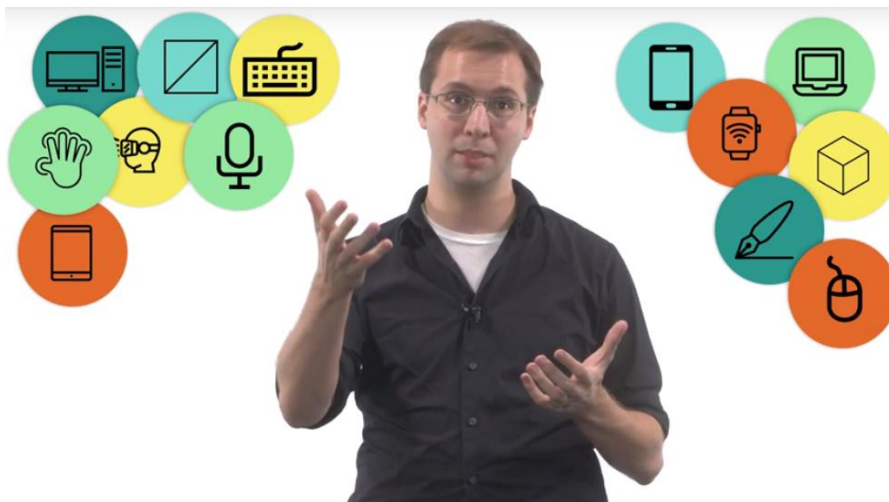
3.4 Design Alternatives

Compiled by Shipra De, Summer 2017

Introduction to Design Alternatives



[MUSIC]. When we've developed a good understanding of the needs of our user, it's time to move on to the second phase of the design life cycle, design alternatives. This is when we start to brainstorm how to accomplish the task we've been investigating. The problem here is that design is very hard, it's hard for a number of reasons. The number of choices we have to make, and things we have to control is more expansive than ever before.



Are we designing for desktops, laptops, tablets, smart phones, smart watches, augmented reality, virtual reality, 2D, 3D, gesture input, pen input, keyboard input, mouse input, voice input?

(c) 2017 by David Joyner, all rights reserved.

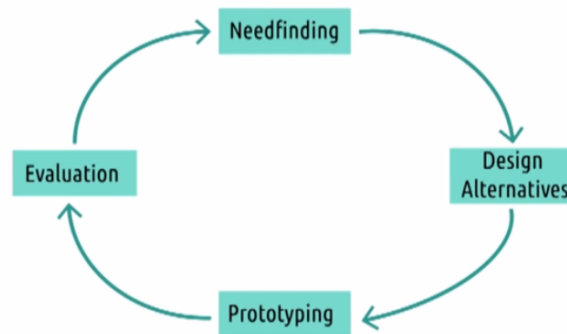


In this lesson, we're going to talk about how to generate ideas for designs.



And then we'll chat about how to explore those ideas a bit further to figure out what you want to actually pursue.

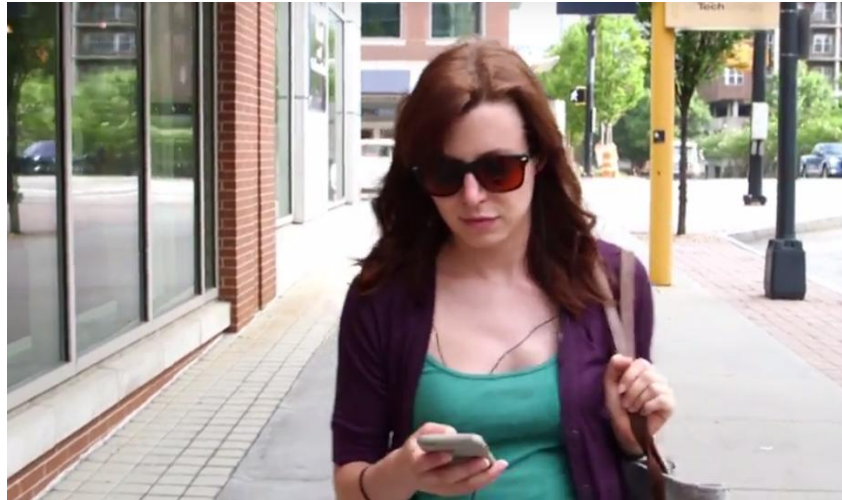
The Second Biggest Mistake



The biggest mistake that a designer can make is jumping straight to designing an interface without understanding the users or understanding the task. The second biggest mistake though is settling on a single design idea or a single genre of design ideas too early.



This can take on multiple forms. One form is staying too allegiant to existing designs or products. Take the thermostat, for example, again. If you settled on tweaking the existing design of a thermostat you would never invent the Nest. So if you're working on improving an existing interface, try to actually distance yourself from the existing solutions, at least initially during the brainstorming session. But this is also a problem if you're designing interfaces for new tasks as well.



Imagine for instance, that while you were observing people exercising, you started sketching interface ideas like how to make the buttons big enough or what buttons need to be featured prominently. In doing so, you're getting tunnel vision and missing out on any design alternatives that might involve voice or gesture control. So the second biggest mistake you can make is focusing too strongly on one alternative from the very beginning, instead of exploring the entire range of possible design alternatives.



The reason why this is such a common mistake, is that there's this natural tendency to think of it as a waste of time to develop interfaces you're not going to end up using. You think you can get it done faster just by picking one early on and sticking to it. But flushing out ideas for interfaces you don't end up using isn't a waste of time, because by doing so you continue to learn more about the problem. The experience of exploring those ideas that you leave behind will make you a better designer for the ideas that you do choose to pursue. In all likelihood your ultimate design will be some combination of the design alternatives that you explored earlier. So, take my security system for an example. There are two ways of interacting with it, the key pad and the key chain. Two different designs that, in this particular instance, integrated just fine. Different alternatives won't always integrate side by side this

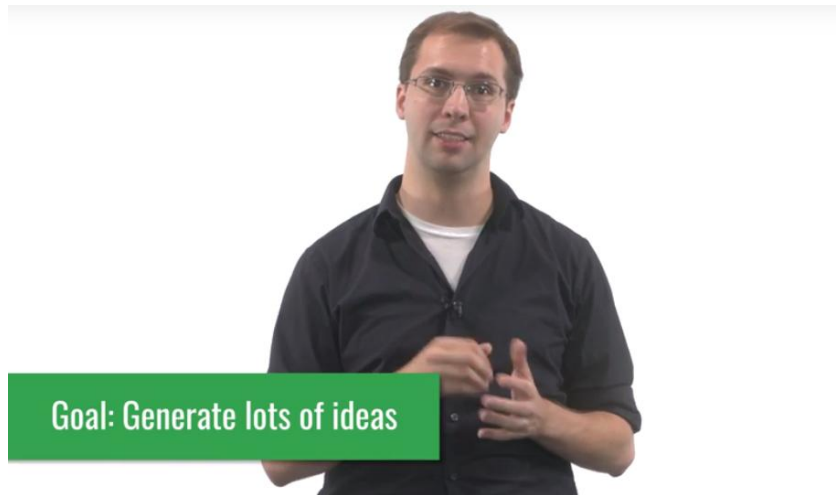
easily, but the design process as a whole is an iterative process of brainstorming, combining, abandoning, revising and improving your ideas, and that requires you start with several ideas in the first place.

The Design Space

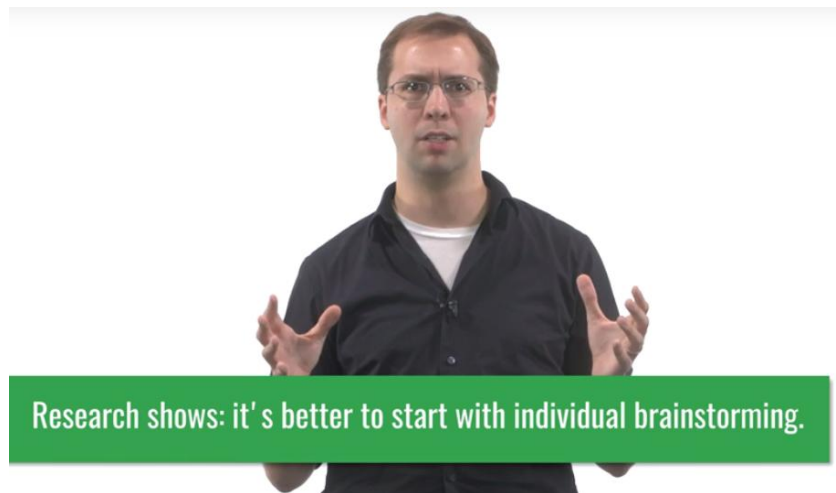


When we talk about the problem we're solving here we define the problem space as disabling a security system as we enter a home. We defined our problem as far as possible away from the current interfaces for doing it. The design space on the other hand is the area in which we design our solutions to this problem. The current design space for this problem is wall mounted devices and portable devices like my key chain. But as we design, the space of possible ideas might expand. For example, as we go along we might be interested in voice interfaces or interfaces with our mobile phones or wearable devices. Our goal during the design alternative phase is to explore the possible design space. We don't want to narrow down too early by sticking devices on walls or devices on keychains. We want to brainstorm lots of possible approaches, and grow a large space of possible designs.

Individual Brainstorming 1



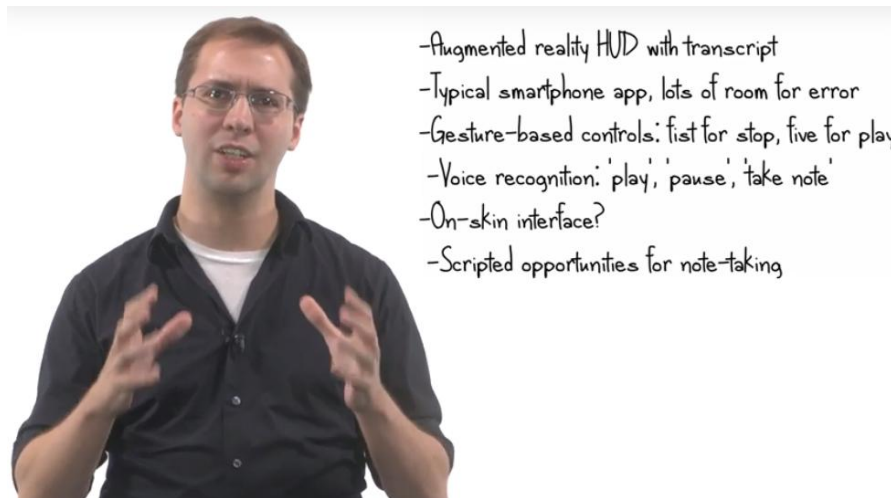
When you first start brainstorming, your goal is to generate a lot of ideas. These ideas can be very short, very high-level, and very general. Your goal is just to generate an expanse of them. They don't even have to be ideas for interfaces: just any idea for solving the problem. If you look online, you'll find numerous great guides to how to brainstorm ideas.



One of the most interesting takeaways is that research generally indicates that it's better to start with individual brainstorming. That's non-intuitive, though -- so often we hold meetings for brainstorming, but it should start individually. That's because brainstorming is most effective when it initially just generates a lot of ideas, but groups tend to coalesce around ideas pretty early.



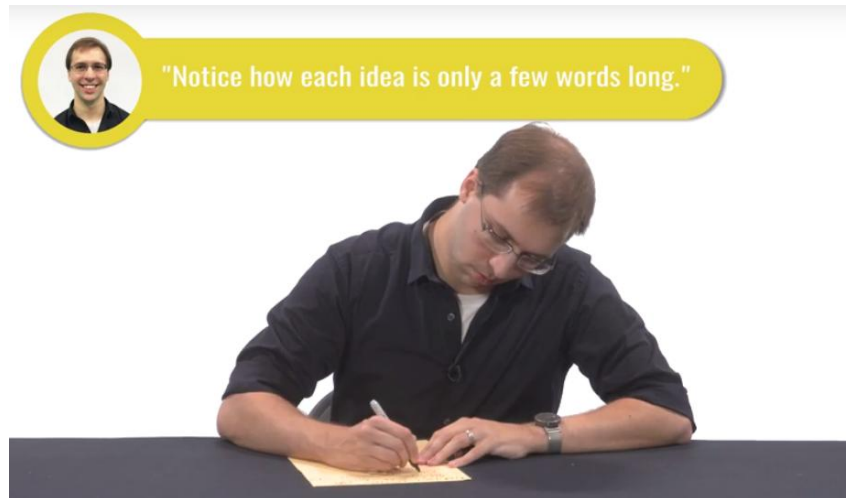
So, start out individually. Generate a lot of ideas. Each idea needs only be a few words or a sentence. Don't worry right now if they're good or bad. Write down everything.



Think about how you'd design with different types of interactions, like gestures, voice, or touch. Think about how you'd design for different interfaces, like smart watches, tablets, augmented reality. Think about how you'd design for different audiences, novices and experts, kids and adults. Get silly with it, some of the best ideas start as silly ideas. How would you design this for your dog, for your cat? How would you design this for someone with three arms? With one arm? Go nuts. Your goal is to generate a lot of ideas. These are going to get loaded into your mind and they'll crop up in interesting ways throughout the rest of the design process. That's why it's important to generate a lot: you never know what will come up.

Individual Brainstorming 2

So, I'm going to demonstrate this for you real quick. I'm going to brainstorm ideas for our problem of allowing exercisers to consume books and take notes. So, I have my paper for brainstorming, so please enjoy this 30 minute video of me sitting here writing at a desk.





"Notice the page is getting messy -- that's good!"

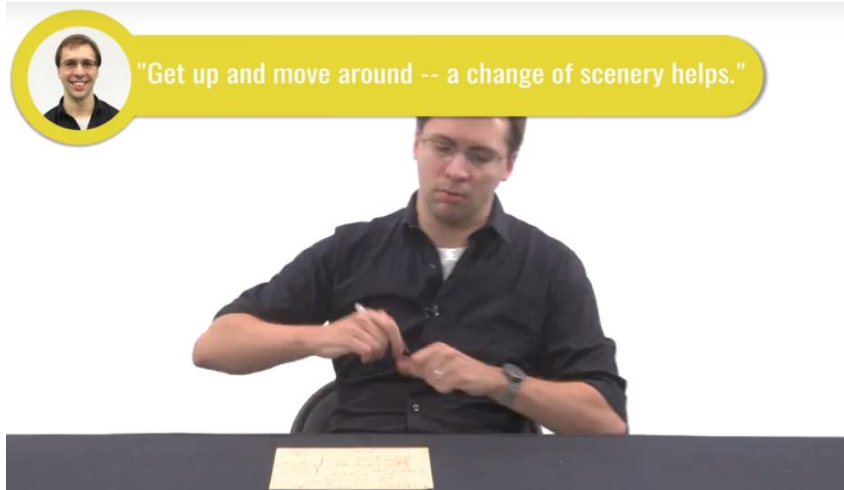


"Don't be afraid to leave and come back to it later."



"Don't cross anything out or throw anything away yet, keep going!"





[MUSIC] Here's my list of ideas. As you might be able to tell, it gets kind of crazy. It's kind of all over the place. You can kind of trace through my entire reasoning process on here. Also, some of the ideas are somewhat straightforward. I've got voice commands, I've got gestures, I've got voice transcription. I kind of tried to separate it out into feedback methods and also the way the user actually interacts because we could kind of combine those. Some of these are actually pretty crazy. I've got an on skin interface. So I've seen some prototypes for things that would let you actually just press on your skin to do different kinds of interactions. I've also got augmented reality, like Google Glass.



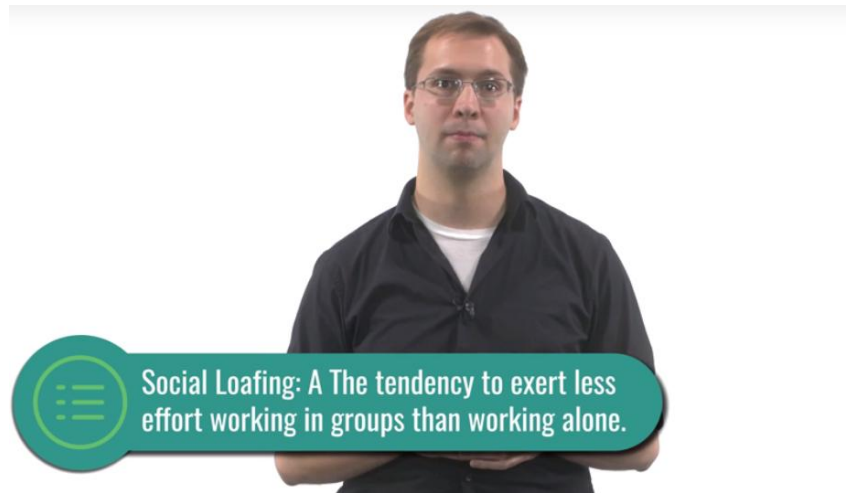
I've got a portable keyboard, like a Twiddler, so notice that this is kind of a mess. That's a good thing. Lists are fine, but chances are, a lot of your ideas are related to each other. Notice also that I never crumpled up my piece of paper, I never threw it away. I crossed one thing out, that's because I wrote the wrong word. Really, at this stage, you don't want to reject any ideas. Your goal is just to kind of free form brainstorm and get all your thoughts out there.

5 Tips: Individual Brainstorming

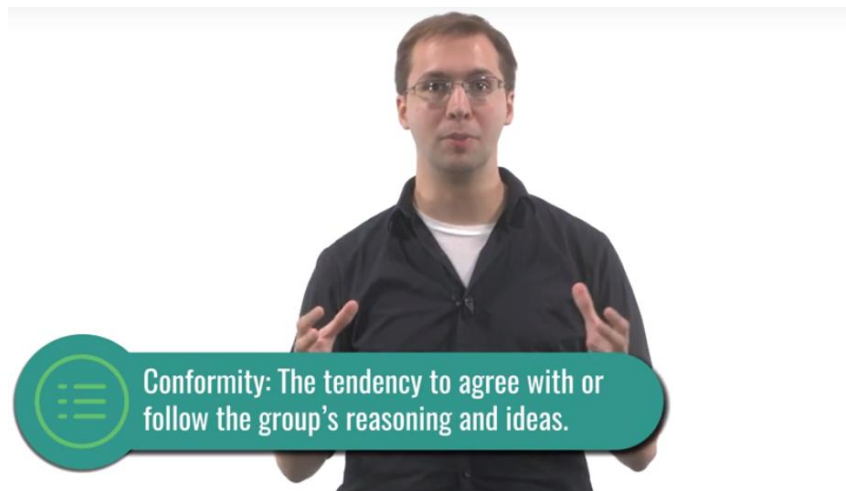


Here are five quick tips for effective individual brain storming. Number one, write down the core problem. Keep this visible. You want to let your mind enter a divergent thinking mode but you also want to remain grounded in the problem. Writing down the problem and keeping it available will help you remain focused while remaining creative. Number two. Constrain yourself. Decide you want at least on idea in a number of different categories. Personally, I try to make sure I have at least three ideas that use nontraditional interaction methods, like touch and voice. You can constrain yourself in strange ways too. Force yourself to think of solutions that are too expensive or not physically possible. The act of thinking in these directions will help you out later. Number three. Aim for 20. Don't stop until you have twenty ideas. These ideas don't have to be very well-formed or complex, they can be simply one sentence descriptions of designs you might pursue. This forces you to think through the problem, rather than getting tunnel vision on an early idea. Number 4. Take a break. You don't need to come up with all of these at once and, in fact, you'll probably find it's easier if you leave and come back. I'm not just talking about a ten minute break either. Stop brainstorming and decide to continue a couple days later but be ready to write down new ideas that come to you. Number 5. Divide and conquer. If you're dealing with a problem like helping kids lead healthier lifestyles, divide it into smaller problems and brainstorm solutions to those. If we're designing audio books for exercises, for example, we might divide it into things like the ability to take and review notes, or the ability to control playback hands-free. Divide it like that and brainstorm solutions to each individual little problem.

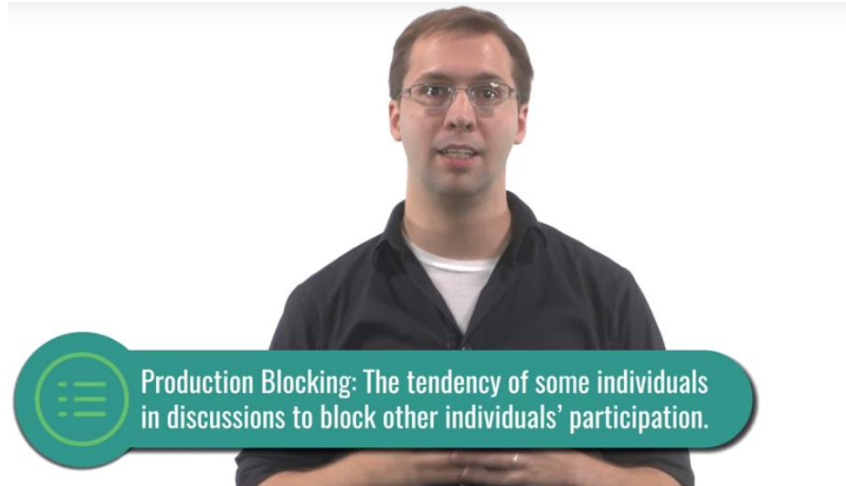
Challenges in Group Brainstorming



Group brain storming presents some significant issues. Thompson in 2008 laid out four behaviors in group brainstorming that can block progress. The first is social loafing. People often don't tend to work as hard in groups as they would individually. It's easy to feel like the responsibility for unproductive brainstorming is shared and deflected. In individual brainstorming, it's clearly on the individual.



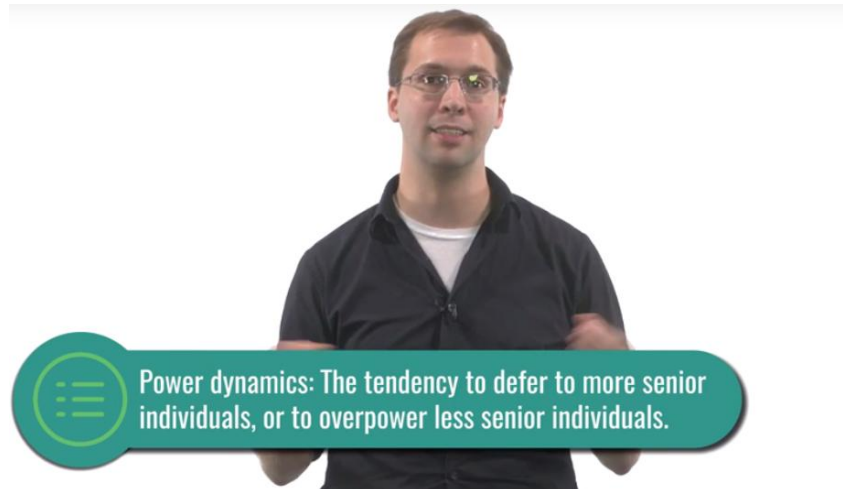
The second blocker is conformity, people in groups tend to want to agree. Studies have shown that group brainstorming leads to convergent thinking. The conversation the group has tends to force participants down the same line of thinking, generating fewer and less varied ideas than the individuals acting alone. During brainstorming, though, the goal is diversion thinking, lots of ideas, lots of creativity.



The third blocker is production blocking. In group brainstorming, there are often individuals who dominate the conversation and make it difficult for others to actually be heard. Their ideas can thus command more weight, not because of the strength of the ideas, but because of the volume of the description.



The fourth blocker is performance matching. People tend to converge in terms of passion and performance, which can lead to a loss of momentum over time. That might be able to get people excited if they're around other excited people initially, but more often than not, it saps the energy of those who enter with enthusiasm. In addition to these four challenges, I would add a fifth.



 Power dynamics: The tendency to defer to more senior individuals, or to overpower less senior individuals.

Group brainstorming may also be prone to power dynamics, or biases. No matter how supportive and collaborative a boss might be, there'll likely always exist a tacit pressure to build on her suggestions, which dampens creative brainstorming. There also exists considerable literature stating that other biases based on gender, age, race, complain to these group sessions as well. Now note that this doesn't mean group brainstorming should be avoided altogether. What it means is that we should enter into group brainstorming with strong ideas of how to address these issues, ideally, after a phase of individual brainstorming has already occurred.

Rules for Group Brainstorming



To have an effective group brainstorming session, we need to have some rules to govern the individual's behavior to address those common challenges. In 1957, Osbourne outline four such rules. Number one, Expressiveness. Any idea that comes to mind, share it out loud, no matter how strange.



Number two, nonevaluation. No criticizing ideas, no evaluating the ideas themselves yet.



Number three, quantity. Brainstorming as many as possible. The more you have, the greater your idea of finding a novel idea.



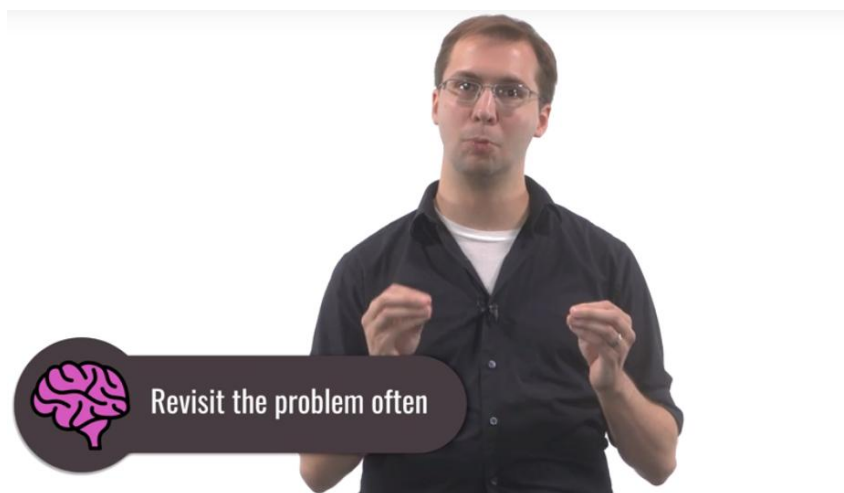
Number four, building. While you shouldn't criticize other's ideas, you should absolutely try to build on them. Then, in 1996, Oxley, Dzindolet and Paulus presented four additional rules.



Number one, stay focused. Keep the goal in mind at all times.



Number two, no explaining ideas. Say the idea and move on. No justifying ideas.



Number three, when you hit a roadblock, revisit the problem. Say it again out loud.



Number four, encourage others. If someone isn't speaking up, encourage them to do so. Note that all eight of these rules prescribe what individuals should do, but they're only effective if every individual does them. So it's good to cover these rules, post them publicly, and call one another on breaking from them.

5 Tips: Group Brainstorming

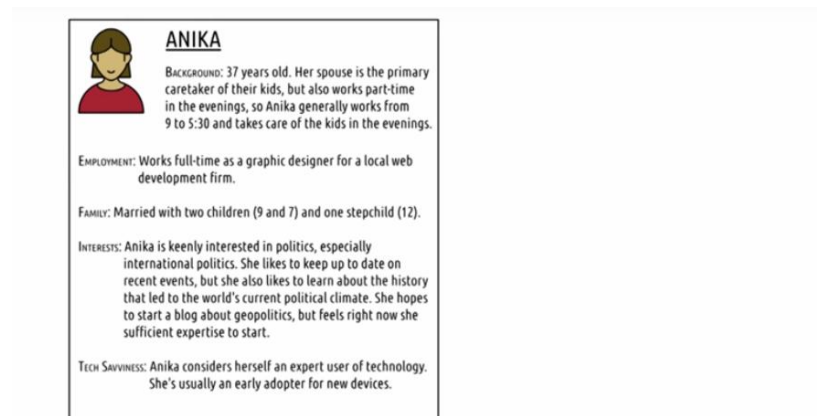


The rules given by Osborn, Oxley, Dzindolet and Paulus are about helping individuals understand how they should act in group brainstorming. Here are a few additional tips though that apply less to the individual participants and more to the design of the activity as a whole. Number one, go through every individual idea. Have participants perform individual brainstorming ahead of time and bring ideas to the group brainstorming session. And explicitly make sure to go through each one. That will help avoid converging on our own idea too early and make sure everyone is heard. Number two, find the optimal size. Social loafing occurs when there's a lack of individual responsibility. When you have so many people that not everyone would get to talk anyway, it's easy for disengagement to occur. I'd say a group brainstorming session should generally not involve more than five people. If more people need to give perspectives than that, then you can have intermediate groups that then send ideas along to a later group. Number three, set clear rules for communication. Get a 20 second timer and when someone starts talking, start it. Once the timer is up, someone else gets to speak. The goal is to ensure that no one can block others ideas by talking too much, whether intentionally or accidentally. Number four, set clear expectations. Enthusiasm starts to wane when people are unsure how long a session will go or what will mark its end. You might set the session to go a certain amount of time or dictate that a certain number of ideas must get generated. No matter how you do it, make sure that people participating can assess where in the brainstorming session they are. Number five. End with ideas, not decisions. It's tempting to want to leave a brainstorming session with a single idea on which to move forward, but that's not the goal. Your brainstorming session should end with several ideas. Then let them percolate in everyone's mind before coming back and choosing the ideas to pursue later.

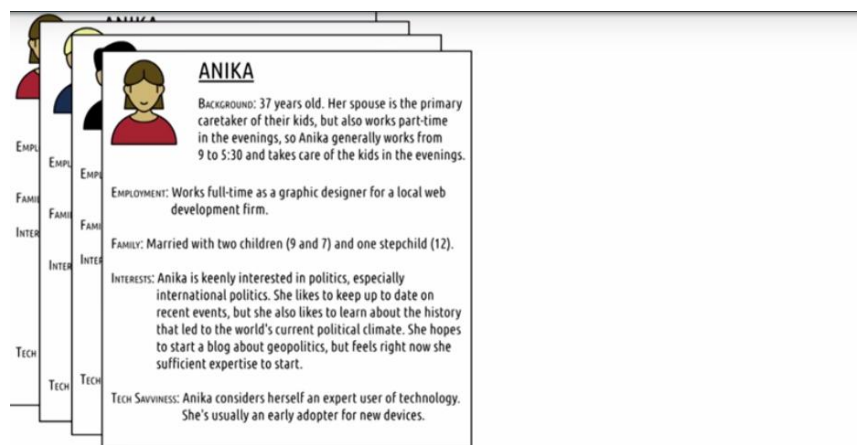
Fleshing Out Ideas

The brainstorming process should lead you to a list of bunch of high level general design alternatives. These are likely just a few words or a sentence each but they described some very general idea of how you might design the interface, to accomplish this task. Your next step is to try to flesh these ideas out into three or four ideas that are worth taking forward to the prototyping stage. Some of the ideas you might be able to dismiss pretty quickly, that's all right. You can't generate good ideas without generating a lot of ideas. Even though you won't end up using all of them. In other places, you might explore an idea a little before dismissing it or you might combine two ideas into a new idea. In the rest of this lesson, we'll give you some thought experiments you can use to evaluate these ideas and decide what to keep, what to combine and what to dismiss.

Personas



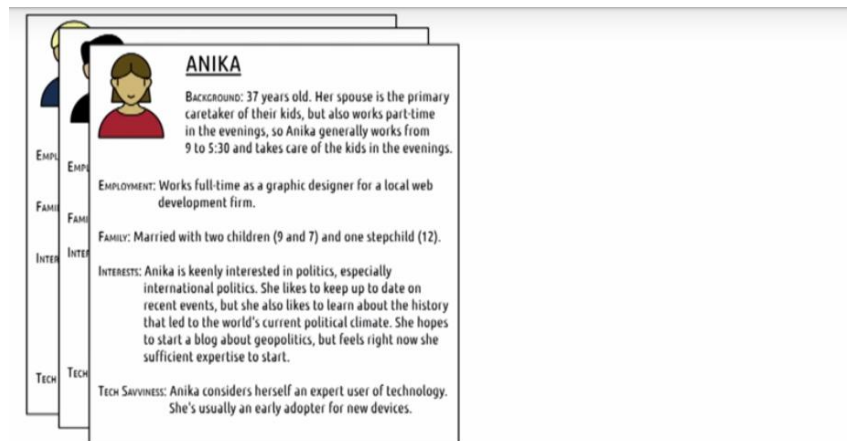
The first common method we can use to flush out design alternatives is called personas. With personas we create actual characters to represent our users. So let's create a persona for the problem of helping exercisers take notes while reading books. We'll start by giving her a name and a face, and then we'll fill out some details. We want to understand who this persona is. We want to be able to mentally simulate her. We want to be able to say, what would Anika do in this situation? What is she thinking when she's about to go exercise? What kind of things might interrupt her? We might want to put in some more domain specific information as well. Like, why does this person exercise? When do they exercise? What kind of books do they like? How are they feeling when they're exercising? Where do they usually exercise?



We want to create at least three or four of these different personas, and perhaps more depending on how many different stakeholders we have for our problem. The important thing is that these should be pretty different people, representing different elements of our designs, different elements of our task, so we can explore its entire range. We don't want to design just for Anika, but we do want to design for

real people. And so we should define personas that represent the range of real people that we care about. That way we can ask our questions like, how would Janet feel about this design alternative? Using this, we can start to extort the space and find the options that has the most appeal.

User Profiles



Personas are meant to give us a small number of characters that we can reason about empathetically. However, it can sometimes also be useful to formulaically generate a large number of user profiles to explore the full design space.

Exercise Expertise	Novice ↔ Expert
Reading Level	Casual ↔ Serious
Motivation	Low ↔ High
Tech Literacy	Low ↔ High
Usage Frequency	Low ↔ High

We can do this by defining a number of different variable about our users and the possibilities within each. So here our few examples, we can ask ourselves, do we care about novice users or expert users or both? Do we care about users that read casually, that read seriously, or both kinds of users? Do we only want to cater to users that are highly motivated to use our app, which can make things a little bit easier on us? Or do we want to assume that it won't take much to stop them from using our app? Can we assume a pretty high-level of technological literacy, or are we trying to cater to more casual users as well? And are we interested in users that are going to use our app all the time, or in users who are going to use our app only occasionally, or both? All of these decisions present some interesting design considerations that we need to keep in mind. For example, for users that are going to use our tool very often, our major consideration is efficiency. We want to make sure they can do what they need to do as quickly as possible. And oftentimes, that might be relying on them to know more about how to use

the app. But if we're designing for users that use our app pretty rarely, we need to make sure to keep all the interactions discoverable and visible. That way every time they come back to the app, it's like the first time they came back to it. They don't need to remember anything from the previous time because we don't know how long it's been since the last time they've used it. If we want to design for both, then we have our work cut out for us. We need to either design very efficient interaction methods that nonetheless are discoverable and visible, or we need to design two sets of interaction methods. One way that's very discoverable and visible, and one way that's very efficient. We see this with our example of the hotkeys for copy and paste. If you don't know how to use them, you have a way of finding them. So it caters to either novice users or users who haven't used the program in awhile. But because you can also do it with simple hotkeys, it caters to those users who use it more frequently and makes it more efficient for those who are going to be doing it a lot. In deciding what to design, we need to understand what groups, what profiles we're designing for, and use that to inform our design decisions. Inexperienced designers often make big mistakes here. They either try to design for everybody, which rarely works, or they design with no one in particular in mind. And so, certain areas of program are good for some users, others are good for other types of users. An entire program as a whole is not good for any particular type of user. So it's very important that we understand the range of users that we're designing for, and that we make sure the range is actually something that we feasibly can design for.

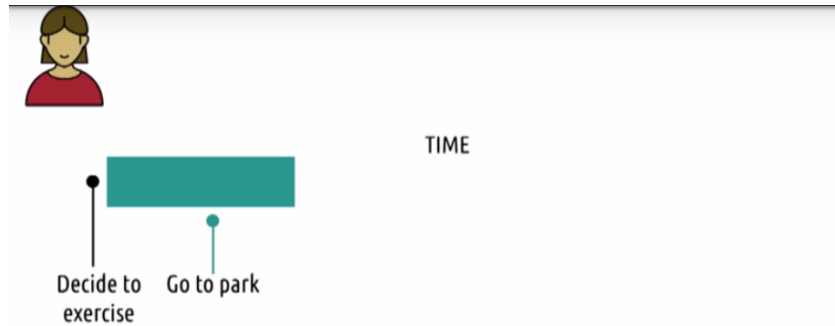
Timeline



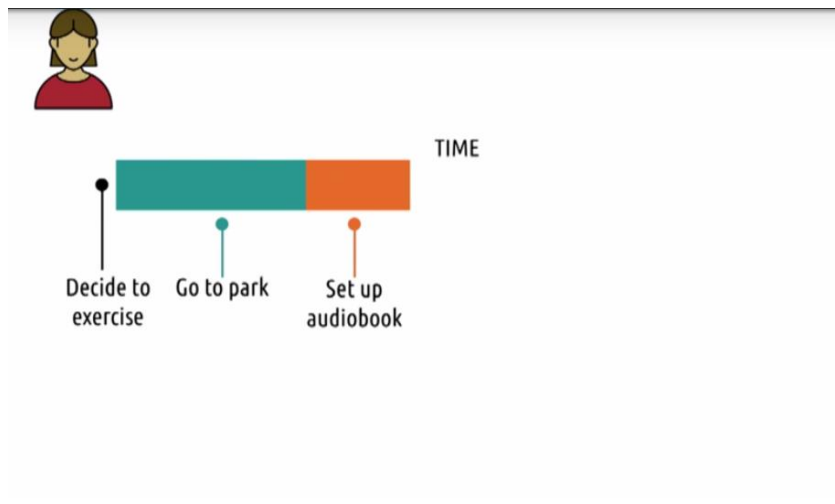
Building on the idea of a persona we can take that person and stretch her out over time and see what is she thinking, what is she doing at various stages of interacting with our interface or interacting with the task at hand. I've also heard this called journey maps although journey maps usually cover much longer periods of time. They cover the entire expanse of the person's life, why they are interested in something and where they are going from here. Timelines can be more narrowed to the specific time which users are interacting with the task or with the program. So our goal is to take that persona and stretched it out overtime.



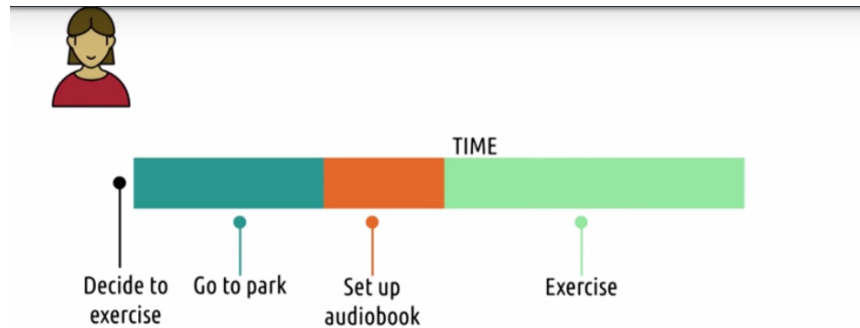
So for our example, what sparks Anika to decide to exercise in the first place? That might be really useful information to know.



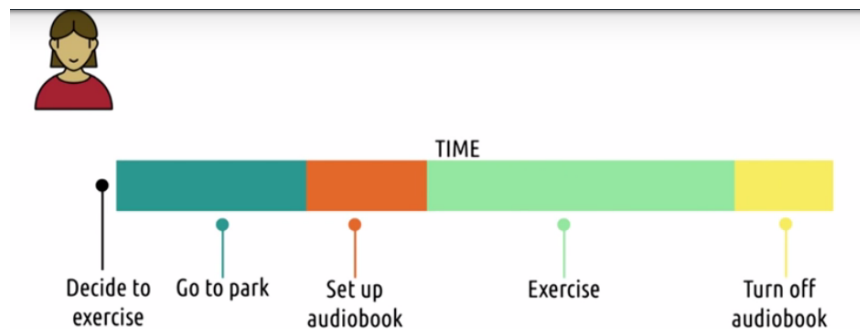
After she decided to exercise, what did she do next? In theory she doesn't just start right there. She goes exercise somewhere she has kind of setup process.



Then what does she do? In this case, maybe she set ups her audiobooks as she actually pushes play, puts her headphones in, and so on.



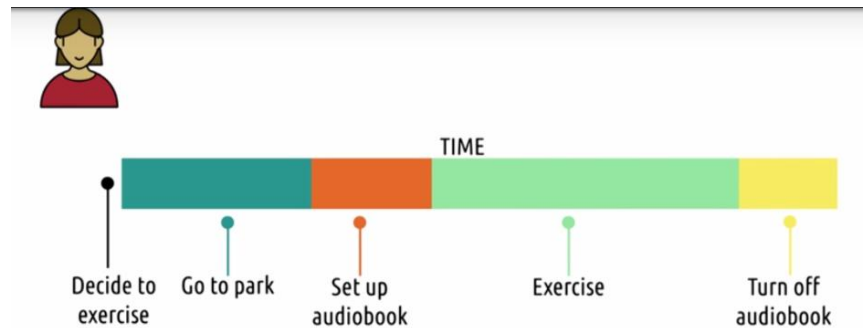
And then there's probably a period of actual exercise in our example.



And then at the end, she turns off the audiobook. The usefulness of drawing this as a timeline is it starts to let us ask some pretty interesting questions. What prompts this person to actually engage in the task in the first place? What actions lead up to the task? How are they feeling at every stage of the task? And can we use that? How would each design alternative impact their experience throughout this process? For example, if our persona for Anika was that she really doesn't like to exercise but she knows she really needs to, then we know her mood during this phase might be kind of glum. We need to design our app with the understanding that she might have kind of low motivation to engage in this at all. If our app is a little bit frustrating to use, then it might turn her off of exercising all together. On the other hand, if Anika really likes to exercise, then maybe she's in a very good mood during this phase. And if she likes exercising on its own, maybe she forgets to even set up the audio book at all. So then we need to design our app with that in mind. We need to design it such that there's something built into it that could maybe remind her that when she gets to a certain location, she meant to start her audio book. So stretching this out as a timeline lets us explore not only who the user is, but also what they're thinking and what they're feeling. And how what we design can integrate with a task that they're participating in. Exploring our different design alternatives in this way allows us to start to

gauge which designs might have the greatest potential to positively impact the user's experience. And they also let us explore what might be different between different users. Our design might need to be different for Anika who loves to exercise, and Paul who hates exercising. This timeline let's us start to explore those more personal elements.

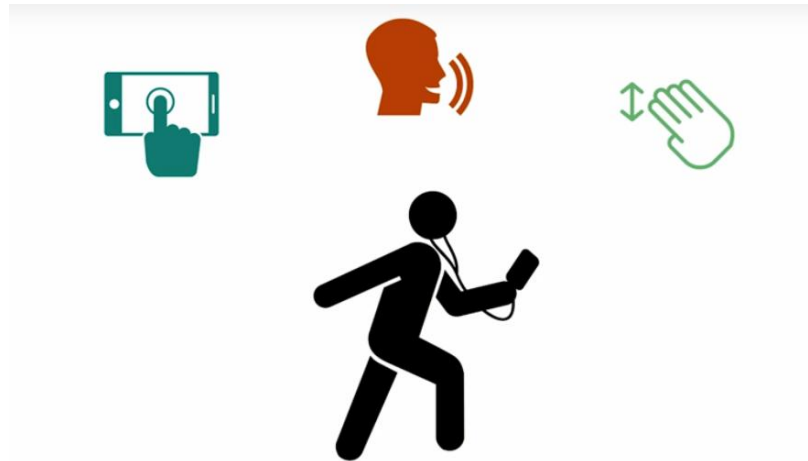
Scenarios and Storyboards



We can create general timelines or routine interactions with our design alternatives, but it's often even more interesting to examine the specific scenarios users will encounter while using our interfaces. Rather than outlining the whole course of interaction, scenarios let us discuss specific kinds of interactions and events that we want to be able to handle. These are sometimes also referred to as storyboards, sequences of diagrams or drawings that outline what happens in a particular scenario. The difference between timelines, and storyboards, and scenarios is that timelines, in my experience at least, tend to be pretty general. They're how a routine interaction with the interface, or a routine interaction with a design alternative goes. Scenarios and storyboards are more specific. They're about a particular person interacting in a particular way, with particular events coming up. So let's build one of these scenarios.

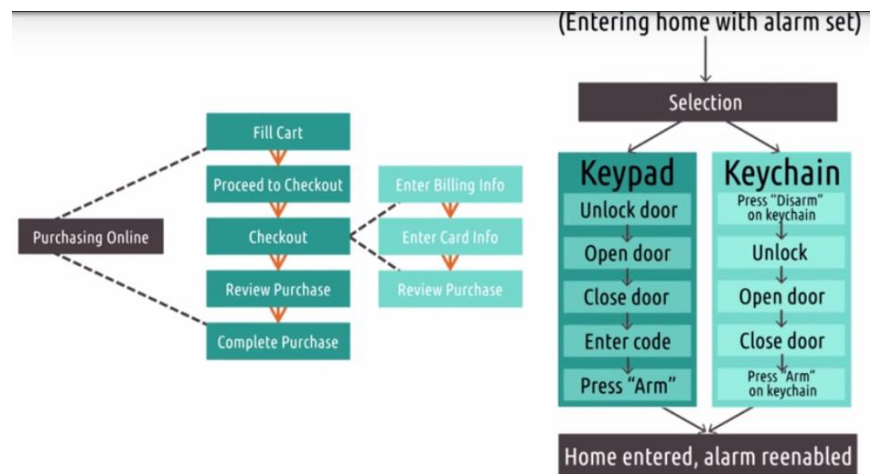


Morgan is out jogging when a fire engine goes by. It's so loud that she misses about 30 seconds of the book. How does she recover from this?




Well we want to go through that question with each of our different design alternatives. For our touch interface, she would need to stop, pull out her phone, turn on the screen, and pause the book. For our voice interface, she would have to wait for the fire engine to finish passing, and then say rewind, because chances are if it's too loud for her to hear her book, it's probably too loud for her phone to hear her voice command. But for a gestural interface she could simply make the gesture that would allow her to pause the book, and then play again when the fire engine is finished passing. Ideally we'd like to outline several such scenarios and explore them for various personas and design alternatives. Of course, now we're reaching three degrees of freedom. So it's not crucial that we explore every possible combination of persona, scenario, and design alternative. This is a more fluid process of exploring what ideas have potential and what ideas are worth exploring further. We might find there are certain combinations of scenarios and personas that we really care about, that completely rule out certain design alternatives. If we really care about allowing her to exercise in a loud area, that goes ahead and tells us that that voice interface might not be the direction we want to go in. Or for another example, if our earlier need finding exercises revealed a significant number of exercisers carry weights or other things in their hands, then gestural interfaces start to look a lot less promising. Now as video technology has gotten more and more ubiquitous, story boards have also taken on a different form in a lot of prototypes that I've seen. There's been an emergence of what is called, video prototyping. Which is basically doing a video mockup of what someone would actually look like interacting with a certain interface, to show to people so that they can see whether or not that would actually be something that would help them in their everyday life. So storyboards, taken to an extreme, could actually be a video mockup of what it would be like to use the finished product.

User Modeling




In our unit on principles, we talk about task analysis, including things like cognitive task analysis or human information processor models. Performing those analysis as part of our need finding also gives us a nice tool for exploring our design alternatives. Using this, we can start to look at how exactly the goals, operators, methods, and selection rules of the Gomes model map up to the ideas of our design alternatives. How does the user achieve each of their goals in each interface? How relatively easy are the goals to achieve between the different design alternatives? With the results of our cognitive task analysis, we can start to ask some deeper questions about what the user is keeping in mind as well. Given what we know about things competing for our user's attention, what are the likelihoods that each interface will work? In some ways, this is a similar process to using personas we outlined earlier, but with a subtle difference. Personas are personal and meant to give us an empathetic view of the user experience. User models are more objective and meant to give us a measurable and comparable view of the user experience. So ideally, the result of this kind of analysis is we would be able to say that the different alternatives have these different phases and these different phases have different efficiencies or different speeds associated with them. So, we could start to say exactly how efficient one design is compared to another.

Quiz: Exercise: Design Alternatives

 Match the advantage to the method.					
	Personas	Scenarios	User Profiles	Timelines	User Modeling
Includes the task context					
Includes the user's context					
Delineates the target audience					
Targets general workflows					
Captures activity over time					
Attempts to capture potential edge cases					

In this lesson we've covered several different ways of developing design alternatives. Each method has its advantages and disadvantages. So, let's start rap the lesson up by exploring this with an exercise. Here are the methods that we've covered and here are some other potential advantages. For each row, mark which of the different methods possesses that advantage. Note that these might be somewhat relative, so your answer might differ from ours and that's perfectly fine.

 Match the advantage to the method.					
	Personas	Scenarios	User Profiles	Timelines	User Modeling
Includes the task context					
Includes the user's context					
Delineates the target audience					
Targets general workflows					
Captures activity over time					
Attempts to capture potential edge cases					

So personally here's my answer. For me scenarios are really the only ones that include the entire task context. You can make the case that personas and timelines do as well, but I tend to think that these are a little bit too separated from the task to really include the context. Personas and user profiles, on the other hand, do include the users context. They include plenty of information about who the user is, why they're doing this task, and what their motivations are. You could make the argument as well, that scenarios and timelines include the user's context. Because the way we describe them, they're instances of the personas being stretched out over a scenario or over time. User profile probably do

the cleanest job of delineating the target audience. With our personas we have kind of a fuzzy idea of who are different users are. But our user profiles really formally articulate the space of your users in which we're interested. As far as general workflows that's what user modeling is really good at. It really outlines the phases or steps or operators that users use in a general sense. You could say the same thing about timelines to a certain extent although timelines are more focused on what the user is thinking and feeling and less on their actual workflow with regard to the task. As far as capturing activity over time, scenarios, timelines, and user modeling all have an element of time in what they analyze. And possibly one of the bigger benefits of using scenarios is they allow us to capture potential edge cases more easily. Timelines, user models, user profiles and personas are all about the general user or the general routine interaction with the task. But scenarios let us pose interesting and novel situations so that we can kind of mentally simulate how our different design alternatives will work with that scenario. For example, we wouldn't say that a fire engine going by Morgan, which is listening to an audiobook, is a routine thing, so it probably wouldn't come up in our timeline or in our user modeling. But we can develop a scenario that explores how is she going to deal with that particular event. And while it might seem a little silly to focus so much on edge cases, as we do more and more design, we start to discover that there are a lot of edge cases. They're all different. But a lot of our time is spent dealing with those unique circumstances that fall pretty far outside the routine interaction with the program.

Exploring Ideas



So let's apply these techniques to some of the ideas that I came up with earlier. The first thing I might do is go ahead and rule out any of the ideas that are just technologically unfeasible. Coming up with those wasn't a waste of time because they're part of a nice broad free flow brainstorming process. But skin based interfaces and augmented reality, probably are not on the table for the immediate future. I might also rule out the options that are unfeasible for some more practical reasons. We might a small team of developers, for example. So, a dedicated wearable device isn't really our expertise. Now the one I might do next is create some timelines, covering a sequence of events in exercising to use to explore these alternatives further. I might notice that the users I observed and talked with, valued efficiency in getting started. They don't want to have to walk through a complex set up process every time they start to exercise. I might also use my user persona's to explore the cognitive load of the users in these different alternatives. They have a lot going on, between monitoring their exercise progress, observing their environment, and listening to the book. So, I'm going to want to keep cognitive load very low. Now granted, we always want to keep cognitive load pretty low, but in this case, the competing tasks are significant enough, that I want to sacrifice features for simplicity, if it keeps that cognitive load pretty manageable. Now based on these timelines and these personas, I would probably end up here with three design alternatives that I want to explore. One is a traditional touch interface, a smartphone app. Then unfortunately it means the user is going to have to pull out their phone whenever they want to take a note. But if I can design it well enough that might not be an issue. I also know that approach gets me a lot of flexibility so it's good to at least explore it. A second approach is gestural interfaces. I know that people aren't usually holding their device while exercising. So it would be great if they had some way of interacting without pulling out their phone. Gestures might let us do that. Now in our gesture recognition is in its infancy, but we might be able delivered smart watched technology or something like a Fitbit to support interaction via gestures. A third approach is a voice interface, I know people generally aren't communicating verbally while exercising, so why not a voice interface? That can even double as the note taking interface. So now that I have three design alternatives that I'm interested in exploring, I would move on to the prototyping stage which is building some version of these that I can test with real users.

Exploring HCI: Design Alternatives

Design alternatives are where you explore different ways to facilitate the user's task. If you've already chosen to focus on a certain area of technology like wearable devices and just general interaction. Then in some ways, you've put the cart before the horse. You've chosen your design before exploring the problem. As a learning experience though, that's perfectly fine. It's fine to say, I want to explore augmented reality and I'm going to find a task that lets me do that. You're still exploring whether or not augmented reality is the right solution for that task. You're just altering the task, if not instead of altering the design if not. For other domains, you might need to make sure to create personas for different stakeholders in healthcare, for instance. We would want to make sure any interface you design takes into consideration nurses, doctors, patients, managers family members and more. So you'd want to create personas for all those different types of people, as well and make sure to explore scenarios that affect each stakeholder.

Conclusion to Design Alternatives



The goal of the design alternative stage of the design life cycle, is to generate lots of ideas, and then synthesize those ideas into a handful worth exploring further.



So we started with some heuristics for generating lots and lots of ideas, through both individual and group brainstorming.



Then we proposed some methods for exploring those ideas, and deciding which one to pursue. Now these were all thought experiments. We haven't actually gotten to the point of designing any of these interfaces yet. That will be the next stage. At the end of the design alternative stage, we want to select a handful of designs that are worth carrying forward and prototyping, to then give the users for actual feedback.

3.5 Prototyping

Compiled by Shipra De, Summer 2017

Introduction to Prototyping



[MUSIC] So we've talked to our users. We've gathered some understanding of what they need. We've created some ideas for how we might address their need and we've mentally simulated those different alternatives.



Now it's time to start actually making things we can put in front of users. This is the prototyping stage. Like brainstorming design alternatives, this involves looking at the different ideas available to us and developing them a bit. But the major distinction is that in prototyping, we want to actually build things we can put in front of users. But that doesn't mean building the entire interface before we ever even

have a user look at it. We want to get user feedback as quickly and rapidly as possible. And build up more sophisticated prototypes over time as we go through the design life cycle.



So we'll start with low fidelity prototypes, things that can be assembled and revised very quickly for rapid feedback from users.

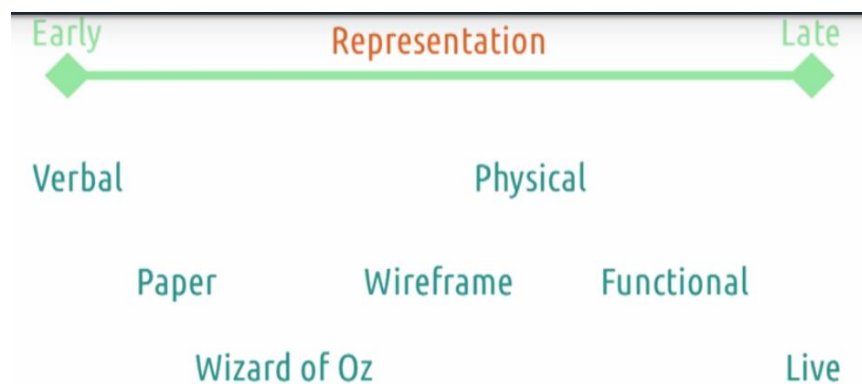


Then we'll work our way towards higher fidelity prototypes, like wire frames or working versions of our interface.

Basics of Prototyping



To discuss prototyping there are a variety of different terms and concepts we need to understand. For the most part, these will apply to where in the prototyping timeline those concepts are used. In the early prototyping, we're doing a very rapid revision on preliminary ideas. This happens on our first few iterations through the design lifecycle. In late prototyping, we're putting the finishing touches on the final design, or revising a design that's already live. This happens when we've already been through several iterations of our design lifecycle. At the various phases, we'll generally use different types of prototypes in evaluations. Now, note that everything I'm about to say is pretty general, there will be lots of exceptions.



The first concept is representation, what is the prototype? Early on, we might be fine with just some textural descriptions or some simple visuals that we've written up on a piece of paper. Later on though, we'll want to make things more visual and maybe even more interactive. We only want to put the work into developing the more complex type of prototypes once we vetted the ideas with prototypes that are easier to build. So in a lot of ways, this is a spectrum of how easy prototypes are to build over time. A verbal prototype is literally just a description, and I can change my description on the fly. A paper prototype is drawn on paper, and similarly, I could ball up the paper, throw it away, and draw a new version pretty quickly. But things like actual functional prototypes that really work, those take a good bit of time. And so we only want to do those once we've already vetted that the ideas that we're going to build actually have some value. You don't want to sink lots of months and lots of engineering resources into building something that actually works. Only to find out there's some feedback you could have gotten just based on a drawing on a piece of paper that would have told you that your idea wasn't a very good one.



This brings us to our second concept, which is fidelity. Fidelity refers to the completeness or the maturity of the prototype. A low-fidelity prototype would be something like paper or simple drawings, very easy to change. A high-fidelity prototype would be something like a wireframe or an actual functional working interface, something that was harder to put together. We want to move from easily changeable low-fidelity prototypes to explore our ideas, to higher-fidelity prototypes to really test them out. Note that fidelity and representation are pretty closely related, low-fidelity is really about a prototype that's pretty far from being complete. And the same thing is true for some of our early methods of prototyping. They describe different ideas, but they very heavily correlate what kinds of representations you're going to use for different levels of fidelity.



Now these different kinds of prototypes also lend themselves to different kinds of evaluation structures. Low fidelity prototypes can fine for evaluating the relative function of an interface, whether or not it can do what's it's designed to do. If a user looks at the interface can they figure out what they're supposed to press? You can prototype that was just a drawing on a piece of paper, as opposed to a real functional prototype. Things like wireframes can be useful in evaluating the relative readability of the interface as well. However, to evaluate actual performance, like how long certain tasks take, or what design leads to more purchases. We generally need a higher fidelity prototype, through more iterations of the design lifecycle. So early on, we're really just evaluating whether or not our prototype even has the potential to do what we want it to do. Can a user physically use it? Can they identify what button to press and when? For that we need additional detail like font size and real screen layout. We need a real prototype that looks the way the final interface will look, even if it doesn't work quite yet. And then, to evaluate performance we really need a prototype that's working, or close to working, to evaluate certain tasks.



And then the final concept we need to understand, is the scope of the interface. Is it a horizontal prototype or a vertical prototype? Horizontal prototypes cover the design as a whole, but in a more shallow way. Vertical prototypes take a small portion of the interaction and prototype it in great detail. So for example, if we were designing Facebook, we might have a vertical prototype specifically for the status-posting screen and a horizontal prototype for the site in general. Now, in my experience, we usually start with horizontal prototypes earlier on, and move toward the deeper vertical prototype later. But in reality, you'll likely move back and forth among these more frequently throughout your iterations through the design lifecycle.

Representation

Fidelity

Evaluation

Scope

So, these are four the main concepts behind prototyping. There are other questions we might ask ourselves as well. Like whether we're prototyping iterative or revolutionary changes, and the extent to which the prototype needs to be executable. But in many ways, those fall under these previous concepts.

Tradeoffs of Prototyping

Prototyping is largely about the tradeoffs we have to make. Low fidelity prototypes like drawings are easy to create and modify, but they aren't as effective for detailed comprehensive evaluations. High fidelity prototypes, like actual working interfaces, can be used for detailed feedback and evaluation, but they're difficult to actually put together.



So, our goal is to maximize these trade-offs. We want to start with easier, low fidelity prototypes to get initial ideas, to evaluate big designs and big plans, and make sure we're on the right track.

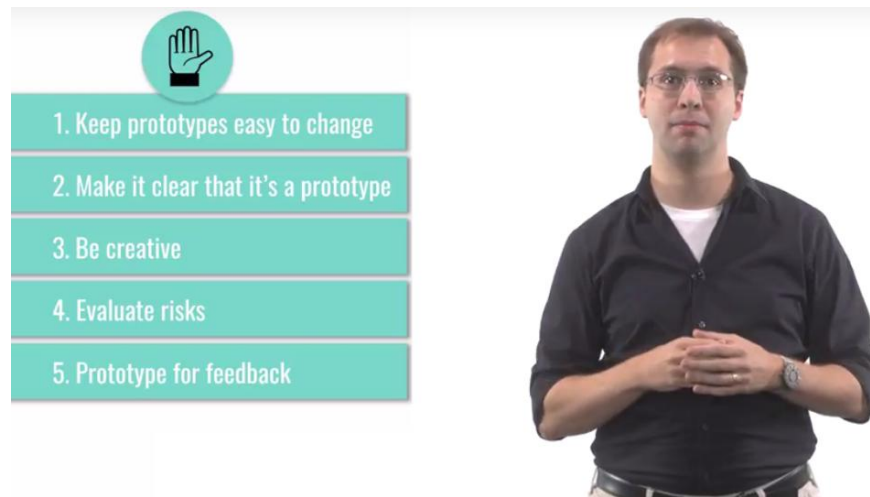


Then as we go along, we can move to the higher fidelity prototypes that take more time assemble because we have initial evidence that our designs are actually sound. It's really important here also, to note that our prototypes are prototypes. They aren't complete interfaces. We've discussed the past that designers often have a tendency to jump straight to designing rather than getting to know their users. That's a big risk here as well because we're designing.



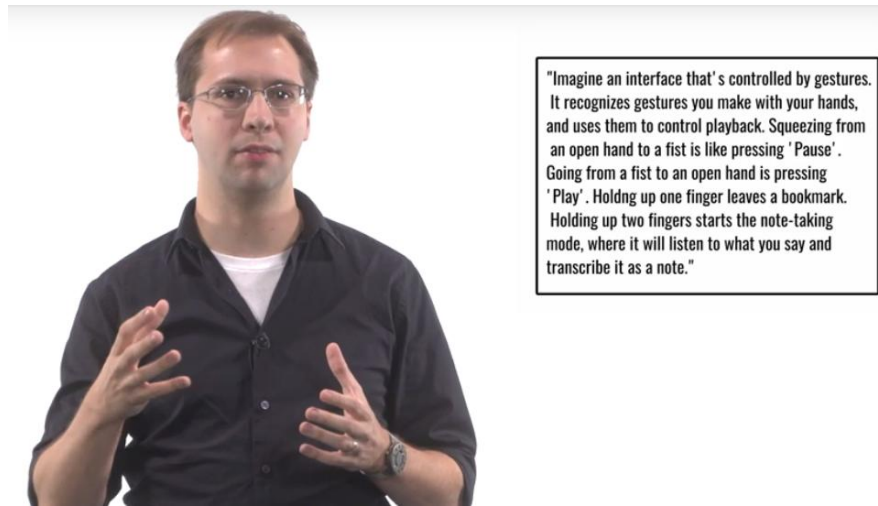
But we're designing specifically to get more feedback. So don't become a runaway train of designing. Design deliberately and get feedback often.

5 Tips: Prototyping



Here are five quick tips for effective prototyping. Number one keep prototypes easy to change, your goal here is to enable to rapid revision and improvement. It's easy to make quick changes to something on paper but it's harder to make it a code or physical prototypes. Number two, make it clear that it's a prototype. If you make a prototype look too good users may focus on superficial elements like colors or font. By letting your prototype look like a prototype you can help them focus on what you're ready to test. Number three, be creative. Your goal is to get feedback. Do whatever it takes to get feedback. Don't let the type of prototype you're designing constrain the type of feedback you can get. If you find your current prototypes don't give you the right kind of feedback, find ones that do. Number 4, evaluate risks. One of the biggest goals of prototyping is to minimize the time spent pursuing bad designs by getting feedback on them early. How much would you lose if you found that users hate the parts of your design that they haven't seen yet? Whenever that answer gets to be longer than a couple hours, try to get feedback to make sure you're not wasting time. Number five, [SOUND] prototype for feedback. The goal of a prototype is to get feedback. You could spend a lot of time focusing on details like font selection and color choice, I know I do, but that's probably not the feedback you need when you're exploring your big alternatives. Prototype for the kind of feedback you want to get.

Verbal Prototypes



At the very simplest, we have verbal prototypes. That means we're literally just describing the design we have in mind to our user. That's probably the lowest fidelity prototype possible. It's literally just telling the user the same thing we tell our co-designers. So it's extremely easy to do, although it can be hard to do effectively. Social desirability bias is big here because it's difficult to describe our idea in a way that allows the participant to feel comfortable disagreeing with us. So we need to make sure to ask for specific and critical feedback. At the same time though, how do we really know that the user understands the desire we're describing? We're working towards becoming experts in the areas in which we're designing, and we don't want to fall victim to expert blind spot by assuming our explanation makes sense to a novice. For that reason, analogies can be powerful tools for explaining prototypes. Describe your interface in terms of other tools the user might already know about. So for example, imagine I was pitching the idea of InstaCart, a grocery delivery company. I might've described it like, it's like Uber for groceries. Uber is a service kind of like taxis, and InstaCart is kind of like a taxi for groceries. That way of describing it in terms of an analogy to another interface can be a powerful way of helping your participant understand your idea more quickly.

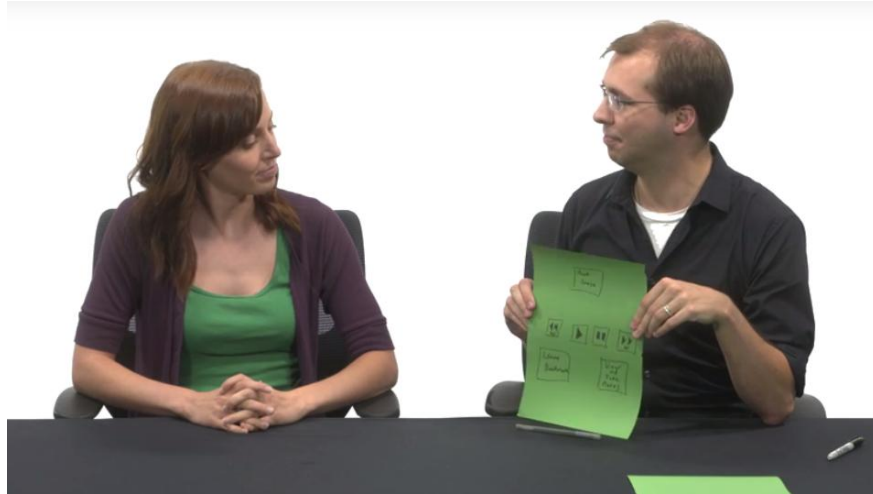
Paper Prototyping



One step above just ‘describing our ideas’ to our user in a verbal prototype would be drawing them out. This is what we call a paper prototype. We could do this for anything from designing an on-screen interface to designing the placement of controls in a vehicle. Let’s go back to our example of designing a way for exercisers to consume and take notes on audiobooks. Let’s imagine one of my design alternatives was just for a very easy-to-use app so that the hassle of pulling out the phone isn’t too distracting. I might start this process simply by drawing up a prototype on paper.



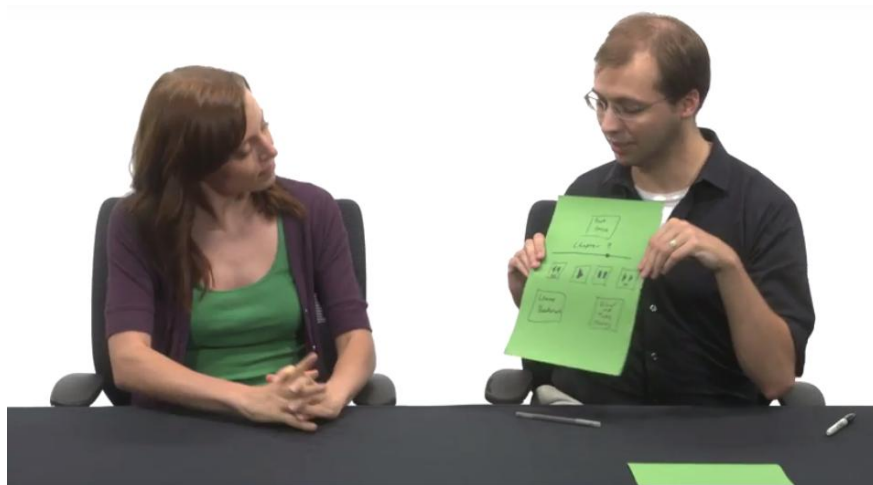
Now I have a paper prototype. Now, I can talk to my user and ask her thoughts on it. We’ll talk more about the kinds of thoughts I’ll ask for when we discuss evaluation. But generally, I can say: hey Morgan, what about this design?



>> Oh. Looks pretty good. It's straightforward. There's play. There's fast forward. You know I would like a way to see where I am in the book, though.

>> That makes sense.

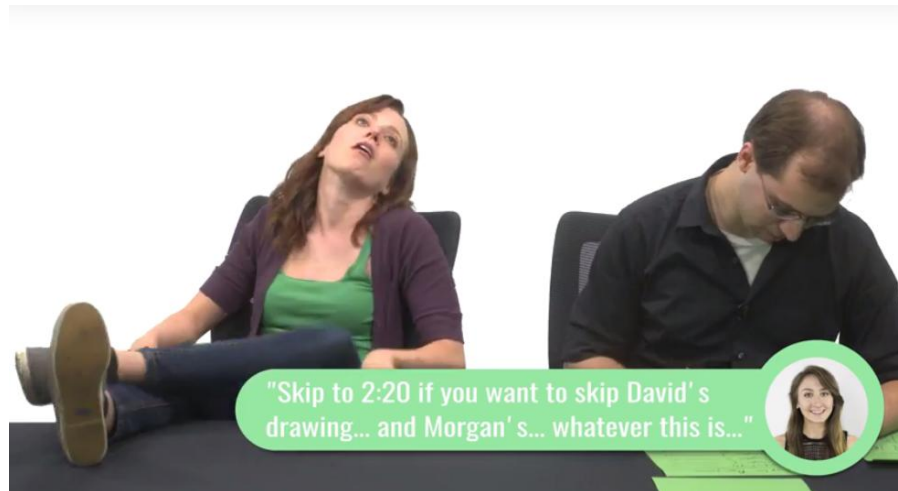
And notice that she didn't comment on color or font or anything like that not because I said "hey, I'm ignoring font right now" but because it's pretty obvious that I'm not caring about font right now. The nature of the paper prototype is it tells the user what kind of feedback we're looking for. We're looking for pretty basic layout information. Now because this is on paper, I can actually immediately revise my prototype and incorporate things that she suggested.



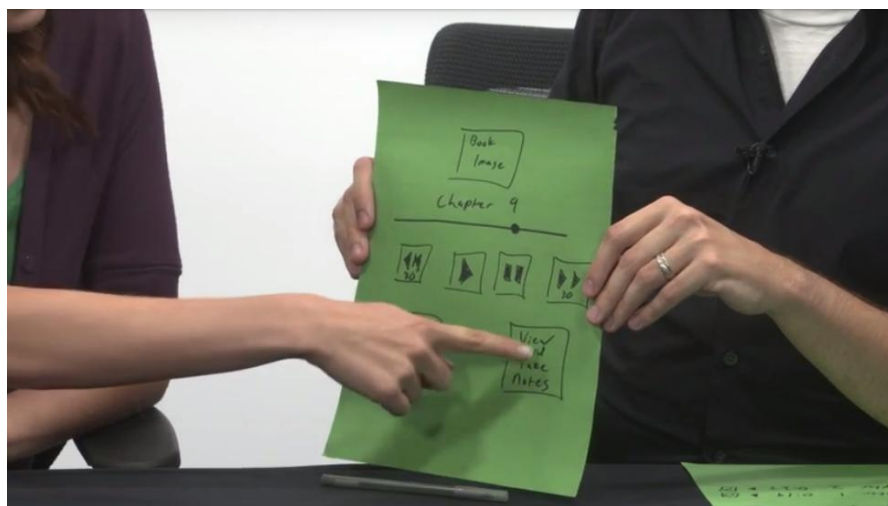
Now, I have a revision based on her feedback, and I can ask: hey, how's that?

>> Looks great, David.

Now paper prototyping isn't only useful for testing out just single interface designs. You can also do some interaction with it. Watch.

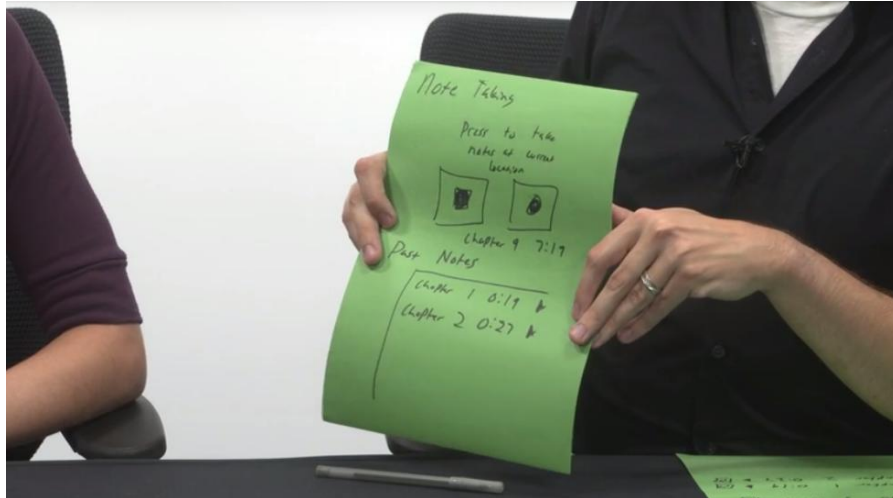


So here I have four screens prototyped. And so, when I give this to Morgan, I can say things like: imagine you're viewing this screen and you want to take a note to attach to your current place in the book. What would you do?



>> Probably press the view the notes button.

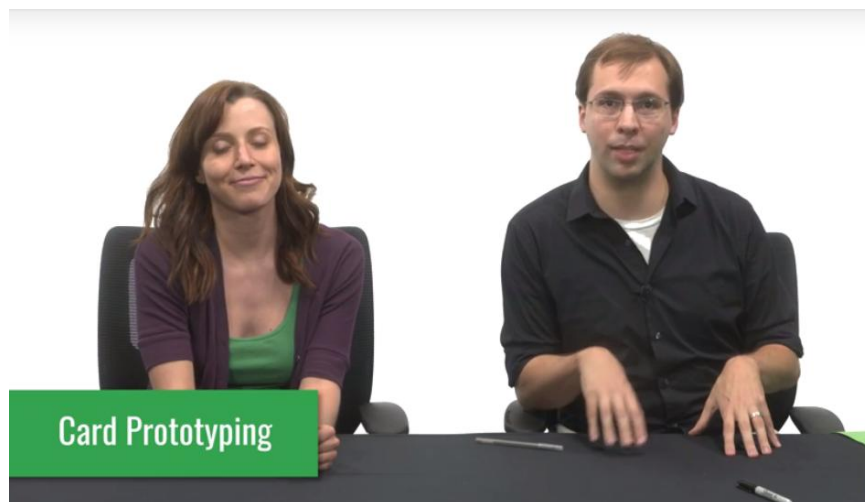
Makes sense. After you press that what you're going to see is this screen for note-taking. What would you do then.



>> Uh... Well, if I want to take a note then I would press the record button.

Yeah! Exactly. And then it would start recording you and it would transcribe what you say and at the end you would press stop to continue.

Interestingly, just doing this I've already noticed that there's really no reason to make her jump to a separate note-taking screen when she want to take a note while listening. It should actually just be a note-taking button here on the main screen.



So just like this I get to walk through some of the interaction with the app on paper and get some ideas like that on how to improve it. This is also called card-based prototyping. The idea is each screen is on a different card and I can quickly sub those cards in and out so we can simulate what it would be like to use the real application. So, that way, I can prototype a decent amount of the interface's interaction with pretty low prototyping effort.

>> Neat.

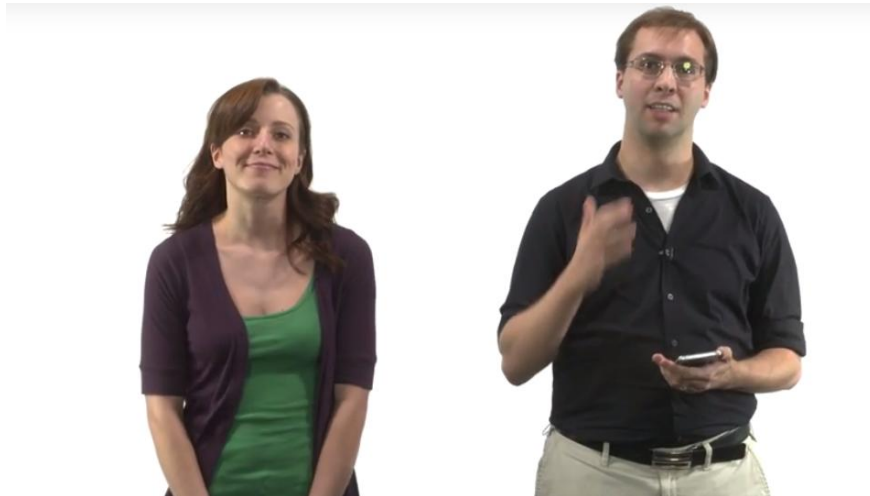
Wizard of Oz

Paper prototyping is great when we're designing flat interfaces for screens. But what if you're designing a voice interface or a gesture interface? How do you prototype that? One way is called Wizard of Oz prototyping. Wizard of Oz prototyping is named as a reference to the famous scene from the classic movie of the same name.



>> Whoa, pay no attention to that man behind the curtain.

The idea here is, that we, behind the curtain, do the things that the interface would do once it's actually implemented. That way we can test out the interactions that we plan to design and see how well they'll work.



So I have Morgan, and we're going to do a Wizard of Oz prototype for an interface that will allow exercisers to consume and take notes on audiobooks. So, I'll start by briefly telling her how the interface would work. Also, this interface is run by voice command. I'll simulate it on my phone. Also you'll say,

play to play the book, pause to pause the book. You'll say, note to enter a note taking view where it will transcribe what you say. And bookmark to just drop a bookmark wherever you are but without pausing. So, whenever you're ready...

>> Let's do it.

>>>> Our wealth consists of desirable things,

>> Pause.

>>>> that is things

>> Play.

>>>> To satisfy human wants directly or indirectly.

>>>> Book mark.

>>>> But not all desirable things are reckoned as well.

>> Note. This book is so good.

I just realized actually I need a way for you stop the note when you're done. So say close note when you're done taking your note.

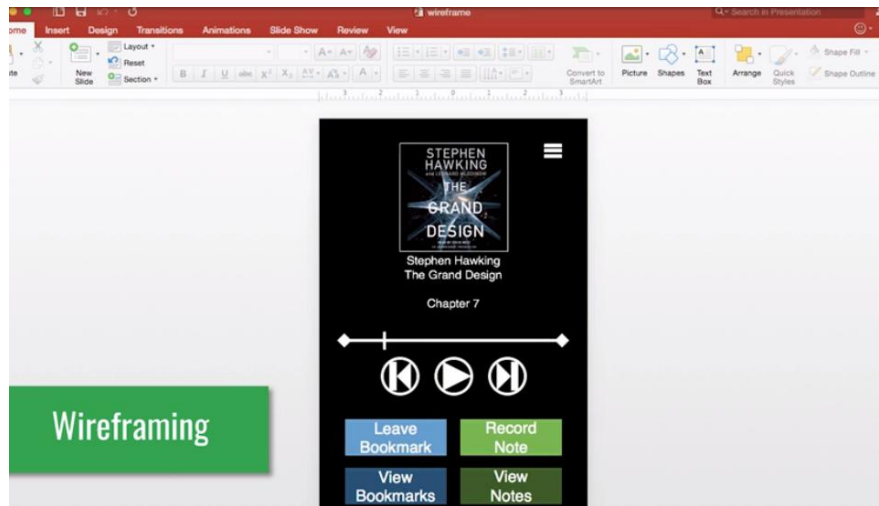
>> Close note.

All right, you can stop. [LAUGH]. Now based on this prototype, I can also ask for some feedback. So Morgan do you think should automatically start playing again when you stop that bookmark? Or should it, what should it do?

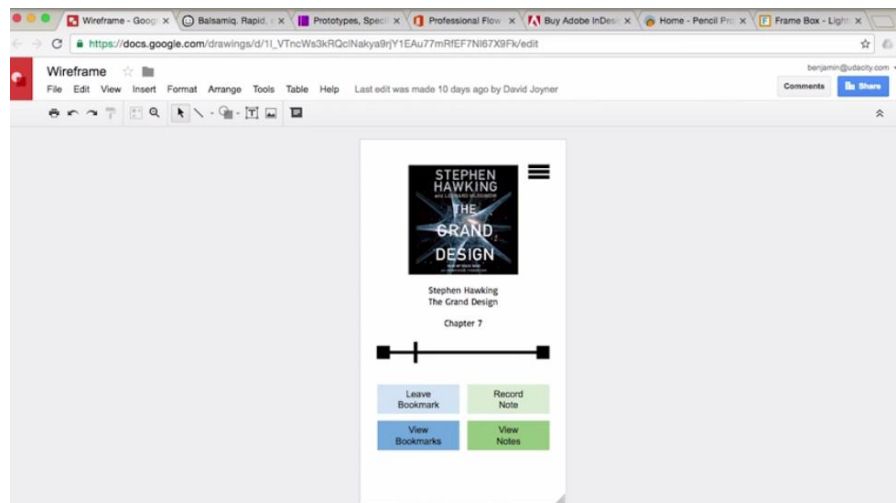
>> Well okay, so I think I'd actually like it to start playing five seconds back. Because I imagine saying, note, is going to step over some of the content.

Yeah, that makes sense, and we can go through this and quickly test out different ideas for how the interaction might work. In practice, Wizard of Oz prototypes can actually get very complex. You can have entire programs that work by having a person supply the requested input at the right time. But as a concept a Wizard of Oz prototype is a prototype where the user can interact authentically with the system. While a human supplies the functionality that hasn't yet been implemented.

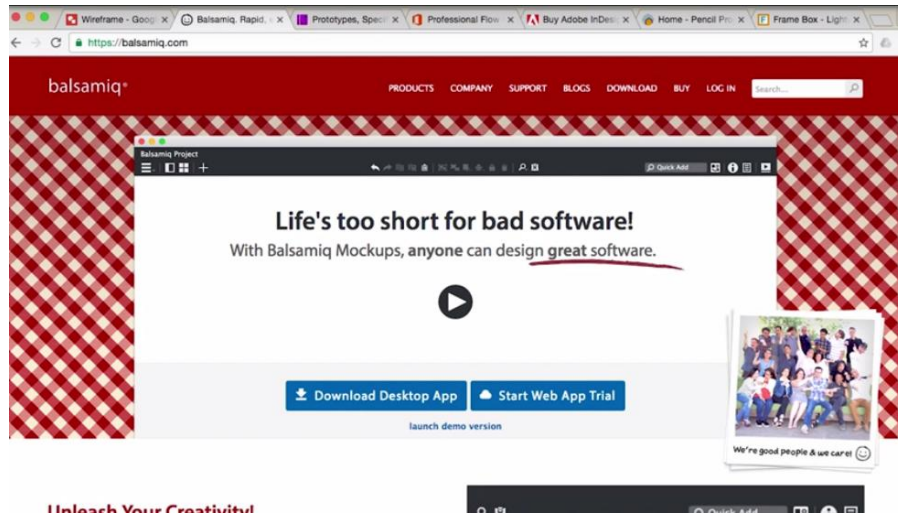
Wireframing



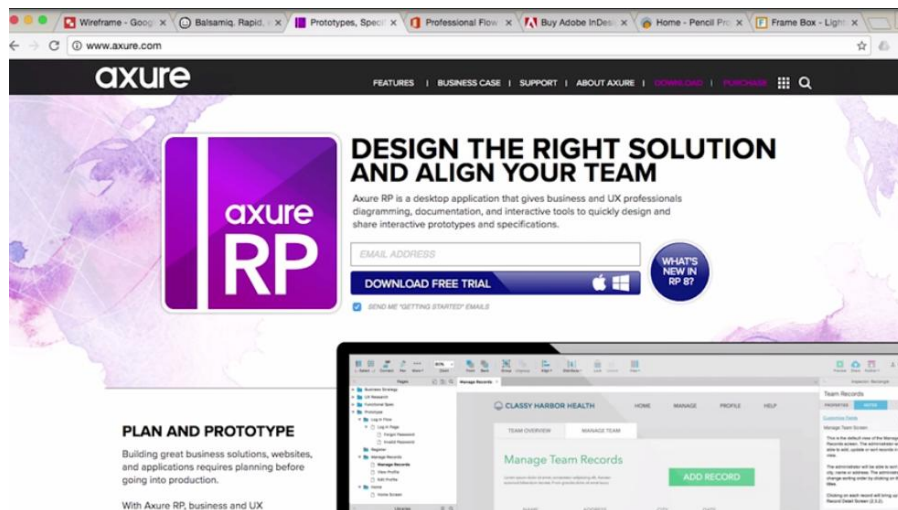
Paper prototyping involved drawing things simply on paper and it can be really for experimenting with overall layouts, especially because it tends to be a lot easier to revise and tweak those pretty naturally. After some feedback though, you'll likely want to start formalizing your designs a little bit more. One way of doing this would be called wire framing. In wire framing we use some more detailed tools to mock up what an interface might look like. For example, my paper prototype from earlier might become a wire frame like this. This lets us experiment with some additional details like font size, colors, and the challenges of screen real-estate. Now there are lots of tools out there for wire framing that come equipped with built-in common widgets and layouts, but you can also do some rudimentary wireframing in something as simple as PowerPoint.



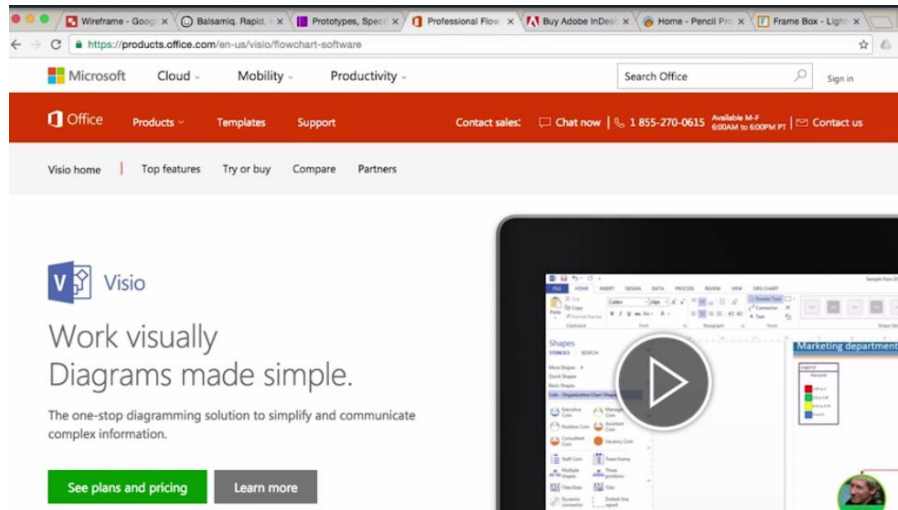
Google drawings can be used the same way as well. So you don't need to get super fancy, although if you do a lot of wire framing you'll probably want to find a more streamlined tool.



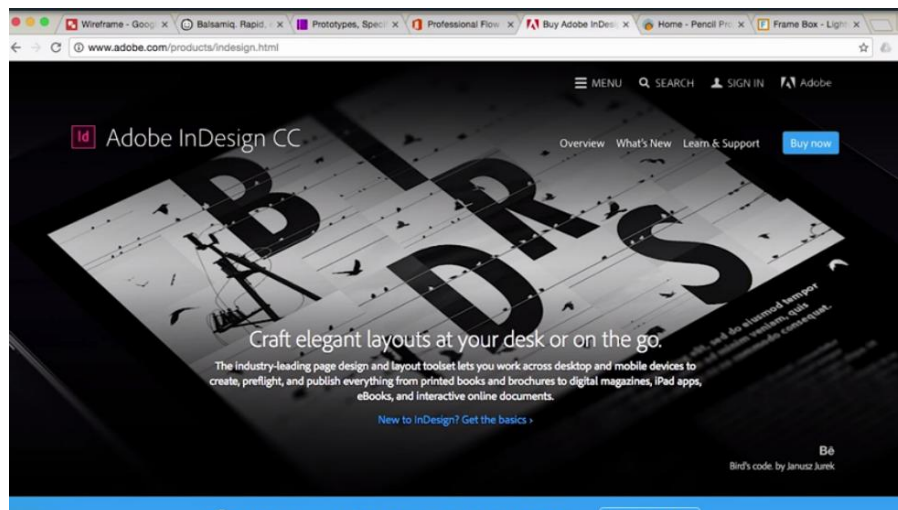
Some of the more popular paid products include Balsamiq...



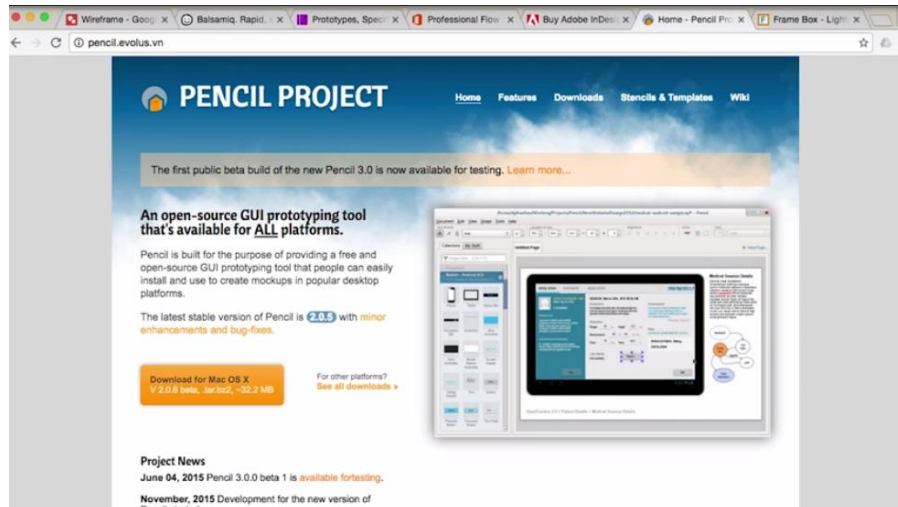
...and Axure. These are both targeted at professionals working in user interface or user experience design. Especially on teams and collaborating with a lot of people.



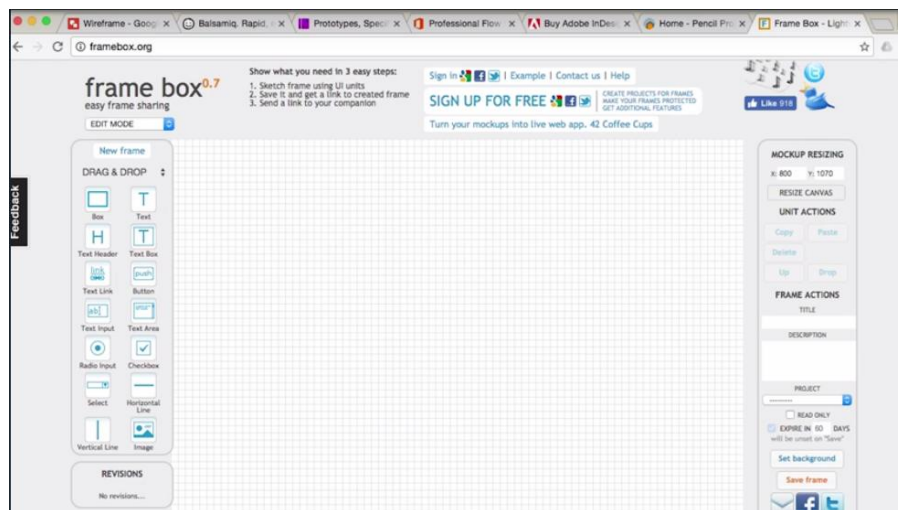
If you're familiar with the suites of tools from either Microsoft or Adobe, then Visio...



...or InDesign might also be great options for you to use because you're already somewhat familiar with those interfaces.



But you don't actually have to buy a tool to do good wireframing. There exists some free to use tools out there as well, like the Pencil Project...



...and Frame Box. Those are great to use, especially if you're just getting started. And of course, these are just the popular ones that I know about right now. There are almost certainly more out there that I'm not familiar with and more will surely come available, so check with your classmates or colleagues to see what they would recommend. I'm personally really excited to see what kind of prototyping options emerge for areas like virtual reality and augmented reality, where you can't really prototype on a 2D canvas like these.

Physical Prototypes




Wire framing is great for prototyping on-screen interfaces, but again, what if you're working on something more physical or three-dimensional? In that case, you might want to construct a physical prototype. But let's be clear, it doesn't have to actually work. That's where a lot of designers get tripped up. They think to get good feedback on a design they have to have a working version, but you don't. There are lots of elements you can test without actually implementing anything. So let's go back to our example of designing a way for exercisers to take notes on audio books.



One of my alternatives might be a Bluetooth device that synchronizes with the phone with buttons for different interactions. The individual will hold this while exercising and interact by pressing different buttons for play or pause or take a note. I've prototyped this just by taking my car's key fob, and we could just say that, pretend this button does this and this button does this. It's probably not the exact shape that I want, but it's pretty close. It's probably about the same size. And I can test the general idea of pressing buttons while exercising with this. And I can actually do a lot of testing with this. I can tell Morgan how it works and watch carefully to see if the buttons she presses are the right ones to


evaluate the intuitiveness of the interface. Or I could just ask her to go running while holding it and give me feedback on whether or not holding something physical like this in her hand throws off her routine at all. I can do a lot of prototyping without a working version.

Quiz: Exercise: Prototyping Pros and Cons

 Match the advantage to the method.

	Verbal Prototypes	Paper Prototypes	Card Prototypes	Wizard of Oz	Wireframing	Physical Prototypes
Revisable during interaction						
Disguises superficial details						
Simulates user interaction						
Easily distributable to remote users						
Supports prototyping look and feel						
Allows mobility during evaluation						

In this lesson we've covered various different methods for prototyping. Each method has its advantages and disadvantages. So let's start to wrap up the lesson by exploring this with another exercise. Here are the methods that we've covered and here are some of the potential advantages. For each row, mark the column to which that advantage applies. Note that as always, these are somewhat relative, so your answer might differ from ours.

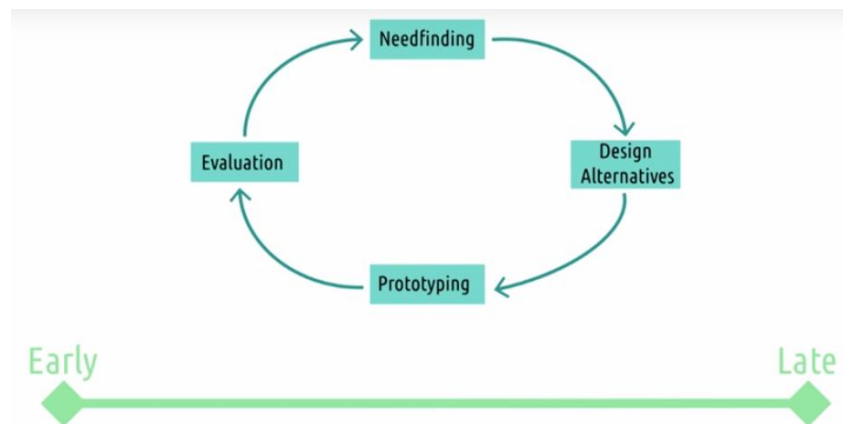
 Match the advantage to the method.

	Verbal Prototypes	Paper Prototypes	Card Prototypes	Wizard of Oz	Wireframing	Physical Prototypes
Revisable during interaction						
Disguises superficial details						
Simulates user interaction						
Easily distributable to remote users						
Supports prototyping look and feel						
Allows mobility during evaluation						

So here are my answers. First when we're talking about things that are revisable during interaction, we're talking about things that I as the experimenter can get feedback from my user and immediately change my prototype. So if they say that that button label doesn't really makes sense, I can cross out that button label and immediately change it. That makes sense for prototypes that are very low fidelity. Verbal prototypes, I can immediately say okay, then let's make it the way you just described. Paper prototypes or card prototypes, I could quickly erase or cross out something on my prototype and change it. Wizard of Oz is similar. Since I'm running what's going on behind the scenes, I can just

change the way I'm running it. Those four prototypes, because they're more low fidelity, also disguises superficial details. No one is going to look at a prototype that I drew by hand and say they don't like the font. No one is going to listen to me run a Wizard of Oz prototype for a voice interface and say, I don't like the voice that you're using. These help us focus on the overall patterns of interaction and disguise some of the superficial elements that users would often have a tendency to get distracted by. However, as we prototype, we need to move from designing interfaces to designing interactions. Verbal prototypes and paper prototypes don't really cover interactions, they cover showing something and asking the user what they think, but they don't really go further than that. Card prototypes, Wizard of Oz prototypes, to a certain extent wireframing and to a certain extent physical prototypes all let us actually simulate the user interaction. With a card prototype, we're actually saying if you did that, then you would see this, so they can walk through the pattern of interaction. Wizard of Oz, we can simply call out or describe or simulate, this is what would happen if you do what you just described. Now, wireframing you could do more like a paper prototype, where it's just a simple wire frame, but more generally, we use wire frames when we're ready to actually show different interfaces and the movement between them. Similarly with physical prototypes, the main reason why we would do a physical prototype is to hand a user, and say, pretend you're jogging, or pretend you're working in your office. How would this interact with what you're actually doing? We're simulating the way they would physically use it. Now among all of these, the wire frames are really the ones that are most easily distributable to remote users. You can make an argument that we can send scans for a paper prototypes but generally, a paper prototype isn't just about what's on paper. It's also about the conversations and descriptions that we're having around it and asking users what they think about certain elements. Whereas a wire frame is more about a general impression that users get. You can make the argument that paper prototypes can be sent easily, as well. But for me, I would only share wire frames with remote users. Now prototyping look and feel is really just the inverse of disguise and superficial details. Look and feel is really about those superficial elements that have a significant user impact, but are more easily modifiable within an overall pattern of functional interaction. So just as the earlier low fidelity prototypes support disguising details, the later ones support prototyping look and feel. As computers become more ubiquitous, and users are moving around while interacting with interfaces more and more, allowing mobility is really valuable. Wizard of Oz, since we're just calling things out to the user, let them move around, and same with physical prototypes. We can actually hand them to a user and have them physically interact, the way they would with the actual interface

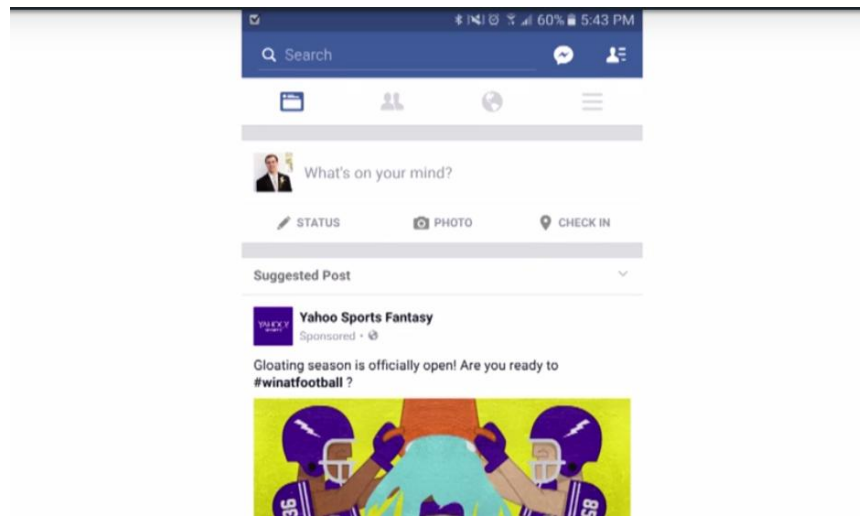
Design Life Cycle Revisited



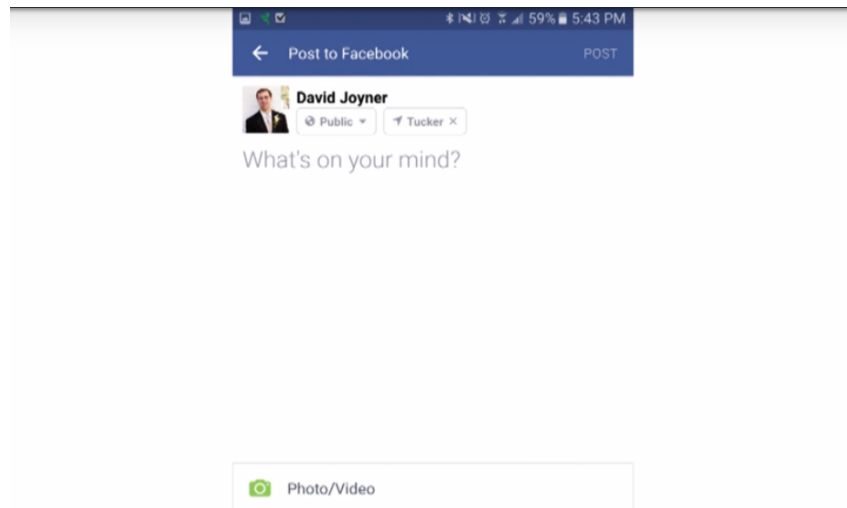
At this point, there's a risk of a major misconception that we should cut off right now. We started with need finding, then develop some design alternatives, and now we're prototyping. We've talked about how prototyping follows a timeline to low fidelity to high fidelity prototypes, from early to late prototyping. We might think that we move on to evaluation when we're done prototyping. That's not the way the design life cycle works though. We go through this cycle several times for a single design and a single prototype corresponds to a single iteration through the cycle. So we did some initial needfinding, we brainstormed some alternatives, and we prototyped those alternatives on paper. We don't jump straight from doing them on paper to doing them via wire framing or doing a functional prototype. We take those prototypes and we use them for evaluation. We evaluate those paper prototypes with real people. The results of that evaluation tell us if we need to go back and understand the task even better. Those results help us reflect on our alternatives as a whole, maybe come up with some new ones. Then, equipped with the results of that evaluation, that additional needfinding, and that additional brainstorming, we return to the prototyping phase. If our prototype seemed to be pretty successful and pretty sound, then maybe it's time to raise the fidelity of it. Maybe we take it from a paper prototype and actually do some wire frames, or do a car prototype around the actual interaction. If it wasn't very successful though, when we reach back here, we're going to do a different paper prototype, or a different low fidelity prototype, and then go to evaluation again. Each time we develop a new prototype we go through the same cycle again. Now that might sound extremely slow and deliberate but we also go through this on a very different time scales too. So for example, after we've gone through needfinding and designing alternative brainstorming, we can develop a paper prototype. We give it to a user and get their evaluation. They say that they don't like it. We ask them why, we ask them to describe what about their task isn't supported by that interface. That's in some ways another needfinding stage. Then we brainstorm real quick how we could resolve that. Maybe we just do that while we're sitting with that user and think it didn't support this element of what they described, but I could add that pretty quickly just by making this button or this function more visible. Now we very quickly have a new prototyping just by sketching out that addition to that paper prototype and now we can do it again. This cycle could take one minute. We could take one prototype, put it in front of a user,

get their evaluation, figure out what they liked and didn't like, brainstorm a way to fix that, and then immediately revise it and try it again. We can go through this very, very quickly. We could also go through this very slowly, we could have prototypes that take months to develop. And generally that's why we only want to do that after we've gone through the cycle a few times. Because if we're going to take months to develop a prototype, we want to make sure we're probably going to get some pretty good evaluations on it. And we can make sure of that by prototyping the elements in lower fidelity first.

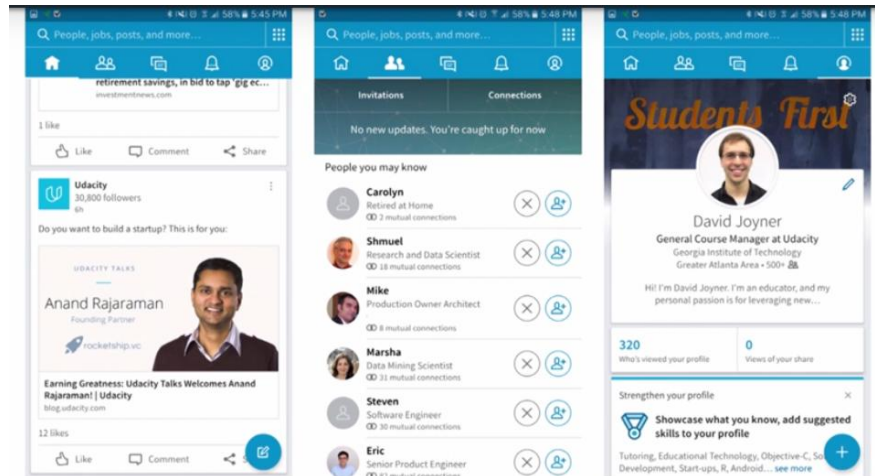
Multi-Level Prototyping



There's one other misconception that I've seen in some designers I've worked with that I feel is also worth explicitly acknowledging. All your prototypes don't have to be at the same level, at the same time. Take Facebook for example. Facebook is a complete app already implemented.



Imagine that Facebook wanted to redesign their status update box, which they've done pretty recently and have probably done since I recorded this. Just because the interface is complete in other ways doesn't mean that all future prototyping efforts need to be similarly high fidelity. They don't need to implement an entire new status composition screen just to prototype it. They can prototype it in lower fidelity with sketches, or wire frames, put that in front of users, get their feedback, before ever actually implementing it into a functional prototype or a working part of the website. This applies particularly strongly to the design of apps or programs with multiple functions.



So take something like the LinkedIn app. It has a number of different functions like editing your own profile, or connecting with others, or browsing your news feed. Each of these individual screens has its own tasks and interactions. And moving amongst them, is itself a task or a type of interaction. Trying to design all the screens and the transitions among them all at the same time is likely far too much. So we could take the bottom-up approach, where we would design the individual screens first, and then design the app experience as a whole. Or we might take the top-down approach and design the overall experience of moving between these different screens, and then design the contents of the individual screens. The point of this is that at any time, prototyping can and should exist at multiple levels of fidelity.

Exploring HCI: Prototyping

If you're working in an application area that relies on traditional screens and input methods, your prototyping process might be pretty straight forward. You'll go from paper prototypes, to wireframes, to exploring iteratively more complete versions of the final interface. For a lot of emerging domains though, you'll have to get somewhat creative with your prototyping. For things like to stroll or voice interaction, you can likely use Wizard of Oz prototyping by having a human interpret the actions or commands that will ultimately be interpreted by the computer. For augmented reality or wearable devices though, you might have to get even more creative. So, take a second and brainstorm how you might go about prototyping in your chosen field. Remember, your goal is to get feedback on your ideas with the user early. What can you create that will get you to that feedback?

Conclusion to Prototyping



In this lesson, we've talked about several methods for prototyping. Our goal is to employ a lot of methods to get feedback rapidly, and iterate quickly on our designs. Through that process, we can work our way towards creating our ultimate interface. The main goal of this prototyping process has to been to create designs we can evaluate with real users. We're obviously not going to deploy a hand draw interface to real customers, its value is in its ability to get us feedback. That's what the entire design life cycle has been leading towards, evaluation, evaluation our ideas, evaluating our prototypes, evaluation our designs. That user evaluation is the key to user centered design. Focusing on user evaluation insures that our focus is always on the user's needs and experiences.



So now that we've researched the user's needs, brainstormed some design alternatives, and created some sharable prototypes, let's move on to actual evaluation.

3.6 Evaluation

Compiled by Shipra De, Summer 2017

Introduction to Evaluation



The heart of user-centered design is getting frequent feedback from the users. That's where evaluation comes into play. Evaluation is where we take what we've designed and put it in front of users to get their feedback. But just as different prototypes serve different functions at different stages of the design process, so also our methods for evaluation need to match as well.



Early on, we want more qualitative feedback. We want to know what they like, what they don't like, whether it's readable, whether it's understandable. Later on, we want to know if it's usable. Does it actually minimize your workload? Is it intuitive? Is it easy to learn?

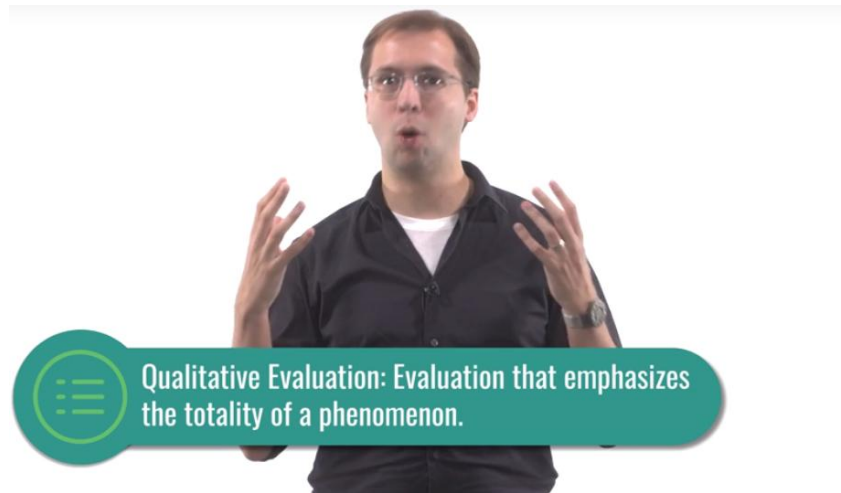


Then at the end, we might want to know something more quantitative. We might want to actually measure, for example, whether the time to complete a task has changed, or whether the number of sales has increased.

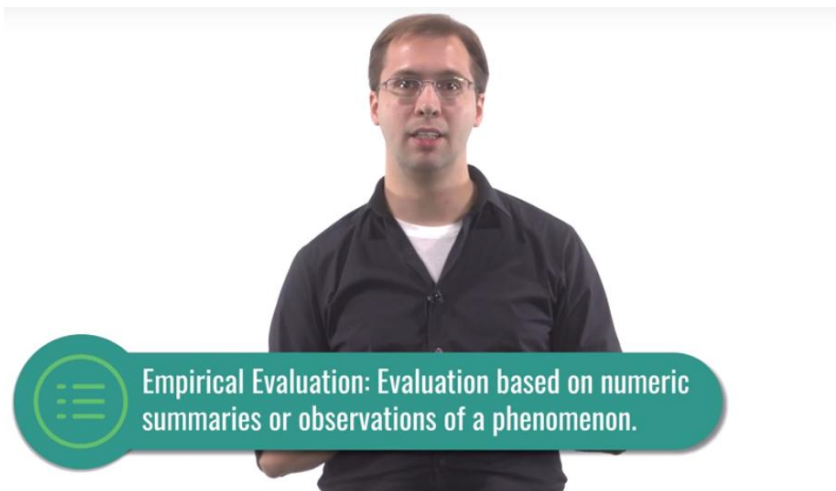


Along the way, we might also want to iterate even more quickly by predicting what the results of user evaluation will be. The type of evaluation we employ is tightly related to where we are in our design process. So in this lesson, we'll discuss the different methods for performing evaluation to get the feedback we need when we need it.

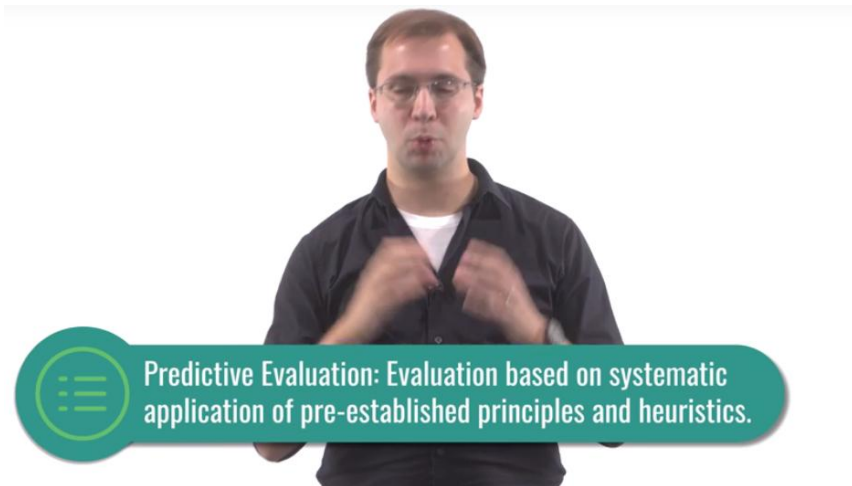
Three Types of Evaluation



There are a lot of ways to evaluate interfaces. So to organize our discussion of evaluation, I've broken these into three categories. The first is qualitative evaluation. This is where we want to get qualitative feedback from users. What do they like, what do they dislike, what's easy, what's hard. We'll get that information through some methods very similar, in fact identical, to our methods for need finding.



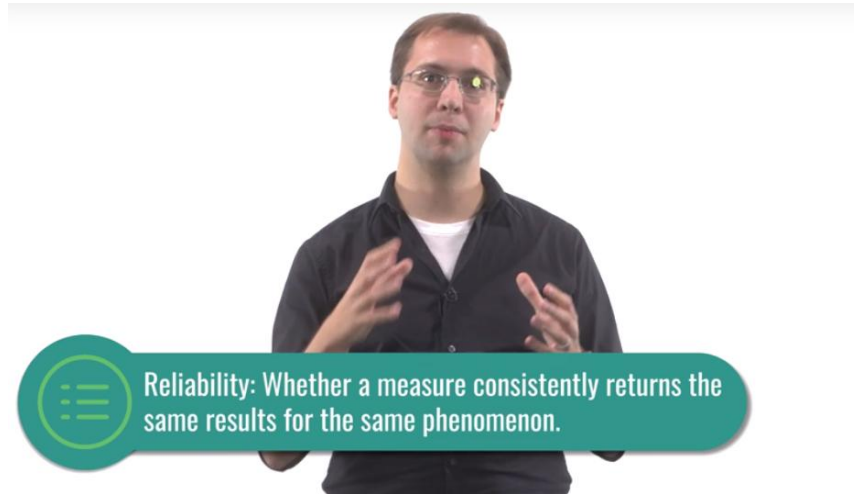
The second is empirical evaluation. This is where we actually want to do some controlled experiments and evaluate the results quantitatively. For that, we need many more participants, and we also want to make sure we addressed the big qualitative feedback first.



The third is predictive evaluation. Predictive evaluation is specifically evaluation without users. In user centered design, this is obviously not our favorite kind of evaluation. Evaluation with real users though is oftentimes slow and its really expensive. So it's useful for us to have ways we can do some simple evaluation on a day to day basis. So we'll structure our discussion of evaluation around these three general categories.

Evaluation Terminology

Before we begin there is some vocabulary we need to cover to understand evaluation. These things especially apply to the data that we gather during an evaluation. While they're particularly relevant for gathering quantitative data. They're useful in discussing our other kinds of data as well.



The first term is reliability. Reliability refers to whether or not some assessment of some phenomenon is consistent over time. So for example, Amanda, what time is it?

>> It's about 2:30.

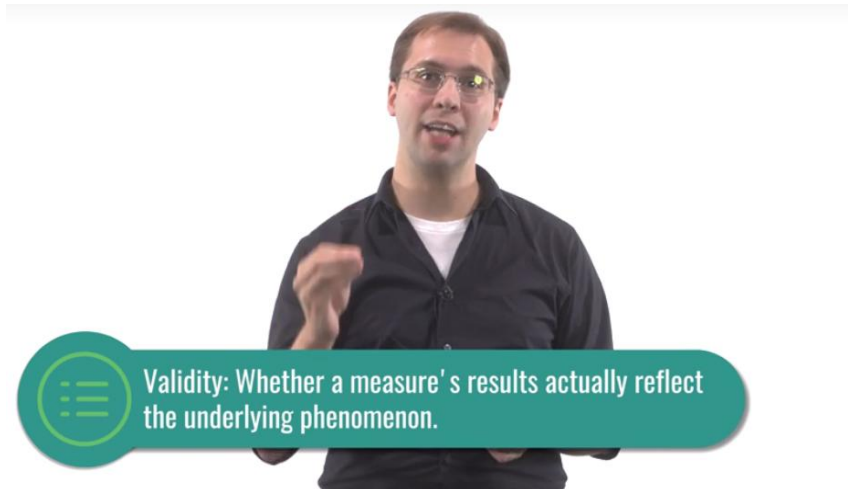
Amanda, what time is it?

>> It's about 2:30.

Amanda, what time is it.

>> It's 2:30.

Amanda is a very reliable assessment of the time. Every time I ask, she gives me the same time. We want that in an assessment measure. We want it to be reliable across multiple trials. Otherwise its conclusions are random, and just not very useful.



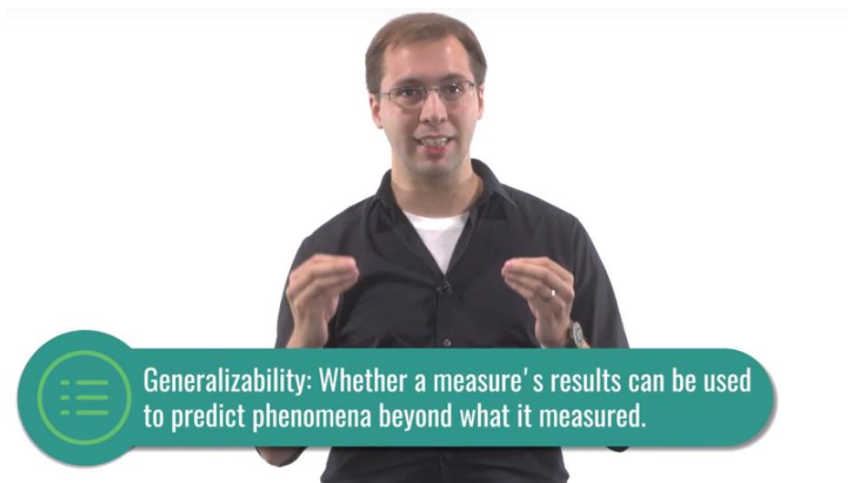
A second principle is validity. Validity refers to how accurately an assessment measures reality. An assessment could be completely reliable but completely inaccurate. So for example, Amanda what time is it?

>> Oh, my goodness, it's 2:30.

Actually it's 1:30.

>> Oh, shoot.

So while Amanda was a reliable time keeper, she wasn't a very valid one. Her time wasn't correct, even though it was consistent.



Validity is closely connected to a principle called generalizability. Generalizability is the extent to which we can apply lessons we learned in our evaluation to broader audiences of people. So, for example, we might find that the kinds of people that volunteer for usability studies have different preferences than the regular user. So the conclusions we find in those volunteers might not be generalizable in measuring what we want to measure.



And finally, one last term we want to know is to understand its precision. Precision is a measurement of how specific some assessment is. So, for example, Amanda, what time is it?

>> Well, apparently, it's 1:30.

Actually it's 1:31.

>> Ah, come on.

But in this case, no one's really going to say that Amanda was wrong in saying that it was 1:30. She just wasn't as precise. I could just accurately say it's 1:31 and 27 seconds, but that's probably more precision than we need.



As we describe the different kinds of data we can gather during evaluation, keep these things in mind. If we were to conduct the same procedure again, how likely is it that we'd get the same results? That's reliability. How accurately does our data actually capture the real world phenomenon that we care about? That's validity. To what extent can we apply these conclusions to people that weren't in the

evaluation? That's generalizability. And how specific are our conclusions and observations? That's precision.

5 Tips: What to Evaluate



In designing evaluations it's critical that we define what we're evaluating. Without that we generally tend to bottom out in vague assessments about whether or not our users like our interface. So here are five quick tips on what you might choose to evaluate. Number 1, efficiency. How long does it take users to accomplish certain tasks? That's one of the classic metrics for evaluating interfaces. Can one interface accomplish a task in fewer actions or in less time than another? You might test this with predictive models or you might actually time users in completing these tasks. Still though, this paints a pretty narrow picture of usability. Number 2, accuracy. How many errors do users commit while accomplishing a task? That's typically a pretty empirical question, although we could address it qualitatively as well. Ideally, we want an interface that reduces the number of errors a user commits while performing a task. Both efficiency and accuracy, however, examine the narrow setting of an expert user using an interface. So that brings us to our next metric. Number 3, learnability. Sitting a user down in front of the interface, define some standard for expertise. How long does it take the user to hit that level of expertise? Expertise here might range from performing a particular action to something like creating an entire document. Number 4, memorability. Similar to learnability, memorability refers to the user's ability to remember how to use an interface over time. Imagine you have a user learn an interface, then leave and come back a week later. How much do they remember? Ideally, you want interfaces that need only be learned once, which means high memorability. Number 5, satisfaction. When we forget to look at our other metrics, we bottom out in a general notion of satisfaction, but that doesn't mean it's unimportant. We do need to operationalize it though. Experience is thing like users' enjoyment of the system, or the cognitive load experience while using the system. To avoid social desirability bias, you might want to evaluate this in creative ways like, finding out how many participants actually download an app they tested after the session is over. Regardless of what you choose to evaluate, it's important that you very clearly articulate at the beginning what you're evaluating, what data you're gathering, and what analysis you will use. These three things should match up to address your research questions.

Evaluation Timeline



When we discussed prototyping, we talked about how over time the nature of our prototypes get higher and higher fidelity. Something similar happens with evaluation. Over time, the evaluation method we'll use will change.

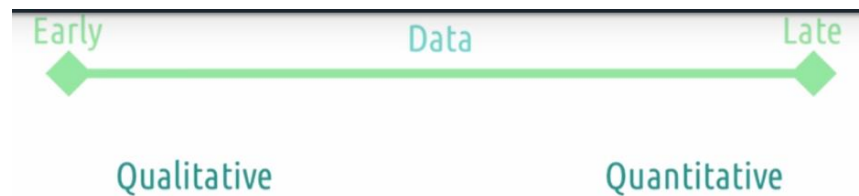


Throughout most of our design process our evaluations are formative. Meaning their primary purpose is to help us redesign and improve our interface. At the end, though, we might want to do something more summative to conclude the design process, especially if we want to demonstrate that the new interface is better. Formative evaluation is evaluation with the intention of improving the interface going forward. Summative is with the intention of conclusively saying at the end what the difference was. In reality, hopefully we never do summative evaluation. Hopefully our evaluations are always with the purpose of revising our interface and making it better over time. But in practice, there might come times when you need to demonstrate a very clear quantitative difference.

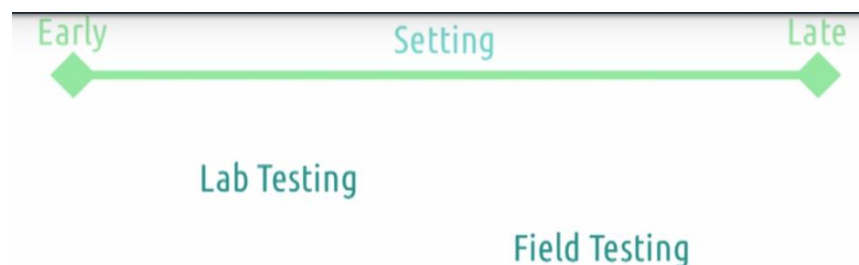


And because of this difference, our early evaluations tend to be more qualitative. Qualitative evaluations tend to be more interpretative and informal. Their goal is to help us improve or understand the task. Our later evaluations are likely more empirical, controlled, and formal. Their goal is to demonstrate or assess change. So while formative evaluation and summative evaluation were the purposes of our evaluations, qualitative evaluations and empirical evaluations are ways to actually fulfill those purposes. Predictive evaluation is a little outside the spectrum, so we'll talk about that as well. As far as this is concerned, predictive evaluations tend to be very similar to qualitative evaluations. They

inform how we revise and improve our interfaces over time. These three categories actually form the bulk of what we'll talk about in this lesson.

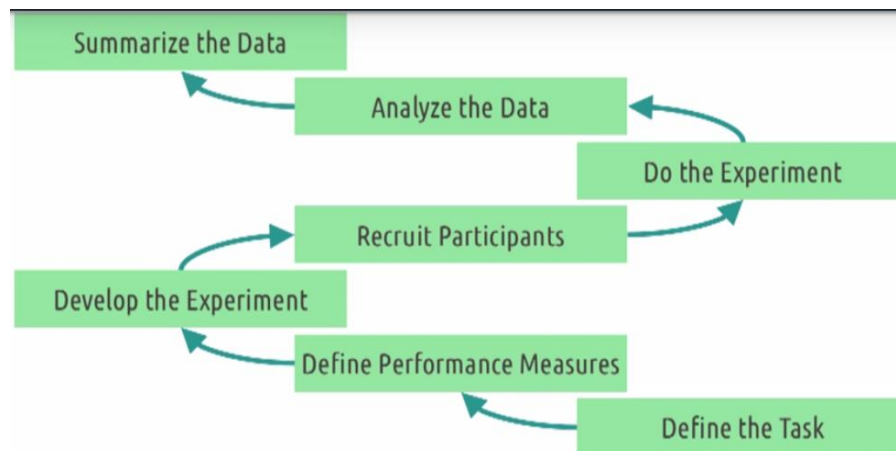


Recall also that earlier we talked about the difference between qualitative and quantitative data. As you've probably realized, if qualitative evaluation occurs early, an empirical evaluation occurs late. And chances are, we're using qualitative data more early, and quantitative data more late. In reality, qualitative data is really always useful to improve our interfaces, whereas quantitative data, while always useful, really can only arise when we have pretty rigorous evaluations.

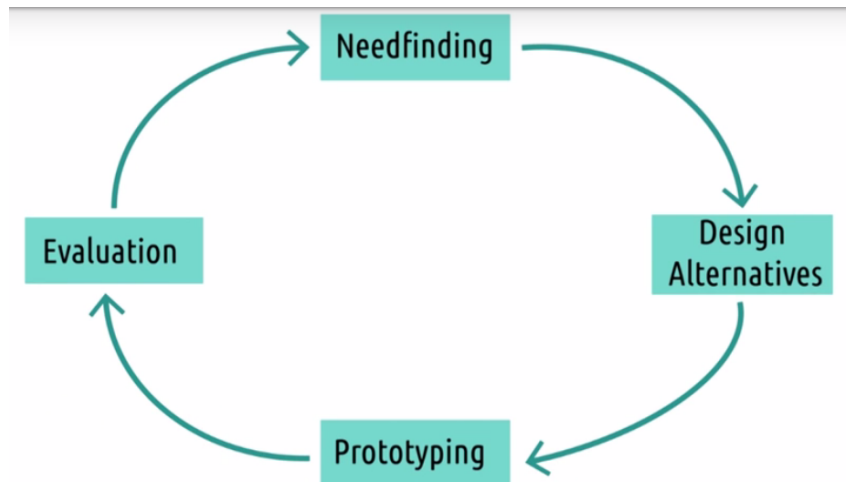


And then one last area we can look at is where the evaluation takes place. In a controlled lab setting or actually out in the field. Generally when we're testing our early low fidelity interfaces, we probably want to do it in a lab setting as opposed to out in the wild. We want to bring participants into our lab and actually describe what we're going for, the rationale behind certain decisions, and get their feedback. Later on we might want to do real field testing where we give users a somewhat working prototype, or something resembling a working prototype. And they can actually reflect on it as they go about their regular lives, participating in whatever task that interface is supposed to help with. This helps us focus exclusively on the interface early on, and in transition to focusing on the interface in context later. But of course we want to also think about the context early on. We could for example, develop a very navigation app that works great when we test in our lab, because it demands a very high cognitive load. But doesn't work at all out in the field because when participants are actually driving, they can't spare that cognitive load to focus on our app. Now of course none of these are hard and fast rules. We'll very likely often do qualitative evaluation late or maybe do some field testing early. But as general principles, this is probably the order in which we want to think about our different evaluation styles.

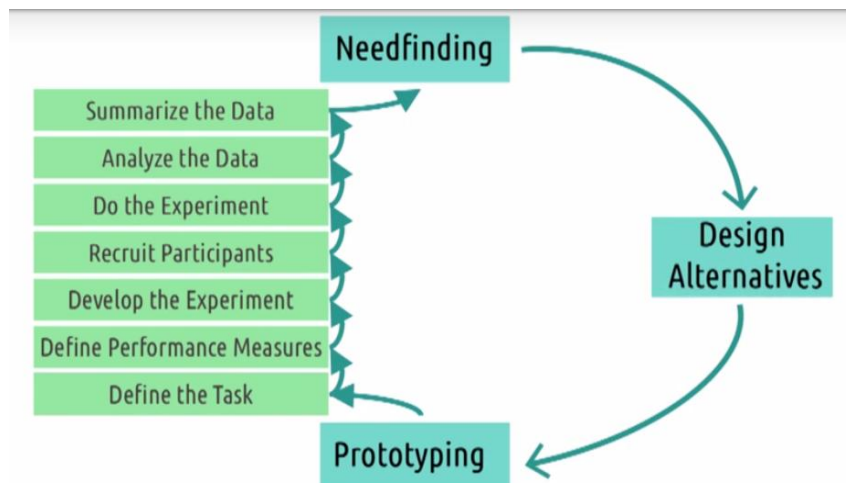
Evaluation Design



Regardless of the type of evaluation you're planning to perform, there's a series of steps to perform to ensure that the evaluation is actually useful. First, we want to clearly define the task that we're examining. Depending on your place in the design process this can be very large or very small. If we were designing Facebook, it can be as simple as posting a status update, or as complicated as navigating amongst and using several different pages. It could involve context and constraints like taking notes while running, or looking up a restaurant address without touching the screen. Whatever it is, we want to start by clearly identifying what task we're going to investigate. Second, we want to define our performance measures. How are we going to evaluate the user's performance? Qualitatively, it could be based on their spoken or written feedback about the experience. Quantitatively, we can measure efficiency in certain activities or count the number of mistakes. Defining performance measures helps us avoid confirmation bias. It makes sure we don't just pick out whatever observations or data confirm our hypotheses, or say that we have a good interface. It forces us to look at it objectively. Third, we develop the experiment. How will we find user's performance on the performance measures? If we're looking qualitatively will we have them think out loud while they're using the tool? Or will we have them do a survey after they're done? If we're looking quantitatively what will we measure, what will we control, and what will we vary? This is also where we ask questions about whether our assessment measures are reliable and valid. And whether the users we're testing are generalizable. Fourth, we recruit the participants. As part of the ethics process, we make sure we're recruiting participants who are aware of their rights and contributing willingly. Then fifth, we do the experiment. We have them walk-through what we outline when we develop the experiment. Sixth, we analyze the data. We focus on what the data tells us about our performance measures. It's important that we stay close to what we outlined initially. It can be tempting to just look for whatever supports are design but we want to be impartial. If we find some evidence that suggests our interface is good in ways we didn't anticipate, we can always do a follow up experiment to test if we're right. Seventh, we summarize the data in a way that informs our on-going design process. What did our data say was working? What could be improved? How can we take the results of this experiment and use it to then revise our interface?



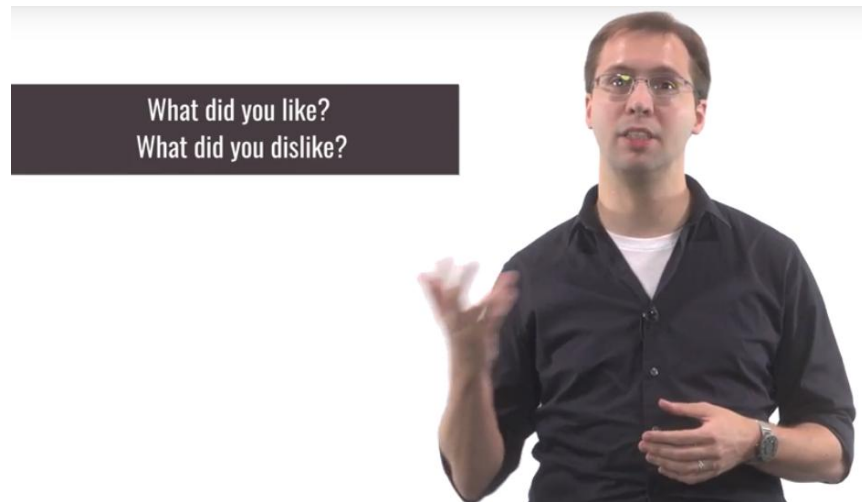
The results of this experiment then become part of our design life cycle. We investigated user needs, develop alternatives, made a prototype and put the prototype in front of users.



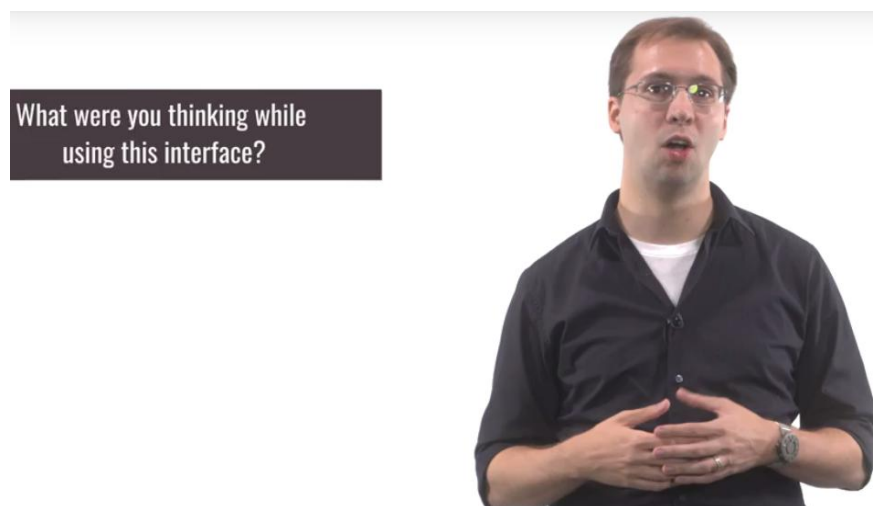
To put the prototype in front of users, we walked through this experimental method. We defined the task, defined the performance measures, developed the experiment, recruited them, did the experiment, analyzed our data and summarized our data. Based on the experience, we now have the data necessary to develop a better understanding of the user's needs, to revisit our earlier design alternatives and to either improve our prototypes by increasing their fidelity or by revising them based on what we just learned. Regardless of whether we're doing qualitative, empirical, or predictive evaluation, these steps remain largely the same. Those different types of evaluation just fill in the experiment that we develop, and they inform our performance measure, data analysis, and summaries.

Qualitative Evaluation


Qualitative evaluation involves getting qualitative feedback from the user. There are a lot of qualitative questions we want to ask throughout the design process.



What did you like? What did you dislike?



What were you thinking while using this interface?



What was your goal when you took that particular action?

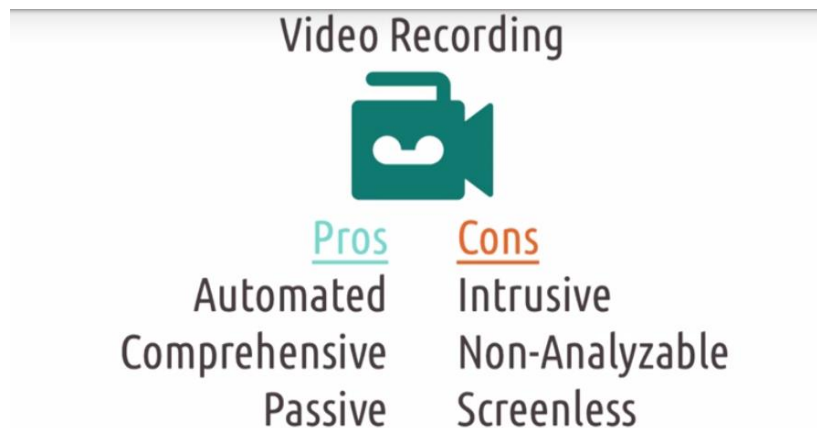
What was your goal when you took that particular action? Now if this sounds familiar, it's because it should be.



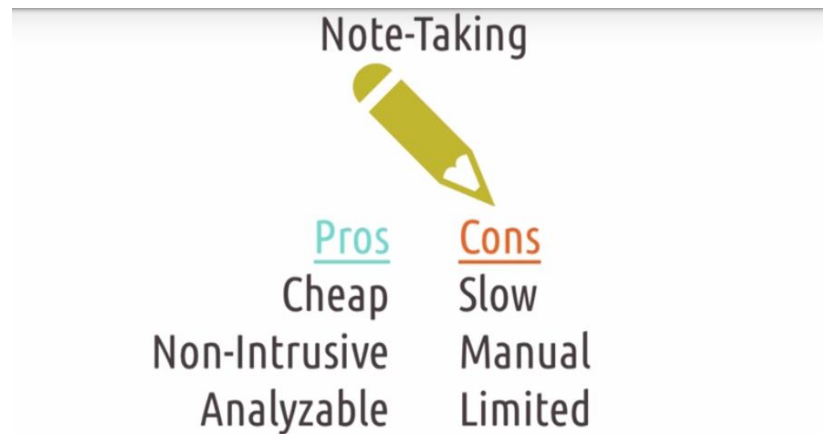
Interviews
Think Aloud Protocols
Focus Groups
Surveys
Post Event Protocols

The methods we use for qualitative evaluation are very similar to the methods we used for need finding. Interviews, think aloud protocols, focus groups, surveys, post event protocols. We use those methods to get information about the task in the first place, and now we can use these techniques to get feedback on how our prototype changes the task.

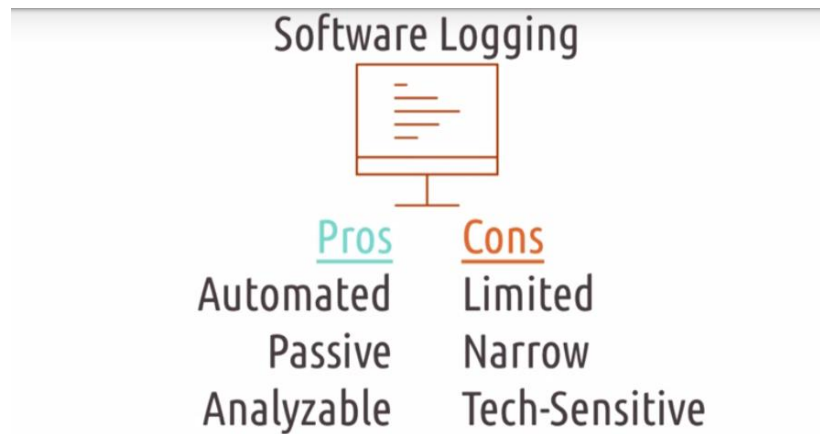
Capturing Qualitative Evaluation



With qualitative research, we want to capture as much of the session as possible. Because things could come up that we don't anticipate. And we'd like to look at them again later. So how do we do that? One way is to actually record the session. The pros of recording a session are that it's automated, it's comprehensive, and it's passive. Automated means that it runs automatically in the background. Comprehensive means that it captures everything that happens during the session. And passive means that it lets us focus on administering the session instead of capturing it. The cons though, are that it's intrusive, it's difficult to analyze, and it's screenless. Intrusive means that many participants are uncomfortable being videotaped. It creates oppression knowing that every question or every mistake is going to be captured and analyzed by researchers later. Video is also very difficult to analyze. It requires a person to come later and watch every minute of video, usually several times, in order to code and pull out what was actually relevant in that session. And video recording often has difficulty capturing interactions on-screen. We can film what a person is doing on a keyboard or with a mouse, but it is difficult to then see how that translates to on-screen actions. Now some of these issues can be resolved, of course. We can do video capture on the screen synchronize it with a video recording. But if we're dealing with children, or at risk populations, or with some delicate subject matter, the intrusiveness can be overwhelming. And if we want to do a lot of complex sessions, the difficulty in analyzing that data can also be overwhelming. For my dissertation work I captured about 200 hours of video, and that's probably why it took me an extra year to graduate. It takes a lot of time to go through all that video.



So instead we can also focus on note-taking. The benefits of note-taking are that it's very cheap, it's not intrusive, and it is analyzable. It's cheap because we don't have to buy expensive cameras or equipment, we just have our pens and papers or our laptops, or anything like that. And can just do it using equipment we already have available to us. It's not intrusive, in that it only captures what we decide to capture. If a participant is uncomfortable asking questions or makes a silly mistake with the interface, we don't necessarily have to capture that, and that can make the participant feel a little bit more comfortable being themselves. And it's a lot easier to analyze notes. You can scroll through and read the notes on a one hour session in only a few minutes. But analyzing that same session in video is certainly going to take at least an hour, if not more, to watch it more than once. But of course there are draw backs too. Taking those can be a very slow process, meaning that we can't keep up with the dynamic interactions that we're evaluating. It's also manual which means that we actually have to focus on actively taking notes, which gets in the way of administering the session. If you're going to use note taking, you probably want to actually have two people involved. One person running the session, and one person taking notes. And finally, it's limited in what it captures. It might not capture some of the movements or the motions that a person does when interacting with an interface. It doesn't capture how long they hesitate before deciding what to do next. We can write all that down of course, but we're going to run into the limitation of how fast we can take notes. It would be nearly impossible to simultaneously take notes on what questions the user is asking, how long they're taking to do things, and what kind of mistakes they're making. Especially if we're also responsible for administering the session at the same.



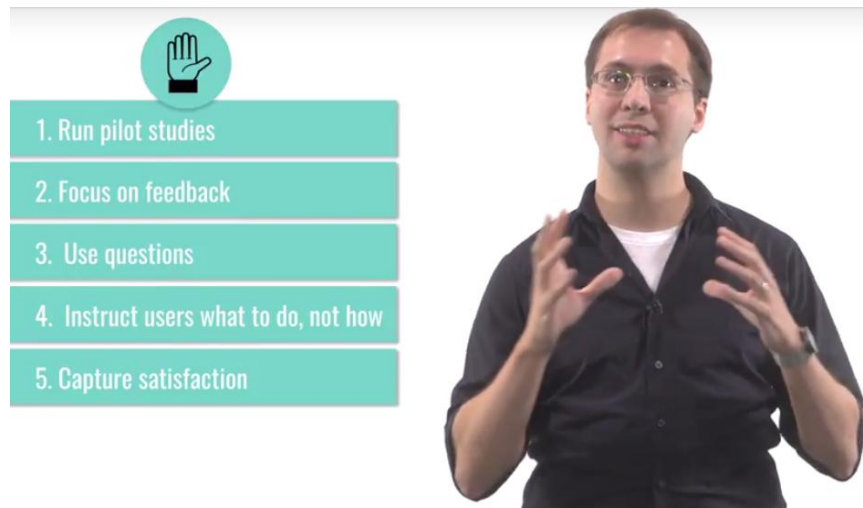
A third approach if we're designing software, is to actually log the behavior inside the software. This is in some ways, the best of both worlds. Like video capture, it's automatic and passive, but like note taking, it's analyzable. Because it's run to the system itself, it automatically captures everything that it knows how to capture, and it does so without our active invention. But it likely does so in a data or text format, that we can then either analyze manually by reading through it, or even with some more complicated data analytics methods. So in some ways, it captures the pros from both note-taking and video capture. But it also has its drawbacks as well. Especially, it's very limited. We can only capture those things that are actually expressed inside the software. Things like the questions that a participant asks wouldn't naturally be captured by software logging. Similarly, it only captures a narrow slice of the interaction. It only captures what the user actually does on the screen. It doesn't capture how long they look at something. We might be able to infer that by looking at the time between interactions, but it's difficult to know if that hesitation was because they couldn't decide what to do, or because someone was making noise outside, or something else was going on. And finally, it's also very tech sensitive. We really have to have a working prototype, in order to use software logging. But remember, many of our prototypes don't work yet. You can't do software logging on a paper prototype, or a card prototype, or a Wizard of Oz prototype. This only really works once we've reached a certain level of fidelity with our interfaces.



So in selecting a way to capture your qualitative evaluation, ask yourself, will the subjects find being captured on camera intrusive? Do I need to capture what happens on screen? How difficult will this data be to analyze? It's tempting, especially for novices, to focus on just capturing as much as possible during the session. But during the session is when you can capture data in a way that's going to make

your analysis easier. So think about the analysis that you want to do, when deciding how to capture your sessions.

5 Tips: Qualitative Evaluation



Here are five quick tips for conducting successful evaluations. Number one, run pilot studies. Recruiting participants is hard. You want to make sure that once you start working with real users, you're ready to gather really useful data. So try your experiment with friends or family or coworkers before trying it out with real users to iron out the kinks in your designs and your directions. Number two, focus on feedback. It's tempting in qualitative evaluations to spend too much time trying to teach this one user. If the user criticizes an element of the prototype, you don't need to explain to them the rationale. Your goal is to get feedback to design the next interface, not to just teach this one current user. Number three, use questions when users get stuck. That way, you get some information on why they're stuck and what they're thinking. Those questions can also be used to guide users to how they should use it, to make the session seem less instructional. Number four, tell users what to do, but not how to do it. This doesn't always apply, but most often we want to design interfaces that users can use without any real instruction whatsoever. So when performing qualitative evaluation, give them instruction on what to accomplish, but let them try to figure out how to do it. If they try to do it differently than what you expect, then you know how to design the next interface. Number five, capture satisfaction. Sometimes we can get so distracted by whether or not users can use our interface that we forget to ask them whether or not they like using our interface. So make sure to capture user satisfaction in your qualitative evaluation.

Empirical Evaluation

In empirical evaluation, we're trying to evaluate something formal, and most often that means something numeric. It could be something explicitly numeric, like what layout of buttons leads to more purchases or what gestures are most efficient to use. There could also be some interpretation involved, though, like counting errors or coding survey responses.



The goal of empirical evaluation: strong conclusions!

The overall goal, though, is to come to something verifiable and conclusive. In industry this is often useful in comparing designs or in demonstrating improvement. In research, though, this is even more important because this is how we build new theories of how people think when they're using interfaces. If we wanted to prove that gestural interaction has a tougher learning curve than voice interaction or that an audio interface is just as usable as a visual one, we would need to do empirical evaluation between the interfaces.

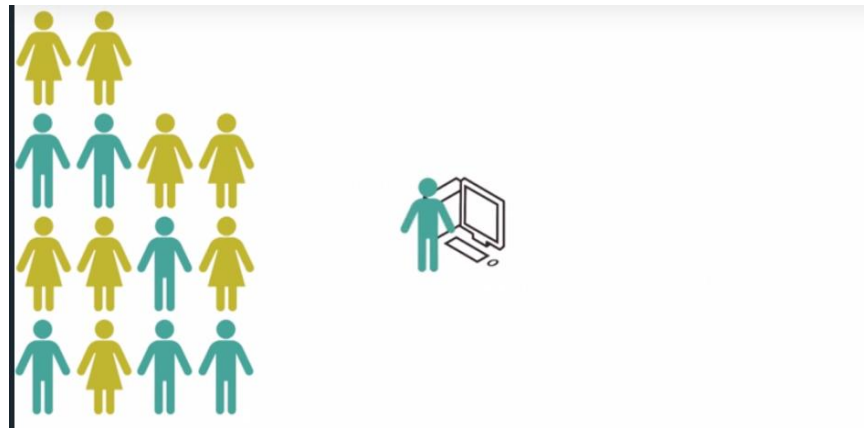


Most empirical evaluations are comparisons.

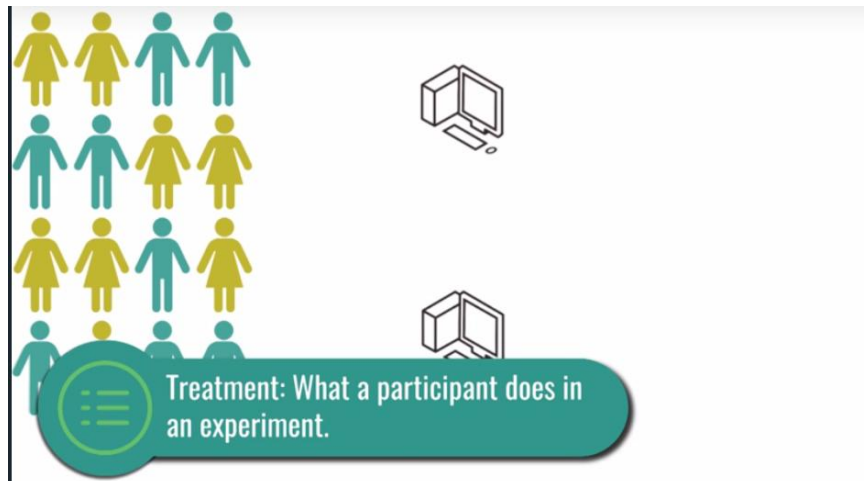
Most empirical evaluations are going to be comparisons. We can do quantitative analysis without doing comparisons, but it usually isn't necessary. The biggest benefit of quantitative analysis is its ability to

have us perform objective comparisons. So with empirical evaluation, our overall question is: how can we show there is a difference between designs?

Designing Empirical Evaluations



When we do qualitative evaluations, we effectively would just bring in participants one at a time or in groups, go through an interview protocol or script, and move along. Empirical evaluation is different, though.



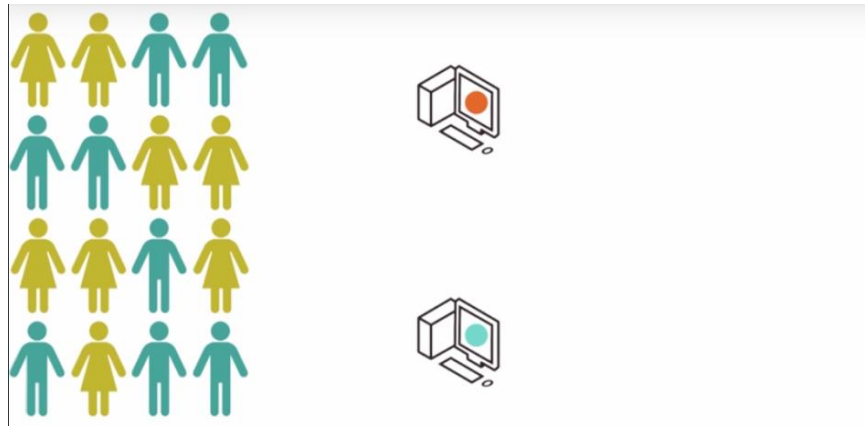
Here, we have multiple conditions, which we call treatments. These treatments could be different interfaces, different designs, different colors, whatever we're interested in investigating. Our goal here is to investigate the comparison between the treatments, and end up with a conclusion about how they're different. However, we have to be careful to make sure that the differences we observe are really due to the differences between the treatments, not due to other factors.



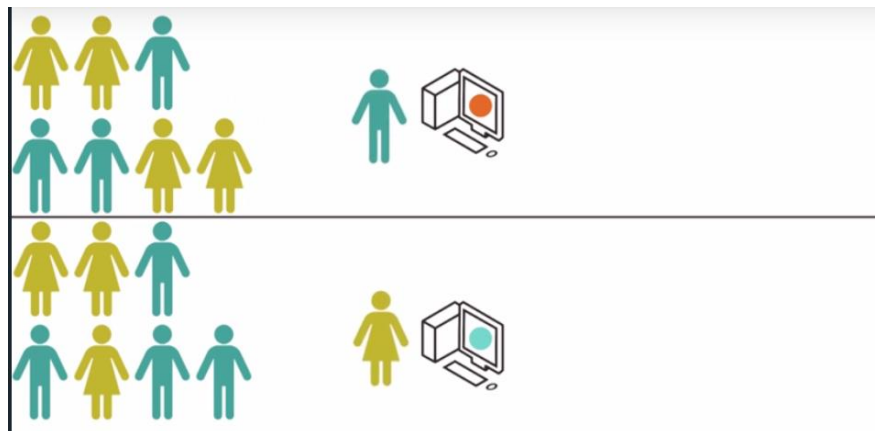
For example, imagine we were testing the difference between two logos, and we wanted to know what worked better: orange or teal. However, we also make one a circle while the other is a triangle. In the end, we wouldn't be able to comment on orange vs. teal, we could only comment on orange circle vs. teal triangle.



To make a judgment about the color, we need to make sure the color is the only thing we're comparing. Of course, here this sounds silly. In practice, though, differences can be more subtle. If you were testing different layouts, you might miss that one loads a bit faster, or one uses prettier images. And that could actually account for any differences you might observe.



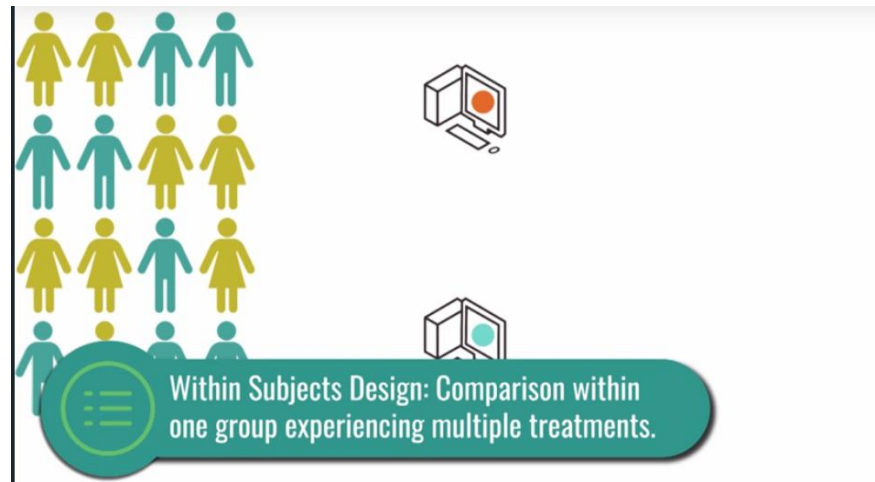
Once we've designed the treatments, it's time to design the experiment. Our first question is: what do participants do? Does each participant participate in one treatment, or both? If each participant only participates in one treatment, then our next step is easy.



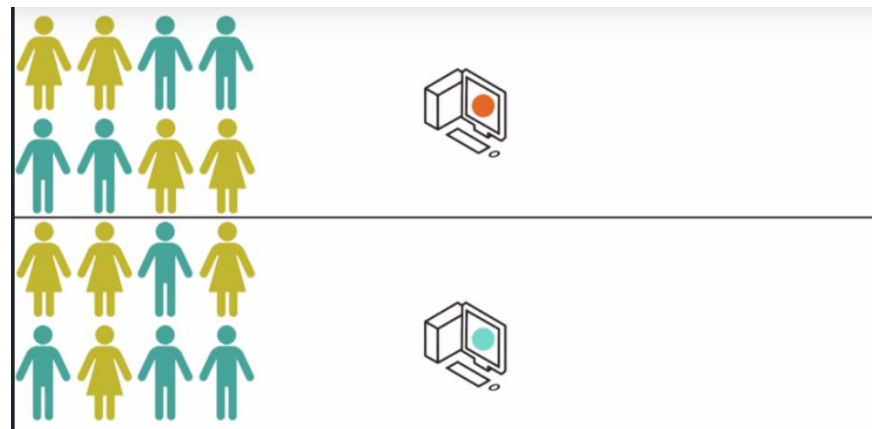
We split the randomly participants into two groups, and one-by-one, we have them go through their treatment. At the end, we have the data from participants in one group to compare to data from participants in the other group..



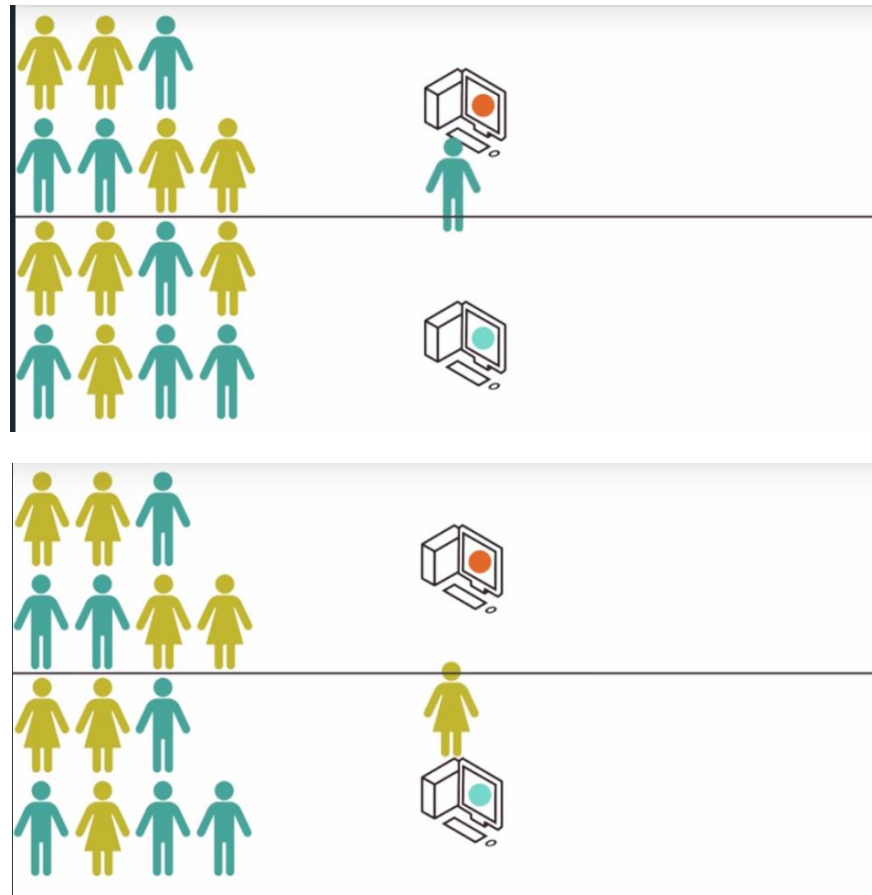
This is a between subjects design. We're comparing the data from one group to the other group.



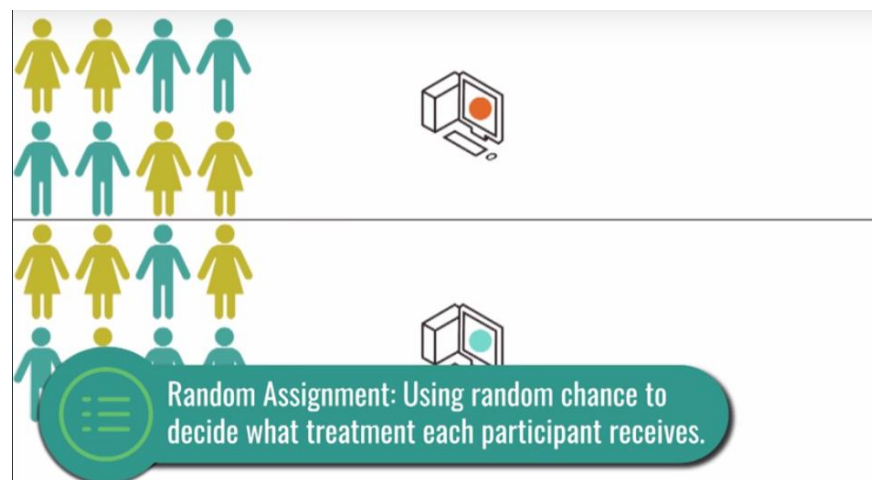
There's a second option, though. We can also do a within-subjects experiment. With a within-subjects experiment, each participant participates in *both* treatments. However, a major lurking variable could potentially be which treatment each participant sees first, so we still have to randomly assign participants to treatment groups.



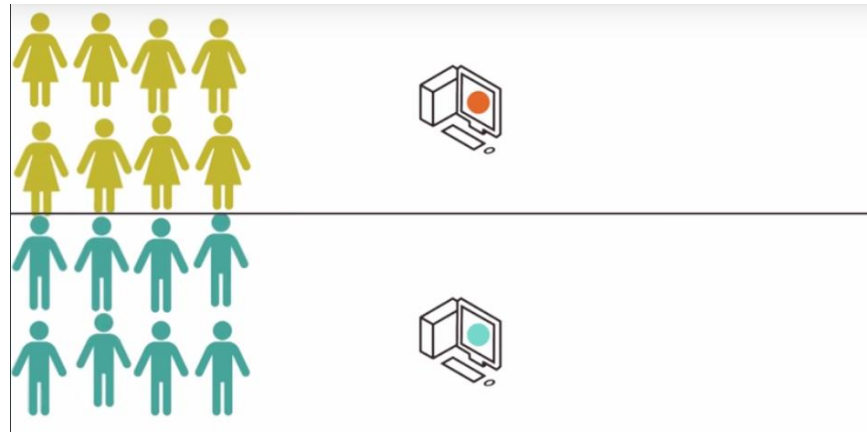
But instead of assigning participants to *which* treatment they're receiving, we're randomly assigning them to what order they'll receive the treatments in.



That way, if the order that participants receive the treatments in matters, we'll see it. Within-subjects is beneficial because it allows us to gather twice as much data if our participant pool is limited: here, each interface would be used by 16 participants instead of just 8. It also allows us to do within-subjects comparisons, seeing how each individual participant was affected instead of the groups as a whole. That can help us identify some more subtle effects, like if different people had different strengths. However, within-subjects requires more of our subjects' time, which can be a big problem if the treatments are themselves pretty long.

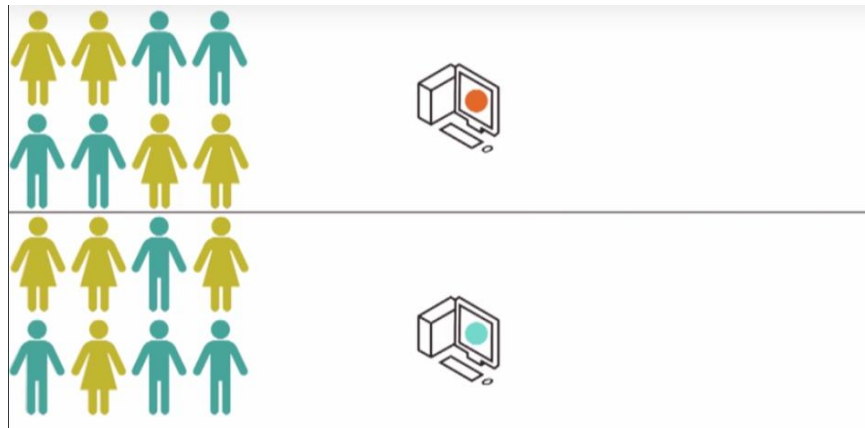


Throughout this example, we've also glossed over an important detail: random assignment. Random assignment to treatments help us control for other biases.



Imagine if all the smartest participants, or all the women were all placed in one group and received the same treatment. That would clearly affect our results. So, we randomly assign people to groups. That might also sound obvious, but imagine if your treatment involved a lot of physical set-up. It would be tempting to run the first eight participants on one set-up, and the second eight on the other. But what if that means the more punctual participants were in the first condition? Or what if you got better at administering the experiment during the first condition, so that participants in the second condition had a generally smoother experience? All of these are lurking variables that are controlled in part by random assignment to groups.

Hypothesis Testing




Let's pretend this is a reaction time study because that gives us nice numeric data. We're curious, what color should we use to alert a driver in a car that they're starting to leave their lane? We run half of them with red, half them with green, in a between-subjects design.

Orange	Green
0.34	0.32
0.31	0.31
0.24	0.23
0.27	0.21
0.28	0.29
0.27	0.31
0.34	0.37
0.21	0.39

As a result, they've generated some data. These are each participant's reaction time. Our goal is to compare this data and decide which is better. So how do we do that?

Orange	Green
0.34	0.32
0.31	0.31
0.24	0.23
0.27	0.21
0.28	0.29
0.27	0.31
0.34	0.37
0.21	0.39
0.28	0.30

Well, you might just average it. So it looks like orange is smaller than green, right? So the orange is better. Eh, not so fast (no pun intended). These numbers are very close. It's entirely possible this difference arose just by random chance. They're not likely going to be exactly equal in any trial: the question is, are they different enough to conclude that they're really different? And in fact, in this case they definitely aren't if you run the numbers.

Orange		Green
0.34	$H_{\text{NULL}}:$	0.32
0.31	$\mu_{\text{ORANGE}} = \mu_{\text{GREEN}}$	0.31
0.24	(The means are equal)	0.23
0.27	$H_{\text{ALTERNATIVE}}:$	0.21
0.28	$\mu_{\text{ORANGE}} \neq \mu_{\text{GREEN}}$	0.29
0.27	(The means are not equal)	0.31
0.34		0.37
 Hypothesis Testing: Testing whether or not the data allows us to conclude a difference exists.		0.39
		0.30

The process for doing this rigorously is called hypothesis testing. Whenever we're trying to prove something, we initially hypothesize that the opposite is true. So if we're trying to prove that one of these options is better than the other, we initially hypothesize that actually the two things are equal. That's the null hypothesis. It's the hypothesis that we accept if we can't find sufficient data to support the alternative hypothesis. So we want to see if this difference is big enough to accept the alternative hypothesis instead of the null hypothesis. We generally accept the alternative hypothesis if there is less than a 5% chance that the difference could have arisen by random chance. In that case, we say that the difference is "statistically significant". So here, there's probably a pretty good chance that this difference could arise by random chance, not because orange is actually any better than green.

Orange		Green
0.34	$H_{\text{NULL}}:$	0.41
0.31	$\mu_{\text{ORANGE}} = \mu_{\text{GREEN}}$	0.42
0.24	(The means are equal)	0.41
0.27		0.43
0.28	$H_{\text{ALTERNATIVE}}:$	0.41
0.27	$\mu_{\text{ORANGE}} \neq \mu_{\text{GREEN}}$	0.49
0.34	(The means are not equal)	0.49
0.21		0.47
0.28		0.44

But imagine if the data changed. Imagine if these were our observations from green instead of the ones before. Our average has gone up considerably from 0.30 to 0.44. Here, it's unlikely for this difference to be based only on random chance. It's still possible. It's just far less likely. This is the general process of hypothesis testing: assuming that things are the same, and seeing if the data is sufficient to prove that they're different.

Types of Hypothesis Testing

Orange		Green
0.34	$H_{\text{NULL}}:$	0.41
0.31	$\mu_{\text{ORANGE}} = \mu_{\text{GREEN}}$	0.42
0.24	(The means are equal)	0.41
0.27	$H_{\text{ALTERNATIVE}}:$	0.43
0.28	$\mu_{\text{ORANGE}} \neq \mu_{\text{GREEN}}$	0.41
0.27	(The means are not equal)	0.49
0.34		0.49
0.21		0.47
0.28		0.44

In hypothesis testing, we test the hypothesis that differences or trends in some data incurred simply by chance. If there are unlikely to have occurred by chance, we conclude that a difference really does exist. How we do that test, though, differs based on the kind of data we have. Now we won't cover how to actually do these tests in this video but we'll cover how to decide what tests you need to use and provide links to additional information separately.

Orange	Two-sample t-test: Compares means of two unpaired sets of data	Green
0.34		0.41
0.31		0.42
0.24		0.41
0.27		0.43
0.28	Used for:	0.41
0.27	Between-subjects testing	0.49
0.34	with two sets of	0.49
0.21	continuous data	0.47
0.28		0.44

In its simplest form, we compare two sets of data like this using something called a two sample t-test. A two sample t-test compares means of two unpaired sets of data. So we use it for between-subjects testing with two sets of continuous data. Continuous data means each of these measurements are on a continuous scale. And any number within that scale is hypothetically possible. We contrast this with something like a discrete scale, where participants are asked to rate something on a scale of one to five and have to choose a whole number. That will not be continuous data. It's important we only use t-tests when we have continuous data. We have other methods for evaluating discrete data.

Participant #	Orange	Green	Paired t-test: Compares means of two paired sets of data
1	0.34	0.41	
2	0.31	0.42	Used for: Within-subjects testing with two sets of continuous data
3	0.24	0.41	
4	0.27	0.43	
5	0.28	0.41	
6	0.27	0.49	
7	0.34	0.49	
8	0.21	0.47	
	0.28	0.44	

If we use paired data or within-subjects design, then our analysis changes a little bit. So imagine if each of our eight participants, instead of only seeing one treatment, saw both. We would evaluate this with a paired t-test. It compares the means of two paired sets of data. We'll use that for within-subjects testing with two sets of continuous data. The data still has to be continuous. The important thing here is that individual data points came from the same participant. Ideally we want to keep those paired when doing our comparison.

Orange	Two-Sample Binomial Test:	Green
92%	Compares proportions from two sets of data	91%
91%		89%
97%	Used for: Between-subjects testing with two sets of nominal data	85%
89%		87%
91%		86%
85%		90%
98%		83%
93%		93%
92%		88%

The math we're using changes a little bit if we're using proportions instead of just measurements. So imagine if instead of just measuring a reaction time, we were instead measuring what percentage of the time each participant reacted to the stimuli within a certain time threshold. So, for example, maybe 91% of the time this participant reacted in less than four-tenths of a second. It might seem like we evaluate this the same way, but because of the difference in math behind proportions compared to measurements, we have to use a slightly different test. We use a two-sample binomial test. Binomial means that on a give trial, there are only two possible results. So, in this case, the two possible results would be either reacted in less than four-tenths of a second or didn't. A two-sample binomial test compares proportions from two sets of data and it's used for between-subjectstesting with two sets of nominal data. Nominal carries a similar meaning to binomial in this context.

Orange	One-Sample Binomial Test:	
92%	Compares proportion from one set of data to hypothesized value	50%
91%		
97%		
89%		
91%		
85%	Used for:	
98%	Testing nominal data against a hypothesized value	
93%		
92%		

There also exists something called a one sample binomial test. That compares a proportion from one set of data to a single hypothesized value and we use that for testing nominal data against the hypothesized value. So, for example, imagine we just wanted to see if our participants reacted in under four-tenths of a second half the time. We could use a one-sample binomial test just to provide sufficient evidence that participants react in less than four-tenths of a second more than 50% of the time. This is particularly good if you want to see if some decision differs from random chance. So if, for example, you were testing to see if potential customers prefer one logo over the other. And you found that 54% preferred one logo and 46% preferred the other, you might do a test against 50% to see if that difference is just due to random chance. If you flipped a fair coin 100 times, you wouldn't necessarily expect exactly 50 heads. In the same way, if you asked 100 people, you wouldn't necessarily expect exactly 50 to say one thing, if there really was no general opinion. You would want to know if 54% for one side was actually sufficient to conclude that there really is a preference for that side.

Orange	Green	Black	Analysis of Variance (ANOVA):
92%	91%	78%	Compares means from several sets of data
91%	89%	82%	
97%	85%	83%	
89%	87%	82%	
91%	86%	74%	Used for:
85%	90%	79%	Between-subjects testing with three or more sets of continuous data
98%	83%	81%	
93%	93%	81%	
92%	88%	80%	

Now everything we've talked about so far has been about comparing two sets of data. Either within participants or between participants. What if you wanted to compare three, though? What if we tried an orange color, a green color, and a black color for those alerts? You might be tempted to compare orange and green, green and black, and orange and black, in three separate comparisons. The problem is that each time we do an additional comparison, we're raising the chance for a false positive.

Remember, we accept the alternative hypothesis if there's less than a 5% chance it could have occurred by chance. But that means if we did 20 different comparisons, there's a pretty good chance we would find one just by accident.

Orange	Green	Black	<p>Analysis of Variance (ANOVA):</p> <p>Compares means from several sets of data</p> <p>Used for:</p> <p>Between-subjects testing with three or more sets of continuous data</p>
92%	91%	78%	
91%	89%	82%	
97%	85%	83%	
89%	87%	82%	
91%	86%	74%	
85%	90%	79%	
98%	83%	81%	
93%	93%	81%	
92%	8	"If you torture the data long enough, eventually, it will talk."	

This is the origin of the phrase, if you torture the data long enough, eventually it will talk. You can get any data to say anything but that doesn't mean what you're getting it to say is true. So for comparing between more than two treatments, we use something called an ANOVA, an Analysis of Variance. It compares the means from several sets of data and we use it for between-subjects testing with three or more sets of continuous data. The purpose of this is to control for those false positives. In order for ANOVA to conclude that there really is a difference, there needs to be a much more profound difference between treatments. So if you're comparing more than two samples or more than two treatments, you'll want to use an ANOVA instead of just the t-test.

Two-sample t-test

Paired t-test

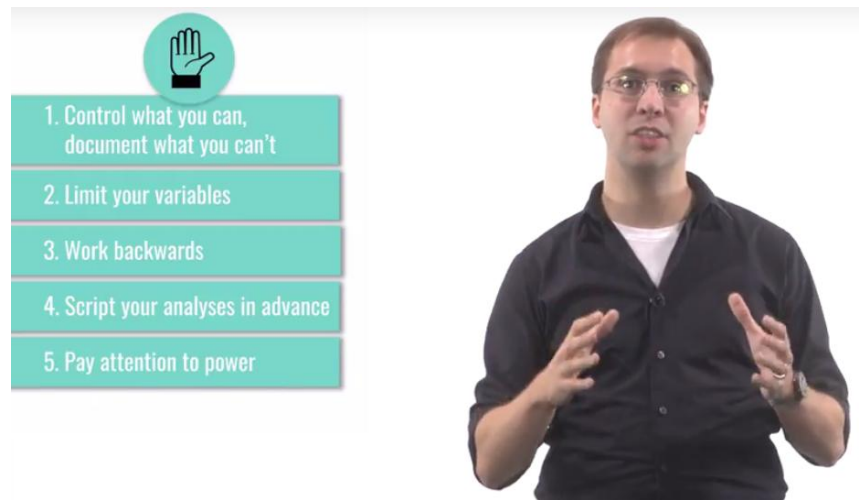
Two-sample Binomial Test

One-sample Binomial Test

Analysis of Variance (ANOVA)

These are just the most common tests you'll see in HCI, in my experience, at least. But there are lots of others. If you find yourself doing something that goes outside these basic tests, let us know and we'll try to find the appropriate analysis for you. We'll also put some links to some resources on common tests used in HCI in the notes below.

5 Tips Empirical Evaluation



Here are five quick tips for doing empirical evaluations. You can actually take entire classes on doing empirical evaluations, but these tips should get you started. Number 1, control what you can, document what you can't. Try to make your treatments as identical as possible. However, if there are systematic differences between them, document and report that. Number 2, limit your variables. It can be tempting to try vary lots of different things and monitor lots of other things. But that just leads to noisy, difficult data that will probably generate some false conclusions. Instead, focus on varying only one or two things and monitor only a handful of things in response. There's nothing at all wrong with only modifying one variable, and only monitoring one variable. Number 3, work backwards in designing your experiment. A common mistake that I've seen is just to gather a bunch of data and figure out how to analyze it later. That's messy, and it doesn't lead to very reliable conclusions. Decide at the start what question you want to answer, then decide the analysis you need to use, and then decide the data that you need to gather. Number 4, script your analyses in advance. Ronald Coase once said, if you torture the data long enough, nature will always confess. What the quote means is if we analyze and reanalyze data enough times, we can always find conclusions. But that doesn't mean that they're actually there. So decide in advance what analysis you'll do, and do it. If it doesn't give you the results that you want, don't just keep reanalyzing that same data until it does. Number 5, pay attention to power. Power refers to the size of a difference that a test can detect. And generally it's very dependent of how many participants you have. If you want to detect only a small effect, then you'll need a lot of participants. If you only care about detecting a big effect, you can usually get by with fewer.

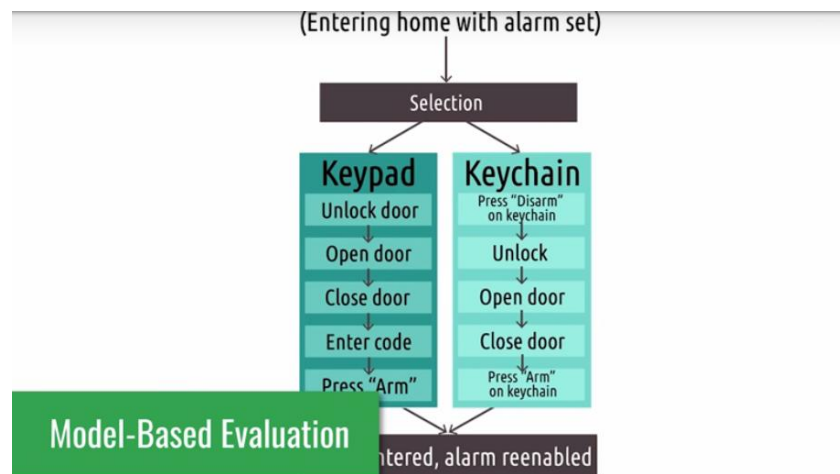
Predictive Evaluation

Predictive evaluation is evaluation we can do without actual users. Now, in user centered design that's not ideal, but predictive evaluation can be more efficient and accessible than actual user evaluation. So it's all right to use it as part of a rapid feedback process. It lets keep the user in mind, even we we're not bringing users into the conversation. The important thing is to make sure we're using it appropriately. Predictive evaluation shouldn't be used where we could be doing qualitative or empirical evaluation. It should only be used where we wouldn't otherwise be doing any evaluation. Effectively, it's better than nothing.

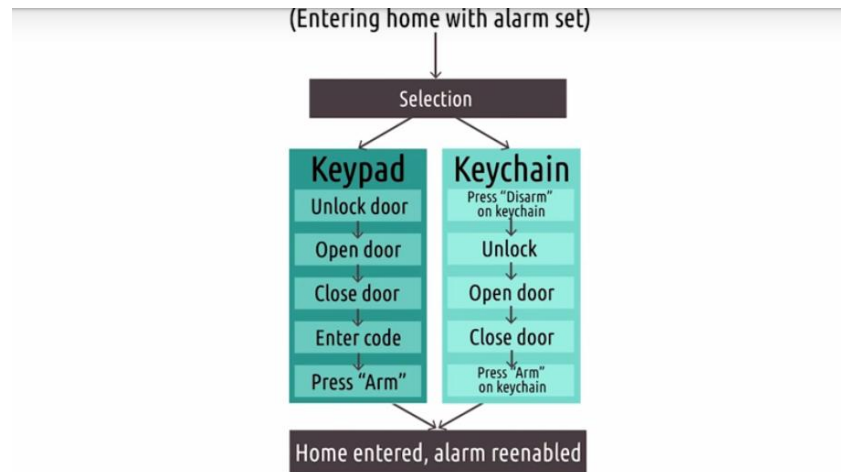
Types of Predictive Evaluation



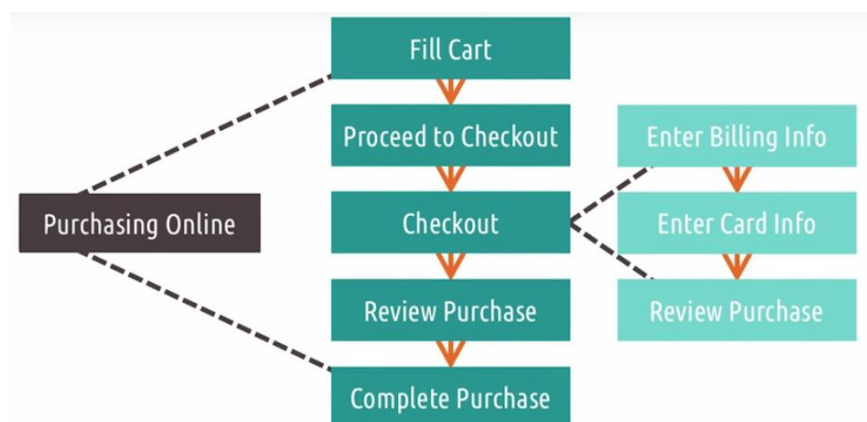
When we talk about design principles, we talk about several heuristics and guidelines we use in designing interfaces. The first method for predictive evaluation is simply to hand our interface and these guidelines to a few experts to evaluate. This is called heuristic evaluation. Each individual evaluator inspects the individual alone and identifies places where the interface violates some heuristic. We might sit with an expert while they perform the evaluation or they might generate a report. Heuristics are useful because they give us small snapshots into the way people might think about our interfaces. If we take these heuristics to an extreme, though, we could go so far as to develop models of the way people think about our interfaces.



During our needfinding exercises, we developed models of our users' tasks. In model-based evaluation, we take these models and trace through it in the context of the interface that we designed.



So let's use a Gomes model for example. Just as we computed a Gomes model for what users did in some context, we could also compute a Gomes model for what they will do in our new interface. Then, we can compare these models side by side to see how our interface changes the task and evaluate whether it aids efficiency. So here the classical way of disabling an alarm was to use a keypad mounted near the door. We could use this Gomes model to evaluate whether or not the new keychain interface was actually more efficient than the keypad interface. We could also use the profiles of users that we developed to evaluate whether the new design meets each criteria.



For example, imagine if we identified this model as applying to users with low motivation to use this interface. Maybe it's people doing purchases that they have to do for work, as opposed to just shopping at their leisure. We can use that to informal evaluation of whether or not the interface relies on high user motivation. If we find that the interface requires users to be more personally driven or to keep more in working memory, then we might find that the users will fail if they don't have high motivation to use the interface. And then we can revise it accordingly.

The State of the Art in Automating Usability Evaluation of User Interfaces

MELODY Y. IVORY AND MARTI A. HEARST

University of California, Berkeley

Usability evaluation is an increasingly important part of the user interface design process. However, usability evaluation can be expensive in terms of time and human resources, and automation is therefore a promising way to augment existing approaches. This article presents an extensive survey of usability evaluation methods, organized according to a new taxonomy that emphasizes the role of automation. The survey analyzes existing techniques, identifies which aspects of usability evaluation automation are likely to be of use in future research, and suggests new ways to expand existing approaches to better support usability evaluation.

Categories and Subject Descriptors: H.1.2 [Information Systems]: User/Machine Systems—human factors; human information processing; H.5.2 [Information Systems]: User Interfaces—benchmarking; evaluation/methodology; graphical user interfaces (GUI)

Simulation-Based Evaluation

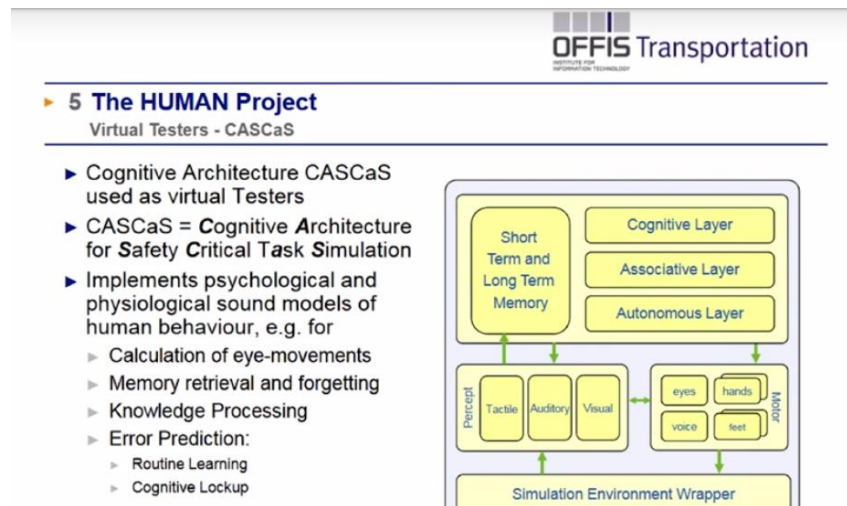
graphical user interfaces, taxonomy, usability

1. INTRODUCTION

Usability is the extent to which a computer

Usability evaluation is an important part of the overall user interface design pro-

If we take model-based evaluation to an extreme, though, we can actually get to the point of simulation-based evaluation. At that point, we might construct an artificially intelligent agent that interacts with our interface in a way that a human would. Melody Ivory and Marti Hearst actually did some research on this back in 2001 on The State of the Art in Automating Usability Evaluation of User Interfaces. And that seems like an amazing undertaking given how flexible and varied user interfaces can actually be. Can we really evaluate them automatically?



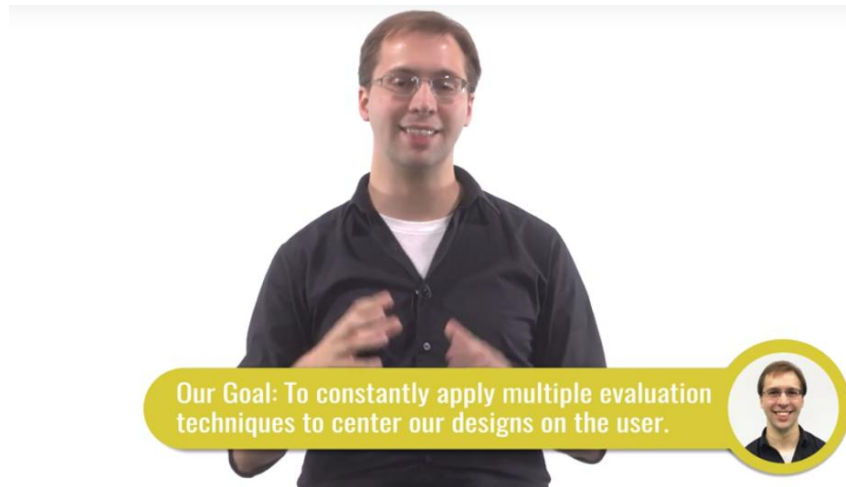
More recently, work has been done to create even more human-like models of users. Like some work done by the Human Centered Design Group at the Institute for Information Technology in Germany. Developing that agent is an enormous task on its own. But if we're working on big long-term project like Facebook, or in a high-stakes environment like air traffic control, having a simulation of a human that we can run hundreds of thousands of times on different interface prototypes would be extremely useful.

Cognitive Walkthroughs




The most common type of predictive evaluation you'll encounter most likely is the cognitive walkthrough. In a cognitive walkthrough, we step through the process of interacting with an interface, mentally simulating at each stage what the user is seeing, thinking, and doing. To do this, we start by constructing specific tasks that can be completed within our prototype. So I'm going to try this with the card prototype that I used with Morgan earlier. I start with some goal in mind. So, right now my goal is to leave a note. I look at the interface and try to imagine myself as a novice user. Will they know what to do? Here, there's a button that says "View and take a Note", so I infer that's what they would decide to do. I tap that, and what is the response that I get? The system pauses playback, and it gives me this note-taking screen. I go through the system like this, predicting what actions the user will take and noting the response the system will give. At every stage of the process, I want to investigate this from the perspective of the gulfs of execution and evaluation. Is it reasonable to expect the user to cross the gulf of execution? Is the right action sufficiently obvious? Is the response to the action the one the user would expect? On the other side, is it reasonable to expect the feedback to cross the gulf of evaluation? Does the feedback show the user what happened? Does the feedback confirm the user chose the right action? The weakness of cognitive walkthroughs is that we're the designers, so it likely seems to us that the design is fine. After all, we designed it. But if you can sufficiently put yourself in the user's shoes, you can start to uncover some really useful takeaways. Here, for example, from this cognitive walkthrough, I've noticed that there isn't sufficient feedback when the user has finished leaving a note. The system just stops recording and resumes playback, which doesn't confirm that the note is received. And right now that might be a minor issue since there's implicit feedback: the only way playback resumes is if the note is received. But I'm also now realizing that it's quite likely that users might start to leave notes, then decide to cancel them. So, they both need a cancel option, and they need feedback to indicate whether the note was completed and saved or canceled. I got that feedback just out of a cognitive walkthrough of the interface as is. So, if you can put yourself in a novice's shoes enough, you can find some really good feedback without involving real users.

Evaluating Prototypes




When we discussed prototypes for our design for an audiobook tool for exercisers, we briefly showed the evaluation stage with Morgan actually using it. Let's look at that in a little more depth, though. What were we evaluating? At any stage of the process, we could have been performing qualitative evaluation. We asked Morgan how easy or hard things were to do, how much she enjoyed using the interface, and what her thought process was in interacting with certain prototypes. We could have also performed some quantitative analysis. When she used the card-based prototype, for example, we could have measured the amount of time it took her to decide what to do, or counted the number of errors she committed. We could do the same kind of thing in the Wizard of Oz prototype. We could call out to Morgan commands like "Press Play" and "Place a Bookmark" and see how long it takes her to execute the command or how many errors she commits along the way. Between opportunities to work with Morgan, we might also use some predictive evaluation to ensure we keep her in mind while designing. Our goal is to apply multiple evaluation techniques to constantly center our designs around the user. That's why evaluation is a foundation of user-centered design -- just like we wanted to understand the user and the task before beginning to design, we also want to understand how the user relates to the design at every stage of the design life cycle.

Quiz: Exercise: Evaluation Pros and Cons

 Match the advantage to the method.			
	Qualitative	Empirical	Predictive
Does not require any actual users			
Identifies provable advantages			
Informs ongoing design decisions			
Investigates the participant's thought process			
Provides generalizable conclusions			
Draws conclusions from actual participants			

In this lesson, we've covered three different types of evaluation. Qualitative, empirical, and predictive. Each method has its advantages and disadvantages. Let's start to wrap this lesson up by exploring those advantages with an exercise. Here are the methods that we've covered. And here are some potential advantages. For each row, mark the column to which that advantage applies. Note that again, these might be somewhat relative, so your answer will probably differ a bit from ours. You can go ahead and skip to the exercise if you don't want to hear me read these. Our advantages are, does not require any actual users, identifies provable advantages. Informs ongoing design decisions, investigates the participants thought process. Provides generalizable conclusions, and draws conclusions from actual participants.

 Match the advantage to the method.			
	Qualitative	Empirical	Predictive
Does not require any actual users			
Identifies provable advantages			
Informs ongoing design decisions			
Investigates the participant's thought process			
Provides generalizable conclusions			
Draws conclusions from actual participants			

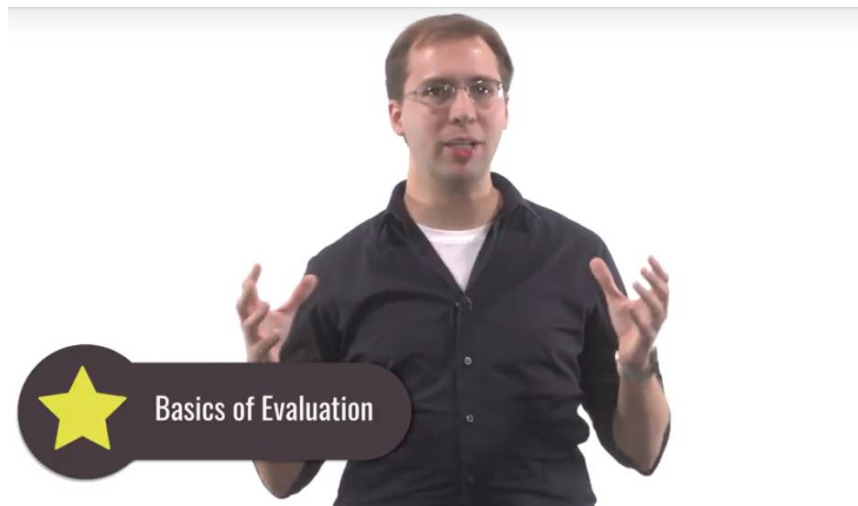
Here would be my answers to this exercise. These are a little bit more objective than some of our exercises in the past. First, if it does not require any actual users, predictive evaluation is the only evaluation we can do without involving users in the evaluation process. That's both its biggest strength

and its biggest weakness. For identifying provable advantages, only empirical evaluation can reliably generate generalizable conclusions, generalizable advantages, because it's the only one who does it numerically. As far as informing ongoing design decisions is concerned, that's definitely the case for qualitative and predictive evaluation. I've left it unmarked for empirical evaluation simply because we usually do this towards the end of our design life cycle, although we also know that the design life cycle never really ends. So eventually, empirical evaluation could be used to inform ongoing design decisions. It's just not involved in the earlier cycles though the design life cycle. As far as investing the participant's thought process, again, empirical evaluation doesn't really do that. It only accesses participants performance numerically. Qualitative evaluation definitely does this, because it actually asks users to think out loud and describe their thought process. And really, predictive evaluation tries to investigate the participant's thought process, just in a lower overhead, or lower cost kind of way. It does so by having experts in usability design simulate the participant's thought process, and comment on it from the perspective of some preset heuristics. Similar to how only empirical evaluation can identify provable advantages, it's also the only one that can provide generalizable conclusions, again because it uses numbers. And finally, qualitative and empirical evaluations both draw conclusions from actual participants. This is the inverse of predictive evaluations, lack of requirement for actual users.

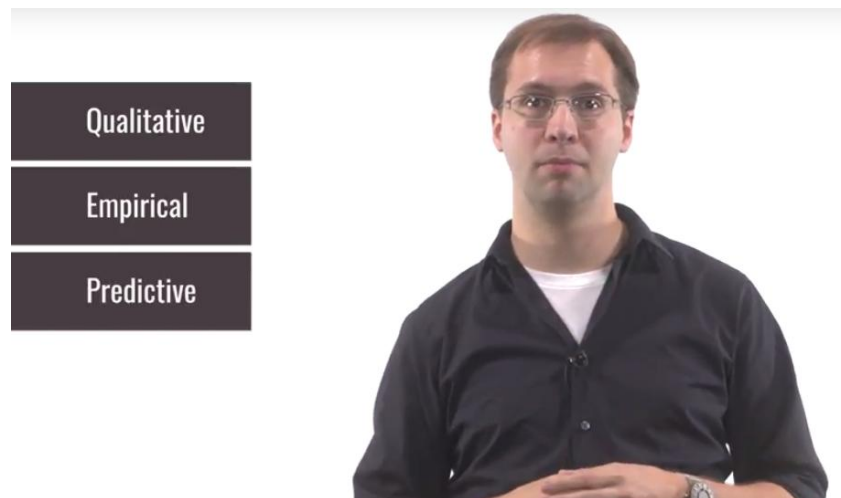
Exploring HCI: Evaluation

To succeed in HCI, you need a good evaluation plan. In industries like healthcare and education, that's initially going to involve getting some time with experts outside the real context of the task. That's bringing in doctors, bringing in nurses, bringing in patients, and exploring their thoughts on the prototypes that you've designed. In some places like education, you might be able to evaluate with real users even before the interface is ready. But in others, like healthcare, the stakes are high enough that you'll only want real users using the interface when you're certain of its effectiveness and reliability. In some emerging areas, you'll be fighting multiple questions in evaluation. Take virtual reality for example. Most people you encounter haven't used virtual reality before, there's going to be a learning curve. How are you going to determine whether the learning curve is acceptable or not? If the user runs into difficulties, how can you tell if those come from your interface, or if they're part of the fundamental VR learning experience? So take a moment to brainstorm your evaluation approach for your chosen application area. What kinds of evaluations would you choose, and why?

Conclusion to Evaluation



In this lesson we've discussed the basics of evaluation. Evaluation is a massive topic to cover though. You could take entire classes on evaluation. Heck, you could take entire classes only on specific types of evaluation.



Our goal here has been to give you enough information to know what to look into further, and when. We want you to understand when to use qualitative evaluation, when to use empirical evaluation, and when to use predictive evaluation. We want you to understand, within those categories, what the different options are. That way, when you're ready to begin evaluation, you know what you should look into doing.

3.7 HCI and Agile Development

Compiled by Shipra De, Summer 2017

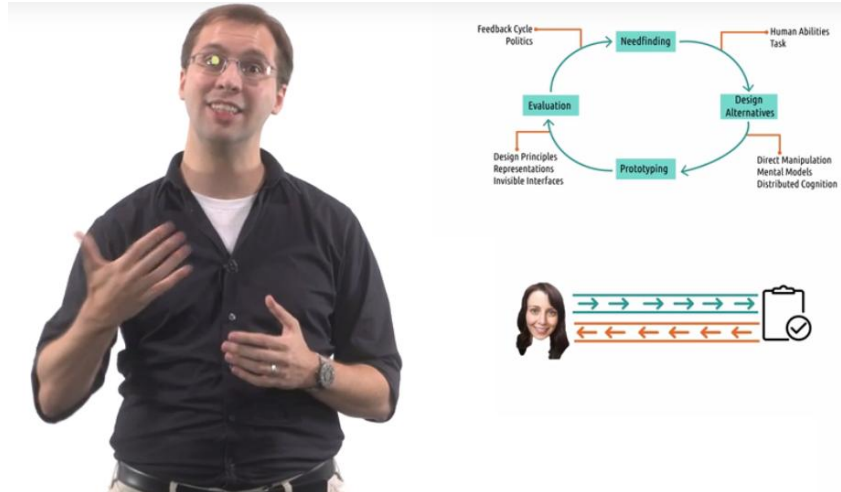
Introduction to Agile Methods



The content we've covered so far was developed over the course of several decades of research in HCI and human factors, and it's all still applicable today as well. At the same time, new technologies and new eras call for new principles and new workflows. And specifically, the advent of the Internet ushered in new methods for HCI.



Many software developers now adopt an Agile workflow which emphasizes earlier delivery, more continuous improvement, and rapid feedback cycles.

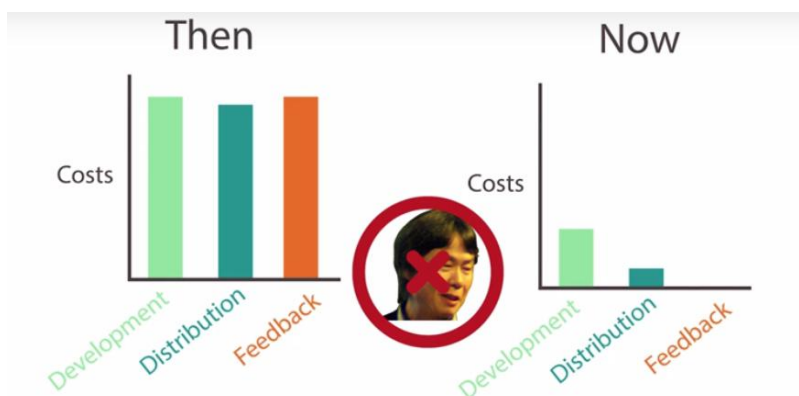


For those of us here in HCI, that's actually really exciting. We love feedback cycles. We love building them for our users, and we love engaging in them ourselves. It's also a scary prospect though. We've discussed long proto-typing processes that moved from paper, to wire frames, to live demos. Involving lots of users, in slow qualitative methodologies. And those things are still very valuable, but now a days, sometimes we just want to build something really fast and get it in front of real users. So in this lesson, we'll talk about how we might use agile development methods to engage in quicker feedback cycles.

The Demand for Rapid HCI

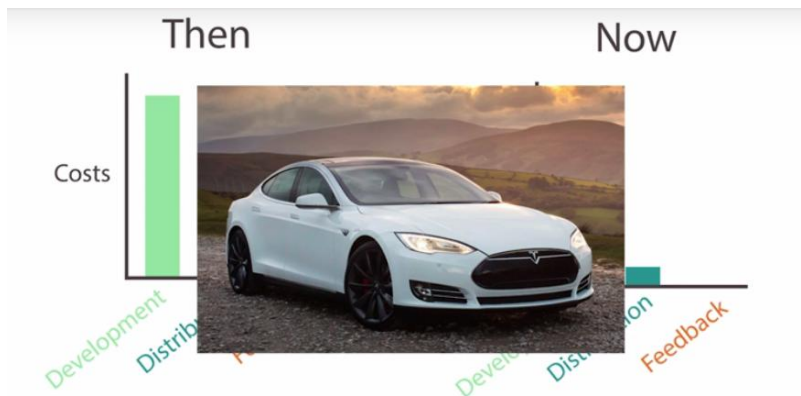


Where did these changes come from? We can think of them in terms of some of the costs associated with elements of the design life cycle. Think back to before the age of the internet. Developing software was very expensive, it required a very specialized skill set. Software distribution was done the same way we sold coffee mugs or bananas. You'd go to the store, and you'd physically buy the software. That distribution method was expensive as well. And if you ship software that was hard to use, the cost of fixing it was enormous. You had to mail each individual person an update disk. And then the only way to get user feedback, or even to find out if it was usable, was the same way you would do it before distribution, by having users come in for testing. All this meant there was an enormous need to get it right the first time. If you didn't, it would be difficult to fix the actual software, difficult to get the fix to users, and difficult to find out that a fix was even needed. Shigeru Miyamoto, the creator of Nintendo's best video game franchises, described this in terms of video games by saying, a delayed game is eventually good, but a rushed game is forever bad. The same applied to software.

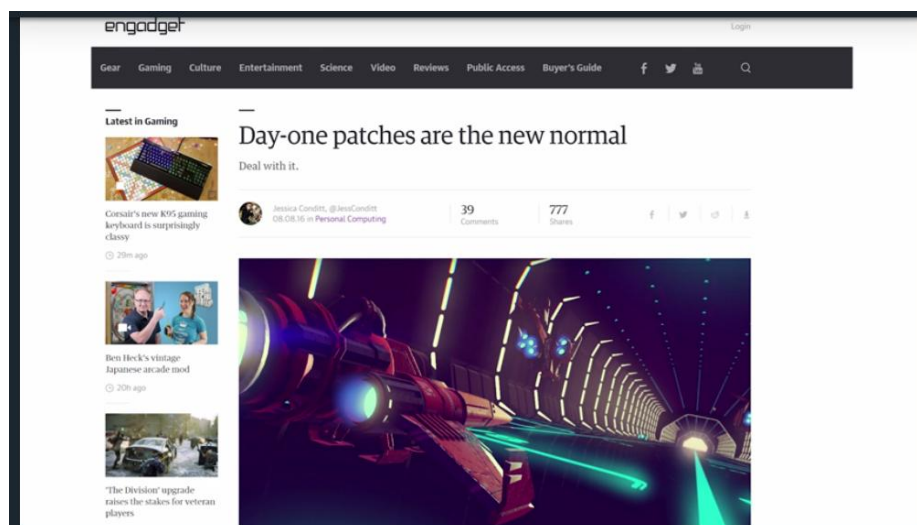


Fast forward to now though, is that still true? Development isn't cheap now, but it is cheaper than it used to be. A single person can develop in a day, what would've taken a team of people months to do 20 years ago, thanks to advances in hardware, programming languages, and the available libraries. You can look at all the imitators of popular games on either the Android or the iPhone App Store to quickly

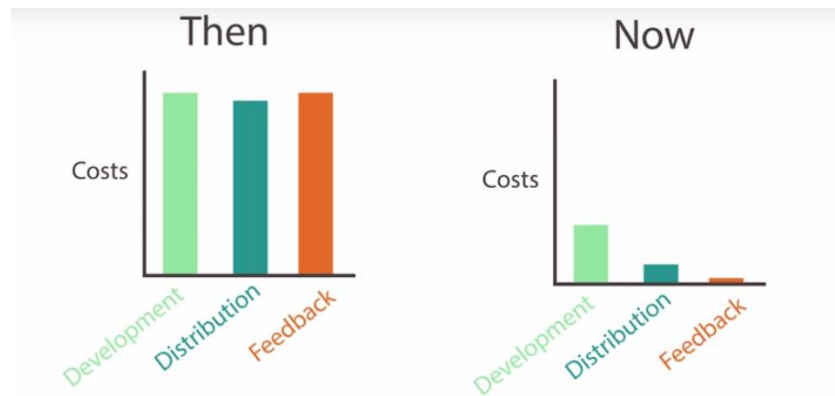
see how much development costs have come down. it's suddenly feasible to churn out a really quick imitator when something becomes popular. But more importantly, distribution for software is now essentially free. And updating a software is essentially free as well. Every day you can download new apps and have them update automatically in the background. If you release something that has a bug in it, you can fix it and roll out the fix immediately. Miyamoto's quote is no longer really accurate because it is possible to fix games after they're released.



Tesla, for example, regularly pushes software updates to its cars via the internet.



And in the video game industry, day one patches that fix glitches on the very first day of release have pretty much become the standard.



And perhaps most importantly, we can gather usage data from live users automatically, and essentially for free as well. And it isn't just usage data, it's product reviews, error reports, buzz on the internet. Lots of feedback about our applications now comes naturally without us having to spend any money to gather it. What all this means, is there is now more incentive to build something fast and get it to users to start getting real feedback as early as possible.

Now make no mistake, this isn't justification to throw out the entire design life cycle. The majority of design and research still goes through with a longer process. You need several iterations through the full design life cycle for big websites, complex apps, anything involving designing hardware. Anything involving a high profile first impression. And really anything involving anything even somewhat high in stakes. But that said, there exists a new niche for rapid development. Maybe you came up with an idea for a simple Android game. In the time it would take you to go through this longer process, you could probably implement the game, and get it in front of real users, and get a lot more feedback. That's what we're discussing here. How do you take the principles we've covered so far and apply them to a rapid, agile development process?

Quiz: Exercise: When to Go Agile



Which of the following applications would lend itself to an agile development process?

- ☒ A camera interface for aiding MOOC recording
- ☒ A tool for helping doctors visualize patient info in surgery
- ☒ A smartwatch game to play in short five-minute sessions
- ☒ A wearable device for mobile keyboard entry
- ☒ A mobile app for aggregating newsfeeds across networks
- ☒ A navigation app for the console of an electric car

Before I describe the current ideas behind when to go for an Agile development process, let's see what you think. Here are six possible applications we might develop. Which of these would lend itself to an agile development process?



Which of the following applications would lend itself to an agile development process?

- ☐ A camera interface for aiding MOOC recording
- ☐ A tool for helping doctors visualize patient info in surgery
- ☒ A smartwatch game to play in short five-minute sessions
- ☐ A wearable device for mobile keyboard entry
- ☒ A mobile app for aggregating newsfeeds across networks
- ☐ A navigation app for the console of an electric car

Here would be my answers. The two areas that I think are good candidates for an agile development process are the two that use existing devices and don't have high stakes associated with them. In both these cases, rolling out updates wouldn't be terribly difficult, and we haven't lost a whole lot by initially having a product that has some bugs in it. A camera interface for aiding MOOC recording would be a good candidate, if the camera environment was easier to program for, but programming for a camera isn't like programming for an app store, or for a desktop environment. I actually don't even know how you go about it. So for us, a camera interface for aiding MOOC recording probably wouldn't be a great candidate, because we don't have access to that platform. And remember, our goal is to get products in front of real users as soon as possible. Now of course, that all changes if we're actually working for a camera company and we do have access to that platform. The second one is more fundamental though.

A tool for helping doctors visualize patient information in surgery. There are really high stakes behind that, if you visualize something in a way that's a little bit misleading, someone could die. So you probably don't want take an agile development process for that. For a wearable device for mobile keyboard entry. Wearable devices are expensive to produce. When you're actually producing the physical device, you want to be sure it's going to work pretty well. And similarly devices aren't easy to update the way software is. So a wearable device is probably not a good candidate for agile development process. And finally, a navigation app for the console of an electric car, I said isn't a good candidate although you might disagree. Personally, I would say that the stakes are high enough for a navigation app, that you probably want to be pretty sure that you're going to have a good product before you roll it out to users. They might take a wrong turn or end up in the wrong neighborhood or miss an appointment based on some mistakes that we make. And I would consider that sufficiently high stakes to avoid a faster development process. And plus not all electric cars are like Tesla. Some of them actually have to have you bring the car to the factory or to the repair shop to get an update. So the cost of rolling out updates can be more significant there as well.

When to Go Agile

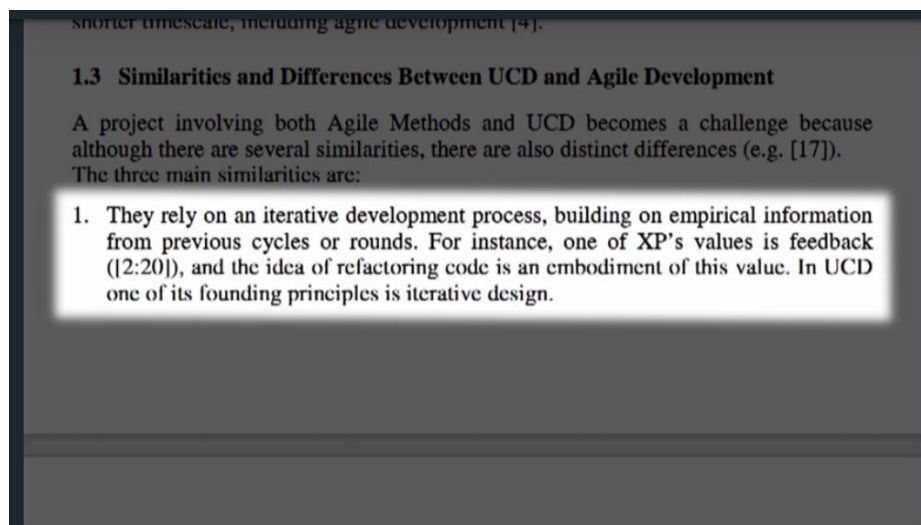
	Traditional	Agile
Criticality is...	High	Low
Requirements change...	Rarely	Frequently
Team size is...	Large	Small
Team embraces...	Order	Change

So when should you consider using these more agile methodologies? Lots of software development theorists have explored this space. Boehm and Turner specifically suggest that agile development can only be used in certain circumstances. First, they say, it must be an environment with low criticality. By its nature, agile development means letting the users do some of the testing. So you don't want to use it in environments where bugs or poor usability are going to lead to major repercussions. Healthcare or financial investing wouldn't be great places for agile development, generally speaking. Although there have been efforts to create standards that would allow the methodology to apply, without compromising security and safety. But for things like smartphone games and social media apps, the criticality is sufficiently low. Second, it should really be a place where requirements change often. One of the benefits of an agile process is they allow teams to adjust quickly to changing expectations or needs. A thermostat, for example, doesn't change its requirements very often. A site like Udacity though, is constantly adjusting to new student interests or student needs. Now these two components apply to the types of problems we're working on. If we're working on an interface that would lend itself to a more agile process, we also must set up the team to work well within an agile process. That means small teams that are comfortable with change. As opposed to large teams that thrive on order. So generally, agile processes can be good in some cases with the right people, but poor in many others.

Paper Spotlight: "Towards a Framework..."



In 2006, Stephanie Chamberlain, Helen Sharp, and Neil Maiden investigated the conflicts and opportunities of applying agile development to user-centered design. They found interestingly that the two actually had a significant overlap.



Both agile development and user-centered design emphasized iterative development processes building on feedback from previous rounds. That's the entire design life cycle that we've talked about. That's at the core of both agile development and user-centered design.

2. Agile techniques place an emphasis on the user, encouraging participation throughout the development process. For instance, in Scrum, user evaluation of the product is encouraged on a monthly basis as users are ideally present during the sprint review ([16:54]) and the “Product Owner” is responsible for the requirements and feature prioritisation for the product. A second founding principle of UCD, is early and continual focus on users.
3. Both approaches emphasise the importance of team coherence. Beck states that one of the purposes of the planning game is to “bring the team together” ([2:85]). One of the features of the UCD approach is that the whole team should have the user in mind while developing the product.

The two main differences are:

1. UCD advocates maintain that certain design products are required to support communication with developers, while agile methods seek minimal documentation.
2. UCD encourages the team to understand their users as much as possible before the

Both methodologies also place a heavy emphasis on the user's role in the development process.

2. Agile techniques place an emphasis on the user, encouraging participation throughout the development process. For instance, in Scrum, user evaluation of the product is encouraged on a monthly basis as users are ideally present during the sprint review ([16:54]) and the “Product Owner” is responsible for the requirements and feature prioritisation for the product. A second founding principle of UCD, is early and continual focus on users.
3. Both approaches emphasise the importance of team coherence. Beck states that one of the purposes of the planning game is to “bring the team together” ([2:85]). One of the features of the UCD approach is that the whole team should have the user in mind while developing the product.

The two main differences are:

1. UCD advocates maintain that certain design products are required to support communication with developers, while agile methods seek minimal documentation.
2. UCD encourages the team to understand their users as much as possible before the

And both also emphasize the importance of team coherence. So it seems that agile methods and user-centered design agree on the most fundamental element, the importance of the user.

ments and feature prioritisation for the product. A second founding principle of UCD, is early and continual focus on users.

3. Both approaches emphasise the importance of team coherence. Beck states that one of the purposes of the planning game is to “bring the team together” (12:85). One of the features of the UCD approach is that the whole team should have the user in mind while developing the product.

The two main differences are:

1. UCD advocates maintain that certain design products are required to support communication with developers, while agile methods seek minimal documentation.
2. UCD encourages the team to understand their users as much as possible before the product build begins, whereas agile methods are largely against an up-front period of investigation at the expense of writing code.

2 Fieldwork

2.1 Method

Three project teams in one organisation were observed for around 2-4 hours per week

By comparison, the conflicts are actually relatively light, at least in my opinion. User-centered design disagrees with agile development on the importance of documentation and the importance of doing research prior to the design work actually beginning. But, clearly, the methodologies have the same objectives. They just disagree on how to best achieve them.

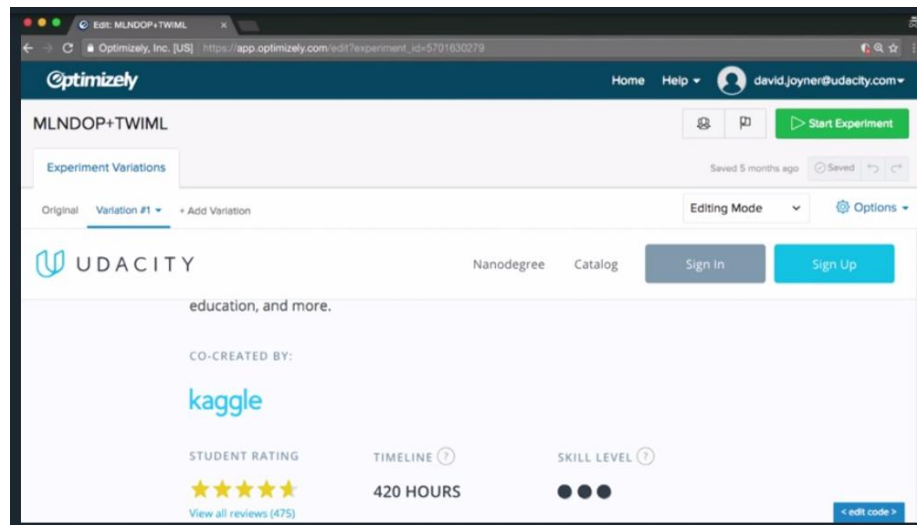
5 Five Principles for Integrating UCD and Agile Development

Based on our observations and the themes discovered, we have evolved a set of five principles which are significant where UCD and agile methods are to be integrated:

1. **User Involvement** – the user should be involved in the development process but also supported by a number of other roles within the team, such as having a proxy user on the team.
2. **Collaboration and Culture** – the designers and developers must be willing to communicate and work together extremely closely, on a day to day basis. Likewise the customer should also be an active member of the team not just a passive bystander.
3. **Prototyping** – the designers must be willing to “feed the developers” with prototypes and user feedback on a cycle that works for everyone involved.
4. **Project Lifecycle** – UCD practitioners must be given ample time in order to discover the basic needs of their users before any code gets released into the shared coding environment.
5. **Project Management** – Finally, the agile/UCD integration must exist within a cohesive project management framework that facilitates without being overly bureaucratic or prescriptive.

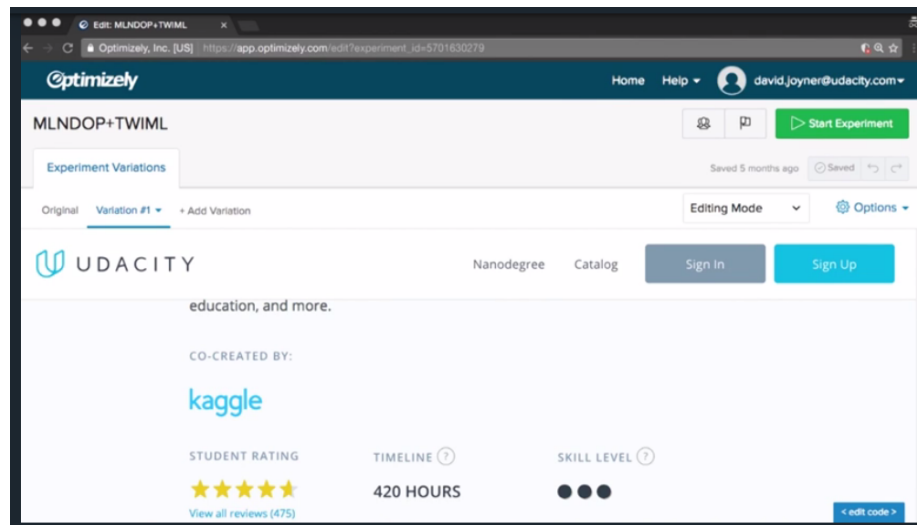
As a result, the authors advocate five principles for integrating user-centered design and agile development. Two of these were shared between the methodologies in the first place, high user involvement and close team collaboration. User-centered designs' emphasis on prototyping and the design life cycle shows that by proposing that design is run a sprint ahead of developers to perform the research necessary for user-centered design. To facilitate this, strong project management is necessary.

Live Prototyping

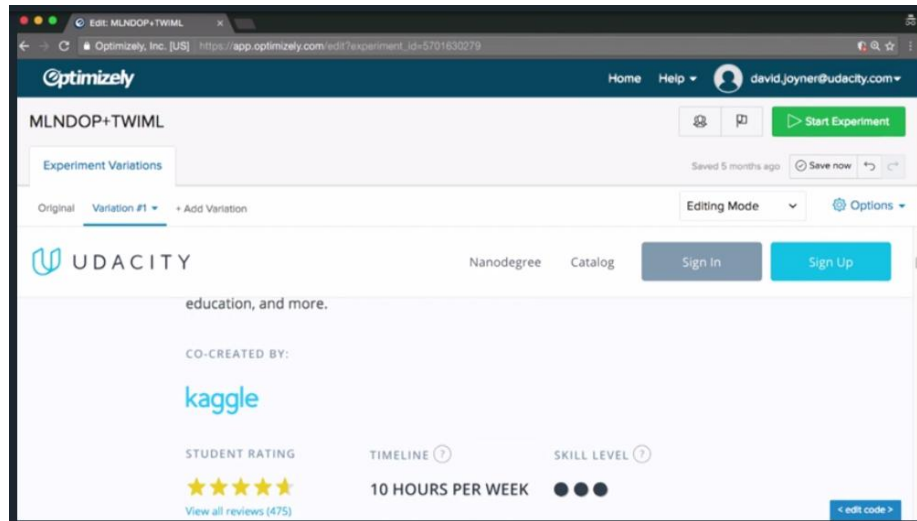


One application of Agile development in HCI is the kind of new idea of live prototyping. Live prototyping is a bit of an oxymoron, and the fact that it's an oxymoron speaks to how far along prototyping tools have come. We've gotten to the point in some areas of development where constructing actual working interfaces is just as easy as constructing prototypes. So here's one example of this, it's a tool we use at Udacity called Optimizely. It allows for drag and drop creation of real working webpages. The interface is very similar to many of the wire-frame tools out there, and yet this website is actually live. I can just click a button and this site goes public. Why bother constructing prototypes before constructing my final interface, when constructing the final interface is as easy as constructing prototypes? Of course, this only addresses one of the reasons we construct prototypes. We don't just construct them because they're usually easier, we also construct them to get feedback before we roll out a bad design to everyone. But when we get to the point of making small little tweaks or small revisions, or if we have a lot of experience with designing interfaces in the first place, this might not be a bad place to start. It's especially true if the cost of failure is relatively low, and if the possible benefit of success is particularly high. I would argue that's definitely the case for any kind of e-commerce site. The cost of failure is maybe losing a few sales but the possible benefit is gaining more sales for a much longer time period. I'm sure anyone would risk having fewer sales on one day for the possible reward of having more sales every subsequent day.

A/B Testing

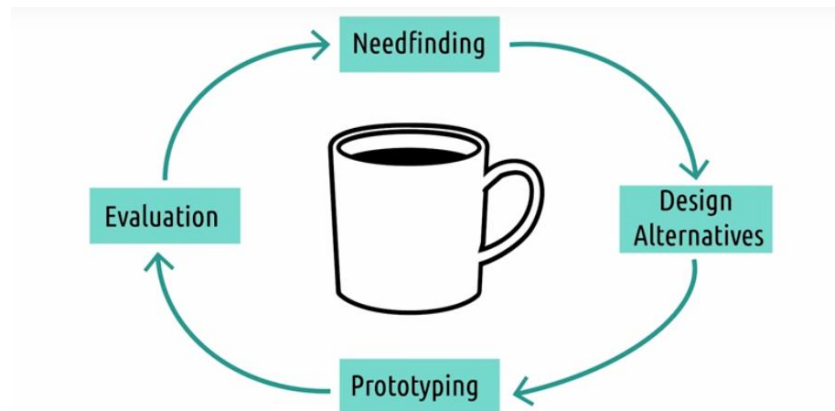


So in some contexts, it's now no harder to construct an actual interface than it is to construct a prototype, so we might skip the prototyping phase altogether. However, prototypes also allowed us to gather feedback from users. Even though we can now easily construct an interface, we don't want to immediately roll out a completely untested interface to everyone who visits our site. We might be able to fix it quickly, but we're still eroding user trust in us and wasting our user's time. That's where the second facet of this comes in, AB testing. AB testing is the name given to rapid software testing between typically two alternatives, A and B. Statistically it's not any different from T-tests. What makes AB testing unique is that we're usually rapidly testing small changes with real users. We usually do it by rolling out the B version, the new version to only a small number of users, and ensuring that nothing goes terribly wrong, or there's not a dramatic dip in performance. That way we can make sure a change is positive, or at least neutral, before rolling it out to everyone, but look where testing feedback coming in here. They're coming automatically with the real users during normal usage of our tool. There's no added cost to recruiting participants and the feedback is received instantly. So for a quick example, this is the overview page for one of Udacity's programs and it provides a timeline the students should dedicate to the program in terms of number of hours. Is number of hours the best way to display this? I don't know, we could find out. Instead of showing 420 hours maybe I say this as 20 hours per week. In this interface all I have to do is edit it and I immediately have a new version of this interface that I can try out.



Now I can click Start Experiment and try this out. I could find out. Does phrasing this as ten hours per week, does it increase the number? Does it decrease the number? If it decreases it, I can very quickly roll this back. If it increases it, I can very quickly roll this out to everybody. I'm going through the same design life cycle. I understand that the need is for the user to know where the timeline is. I've got a design in mind, which is to show the timeline in number of hours per week. I prototype it. It just happens to be here that the prototype is live. And I immediately roll it out. I look at how users use it, I evaluate it, and I decide if I want to roll back that change, or roll it out to everybody. I can go through a microcosm, a very rapid iteration to really design life cycle by using live prototyping and AB testing.

Agile HCI in the Design Life Cycle



Agile development techniques don't replace the design lifecycle, they just caffeinate it. We're still doing needfinding, but we're probably just doing it a little bit more tacitly by reading user feedback or checking out interaction logs. We're still brainstorming design alternatives. But we're really just coming up with them in our head, because we then immediately move them on to prototyping. And our prototypes are still just prototypes, they just happen to work. And we're still doing evaluation by rolling our changes out to only certain participants first to make sure the response is good. And the results of that evaluation then feed the same process over and over again. So taking an agile approach to the design lifecycle really doesn't change the cycle itself. It just changes the rate at which we go through it, and the types of prototypes, and the types of evaluation that we actually do. And remember also that Chamberlain, Sharp, and Maiden advocated still doing the initial needfinding step. Rarely will we go from no interface at all to a working prototype quite as quickly as we go through revisions of those working prototypes. And so it's useful to do an initial needfinding phase the way we normally would do it, and then bursting into a more agile revision process once we have our working prototype to actually tweak and modify.

5 Tips: Mitigating Risk in HCI and Agile Development



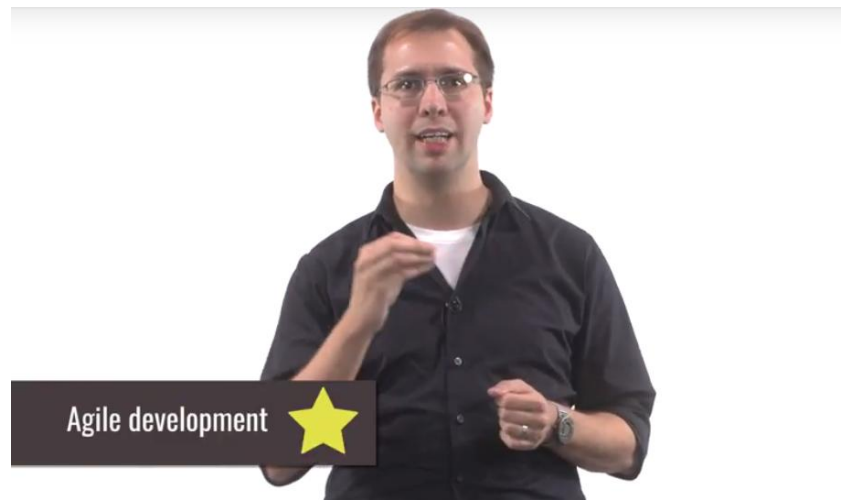
Here are five quick tips for using HCI and agile development together, especially for mitigating the risks to the experience presented by this more agile development process. Number one, [SOUND] start more traditional, start with a more traditional need, finding, and prototyping process, and shift to more agile development once you have something up and running. Jacob Nielsen describes this as doing some foundational user research. Once you have something up and running, you have a way of probing the user experience further. But you need something solid to begin with, and that comes from the more traditional process. Number two, focus on small changes. Notice that when I was doing live prototyping and A-B testing, I was making a small change to an existing interface. Not building an entire new site from scratch. Number three, adopt a parallel track method. Agile development often uses short two week sprints in development. Under that setup have the HCI research one sprint ahead of the implementation. The HCI team can do two week sprints of need finding, prototyping, and low fidelity evaluation and then hand the results to the development team for their next sprint. Number four, be careful with consistency. One of our design principals was consistency both within our interfaces and across interface design as a whole. If your interface caters to frequent visitors or users, you'll want to be conservative in how often you mess with their expectations. If you're designing for something like a museum kiosk, though, you can be more liberal in your frequent changes. Number five, nest your design cycles. In agile development you go through many small design cycles rapidly and each cycle gives you a tiny bit of new information. Take all that new information you gather and use it in the context of a broader, more traditional design cycle, aimed at long-term substantive improvements instead of small optimizations.

Exploring HCI: Agile Development

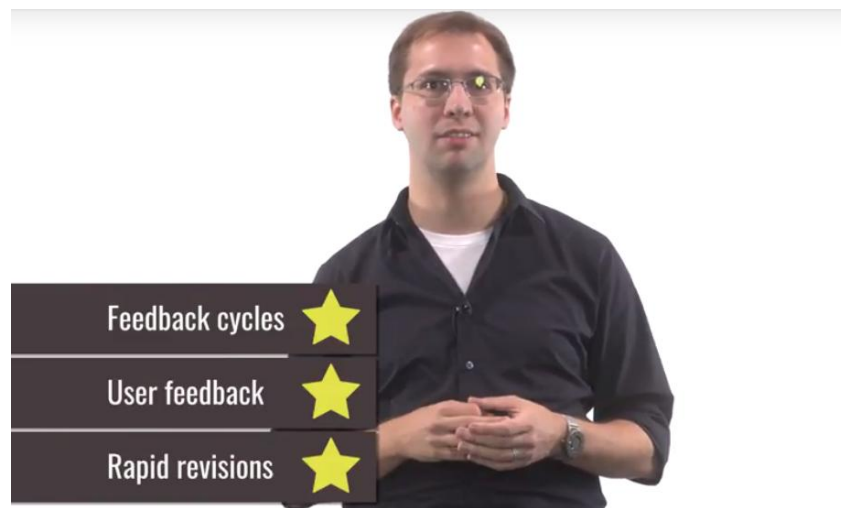


Does the area of HCI on which you chose to focus lend itself naturally to agile development? There are a lot of questions to ask in that area. Are you working in a high stakes area like healthcare or autonomous vehicles? What's the cost of failure? If it's high, you might want to avoid agile development. After all, it's built in large part around learning from the real failures of real users. If that's a user unfairly failing to reach the next of a game, that's probably fine. If that's a doctor entering the wrong dosage of a medication into a new interface, that's not fine. You also need to think of development costs. Agile development relies on being able to get a product up and out the door quickly, and change it frequently. If any part of your design is reliant on the hardware, then agile development presents challenges. It might be easy to roll out a software update to improve a car screen interface, but you can't download a car to fix a hardware problem. Now take a moment, and think about whether agile development would be right for the area of application that you chose.

Conclusion to Agile Methods



In this lesson we've covered a small glimpse of how HCI can work in a more agile development environment. In many ways they're a nice match.



Both emphasize feedback cycles, both emphasize getting user feedback, and both emphasize rapid changes. But while HCI traditionally has done these behind the scenes before reaching real users, Agile emphasizes doing these live. Now it's important to note, I've only provided a narrow glimpse into what Agile development is all about. I've discussed how HCI matches with the theory and the goals of Agile development, but Agile is a more complex suite of workflows and stakeholders. I really recommend reading more about it before you try to take an Agile approach to HCI, or before you try to integrate interaction design into an existing Agile team. As you do though, I think you'll notice that there can be a really nice compatibility between the two.

3.8 Conclusion to Methods

Compiled by Shipra De, Summer 2017

Introduction



In this unit we've discussed the HCI research methods that form the design life cycle, an iterative process between needfinding, brainstorming design alternatives, prototyping, and evaluation with real users.



We've also discussed the ethics behind this kind of research.

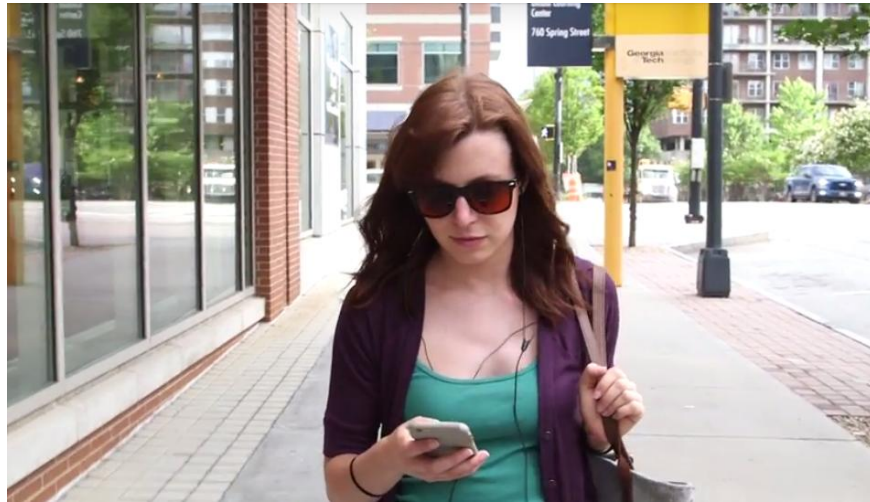


And how it applies to some more modern agile software development methodologies. In this wrap-up lesson, we want to explore a couple examples of the full design life cycle in action.



We also want to tie it into the design principles unit and explore how we can use the design principles and research methods in conjunction with one another.

Designing Audiobooks for Exercisers 1



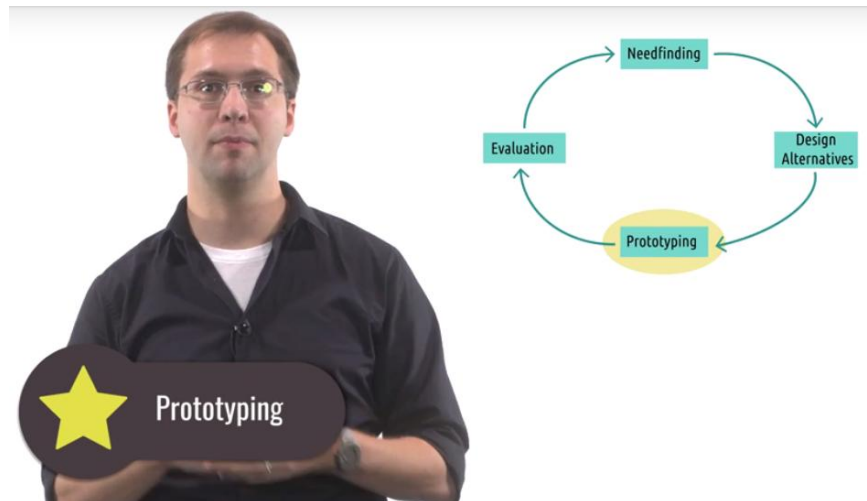
Throughout this unit we've used the running example of designing an audiobook app that would let people who are exercising interact with books in all the ways you or I might while sitting and reading. That means being able to leave bookmarks, take notes, and so on.



We discussed doing our foundational needfinding, going to a park and observing people exercising. We talked about doing some interviewing and surveys to find out more targeted information about what people wanted and needed.



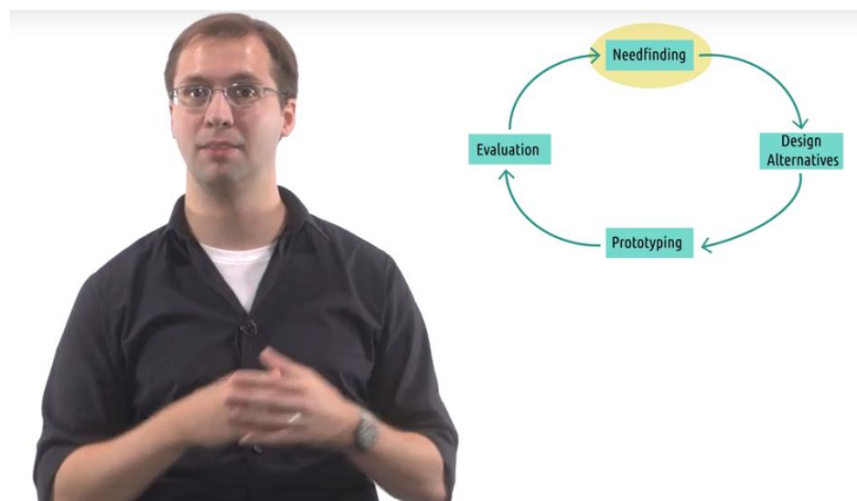
Then, based on that, we brainstormed a whole lot of alternatives. We thought about those alternatives in terms of different scenarios and personas to settle on those with the most potential.



Then, we took those alternatives and prototyped a few of them. Specifically, we constructed Wizard of Oz prototypes for voice and gesture interfaces and paper prototypes for on-screen interfaces.



Then, we put those in front of our users -- well, a user, in my case, but you would use more users, and we got some initial feedback. So, at the end of one iteration of the design life cycle, we have three different low-fidelity prototypes, each with some feedback on how effectively they work. But as you can tell, we're not done yet. We don't have an app.



What's next? Next, we go through another phase of the design life cycle.

Designing Audiobooks for Exercisers 2



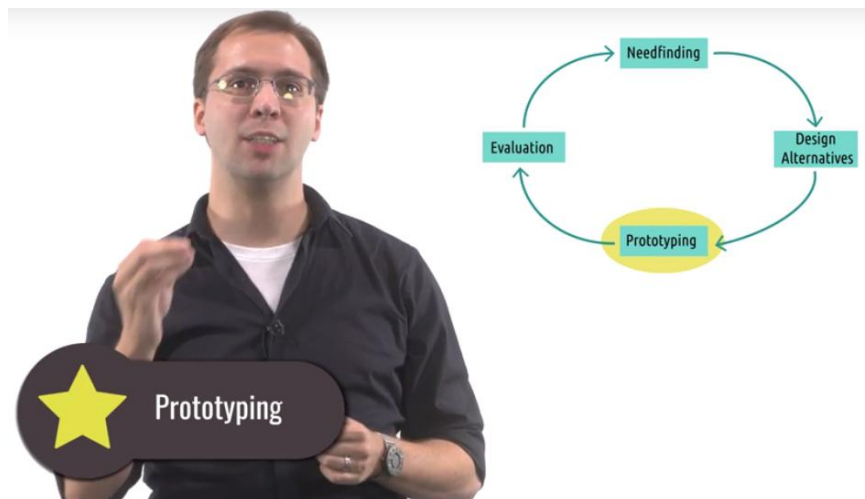
We take the results of our initial iteration through the design cycle and use the results to return to the needfinding process. That's not to say we need to redo everything from scratch, but our prototypes and evaluation have now increased our understanding of the problem. There are things we learn by prototyping and evaluating about the task itself. In this case, we could have learned that even for exercisers with their hands free, gestures are still tough because they're moving around so much. The evaluation process may have also given us new questions we want to ask users to understand the task better. For example, Morgan mentioned needing to be able to rewind. We might want to know how common a problem that is.



In many ways, synthesizing our experiences with the evaluation is our next needfinding process.



We then move on to design alternatives stage. Again, that doesn't mean starting from scratch and coming up with all new ideas. Here it means expanding on our current ideas, fleshing them out a bit more, and brainstorming them in terms of those personas and scenarios we used previously. We might also come up with whole new ideas here based on our first iteration.



Then, more prototyping. At this point, we might discover that as we try to increase the fidelity of our prototypes, the technology or resources aren't quite there yet. For example, while the gesture interface might have been promising in the Wizard of Oz prototype, we don't yet have the technology to recognize gestures that way on the go. Or we might have found that the expense related to the prototype is unfeasible, or the realizing the prototype would require violating some of our other user needs. So, for example, we could do gesture recognition if we had users hold a physical device that could recognize gestures, but that might be too expensive to produce, or it might conflict with our audience's need for a hands-free system. So we move on with the prototypes that we can build, with the goal of getting to the feedback stage as quickly as possible. For voice recognition, instead of trying to build a full voice recognition system, maybe we just build a system that can recognize very simplistic

voice commands. Instead of recognizing words, maybe it just recognizes the number of utterances if that's easier to build. For the screen, maybe we build a wireframe prototype that moves between different screens on a phone, but we don't connect it to a real system. We still have someone run alongside the exerciser and play the book according to their commands. That way we focus on usability instead of things like integration with audiobook apps or voice-to-text transcription, things that take a lot of work to get right and might end up unnecessary if we find that the prototype isn't actually useful.



Then, we evaluate again. This time, we probably get a little more objective. We still want data on the qualitative user experience, but we also want data on things like: how long does it take a user to perform the desired actions in the interface? What prevents them from working with the interface? Imagine that we found, for instance, that for many exercisers, they go through places that are too loud for voice commands to work. Or, we find that the time it takes to pull out the interface and interact is too distracting. That information is once again useful to our ongoing iteration. At the end of that process, we again have some higher-fidelity prototypes, but no product yet. So, we go again.

Designing Audiobooks for Exercisers 3



At the end of the last iteration through the design cycle, we had two interface prototypes, each with significant weaknesses. Our voice command interface struggled in loud areas where exercises are often exercising, and our screen-based interface presented too high a gulf of execution. But notice how far we've come at this point. We now have a pretty complete and nuanced view of the task and our possible solutions. Now, let's go through one more iteration to get to something we can actually implement and deploy.



Our needfinding has come along to the point of understanding that completely hands-free interfaces are more usable, but we also know that gesture-based is technologically unfeasible and voice-based isn't perfectly reliable.



Now we might come up with a new alternative. A hybrid system. The voice interaction and on-screen touch interaction aren't incompatible with one another. Our new alternative is to develop a system that supports both, allowing users to use voice commands most of the time, but default to touch commands in situations where the voice commands don't work. So they always have full functionality, but usability is still maximized.



So, we create a new prototype, basically merging our two from the previous iteration. They're still reasonably low fidelity because we haven't tested this combination yet, and the next stage of sophistication is going to be expensive. So, we want to make sure it's worth pursuing.

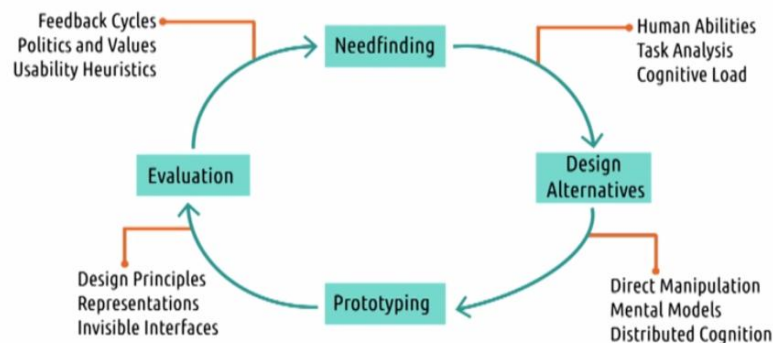


Then, we evaluate that with users, and we find it's good enough to go ahead and move forward with producing it.

Designing Audiobooks for Exercisers 4

So that's the end, right? We went through a few iterations of the design life cycle getting iteratively more high-fidelity and rigorous with our evaluation. Finally, we have a design we like. We implement it fully, submit it to the app store, and sit back while the money rolls in. Not exactly. Now instead of having a handful of users we bring in to use our interface, we have hundreds of users using it in ways we never expected. And now the cycle begins again. We have data we're automatically collected either through usage tracking or error logs. We have user reviews or feedback they submit. So, we jump back into needfinding using the data we have available to us. We might find subtle needs, like the need for more control over rewinding and fast forwarding. We might move on and prototype that with commands like 'back 5' and 'back 15'. We might uncover more novel new needs as well: we find there's a significant contingent of people using the interface while driving. It's similar in that it's another place where people's hands and eyes are occupied, but it has its own unique needs as well, like the ability to run alongside a navigation app. So the process starts again, this time with live users' data. And in general, it never really ends. Nowadays, you very rarely see interfaces, apps, programs, or web sites that are intentionally put up once and never changed. That might happen because the designers got busy or the company went out of business, but it's rarely one-off by design. And as the design evolves over time with real data, you'll start to see nested feedback cycles: week to week small additions give way to month-to-month updates and year-to-year reinventions. In many ways, your interface becomes like a child: you watch it grow up and take on a life of its own.

Research Methods Meet Design Principles



The design principles we describe in our other unit are deeply integrated throughout this design life cycle. They don't supplant it -- you won't be making any great designs just by applying principles -- but they can streamline things. In many ways, design principles capture takeaways and conclusions found by this design life cycle in the past in ways that can be transferred to new tasks or new interfaces. Many of our needs are driven by our current understanding of human abilities. Task analysis allows us to describe those needs, those tasks, in formal ways to equip the interface design process. And cognitive load lets us keep in mind how much users are asked to do at a time. Direct manipulation gives us a family of techniques that we want to emphasize in coming up with our design alternatives. Mental models provide us an understanding of how the design alternatives might mesh with the user's understanding of the task. And distributed cognition gives us a view on interface design that lends itself to design at a larger level of granularity. Here we're designing systems, not just interfaces. Design principles in general give us some great rules of thumb to use when creating our initial prototypes and designs. Our understanding of representations ensures that the prototypes we create match with users' mental models that we uncovered before. Invisible interfaces help us remember that the interface should be the conduit between the user and the task, not the focus of attention itself. Then the vocabulary of the feedback cycle, the gulfs of execution and evaluation, give us ways to evaluate and describe our evaluations of the interfaces that we design. The notion of politics and values in interfaces allow us to evaluate the interface not just in terms of its usable interactions, but in the types of society it creates or preserves. And those usability heuristics that we applied to our prototyping are also a way of evaluating our interface and mentally simulating what a user will be thinking while using our creation. These principles of HCI were all found through many years of going through the design life cycle, creating different interfaces, and exploring and evaluating their impact. By leveraging those lessons, we can speed to usable interfaces much faster. But applying those lessons doesn't remove the need to go talk to real users.

Exploring HCI: HCI Methods

Over the past several lessons, you've been exploring how the design life cycle applies to the area of HCI that you chose to explore. Now that we've reached the end of the unit, take a moment and reflect on the life cycle you developed. How feasible would it be to actually execute? What would you need? What kind of users do you need? How many? When do you need them? There are right answers here, of course: ideally, you'll need users early and often. That's what user-centered design is all about. In educational technology, that might mean having some teachers, students, and parents you can contact frequently. In computer-supported cooperative work, that might mean having a community you can visit often to see the new developments. In ubiquitous computing, that might mean going as far as having someone who specializes in low-fidelity 3D prototypes to quickly spin up new ideas for testing. Now that you understand the various phases of the design life cycle, take a moment and reflect on how you'll use it iteratively as a whole in your chosen area of HCI.

Approaches to User-Centered Design

At a minimum, user-centered design advocates involving users throughout the process through surveys, interviews, evaluations, and more that we'll talk about. However, user-centered design can be taken to even greater extremes through a number of approaches beyond what we've covered.



One is called participatory design. In participatory design, all the stakeholders -- including the users themselves -- are involved as part of the design team. They aren't just a source of data, they're actually members of the design team working on the problem. That allows the user perspective to be pretty omnipresent throughout the design process. Of course, there's still a danger there: generally, we are not our user, but in participatory design one of the designers *is* the user... but they're not the only user. So, it's a great way to get a user's perspective, but we must also be careful not to over-represent that one user's view.



A second approach is action research. Action research is a methodology that addresses an immediate problem, and researches it by trying to simultaneously solve it. Data gathered on the success of the

approach is then used to inform the understanding of the problem and future approaches. Most importantly, like participatory design, action research is undertaken by the actual users. For example, a teacher might engage in action research by trying a new activity in his classroom and reflecting on the results, or a manager might use action research by trying a new evaluation system with her employees and noting the changes.

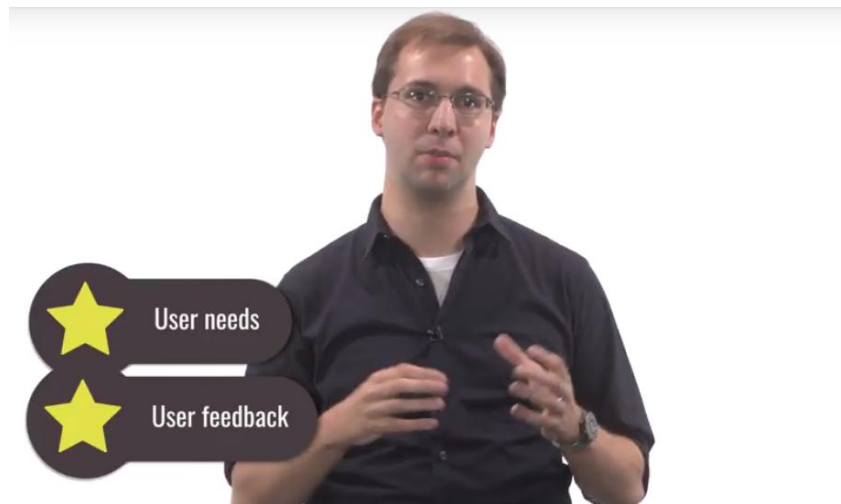


A third approach is design-based research. Design-based research is similar to action research, but it can be done by outside practitioners, as well. It's especially common in learning sciences research. In design-based research, designers create interventions based on current understanding of the theory and the problem, and use the success of those interventions to improve our understanding of the theory or the problem. For example, if we believed a certain intersection had a lot of jaywalkers because the signs had poor visibility, we might interview people at the intersection for their thoughts. Or, we could create a solution that assumes we're correct, and then use it to evaluate whether or not we were correct. If we create a more clearly visible sign and it fixes the problem, then it suggests our initial theory was correct. In all these approaches, notice iteration still plays a strong roll: we never try out just one design and stop. We run through the process, create a design, try it out, and then iterate and improve on it. Interface design is never done: it just gets better and better as time goes on, while also adjusting to new trends and technologies.

Conclusion



This wraps up our conversation on research methods and the design life cycle. The purpose of this is to put a strong focus on user-centered design throughout the process.



We want to start our designs by understanding user needs, then get user feedback throughout the design process. As we do, our understanding of the user and the task improves, and our designs improve along with it. Even after we've released our designs, modern technology allow us to continue that feedback cycle, continually improving our interfaces and further enhancing the user experience.

4.1 Applications: Technology

Compiled by Shipra De, Summer 2017

Introduction to Applications

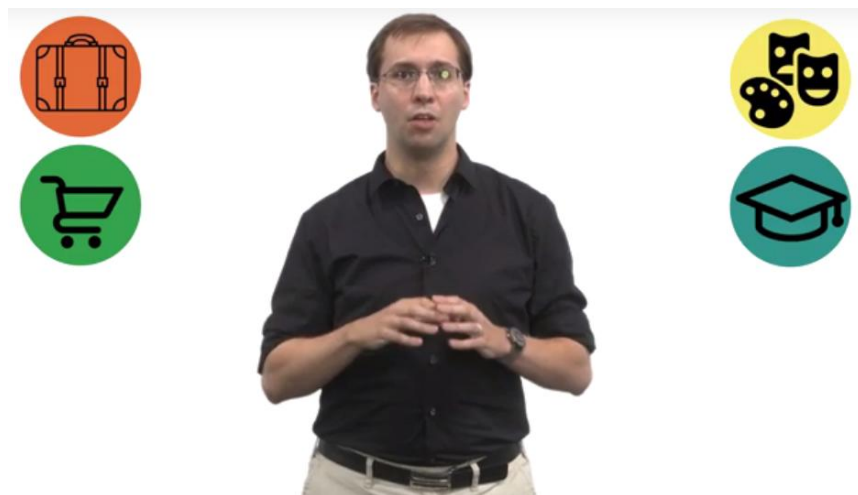


If you're watching this course in the order it was originally produced, you've now learned the foundational principles of HCI and the research methods behind developing interfaces. At the beginning of the course, we also previewed some of the open areas of HCI development and research. Now, what we'd like to do is give you a jump start in looking into the areas that are interesting to you. In the lessons that follow, we'll replay the preview videos for each topic from the beginning of the course, and then provide a small library of information on each topic. We certainly don't expect you to go through every portion of all of these lessons. Find what's interesting to you, and use these materials as your jumping-off point.

Technology: Virtual Reality



The year that I'm recording this is what many have described as the year that virtual reality finally hits the mainstream. By the time you watch this, you'll probably be able to assess whether or not that was true, so come back in time and let me know. Virtual reality is an entire new classification of interaction and visualization and we're definitely still at the beginning of figuring out what we can do with these new tools. You could be one of the ones who figures out the best way to solve motion sickness or how to get proper feedback on gestural interactions.



A lot of the press around virtual reality has been around video games, but that's definitely not the only application. Tourism, commerce, art, education, virtual reality has applications to dozens of spaces.



For example, there is a lab in Michigan that's using virtual reality to treat phobias. They're creating a safe space where people can very authentically and realistically confront their fears. The possible applications of virtual reality are really staggering. So I'd encourage you to check them out as you go through this class.

Virtual Reality Resources

Recommended Academic Resources:

- Lécuyer, A., Lotte, F., Reilly, R. B., Leeb, R., Hirose, M., & Slater, M. (2008). [Brain-computer interfaces, virtual reality, and videogames](#). *IEEE Computer*, 41(10), 66-72.
- Sutcliffe, A., & Gault, B. (2004). [Heuristic evaluation of virtual reality applications](#). *Interacting with Computers*, 16(4), 831-849.
- Riva, G., Baños, R. M., Botella, C., Mantovani, F., & Gaggioli, A. (2016). [Transforming Experience: The Potential of Augmented Reality and Virtual Reality for Enhancing Personal and Clinical Change](#). *Frontiers in Psychiatry*, 7.
- Gugenheimer, J., Wolf, D., Eiriksson, E. R., Maes, P., & Rukzio, E. (2016, October). [Gyrov: Simulating inertia in virtual reality using head worn flywheels](#). In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (pp. 227-232). ACM.
- Bian, D., Wade, J., Warren, Z., & Sarkar, N. (2016, July). [Online Engagement Detection and Task Adaptation in a Virtual Reality Based Driving Simulator for Autism Intervention](#). In *International Conference on Universal Access in Human-Computer Interaction* (pp. 538-547). Springer International Publishing.
- North, M. M., & North, S. M. (2016). [Virtual reality therapy](#). In *Computer-Assisted and Web-Based Innovations in Psychology, Special Education, and Health*, 141.

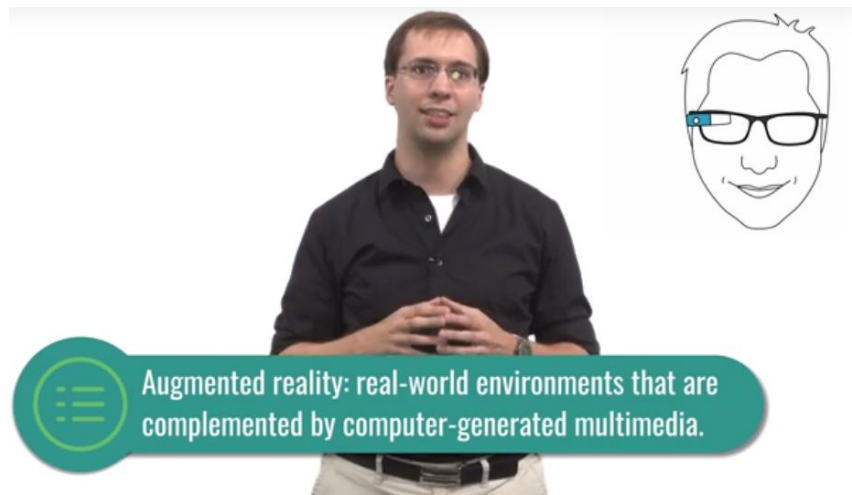
Recommended Books:

- Biocca, F., & Levy, M. R. (Eds.). (2013). *Communication in the age of virtual reality*. Routledge.
- Earnshaw, R. A. (Ed.). (2014). *Virtual reality systems*. Academic Press.

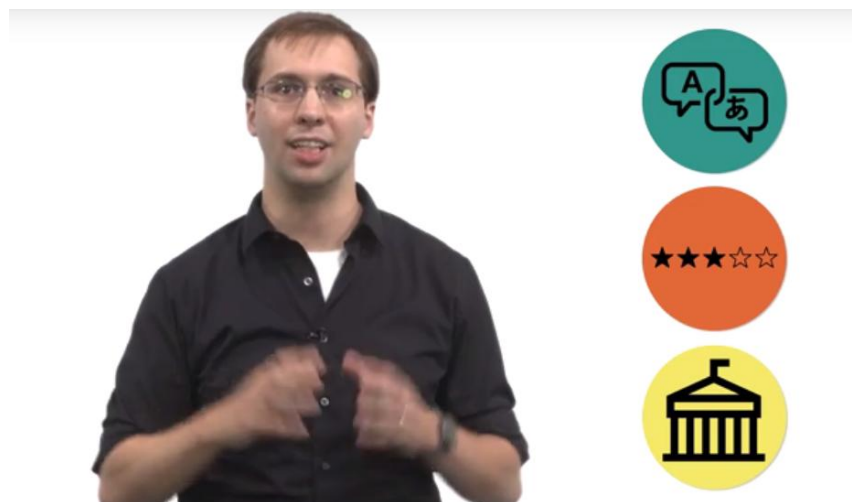
Dedicated Courses:

- [Introduction to Virtual Reality](#), Udacity
- [Make VR Games in Unity with C#](#), Udemy

Technology: Augmented Reality



Virtual reality generally works by replacing the real world's visual, auditory, and sometimes even all factory or kinesthetic stimuli with it's own input. Augmented reality on the other hand, compliments what you see and hear in the real world. So for example, imagine a headset like a Google Glass that automatically overlays directions right on your visual field. If you were driving, it would highlight the route to take, instead of just popping up some visual reminder. The input it provides complements stimuli coming from the real world, and instead of just replacing them. And that creates some enormous challenges, but also some really incredible opportunities as well.



Imagine the devices that can integrate directly into our everyday lives, enhancing our reality. Imagine systems that could, for example, automatically translate text or speech in a foreign language, or could show your reviews for restaurants as you walk down the street. Imagine a system that students could use while touring national parks or museums, that would automatically point out interesting information, custom tailored to that student's own interests. The applications of augmented reality could be truly stunning, but it relies on cameras to take input from the world, and that actually raises

some interesting societal problems. There are questions about what putting cameras everywhere would mean. So keep those in mind when we get to interfaces and politics, in unit two.

Augmented Reality Resources

Recommended Academic Resources:

- Olsson, T., Lagerstam, E., Kärkkäinen, T., & Väänänen-Vainio-Mattila, K. (2013). [Expected user experience of mobile augmented reality services: a user study in the context of shopping centres](#). *Personal and Ubiquitous Computing*, 17(2), 287-304.
- Chang, K. E., Chang, C. T., Hou, H. T., Sung, Y. T., Chao, H. L., & Lee, C. M. (2014). [Development and behavioral pattern analysis of a mobile guide system with augmented reality for painting appreciation instruction in an art museum](#). *Computers & Education*, 71, 185-197.
- Hürst, W., & Van Wezel, C. (2013). [Gesture-based interaction via finger tracking for mobile augmented reality](#). *Multimedia Tools and Applications*, 62(1), 233-258.
- Lv, Z., Halawani, A., Feng, S., Ur Réhman, S., & Li, H. (2015). [Touch-less interactive augmented reality game on vision-based wearable device](#). *Personal and Ubiquitous Computing*, 19(3-4), 551-567.
- Lee, K. (2012). [Augmented reality in education and training](#). *TechTrends*, 56(2), 13-21.
- Suárez-Warden, F., Barrera, S., & Neira, L. (2015). [Communicative Learning for Activity with Students Aided by Augmented Reality within a Real Time Group HCI](#). *Procedia Computer Science*, 75, 226-232.
- Anderson, F., Grossman, T., Matejka, J., & Fitzmaurice, G. (2013, October). [YouMove: enhancing movement training with an augmented reality mirror](#). In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (pp. 311-320). ACM.
- Zhu, E., Hadadgar, A., Masiello, I., & Zary, N. (2014). [Augmented reality in healthcare education: an integrative review](#). *PeerJ*, 2, e469.

Recommended Books:

- Schmalstieg, D. & Hollerer, T. (2016). [Augmented Reality: Principles and Practice](#). Addison-Wesley Professional.
- Craig, A. B. (2013). [Understanding Augmented Reality: Concepts and Applications](#). Newnes.

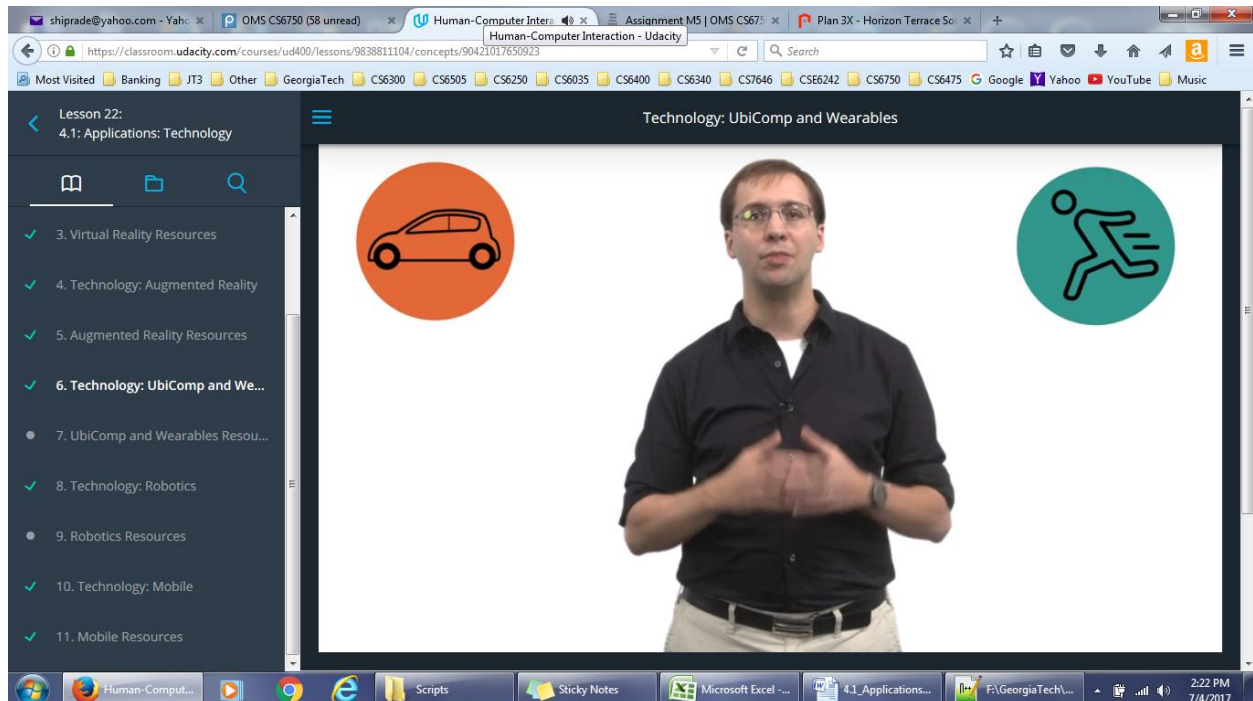
Dedicated Courses:

- [Internet of Things & Augmented Reality Emerging Technologies](#), Yonsei University via Coursera
- [Getting started with Augmented Reality](#), Institut Mines-Télécom

Technology: UbiComp and Wearables



Ubiquitous Computing refers to trend towards embedding computing power in more and more everyday objects. You might also hear it referred to as pervasive computing, and it's deeply related to the emerging idea of an Internet of Things. A few years ago, you wouldn't have found computers in refrigerators and wristwatches, but as microprocessors became cheaper and as the world became increasingly interconnected, computers are becoming more and more ubiquitous.



Modern HCI means thinking about whether someone might use a computer while they're driving a car or going on a run. It means figuring out how to build smart devices that offloads some of the cognitive

load from the user, like refrigerators that track their own contents and deliver advice to the users right at the right time.



This push for increasing pervasiveness has also led to [SOUND] the rise of wearable technologies. Exercise monitors are probably the most common examples of this, but smart watches, Google Glass, augmented reality headsets, and even things like advanced hearing aids and robotic prosthetic limbs, are all examples of wearable technology. This push carries us into areas usually reserved for human factors engineering and industrial design, which exemplifies the increasing role of HCI in the design of new products.

UbiComp and Wearables Resources

Recommended Academic Resources:

- Starner, T. (2013). [Project glass: An extension of the self](#). *IEEE Pervasive Computing*, 12(2), 14-16.
- Lv, Z., Halawani, A., Feng, S., Ur Réhman, S., & Li, H. (2015). [Touch-less interactive augmented reality game on vision-based wearable device](#). *Personal and Ubiquitous Computing*, 19(3-4), 551-567.
- Clear, A. K., Comber, R., Friday, A., Ganglbauer, E., Hazas, M., & Rogers, Y. (2013, September). [Green food technology: UbiComp opportunities for reducing the environmental impacts of food](#). In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing*, 553-558. ACM.
- De Haan, G. (2013, April). [A Vision of the Future of Media Technology Design Education-design and education from HCI to UbiComp](#). In *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research*, 67-72. Open Universiteit, Heerlen.

Recommended Books:

- Krumm, J. (2016). [Ubiquitous Computing Fundamentals](#). CRC Press.
- Sazonov, E. & Neuman, M. (2014). [Wearable Sensors: Fundamentals, Implementation, and Applications](#). Academic Press.

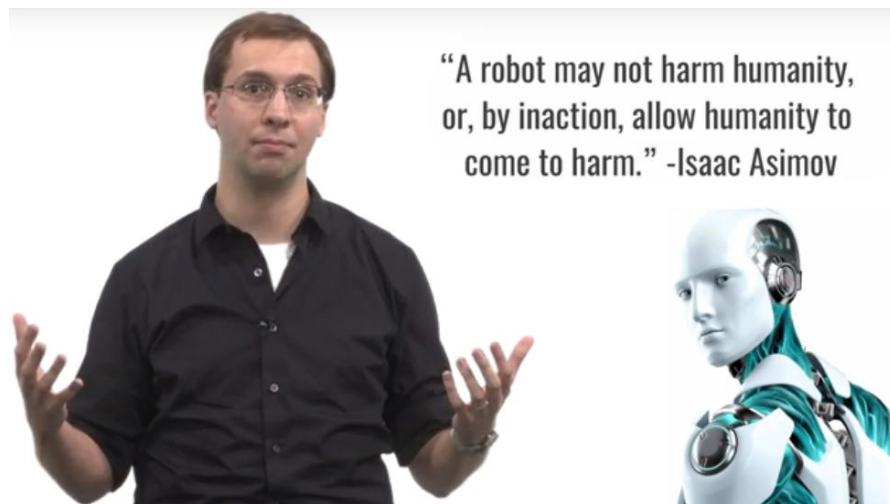
Dedicated Courses:

- [Android Wear Development](#), by Google via Udacity

Technology: Robotics



A lot of the current focus on robotics is on their physical construction and abilities or on the artificial intelligence that underlies their physical forms. But as robotics becomes more and more mainstream, we're going to see the emergence of a new subfield of human-computer interaction, human-robot interaction. The field actually already exists. The first conference on human robot interaction took place in 2006 in Salt Lake City, and several similar conferences have been created since then. Now as robots enter the mainstream, we're going to have to answer some interesting questions about how we interact with them.



For example, how do we ensure that robots don't harm humans through faulty reasoning. How do we integrate robots into our social lives, or do we even need to? As robots are capable of more and more, how do we deal with the loss of demand for human work? Now these questions all lie at the intersection of HCI, artificial intelligence and philosophy in general. But there are some more concrete questions we can answer as well. How do we pragmatically equip robots with the ability to naturally

interact with humans based on things like voice and touch? How do we provide tasks that subtle feedback to humans interacting with robots to confirm their input is being received and properly understood? How do we support humans in teaching things to robots, instead of just programming them? Or alternatively, can we create robots that can teach things to humans? We already see robotics advances applied to things like healthcare and disability services. And I'm really excited to see where you take it next.

Robotics Resources

Recommended Academic Resources:

- Glas, D., Satake, S., Kanda, T., & Hagita, N. (2012, June). [An interaction design framework for social robots](#). In *Robotics: Science and Systems* 7, 89 - 96.
- Toris, R., Kent, D., & Chernova, S. (2014). [The robot management system: A framework for conducting human-robot interaction studies through crowdsourcing](#). *Journal of Human-Robot Interaction*, 3(2), 25-49.
- Beer, J., Fisk, A. D., & Rogers, W. A. (2014). [Toward a framework for levels of robot autonomy in human-robot interaction](#). *Journal of Human-Robot Interaction*, 3(2), 74.
- Schirner, G., Erdogmus, D., Chowdhury, K., & Padir, T. (2013). [The future of human-in-the-loop cyber-physical systems](#). *Computer*, 1, 36-45.
- Ruhland, K., Peters, C. E., Andrist, S., Badler, J. B., Badler, N. I., Gleicher, M., ... & McDonnell, R. (2015, September). [A Review of Eye Gaze in Virtual Agents, Social Robotics and HCI: Behaviour Generation, User Interaction and Perception](#). In *Computer Graphics Forum* 34(6), 299-326.
- Naveed, S., Rao, N. I., & Mertsching, B. (2014). [Multi Robot User Interface Design Based On HCI Principles](#). *International Journal of Human Computer Interaction (IJHCI)*, 5(5), 64.
- Złotowski, J., Proudfoot, D., Yogeewaran, K., & Bartneck, C. (2015). [Anthropomorphism: opportunities and challenges in human-robot interaction](#). *International Journal of Social Robotics*, 7(3), 347-360.
- Broz, F., Nourbakhsh, I., & Simmons, R. (2013). [Planning for human-robot interaction in socially situated tasks](#). *International Journal of Social Robotics*, 5(2), 193-214.

Recommended Books:

- Dautenhahn, K. & Saunders, J. (Eds.) (2011). [New Frontiers in Human-Robot Interaction](#). John Benjamins Publishing Co.
- Rahimi, M. & Karwowski, W. (Eds.) (2003). [Human-Robot Interaction](#). Taylor & Francis.

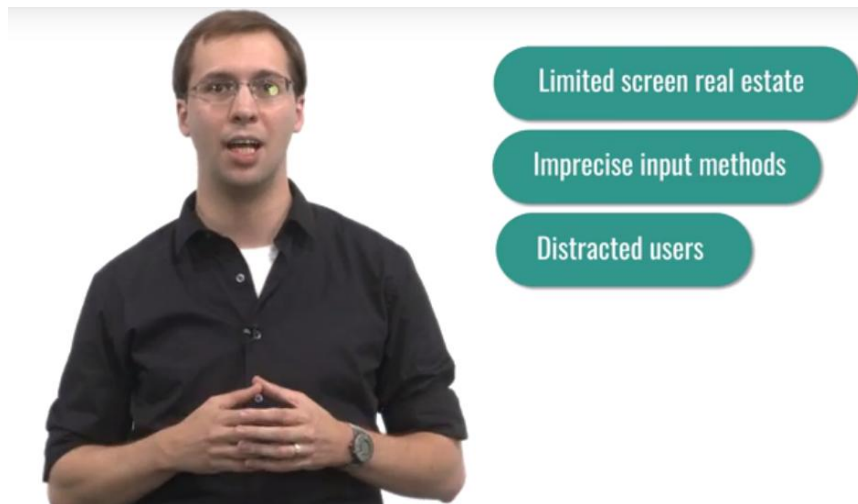
Dedicated Courses:

- [Autonomous Mobile Robots](#), from ETH Zurich via edX
- [Robotics Specialization](#), from University of Pennsylvania via Coursera
- [Artificial Intelligence for Robotics](#), from Georgia Tech via Udacity

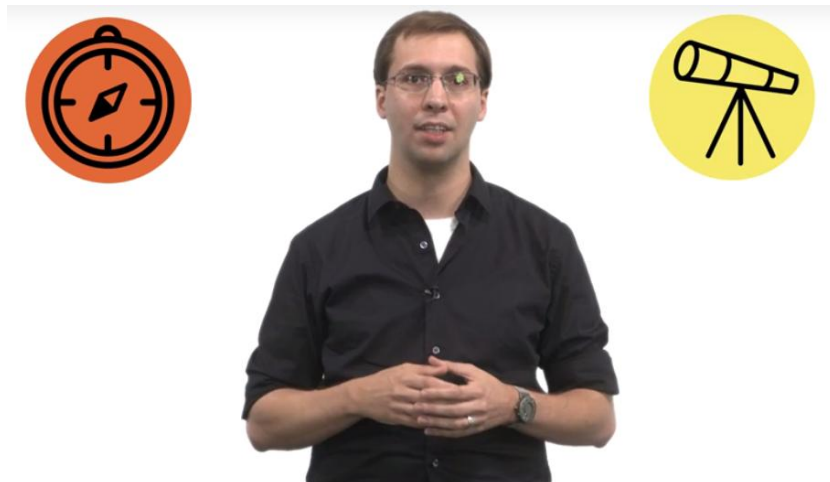
Technology: Mobile



One of the biggest changes to computing over the past several years has been the incredible growth of mobile as a computing platform. We really live in a mobile first world and that introduces some significant design challenges.



Screen real estate is now far more limited, the input methods are less precise and the user is distracted. But mobile computing also presents some really big opportunities for HCI. Thanks in large part to mobile we're no longer interested just in a person sitting in front of a computer. With mobile phones, most people have a computer with them at all times anyway.



We can use that to support experiences from navigation to star gazing. Mobile computing is deeply related to fields like context aware computing, ubiquitous computing and augmented reality, as it possesses the hardware necessary to compliment those efforts. But even on its own, mobile computing presents some fascinating challenges to address.



For me, the big one is that we haven't yet reached a point where we can use mobile phones for all the tasks we do on computers. Smart phones are great for social networking, personal organization, games, and lots of other things. But we haven't yet reached a point where the majority of people would sit down to write an essay, or do some programming on smart phones. Why haven't we? What do we need to do to make smart phones into true replacements for traditional desktop and laptop computers?

Mobile Resources

Recommended Academic Resources:

- Kjeldskov, J., & Paay, J. (2012, September). [A longitudinal review of Mobile HCI research methods](#). In *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services*, 69-78. ACM.
- Kjeldskov, J., & Skov, M. B. (2014, September). [Was it worth the hassle?: ten years of mobile HCI research discussions on lab and field evaluations](#). In *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services*, 43-52. ACM.
- McMillan, D., Morrison, A., & Chalmers, M. (2013, April). [Categorised ethical guidelines for large scale mobile HCI](#). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1853-1862. ACM.
- Sakamoto, D., Komatsu, T., & Igarashi, T. (2013, August). [Voice augmented manipulation: using paralinguistic information to manipulate mobile devices](#). In *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services*, 69-78. ACM.

Conference Proceedings:

- [18th International Conference on Human-Computer Interaction with Mobile Devices and Services](#)
- [17th International Conference on Human-Computer Interaction with Mobile Devices and Services](#)
- [16th International Conference on Human-Computer Interaction with Mobile Devices and Services](#)
- [15th International Conference on Human-Computer Interaction with Mobile Devices and Services](#)

Recommended Books:

- Love, S. (2005). [Understanding Mobile Human-Computer Interaction](#). Elsevier.
- Kjeldskov, J. (2014). [Mobile Interactions in Context: A Designerly Way Toward Digital Ecology](#). Morgan & Claypool Publishers.

Dedicated Courses:

- [UX Design for Mobile Developers](#), by Google via Udacity
- [Mobile User Experience \(UX\) Design](#), by Interaction-Design.org

4.2 Applications: Ideas

Compiled by Shipra De, Summer 2017

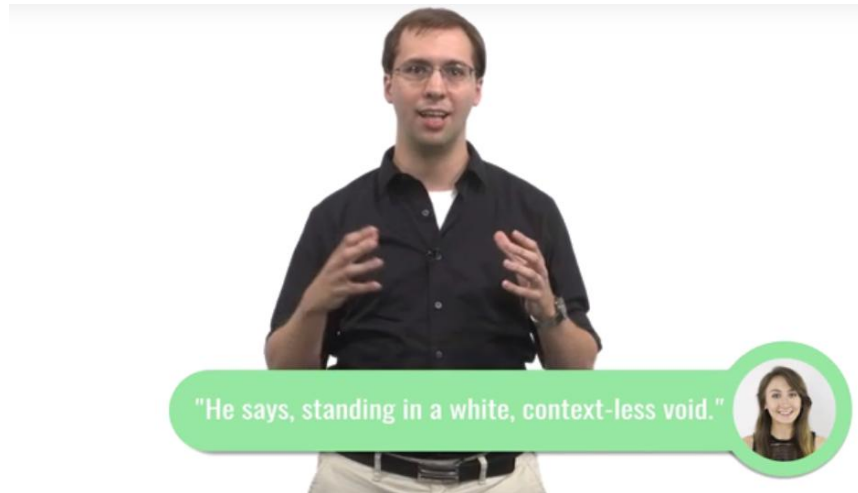
Idea: Context-Sensitive Comp



What time is it?

>> You can go ahead and go to lunch.

Did that exchange make any sense? I asked Amanda for the time and she replied by saying I can go ahead and go get lunch. The text seems completely non-sensical and yet hearing that, you may have filled in the context that makes this conversation logical. You might think that I asked a while ago what time we were breaking for lunch, or maybe I mentioned that I forgot to eat breakfast. Amanda would have that context and she could use it to understand why I'm probably asking for the time. Context is a fundamental part of the way humans interact with other humans. Some lessons we'll talk about even suggest that we are completely incapable of interacting without context.



If context is such a pervasive part of the way humans communicate, then to build good interfaces between humans and computers, we must equip computers with some understanding of context.



That's where context-sensitive computing comes in. Context-sensitive computing attempts to give computer interfaces the contextual knowledge that humans have in their everyday lives. For example, I use my mobile phone differently depending on whether I'm sitting on the couch at home, or using it in my car, or walking around on the sidewalk. Imagine I didn't have to deliberately inform my phone of what mode I was in though. Imagine if it just detected that I was in my car and automatically brought up Google Maps and Audible for me. Services have started to emerge to provide this, but there's an enormous amount of research to be done on context sensitive computing. Especially as it relates to things like wearables, augmented reality, and ubiquitous computing.

Context-Sensitive Computing Resources

Recommended Academic Resources:

- Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999, September). [Towards a better understanding of context and context-awareness](#). In *International Symposium on Handheld and Ubiquitous Computing* (pp. 304-307). Springer Berlin Heidelberg.
- Yau, S. S., Karim, F., Wang, Y., Wang, B., & Gupta, S. K. (2002). [Reconfigurable context-sensitive middleware for pervasive computing](#). *IEEE Pervasive Computing*, 1(3), 33-40.
- Brown, B., & Randell, R. (2004). [Building a context sensitive telephone: some hopes and pitfalls for context sensitive computing](#). *Computer Supported Cooperative Work (CSCW)*, 13(3-4), 329-345.
- Bouabid, A., Lepreux, S., Kolski, C., & Havrez, C. (2014, July). [Context-sensitive and Collaborative application for Distributed User Interfaces on tabletops](#). In *Proceedings of the 2014 Workshop on Distributed User Interfaces and Multimodal Interaction* (pp. 23-26). ACM.

Recommended Books:

- Stojanovic, Dragan (Ed.). (2009). [Context-Aware Mobile and Ubiquitous Computing for Enhanced Usability: Adaptive Technologies and Applications: Adaptive Technologies and Applications](#). Information Science Reference.
- Brezillion, P. & Gonzalez, A. (Eds.). (2014). [Context in Computing: A Cross-Disciplinary Approach for Modeling the Real World](#). Springer.

Dedicated Courses:

- [Out of Context: A Course on Computer Systems That Adapt To, and Learn From, Context](#), from MIT OpenCourseware

Idea: Gesture-Based Interaction



As this course goes on, you'll find that I'm on camera more often than you're accustomed to seeing in a Udacity course. Around half this course takes place with me on camera. There are a couple of reasons for that. The big one is that this is Human Computer Interaction. So it makes sense to put strong emphasis on the Human. But another big one is that when I'm on camera, I can express myself through gestures instead of just word and voice intonations. I can for example make a fist and really drive home and emphasize a point. I can explain that a topic applies to a very narrow portion of the field or a very wide portion of the field. We communicate naturally with gestures every day. In fact, we even have an entire language built out of gestures. So wouldn't it be great if our computers could interpret our gestures as well?



That's the emerging field of Gesture-Based Interaction. You've seen this with things like the Microsoft Connect which has far reaching applications from healthcare to gaming. We've started to see some applications of gesture based interaction on the go as well with wrist bands that react to certain hand motions. Gesture based interaction has enormous potential. The fingers have some of the finest muscle

movements, meaning that a system based on finger movements could support an incredible number of interactions. We might see a day when it's possible to type invisibly in the air in front of you based on system's recognition of the movement in the muscles of your wrist. That might finally allow mobile devices to replace traditional computers altogether.

Gesture-Based Interaction Resources

Recommended Academic Resources:

- Waldherr, S., Romero, R., & Thrun, S. (2000). [A gesture based interface for human-robot interaction](#). *Autonomous Robots*, 9(2), 151-173.
- Hürst, W., & Van Wezel, C. (2013). [Gesture-based interaction via finger tracking for mobile augmented reality](#). *Multimedia Tools and Applications*, 62(1), 233-258.
- Lv, Z., Halawani, A., Feng, S., Ur Réhman, S., & Li, H. (2015). [Touch-less interactive augmented reality game on vision-based wearable device](#). *Personal and Ubiquitous Computing*, 19(3-4), 551-567.
- Steins, C., Gustafson, S., Holz, C., & Baudisch, P. (2013, August). Imaginary devices: gesture-based interaction mimicking traditional input devices.](<http://www.seangustafson.com/static/papers/2013-MobileHCI-Steins-ImaginaryDevices.pdf>) In *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services* (pp. 123-126). ACM.
- Lu, Z., Chen, X., Li, Q., Zhang, X., & Zhou, P. (2014). [A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices](#).pdf) *IEEE Transactions on Human-Machine Systems*, 44(2), 293-299.
- Mazalek, A., Shaer, O., Ullmer, B., & Konkel, M. K. (2014). [Tangible Meets Gestural: Gesture Based Interaction with Active Tokens](#). In *ACM CHI 2014 Workshop on Gesture-Based Interaction Design*, ACM CHI.
- Wilson, A. D. (2004, October). [TouchLight: an imaging touch screen and display for gesture-based interaction](#). In *Proceedings of the 6th International Conference on Multimodal Interfaces* (pp. 69-76). ACM.

Recommended Books:

- Premaratne, P. (2014). [Human Computer Interaction Using Hand Gestures](#). Springer Science & Business Media.
- Ji, Y. & Choi, S. (Eds.). [Advances in Affective and Pleasurable Design](#). AHFE Conference.

Dedicated Courses:

- [Input & Interaction](#), by University of California-San Diego via Coursera

Idea: Pen- and Touch-Based Interaction



I always find it interesting how certain technologies seem to come around full circle. For centuries we only interacted directly with the things that we built and then computers came along. And suddenly we needed interfaces between us and our tasks. Now, computers are trying to actively capture natural ways we've always interacted. Almost every computer I encounter now days has a touch screen. That's a powerful technique for creating simple user interfaces because it shortens the distance between the user and the tasks they're trying to accomplish. Think about someone using a mouse for the first time. He might need to look back and forth from the screen to the mouse, to see how interacting down here, change things he sees up here. With a touch based interface, he interacts the same way he uses things in the real world around him. A challenge can sometimes be a lack of precision, but to make up for that we've also create pen based interaction. Just like a person can use a pen on paper, they can also use a pen on a touch screen. And in fact, you might be quite familiar with that, because most Udacity courses use exactly that technology. They record someone writing on a screen. That gives us the precision necessary to interact very delicately and specifically with our task. And as a result tablet based interaction methods have been used in fields like art and music. Most comics you find on the internet are actually drawn exactly like this, combining the precision of human fingers with the power of computation.

Pen and Touch Interaction Resources

Recommended Academic Resources:

- Moran, T. P., Chiu, P., & Van Melle, W. (1997, October). [Pen-based interaction techniques for organizing material on an electronic whiteboard](#). In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology* (pp. 45-54). ACM.
- Ren, X., & Moriya, S. (2000). [Improving selection performance on pen-based systems: a study of pen-based interaction for selection tasks](#). *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(3), 384-416.
- Vogel, D., & Baudisch, P. (2007, April). [Shift: a technique for operating pen-based interfaces using touch](#). In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 657-666). ACM.
- Wilkinson, G., Kharrufa, A., Hook, J. D., Pursgrove, B., Wood, G., Haeuser, H., ... & Olivier, P. (2016, May). [Expressy: Using a Wrist-worn Inertial Measurement Unit to Add Expressiveness to Touch-based Interactions](#). In *Proceedings of the ACM Conference on Human Factors in Computing Systems: Association for Computing Machinery (ACM)*.
- Hardy, R., & Rukzio, E. (2008, September). [Touch & interact: touch-based interaction of mobile phones with displays](#). In *Proceedings of the 10th International Conference on Human-Computer Interaction with Mobile Devices and Services* (pp. 245-254). ACM.
- Häikiö, J., Wallin, A., Isomursu, M., Ailisto, H., Matinmikko, T., & Huomo, T. (2007, September). [Touch-based user interface for elderly users](#). In *Proceedings of the 9th International Conference on Human-Computer Interaction with Mobile Devices and Services* (pp. 289-296). ACM.

Recommended Books:

- Annett, M. (2014). [The Fundamental Issues of Pen-based Interaction with Tablet Devices](#). University of Alberta.
- Berque, D., Prey, J., & Reed, R. (2006). [The Impact of Tablet PCs and Pen-based Technology on Education: Vignettes, Evaluations, and Future Directions](#). Purdue University Press.
- Reed, R. (2010). [The Impact of Tablet PCs and Pen-based Technology on Education: Going Mainstream](#). Purdue University Press.

Dedicated Courses:

- [Input & Interaction](#), by University of California-San Diego via Coursera

Idea: Information Visualization



One of the biggest trends of the information age is the incredible availability of data. Scientists and researchers use data science and machine learning to look at lots of data and draw conclusions. But often times those conclusions are only useful if we can turnaround and communicate them to ordinary people. That's where information visualization comes in. Now at first glance you might not think of data visualization as an example of HCI. After all, I could draw a data visualization on a napkin and print in a newspaper and there's no computer involved anywhere in that process. But computers give us a powerful way to re-represent data in complex, animated, and interactive ways. We'll put links to some excellent examples in the notes. Now what's particularly notable about data visualization in HCI is the degree with which it fits perfectly with our methodologies for designing good interfaces. One goal of a good interface is to match the user's **mental model** to the reality of the task at hand. In the same way, the goal of information visualization is to match the reader's mental model of the phenomenon to the reality of it. So the same principles we discussed for designing good representations apply directly to designing good visualizations. After all, a visualization is just a representation of data.

Information Visualization Resources

Recommended Academic Resources:

- Stasko, J. T. (1990). Tango: A framework and system for algorithm animation. *Computer*, 23(9), 27-39.
- Fekete, J. D., Van Wijk, J. J., Stasko, J. T., & North, C. (2008). [The value of information visualization](#). In *Information visualization*. Springer Berlin Heidelberg.
- Kapler, T., & Wright, W. (2005). [GeoTime information visualization](#). *Information Visualization*, 4(2), 136-146.
- Gleicher, M., Albers, D., Walker, R., Jusufi, I., Hansen, C. D., & Roberts, J. C. (2011). [Visual comparison for information visualization](#). *Information Visualization*, 10(4), 289-309.
- Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). [A meta-study of algorithm visualization effectiveness](#). *Journal of Visual Languages & Computing*, 13(3), 259-290.

Recommended Books:

- Ware, C. (2012). [Information visualization: perception for design](#). Elsevier.
- Spence, R. (2001). [Information visualization](#). New York: Addison-Wesley.

Dedicated Courses:

- [Data & Visual Analytics](#), from Georgia Tech via Udacity
- [Information Visualization](#), from the University of Indiana

Idea: CSCW



CSCW stands for Computer-Supported Cooperated Work. The field is just what the name says. How do we use computers to support people working together. You're watching this course online. So odds are that you've experienced this closely. Maybe you've worked on a group project with a geographically distributed group. Maybe you've had a job working remotely. Distributed teams are one example of CSCW in action but there are many others. The community often breaks things down into two dimensions.



Time and place. We can think of design as whether or not we're designing for the users in the same time and place or users at different times in different places. This course is an example of designing for different time and different place. You're watching this long after I recorded this, likely from far away from our studio. Work place chat utilities like slack and hipchat would be examples of same time, different place. They allow people to communicate instantly across space, mimicking the real-time

office experience. Now imagine a kiosk at a museum that asks visitors to enter their location to create a map of where everyone comes from. Now that would be different time, same place. Everyone uses the interface in the same place, but across time. And even when we're in the same time and place, computers can still support cooperation. In fact, right now, Amanda's running our camera, Ben's running the teleprompter and I'm standing up here talking at you. These different computers are supporting us in cooperating to create this course. So we can often think of CSCW as mediating cooperation across traditional geographic or temporal borders. But it can also help us with collocated simultaneous cooperation.

CSCW Resources

Recommended Academic Resources:

- Schmidt, K., & Bannon, L. (1992). [Taking CSCW seriously](#). *Computer Supported Cooperative Work (CSCW)*, 1(1-2), 7-40.
- Schmidt, K., & Simonee, C. (1996). [Coordination mechanisms: Towards a conceptual foundation of CSCW systems design](#). *Computer Supported Cooperative Work (CSCW)*, 5(2-3), 155-200.
- Ackerman, M. S. (2000). [The intellectual challenge of CSCW: the gap between social requirements and technical feasibility](#). *Human-Computer Interaction*, 15(2), 179-203.
- Bruckman, A. (1998). [Community support for constructionist learning](#). *Computer Supported Cooperative Work (CSCW)*, 7(1-2), 47-86.
- Bruckman, A., Karahalios, K., Kraut, R. E., Poole, E. S., Thomas, J. C., & Yardi, S. (2010). [Revisiting research ethics in the Facebook era: Challenges in emerging CSCW research](#). In *CSCW 2010: Proceedings of the 2010 Conference on Computer Supported Cooperative Work*.
- Luther, K., Fiesler, C., & Bruckman, A. (2013, February). [Redistributing leadership in online creative collaboration](#). In *Proceedings of the 2013 Conference on Computer-Supported Cooperative Work* (pp. 1007-1022). ACM.

Recommended Books:

- Lubich, H. (1995). [Towards a CSCW Framework for Scientific Cooperation in Europe](#). Springer Science & Business Media.
- Diaper, D. & Sanger, C. (Eds.). (2012). [CSCW in Practice: an Introduction and Case Studies](#).

Idea: Social Computing



Social computing is the portion of HCI that's interested in how computers affect the way we interact and socialize. One thing that falls under this umbrella is the idea of recreating social norms within computational systems. So for example, when you chat online, you might often use emojis or emoticons. Those are virtual recreations of some of the tacit interaction we have with each other on a day-to-day basis. So, for example, these all take on different meanings depending on the emotion provided. Social computing is interested in a lot more than just emojis, of course.



From online gaming and Wikipedia, to social media, to dating websites, social computing is really interested in all areas where computing intersects with our social lives.

Social Computing Resources

Recommended Academic Resources:

- Wang, F. Y., Carley, K. M., Zeng, D., & Mao, W. (2007). [Social computing: From social informatics to social intelligence](#). *IEEE Intelligent Systems*, 22(2), 79-83.
- Parameswaran, M., & Whinston, A. B. (2007). [Research issues in social computing](#). *Journal of the Association for Information Systems*, 8(6), 336.
- Wang, F. Y. (2007). [Toward a paradigm shift in social computing: the ACP approach](#). *IEEE Intelligent Systems*, 22(5), 65-67.
- Vassileva, J. (2012). [Motivating participation in social computing applications: a user modeling perspective](#). *User Modeling and User-Adapted Interaction*, 22(1-2), 177-201.
- Scekic, O., Truong, H. L., & Dustdar, S. (2013). [Incentives and rewarding in social computing](#). *Communications of the ACM*, 56(6), 72-82.
- Luther, K., Fiesler, C., & Bruckman, A. (2013, February). [Redistributing leadership in online creative collaboration](#). In *Proceedings of the 2013 Conference on Computer-Supported Cooperative Work* (pp. 1007-1022). ACM.

Recommended Books:

- Sabhasish, D. (2009). [Social Computing: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications](#). Idea Group.
- Zaphiris, P. & Ang, C. (Eds.) (2009). [Social Computing and Virtual Communities](#). CRC Press.

Dedicated Courses:

- [Social Computing](#), by University of California-San Diego via Coursera

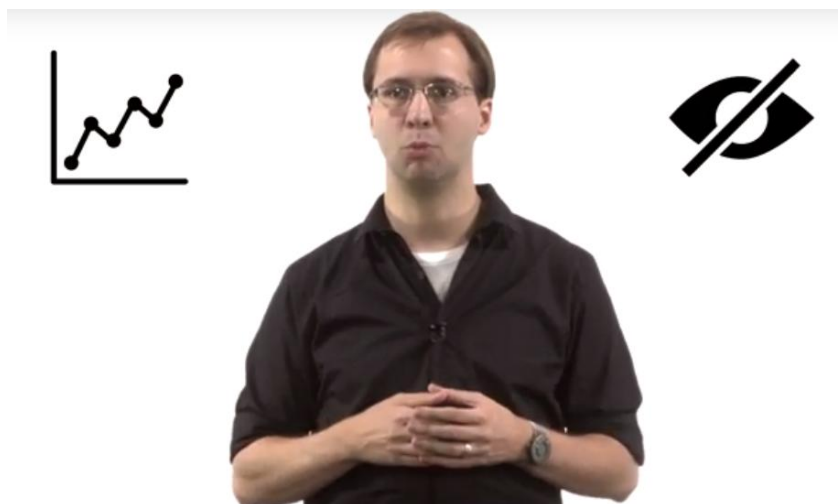
4.2 Applications: Domains

Compiled by Shipra De, Summer 2017

Domain: Special Needs



One of the most exciting application areas for HCI is in helping people with special needs. Computing can help us compensate for disabilities, injuries, aging. Think of a robotic prosthetic, for example. Of course, part of that is engineering, part of it is neuroscience. But it's also important to understand how the person intends to use such a limb in the tasks they need to perform. That's HCI intersecting with robotics.



Or take another example from some work done here at Georgia Tech by Bruce Walker, how do you communicate data to a blind person? We've talked about informational visualization, but if it's a

visualization, it's leaving out a significant portion of the population. So Dr. Walker's sonification lab works on communicating data using sound. A lot of the emerging areas of HCI technology could have extraordinary significance to people with special needs. Imagine virtual reality for people suffering from some form of paralysis. Or imagine using artificial intelligence with context-aware computing to create an autonomous wheelchair. These are projects that would only target a small portion of the population, but the impact of that portion would be absolutely indescribable.

Special Needs Resources

Recommended Academic Resources:

- Abascal, J., & Nicolle, C. (2005). [Moving towards inclusive design guidelines for socially and ethically aware HCI](#). *Interacting with Computers*, 17(5), 484-505.
- Bian, D., Wade, J., Warren, Z., & Sarkar, N. (2016, July). [Online Engagement Detection and Task Adaptation in a Virtual Reality Based Driving Simulator for Autism Intervention](#). In *International Conference on Universal Access in Human-Computer Interaction* (pp. 538-547). Springer International Publishing.
- Biswas, P. (2007). [Simulating HCI for special needs](#). *ACM SIGACCESS Accessibility and Computing*, (89), 7-10.
- Frauenberger, C., Good, J., & Keay-Bright, W. (2011). [Designing technology for children with special needs: bridging perspectives through participatory design](#). *CoDesign*, 7(1), 1-28.

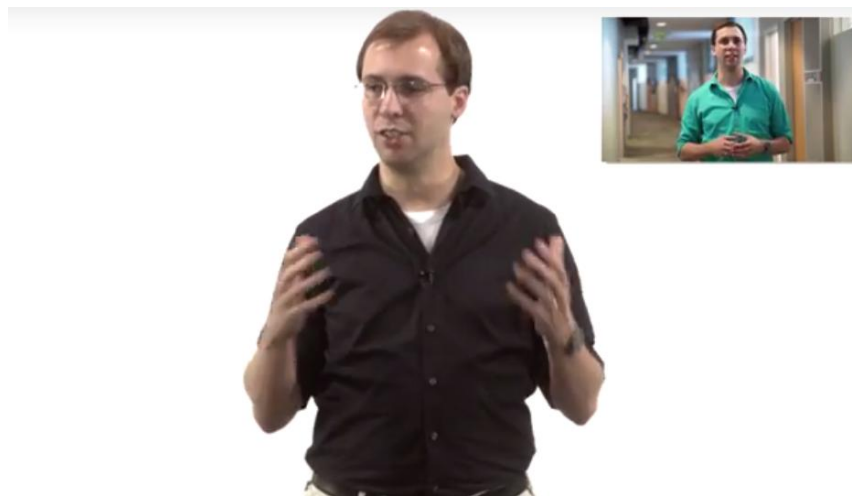
Recommended Books:

- Miesenberger, K., Fels, D., Archambault, D., Penaz, P., & Zagler, W. (Eds.). (2014). [Computers Helping People with Special Needs: 14th International Conference, ICCHP 2014, Paris, France, July 9-11, 2014, Proceedings](#). Springer.
- Antona, M. & Stephanidis, C. (2016). [Universal Access in Human-Computer Interaction. Methods, Techniques, and Best Practices: 10th International Conference, UAHCI 2016, Held as Part of HCI International 2016, Toronto, ON, Canada, July 17-22, 2016, Proceedings](#). Springer.

Domain: Education



Hi, and welcome to educational technology. My name is David Joyner and I'm thrilled to bring you this course.



As you might guess, education is one of my favorite application areas of HCI. In fact, as I'm recording this, I've been teaching educational technology at Georgia Tech for about a year, and a huge portion of designing educational technology is really just straightforward HCI. But education puts some unique twists on the HCI process. Most fascinatingly, education is an area where you might not always want to make things as easy as possible. You might use HCI to introduce some desirable difficulties, some learning experiences for students. But it's important to ensure that the cognitive loads students experience during a learning task is based on the material itself. Not based on trying to figure out our interfaces. The worst thing you can do in HCI for education is raise the student's cognitive load because they're too busy thinking about your interface instead of the subject matter itself. Lots of very noble

efforts in designing technology for education have failed due to poor HCI. So if you're interested in going into educational technology, you'll find a lot of valuable lessons in Human Computer Interaction.

Education Resources

Recommended Academic Resources:

- Rutten, N., van Joolingen, W. R., & van der Veen, J. T. (2012). [The learning effects of computer simulations in science education](#). *Computers & Education*, 58(1), 136-153.
- Tondeur, J., van Braak, J., Sang, G., Voogt, J., Fisser, P., & Ottenbreit-Leftwich, A. (2012). [Preparing pre-service teachers to integrate technology in education: A synthesis of qualitative evidence](#). *Computers & Education*, 59(1), 134-144.
- Lee, K. (2012). [Augmented reality in education and training](#). *TechTrends*, 56(2), 13-21.
- Zhu, E., Hadadgar, A., Masiello, I., & Zary, N. (2014). [Augmented reality in healthcare education: an integrative review](#). *PeerJ*, 2, e469.
- Wu, H. K., Lee, S. W. Y., Chang, H. Y., & Liang, J. C. (2013). [Current status, opportunities and challenges of augmented reality in education](#). *Computers & Education*, 62, 41-49.
- Merchant, Z., Goetz, E. T., Cifuentes, L., Keeney-Kennicutt, W., & Davis, T. J. (2014). [Effectiveness of virtual reality-based instruction on students' learning outcomes in K-12 and higher education: A meta-analysis](#). *Computers & Education*, 70, 29-40.

Recommended Books:

- Tsai, C., Heller, R., Nussbaum, M., & Twining, P. (Eds.). [Computers & Education](#). Elsevier.
- Maddux, C. & Johnson, D. (2013). [Technology in Education: A Twenty-Year Retrospective](#). Routledge.

Dedicated Courses:

- [Educational Technology](#), from Georgia Tech via Udacity
- [EdTechX](#), from MIT via edX
- [Integrating Technology in the K-12 Classroom](#), from the New Teacher Center via Coursera

Domain: Healthcare



A lot of current efforts in healthcare are about processing the massive quantities of data that are recorded everyday. But in order to make that data useful, it has to connect to real people at some point. Maybe it's equipping doctors with tools to more easily visually evaluate and compare different diagnoses. Maybe it's giving patients the tools necessary to monitor their own health and treatment options. Maybe that's information visualization so patients can understand how certain decisions affect their well-being. Maybe it's context aware computing that can detect when patients are about to do something they probably shouldn't do. There are also numerous applications of HCI to personal health like Fitbit for exercise monitoring or MyFitnessPal for tracking your diet. Those interfaces succeed if they're easily usable for users. Ideally, they'd be almost invisible. But perhaps the most fascinating upcoming intersection of HCI and health care is in virtual reality. Virtual reality exercise programs are already pretty common to make living an active lifestyle more fun, but what about virtual reality for therapy? That's actually already happening. We can use virtual reality to help people confront fears and anxieties in a safe, but highly authentic place. Healthcare in general is concerned with the health of humans. And computers are pretty commonly used in modern healthcare. So the applications of human computer interaction to healthcare are really huge.

Healthcare Resources

Recommended Academic Resources:

- Alpay, L., Toussaint, P., & Zwetsloot-Schonk, B. (2004, June). [Supporting healthcare communication enabled by information and communication technology: Can HCI and related cognitive aspects help?](#) In *Proceedings of the Conference on Dutch Directions in HCI* (p. 12). ACM.
- Riche, Y., & Mackay, W. (2005, September). [PeerCare: Challenging the monitoring approach to care for the elderly. In HCI and the older population.](#) In *Proceedings of the 19th British HCI Group Annual Conference*.
- Zhu, E., Hadadgar, A., Masiello, I., & Zary, N. (2014). [Augmented reality in healthcare education: an integrative review.](#) *PeerJ*, 2, e469.
- Ash, J. S., Berg, M., & Coiera, E. (2004). [Some unintended consequences of information technology in health care: the nature of patient care information system-related errors.](#) *Journal of the American Medical Informatics Association*, 11(2), 104-112.
- Harrison, M. I., Koppel, R., & Bar-Lev, S. (2007). [Unintended consequences of information technologies in health care—an interactive sociotechnical analysis.](#) *Journal of the American medical informatics Association*, 14(5), 542-549.

Recommended Books:

- Patel, V., Kannampallil, T., & Kaufman, D. (2015). [Cognitive Informatics for Biomedicine: Human Computer Interaction in Healthcare.](#) Springer.
- Ma, M., Jain, L., & Anderson, P. (2014). [Virtual, Augmented Reality and Serious Games for Healthcare.](#) Springer Science & Business.

Dedicated Courses:

- [Health Informatics on FHIR](#), from Georgia Tech via Coursera
- [Innovating in Health Care](#), from Harvard via edX

Domain Security

Classes on network security are often most concerned with the algorithms and encryption methods that must be safeguarded to ensure secure communications. But the most secure communication strategies in the world are weakened if people just refuse to use them. And historically, we've found people have very little patience for instances where security measures get in the way of them doing their tasks. For security to be useful it has to be usable. If it isn't usable, people just won't use it. XEI can increase the usability of security in a number of ways. For one, it can make those actions simply easier to perform. CAPTCHAs are forms that are meant to ensure users are humans. And they used to involve recognizing letters in complex images, but now they're often as simple as a check-box. The computer recognizes human-like mouse movements and uses that to evaluate whether the user is a human. That makes it much less frustrating to participate in that security activity. But HCI can also make security more usable by visualizing and communicating the need. Many people get frustrated when systems require passwords that meet certain standards or complexity, but that's because it seems arbitrary. If the system instead expresses to the user the rationale behind the requirement, the requirement can be much less frustrating. I've even seen a password form that treats password selection like a game where you're ranked against others for how difficult your password would be to guess. That's a way to incentivize strong password selection making security more usable.

Security Resources

Recommended Academic Resources:

- Zurko, M. E. (2005, December). [User-centered security: Stepping up to the grand challenge](#). In *21st Annual Computer Security Applications Conference (ACSAC'05)*. IEEE.
- Smith, S. W. (2003). [Humans in the loop: Human-computer interaction and security](#). *IEEE Security & privacy*, 1(3), 75-79.
- Patrick, A. S., Long, A. C., & Flinn, S. (2003, April). [HCI and security systems](#). In *CHI'03 Extended Abstracts on Human Factors in Computing Systems* (pp. 1056-1057). ACM.
- Braz, C., & Robert, J. M. (2006, April). [Security and usability: the case of the user authentication methods](#). In *Proceedings of the 18th Conference on l'Interaction Homme-Machine* (pp. 199-203). ACM.
- Sasse, M. A., Brostoff, S., & Weirich, D. (2001). [Transforming the 'weakest link'—a human/computer interaction approach to usable and effective security](#). *BT Technology Journal*, 19(3), 122-131.
- Parveen, N., Beg, R., & Khan, M. H. (2014). [Integrating security and usability at requirement specification process](#). *International Journal of Computer Trends and Technology*, 10(5), 236-240.

Recommended Books:

- Garfinkel, S. & Lipford, H. (2014). [Usable Security: History, Themes, and Challenges](#) Morgan & Claypool Publishers.
- Tryfonas, T. & Askoxylakis, I. (Eds.). (2014). [Human Aspects of Information Security, Privacy, and Trust](#). Springer.

Dedicated Courses:

- [Usable Security](#), from the University of Maryland via Coursera

Domain: Games

Video games are one of the purest examples of HCI. They're actually a great place to study HCI, because so many of the topics we discuss are so salient. For example, we discussed the need for logical mapping between actions and effects. A good game exemplifies that. The actions that the user takes with the controller should feel like they're actually interacting within the game world. We discussed the power of feedback cycles. Video games are near constant feedback cycles as the user performs actions, evaluates the results and adjust accordingly. In fact, if you read through video game reviews you'll find that many of the criticisms are actually criticisms of bad HCI. The controls are tough to use, it's hard to figure out what happened. The penalty for failure is too low or too high. All of these are examples of poor interface design. In gaming though there's such a tight connection between the task and the interface. Their frustrations with a task can help us quickly identify problems with the interface.

Games Resources

Recommended Academic Resources:

- Jørgensen, A. H. (2004, October). [Marrying HCI/Usability and computer games: a preliminary look](#). In *Proceedings of the Third Nordic Conference on Human-Computer Interaction* (pp. 393-396). ACM.
- Lv, Z., Halawani, A., Feng, S., Ur Réhman, S., & Li, H. (2015). [Touch-less interactive augmented reality game on vision-based wearable device](#). *Personal and Ubiquitous Computing*, 19(3-4), 551-567.
- Nijholt, A., Tan, D., Allison, B., del R Milan, J., & Graimann, B. (2008, April). [Brain-computer interfaces for HCI and games](#). In *CHI'08 Extended Abstracts on Human Factors in Computing Systems* (pp. 3925-3928). ACM.
- Zaphiris, P., & Ang, C. S. (2007). [HCI issues in computer games](#). *Interacting with Computers*, 19(2), 135-139.

Recommended Books:

- Bogost, I. (2007). [Persuasive Games: The Expressive Power of Videogames](#). MIT Press.
- Bogost, I. (2011). [How to Do Things With Videogames](#). University of Minnesota Press.

Dedicated Courses:

- [Introduction to Game Design](#), from MIT via edX
- [Game Design and Development Specialization](#), from Michigan State via Coursera

5.1 Course Recap

Compiled by Shipra De, Summer 2017

Introduction



We've reached the end of our HCI course. To close things off, let's briefly recap what we've covered. The purpose of this lesson isn't just to give you an inventory of the course content. The real reason is to give us an excuse to have this cool inception effect over here.



No, really it's to repeat it again to load it into your working memory one more time. After all, we know that the more often you load some content into short-term memory, the more strongly it remains in

long-term memory. That was a principle we learned in the lesson on human abilities, and it covers why we start and end every lesson by repeating the material that will be covered. Each repeat loads it into your short-term memory one more time, further solidifying it. So to close this course, we'll do this one more time, although we hope that you'll come back again and again to watch some of the material when you need a refresher.

Recap: HCI Principles



Our first unit was the unit on design principles. To start that unit off, we began by investigating the process of identifying a task. We discussed how a task is not just the actions that a user performs, but it's the combination of their motivations, their goals, the context, and so on. We emphasized that we're not just interface designers, we're task designers. We design tasks, and then design the interfaces that make those tasks possible. We then explored three views on the user's role in a task. We might view them as an information processor, like another computer in the system. We might view them as a predictor, someone operating a mental model of the system. We might view them as a participant, someone working in a larger context beyond just our interface. Finally, we discussed how the views we take inform the designs we create.

Recap: Feedback Cycle



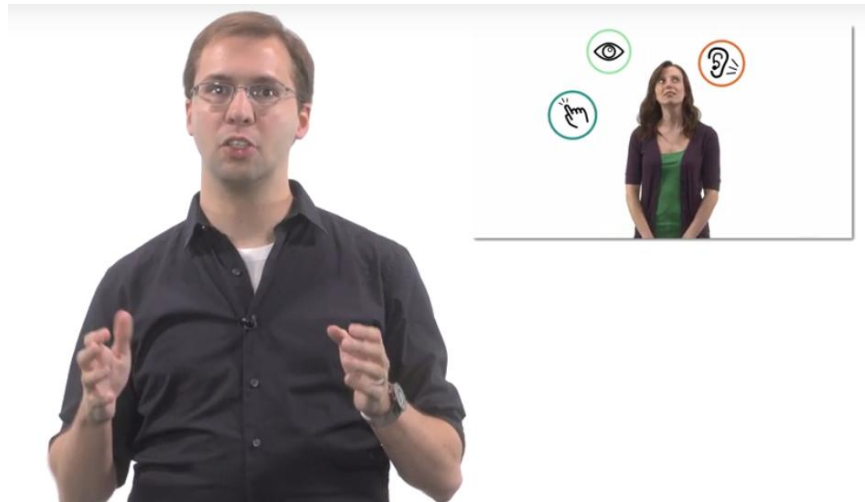
We started the first unit by discussing feedback cycles. Feedback cycles, as we discussed, are ubiquitous in all areas of life. They're how we learn and adapt to our environments, and so they're also how we learn and adapt to the interfaces available to us. We then described the two parts of feedback cycles. Gulfs of execution covered how users go from personal goals to external actions. Gulfs of evaluation covered how the users then evaluated whether the results of those actions met their goals. We can describe basically all of HCI as designing ways to bridge these two gulfs: helping users accomplish their goals more easily, and helping users understand that their goals have been accomplished more quickly.

Recap: Direct Manipulation and Invisible Interfaces



We then moved on to direct manipulation, which was one way to create interfaces with very short gulfs of execution and evaluation. Direct manipulation involved creating interfaces where the user felt like they were directly interacting with the object of the task. Instead of typing commands or selecting operators, they would more physically participate with the interface. The goal of this, and really of any good interface design, was to create interfaces that become invisible. Invisible interfaces are those that allow the user to focus completely on the task instead of on the interface. We noted that nearly any interface can become invisible when the user has enough practice and expertise, but our goal is to create interfaces that vanish sooner by good design.

Recap: Human Abilities



We're discussing human-computer interaction, and that means we have to understand the human portion of the equation. So, we also took a crash course toward some basic psychology. We broke the human down into three systems: perception, cognition, and the motor system. With perception, we covered the strengths and limitations of the human visual, auditory, and kinesthetic senses. We discussed how each can be useful for different kinds of information. Then, we discussed some of the limitations to human cognition. We focused on memory: how many things can be stored at a time, and how things can be stored more permanently. We also discussed the notion of cognitive load. We focused especially on how we should use our interfaces to reduce the user's cognitive load. Finally, we discussed the limitations of the human motor system, especially how those limitations change with age or in the presence of distractions. We're designing for humans, so these limitations and advantages are key to how we design our interfaces.

Recap: Design Principles and Heuristics



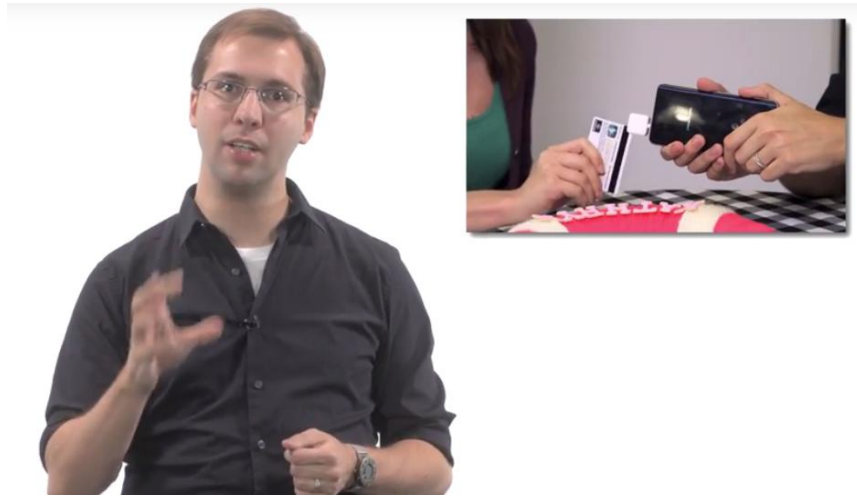
Human-computer interaction has a long and rich history, initially drawing from human factors engineering before becoming a field of its own. During that history, it's developed lots of principles and heuristics for how to design good interfaces. Lots and lots and lots, in fact. Literally thousands. In this lesson, we covered fifteen of the most significant ones, drawn from four sets of design principles. We covered the principles of Don Norman, Jakob Nielsen, Larry Constantine, Lucy Lockwood, and the Centre for Universal Design. Among these, I would argue the most significant principles were Affordances, Mappings, and Constraints. Affordances are parts of interfaces that, by their design, tell the user what to do. A good mapping tells the user what the effect of that interaction will actually be. Constraints ensure that the user only chooses to do the correct things. With those three combined, as well as the other heuristics, we can create interfaces that vanish between the user and the task very quickly.

Recap: Mental Models and Representations



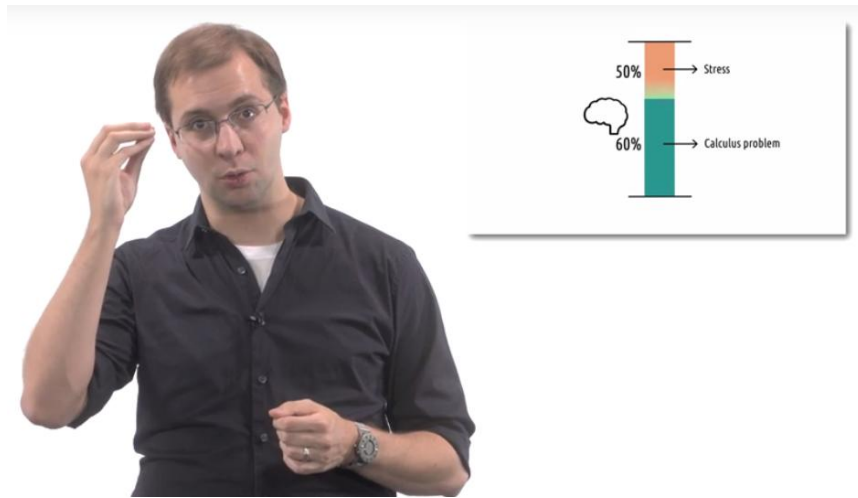
Every one of our users has some mental understanding of their task, as well as where our interfaces fit in that task. We call that their mental model. They use that mental model to simulate and predict the effects of certain actions. Our goal is for the user's mental model to match the reality of the task and the interface. To accomplish that, we try to design representations with clear mappings to the underlying task. That's how we can ensure the user's mental model of the system is accurate and useful. We discussed mistakes, which are errors that occur based on inaccurate mental models, and we discussed slips, which are errors that occur despite accurate mental models. We also talked about a couple of the challenges that can arise in trying to help users build accurate mental models. One of those was called expert blind spot, which occurs when we lose sight of our own expertise and forget what it's like to be a novice. And the other was learned helplessness, which is when users learn that they have no real ability to accomplish their goals because of a broken feedback cycle. In designing representations that lead to accurate mental models, we need to make sure to avoid both of these.

Recap: Task Analysis



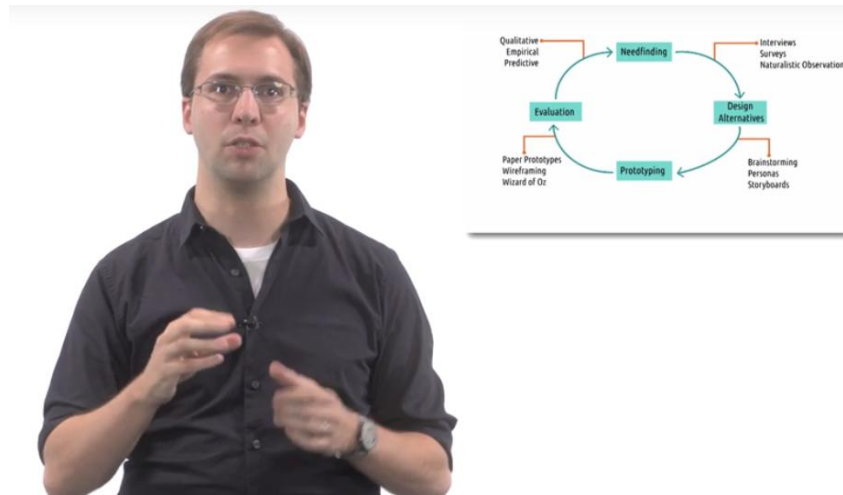
We've discussed repeatedly that HCI is in large part about understanding tasks. As designers, we design tasks that feature interfaces, not just interfaces alone. To accomplish that, it's important to have a very clear understanding of the task for which we're designing. Toward that end, we discussed task analysis. Task analyses are ways of breaking down tasks into formal workflows to aid the design process. We covered two general kinds of task analyses. Information processor models, like the GOMS model, focus on the user's goals, operators, and methods. They're concerned primarily with what we can observe. Cognitive task analyses, on the other hand, try to get inside the user's head and understand the thought process in the task as well. Both of these approaches are valuable to designing good tasks with usable interfaces.

Recap: Distributed Cognition



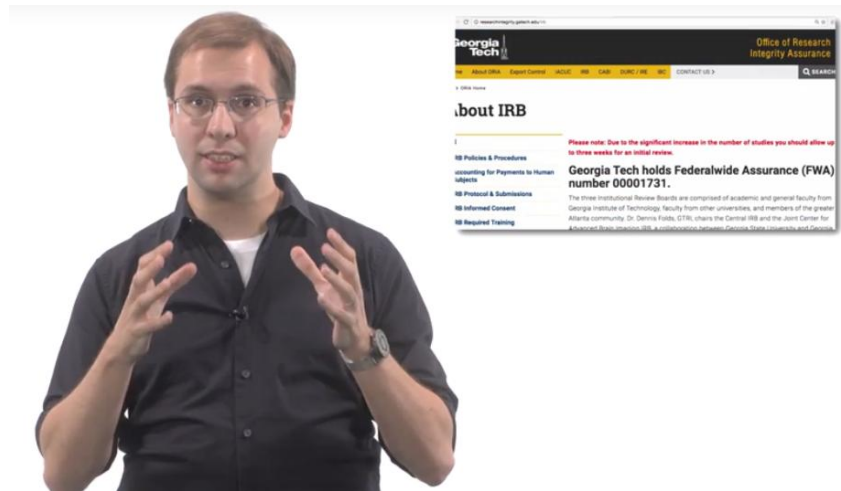
Earlier, we discussed the idea of cognitive load. Cognitive load was the principle that humans have a set amount of cognitive resources, and if they're overloaded, their performance suffers and they get frustrated. So, how do we reduce the user's cognitive load? Well we can make the task easier, sure, but we can also add to their cognitive resources. That's the principle of distributed cognition: the interactions of humans and artifacts together have more cognitive resources than individuals. Devices and interfaces can exhibit cognitive properties like memory and reasoning, offloading those demands from the user. We also discussed three related theories: social cognition, situated action, and activity theory. All three of these put a strong emphasis on the context of a task, whether it be the physical context, social context, or societal context.

Recap: HCI Methods



The design principles unit of the course covered the fundamental principles and ideas developed over decades of work in this space. However, we can't create good interfaces just by applying old principles to new problems. Those old principles can help us make progress much faster, but to design good interfaces, we have to involve the user. That's perhaps the most important principle of HCI: user-centered design. User-centered design advocated keeping the user at the heart of all our design activities. For us, that isn't just the person using the tool, but it's also the people affected by the tool's very existence. To keep the user in mind, we use an iterative design life cycle that focuses on getting feedback from the user early and often. The methods of that life cycle were the core of the methods unit of the course.

Recap: Ethics and Human Research



When we're doing research in HCI, we have access to some pretty sensitive personal data about our participants. There are huge ethical considerations around privacy and coercion that we have to keep in mind when participating in the design life cycle. So, we discussed the role of institutional review board, or IRB, for university research. They oversee studies and make sure we're preserving our participants' rights. They also make sure that the benefits of our research outweigh the risks, and as part of that, they help ensure our methods are sound enough to have benefits in the first place. We also discussed how industry doesn't have the same kind of oversight. However, some companies have partnered with universities to participate with their IRBs, while others have formed internal IRBs. All of this is driven by the need to preserve the rights of our users. That's a key part of user-centered design.

Recap: Needfinding and Requirements Gathering



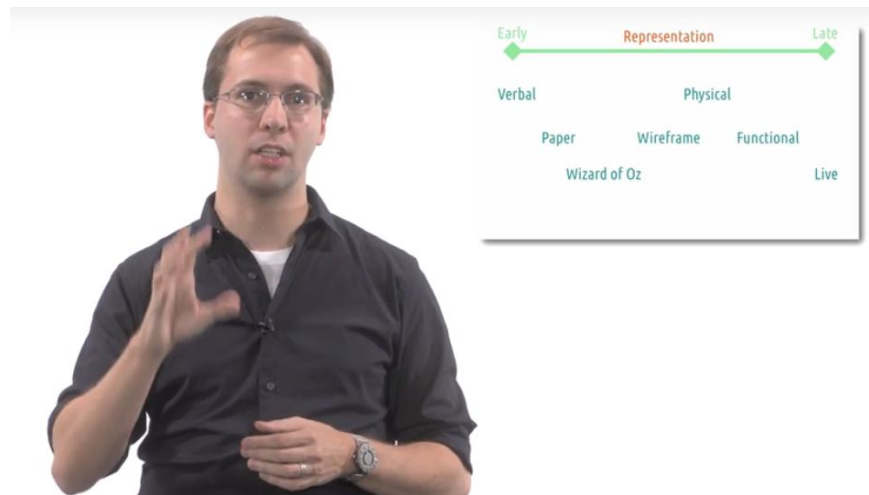
The first stage of the design life cycle was needfinding. Needfinding was how we developed a keen understanding of the needs of our users. One of the biggest mistakes a designer can make is assuming they already understand the user and the task before ever interacting with them. There are several questions about the user we need to answer before we're ready to start designing. So, to get a good understanding of the user and the task, we discussed several methods. We might start with methods that have little direct interaction with users, like watching them in the wild or trying out the task ourselves. We might use those to inform more targeted needfinding exercises, like interviews, focus groups, and surveys. By combining multiple needfinding approaches, we can build a strong model of the user and the task that will help us design usable interfaces.

Recap: Design Alternatives



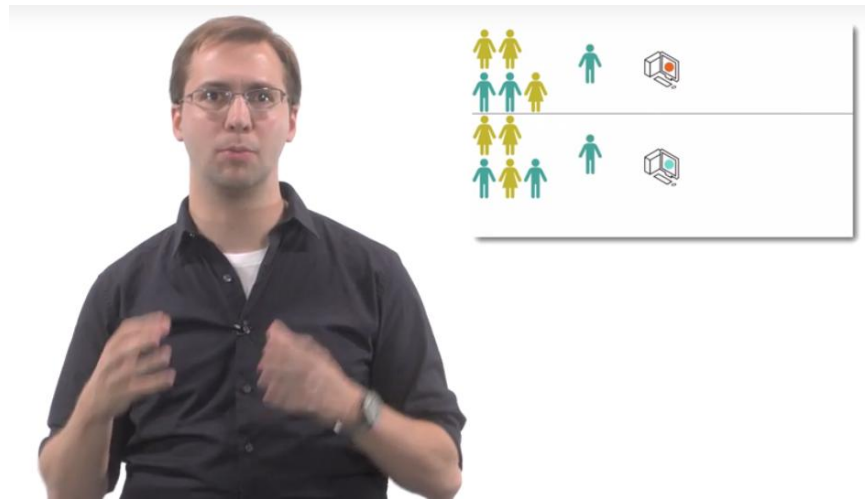
Once we have a solid understanding of the user and the task, we want to start brainstorming possible designs. The important thing here is to make sure we don't get fixated on one idea too early. That sort of tunnel vision risks missing lots of fantastic ideas. So, we want to make sure to engage in a well-defined brainstorming process. I recommend starting that with individual brainstorming, then setting up a group brainstorming session that ensures everyone's ideas are heard. From there, we proposed some different ideas on how to explore those design alternatives, through methods like personas, scenarios, and timelines. Our end goal here was to arrive at a set of designs worth moving on to the prototyping stage.

Recap: Prototyping



In user-centered design, our goal is to get user feedback early and often. So, once we have some design alternatives, our goal is to get them in front of users as quickly as possible. That's the prototyping stage, where we take those design alternatives and build prototypes we can show to users. At first, those prototypes might be very low-fidelity. We might start by just describing or drawing the designs. Those are verbal or paper prototypes. We want to keep our designs easy to revise. We might even revise the prototypes live while working with users. As we get more and more feedback, we build higher-fidelity prototypes to explore more detailed questions about our designs. We might use wireframes or set up live simulated demos. At every stage of the way, we design our prototypes in a way to get the user's view and inform the next iteration of the design life cycle.

Recap: Evaluation



The goal of the design life cycle is to get frequent user feedback. Or, to put it differently, to frequently evaluate our interface ideas. Frequent, rapid feedback cycles are important for users of our interfaces, and they're also important to us as designers of interfaces. That's where evaluation comes into play. Once we've designed an interface, whether it's just an idea in our head or a full-fledged working version, it's time to evaluate it with users. Early on, that may be qualitative evaluation to get the full picture of the user experience. Later, that might be empirical evaluation, to more formally capture the results of the interface. Along the way, we might also employ predictive evaluation to try to anticipate how users will react to our designs. These methods are the foundation of user-centered design: design that features frequent user evaluation of our ideas.

Recap: HCI and Agile Development



Traditional HCI can be a slow, deliberate process, fed by constant interaction with users as we slowly ramp up the fidelity of our prototypes. In the past, that was because every phase of the process was expensive: development, distribution, evaluation, and updates were all expensive. But now, in some contexts, those steps have become much, much cheaper. Some web development can be done with drag-and-drop interfaces. We can now distribute applications to millions of users essentially for free. We can pull back enormous quantities of live data from them. We can push updates to all of them in real time. Given all that, sometimes it might be more prudent to put together working versions quickly to start getting real user data faster. So, we also discussed agile methods for development while keeping an eye on the design life cycle. These methods aren't appropriate for all contexts, especially those with high costs of failure, but for many contexts they can really increase the iteration speed of our design life cycle.

Exploring HCI: Recap

Over the course of our conversations, I've asked you to revisit the area of HCI in which you're most interested during each topic. I've asked you to brainstorm how the various design principles and theories apply to the area you chose. I've asked you to think of a design life cycle that would support developing in that chosen application area. We've also given you lots of information to read about your chosen area. You have all the tools necessary to start developing. I'm looking forward to seeing what you come up with.

Conclusion

One of the famous models for communicating is to tell them what you're going to tell them, tell them, then tell them what you told them. That's what we've done, at several levels of abstraction. At the beginning of the course, we told you the overall structure of the course. Within each unit, we outlined the content of the unit. Within each lesson, we previewed the content of that particular lesson. Then we delivered the content. Then, we summarized each the lesson. Then, we summarized each unit. Now, we've summarized again the course as a whole. So, we're done, right? ...not quite. There are two other useful things to cover: the closely related fields to HCI, and where you might want to go next.

5.2 Related Fields

Compiled by Shipra De, Summer 2017

Introduction



At the beginning of our conversations, we talked about how HCI is part of a broader hierarchy of fields.

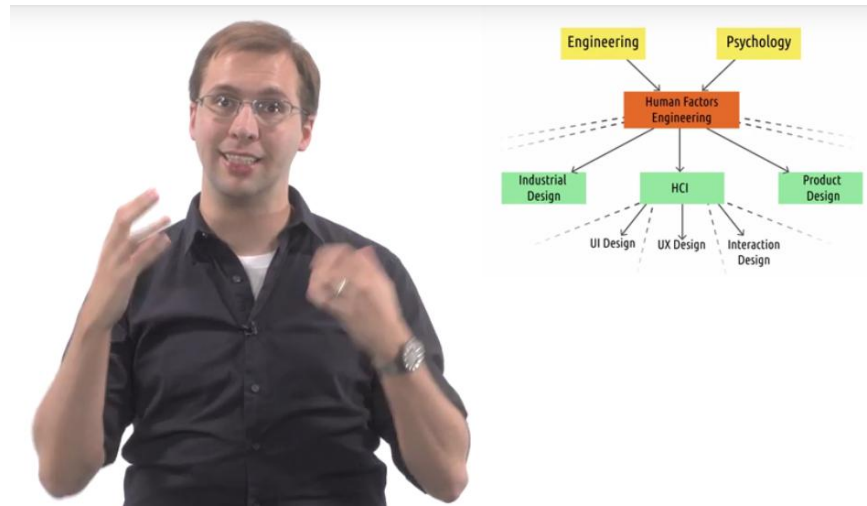


It draws a lot from Human Factors Engineering in many ways, and in fact it's Human Factors Engineering applied specifically to software.



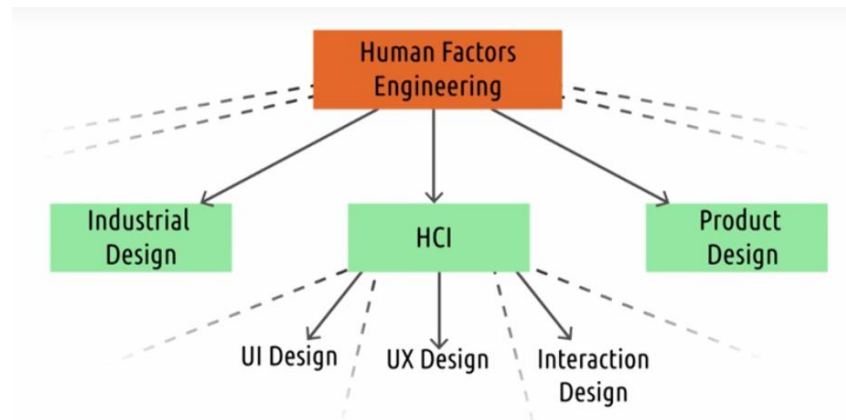
It also has numerous sub-fields, like User Interface Design, User Experience Design, Computer-Supported Cooperative Work, and more. In this lesson, we want to let you know where you might go next in your exploration of HCI in terms of subject matter. Note that these are different from the areas of application of HCI. When we talk about things like virtual reality and educational technology, we're describing fields to which HCI applies. Here, we're talking about HCI subfields themselves.

Human Factors and Industrial Design

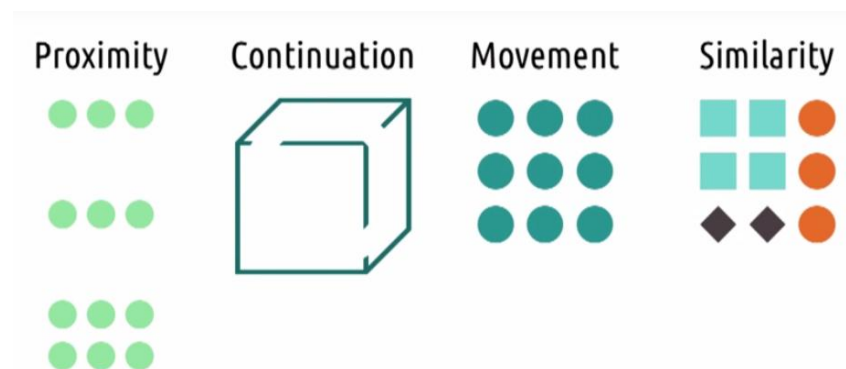


Human-computer interaction was concerned with the interaction between users and tasks as mediated by things equipped with computing power. Nowadays, that's more and more common. It's a relatively recent phenomenon that things like watches and cameras were examples of computers. When a device doesn't have computational power behind it, though, there are still design considerations to make. In fact, many of our principles and many of our methods apply just as well to non-computational interfaces. What makes human factors engineering particularly interesting is that it deals with more constraints, physical constraints. Things like the height of the user or the size of a hand come up in human factors engineering. What's interesting, though, is that as many devices start to have computational resources added to them, human factors engineering and human-computer interaction will start to interact more and more. My watch, for example, has no computational resources to it. It's completely within the human factors area. But smartwatches see some interesting interactions between human factors and HCI. Human factors determines the size of the watch, which determines the size of the battery or the additional sensors that can be placed inside. Those things then influence what we can do on the HCI side. So, if you're dealing with anything related to ubiquitous computing, wearable devices, contextual computing, human factors engineering is a great place to brush up. So, we'll throw you some resources in the notes below.

User Interface Design

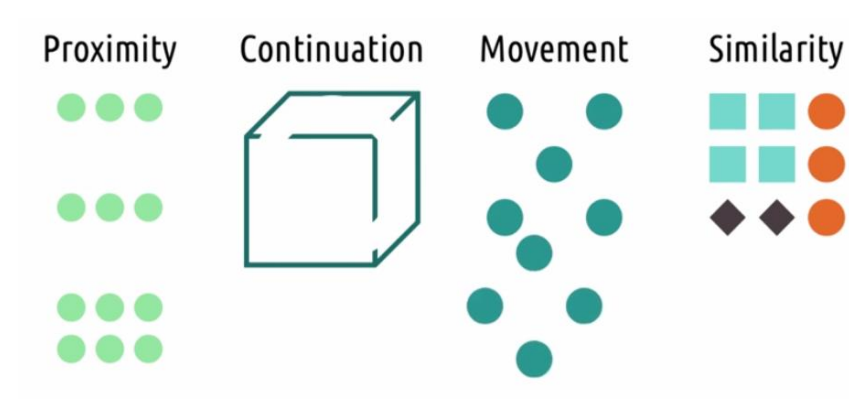


Likely the most significant subfield of HCI is user interface design. Colloquially, user interface design most often refers to design between a user and some rectangular on-screen interface via a computer, a laptop, a smartphone and so on. For a long time, user interface design and HCI were pretty much synonymous because the vast majority of interaction between users and computers happened via a rectangular on screen interface connected to a mouse and keyboard. It's relatively recent that we've started to interaction break out of that literal box. And to a certain extent, the term user interface design captures this as well. Interface doesn't have to mean a screen. Colloquially, though, I find most classes on UI design focus on designing screens and interacting with screens. That's a massive and well-developed field, and in fact a lot of this course's material comes from the user interface design space. There are some more narrow things user interface design is concerned with, though. UI design has its own set of design principles that apply more narrowly to the design of traditional computer software or web sites.



Some of these principles guide how people visually group things together. These are called Gestalt grouping principles. When things are close together we mentally put them into groups. You likely see two groups of three and one group of six. When there are implicit lines we see a continuation. You likely

see this as a cube even though the lines are broken. We also group things together based on similarity. You probably see this as four blue squares, two gray diamonds, and three orange circles. And while right now you see this as a three by three grid of green circles, watch what happens when they move.



Even after they stop moving, you likely still see the diamond that was formed by the moving circles.

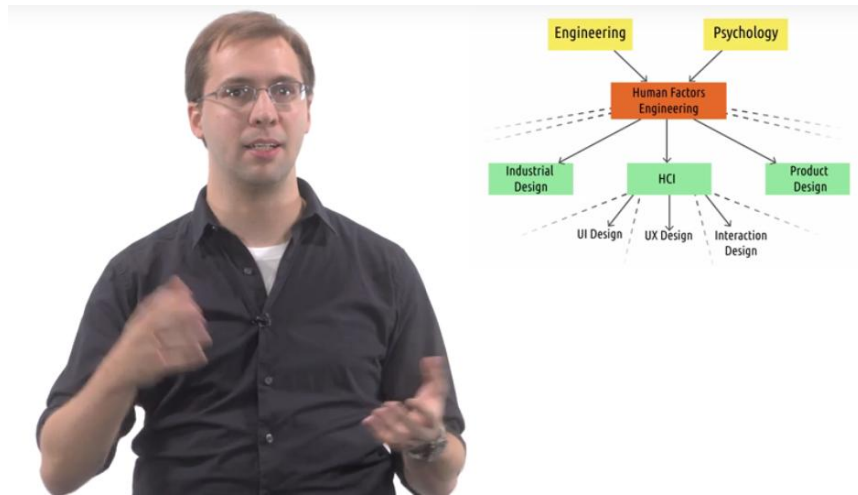


These Gestalt principles in some ways apply the user interface design's emphasis on designing with good grids in mind, just the way magazines and newspapers have done for centuries. We've already discussed this principle a lot and part of it is because this new interface leverages the analogy to the old interface but its value isn't just in the analogy. It's value is also in the way it instantiates the same Gestalt principles that guided the layout of a newspaper



And finally, user interface design touches on one of my favorite topics, typography. Typography is often covered in user interface design courses. So generally speaking, while in HCI we've taken special care to talk about general methods that deal with any kind of computer interface between users and tasks, user interface design zooms more closely in on the design of interfaces on screens. If you want to study UI design some more, we'll add some links to related courses and materials online to the notes below.

User Experience Design



For a lot of people, HCI and user experience design are essentially the same thing. For me, UX design is a little bit more prescriptive, while HCI is a little more descriptive. HCI describes how people learn and interpret interfaces, while UX design prescribes what you should do with that information. But in many ways, that's just applied HCI: the content is the same, the question is how you use it. If you choose to continue your studies into UX design, though, there are a few new things you'll encounter. You'll see an increased focus on the mood and joy of the user: we want them to have a great experience, not just accomplish a task. You'll see more attention paid to the user's motivations behind engaging with the interface. You'll also see a greater emphasis on the interaction between the user, the interface, and the context all around them, as these are all parts of the user experience. These are all things we've talked about, but user experience gets into prescribing these with more specificity. So, if you want some more information on user experience design, we'll put some resources in the notes below.

HCI & Psychology

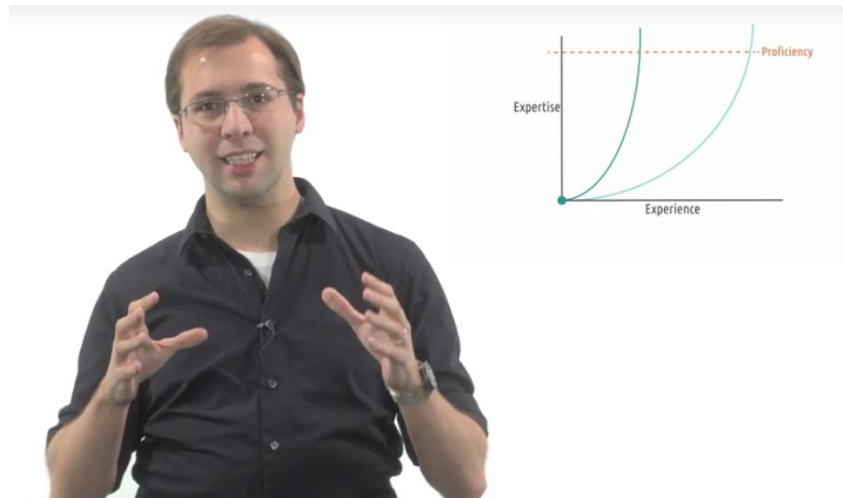


Early in our conversations, we described how HCI is generally about an interaction between design and research. We use research findings to inform our designs, and we use the success of our designs to create new research findings. As you explore more HCI, though, you'll find that there's plenty of room to specialize in one side or the other. Professional designers focus almost exclusively on the creation side, but there are lots of people who focus exclusively on the research side of HCI. Many universities like Georgia Tech and Carnegie Mellon have research faculty dedicated to understanding the way people interact with technology at individual, group, and societal levels. So if you're more interested in understanding how people interact with interfaces than in designing new ones, you might be interested in taking more of a research bent on HCI. This class is built from the research perspective more than the design perspective, so you've already got a great foundation. We'll add some links below if you want to explore some more.

Human-Centered Computing

HCI research broadens to a field called human-centered computing. While much of HCI research is concerned with the immediate interactions between people and computers, human-centered computing is interested more generally with how computers and humans affect each other at a societal level. There's a lot of fascinating research going on in this area. Some of the questions people are addressing are: How did participants in the Arab Spring use tools like Facebook and Twitter to coordinate? How can information visualizations be employed to help people better understand their diet or energy usage? How does access to computing resources influence early childhood education? Now notice that these issues don't necessarily involve designing new interfaces or creating new tools. They involve looking at the way people interact with computers more generally—and not just specific tools, but the ubiquity of technology as a whole. If you're interested more in this, we'll put some materials in the notes below.

Cognitive Science



When we described mental models, we were actually briefly touching on a deep body of literature from the cognitive science field. Cognitive science is the general study of human thought, mental organization, and memory.



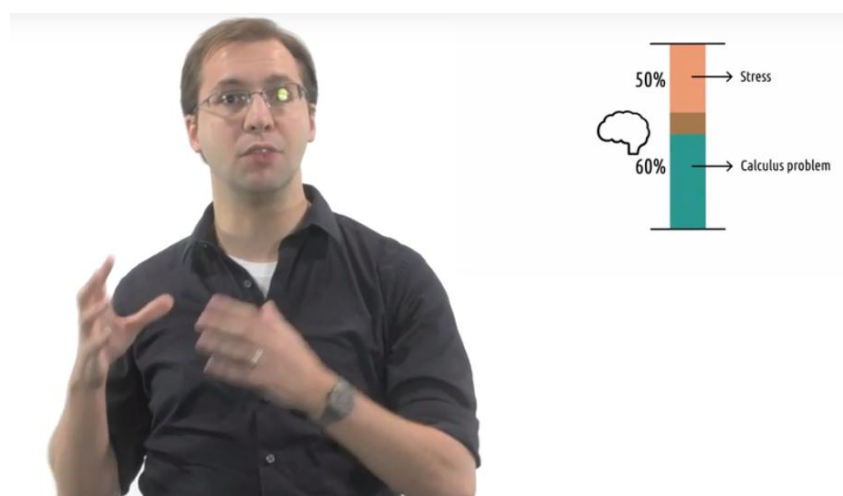
Now, cognitive science isn't a subfield of HCI, but HCI informs a good portion of cognitive science research.



That's because HCI gives us a way to explore aspects of human cognition.



We can design interfaces that assume humans think and process in a certain way.



And we can use the results of those interfaces to further develop our theories. In this way, HCI is a probe into human thought that informs the development of cognitive science, and cognitive science in turn gives us theories on human abilities that inform the interfaces that we design. So if you're interested in studying people more closely using HCI as a probe or tool, we'll put some links to some worthwhile courses to explore.

Computer-Supported Collaboration

When we discussed feedback cycles, we mentioned that the user experience applies not only at the individual level, but also at the group level. Distributed cognition, too, was interested in how interactions between people can be mediated by interfaces, and how the output and knowledge of those interactions can't be attributed narrowly to one particular part of the system but rather to the system as a whole. The ubiquity of human interaction and the potential of computers to mediate interaction between people gives rise to fields that investigate collaboration across interfaces. These fields ask the question like: how can computers be used to allow people to collaborate across distance and time? And how can computers be used to enhance the collaboration of people working together at the same place and at the time? These fields look at how computers can support things like cooperative work and collaborative learning. For example, how does Wikipedia enable people across enormous variations in location and time to work together to capture knowledge? Or how do online courses allow teachers and students to interact and learn asynchronously across distances? Or how can computers be used to facilitate conversations between people with different backgrounds, expertises, or even languages? These are pretty well-developed fields, so if you're like to learn more, we'll put some more information in the notes below.

Intelligent User Interfaces

To close out, my work is at the intersection of artificial intelligence, human-computer interaction, and education. My research is largely on how to use AI to create valuable learning experiences. Setting aside the education part for a second, though, there is also a rich interaction between artificial intelligence and human computer interaction in the form of intelligent user interfaces. This field looks at how we can apply AI techniques to adapting user interfaces to their users. Now an infamous example of this is Clippy, the Microsoft office assistant: he tried to infer what you were working on and give you in-context feedback on it. Intelligent user interfaces have come a long way since then, though. Google Now, for example, is consistently trying to learn from your routine and give you information when you need it. One of my favorite experiences with intelligent user interfaces came from the interaction between Google Maps, Gmail, and Google Calendar. Google Calendar had automatically imported by a restaurant reservation I had made from Gmail, along with the location information. Then, Google Maps, knowing where I was, detected that there was unusual traffic between me and the reservation, and buzzed me to let me know when to leave to arrive on time. I hadn't checked traffic, but I was on time for my reservation because of the intelligence of that user interface. It knew what I needed to know and when I needed to know it. So if you'd like to hear more about the overlap between artificial intelligence and human-computer interaction, we'll put some information in the notes below.

Conclusion

Human-computer interaction is a massive field with lots of sub-domains. This course has been a combination of the fundamental methods and principles of HCI, but there are lots of directions to go from here. In this lesson, I've attempted to give you some idea of where you might look next.



You might be interested in the research side of HCI and exploring more about how technology influences the way people think and act.



You might be interested in the design side and creating excellent user interfaces and experiences.



You might be interested in collaboration and helping people interact across technology.

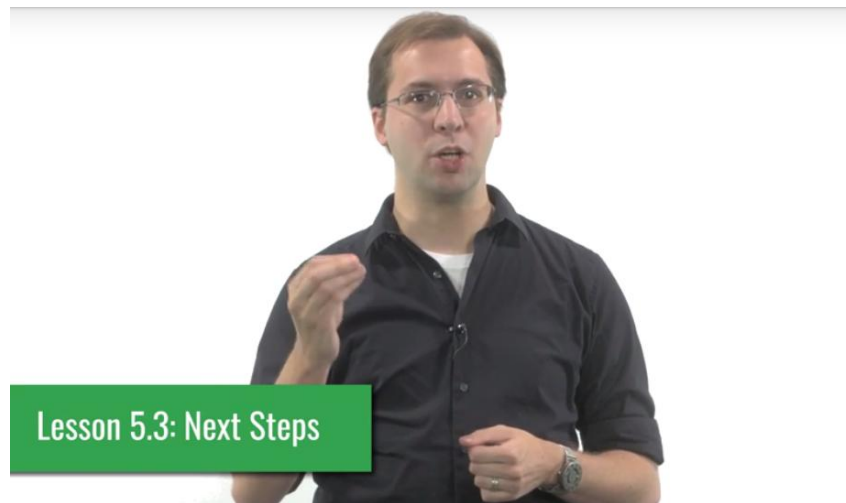


Or you might be interested in artificial intelligence and designing interfaces that adapt to their users' needs. Whatever your interest, there's a rich amount of content in HCI in front of you. The last thing we need to think about is: how to best get that content.

5.3 Next Steps

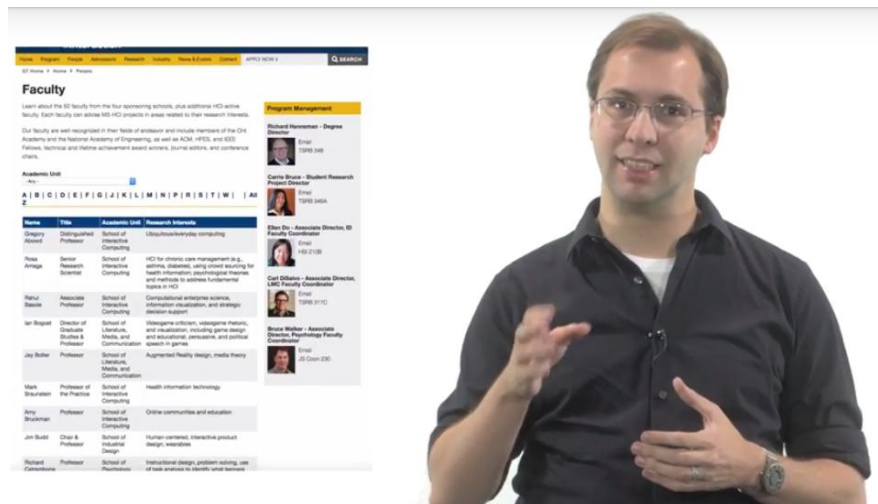
Compiled by Shipra De, Summer 2017

Introduction



To close our journey through HCI, let's take a look at where you might go from here. We've already talked about the different application areas of HCI, like virtual reality and educational technology, and those certainly apply to what you could do next. We've also talk elsewhere about the deeper topics in HCI you might investigate, like intelligent user interfaces or human-centered computing. But to close, let's talk about the formal educational steps you might take going forward to get deeper into HCI.

Ongoing Research



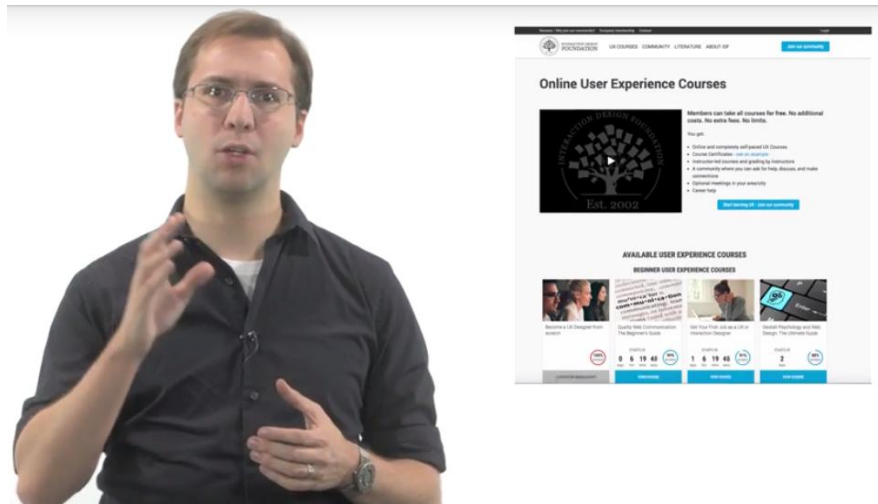
The quickest way to get involved in more HCI if you're at Georgia Tech is to see about joining a professor's research team. On the Georgia Tech HCI faculty listing, you'll find listings for every HCI faculty member along with their research interests. Find someone who's working on the same kinds of things that you're working on, and see if they'd like to let you join one of their ongoing projects. Professors are extremely busy people, but one of the things I love about Georgia Tech is the school's focus on fostering student education and involvement in addition to fostering quality research. So, it's quite likely you'll find someone willing to let you join up and prove that you can contribute. Let's check out a list of what kinds of projects are going on.



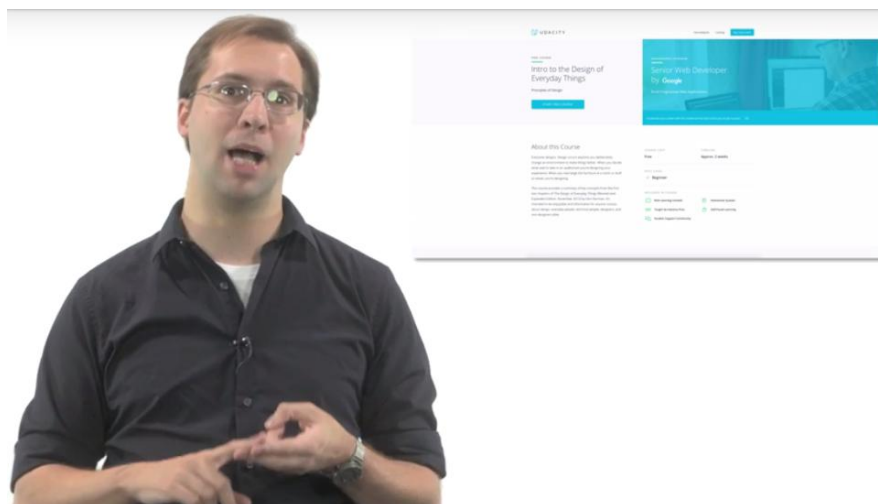
New research projects are coming up all the time, and new faculty are joining every year—that's why I'm not listing names or specific projects. But if any of these domains sound interesting to you, check out the HCI faculty web sites and see if there's anything to which you'd like to contribute.

MOOCs

HCI is a popular topic in emerging MOOCs as well. So if you're looking to continue your HCI education in a slightly more formal way but don't want to shell out the money for a formal degree, there are several great places you can start.



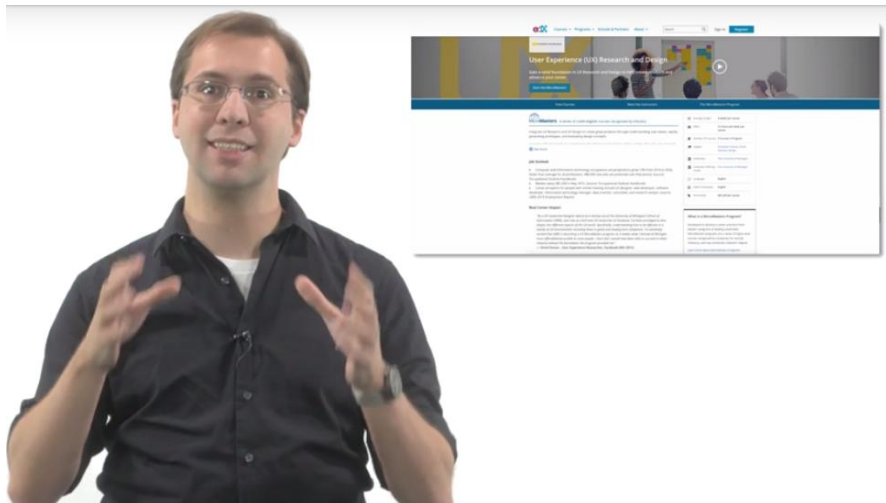
First, Interaction-Design.org is a treasure trove of HCI information. It has a fantastic free open access body of literature to study independently, including additional information on many topics we've covered in this course. The site also runs quite a few of its own closed courses as well.



For more traditional MOOCs, Udacity has free courses on HCI as it applies to product design and mobile app design, as well as a MOOC by Don Norman based on his famous book, Design of Everyday Things.



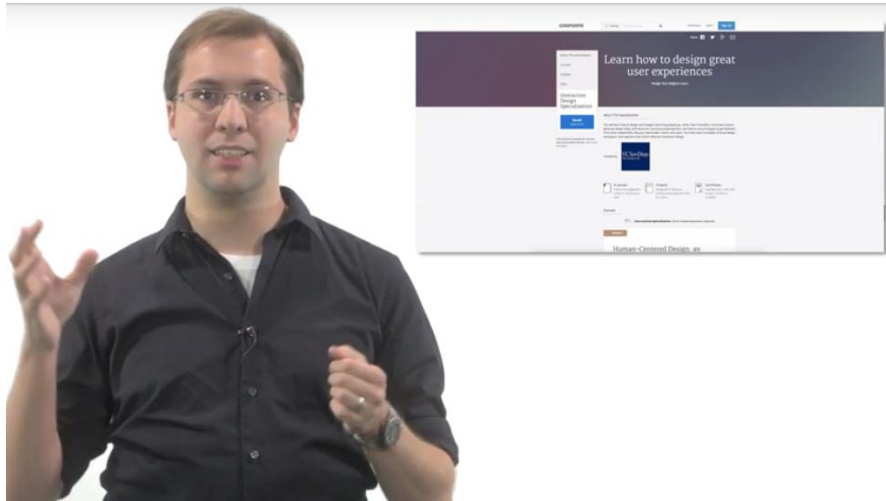
Over at edX.org, MIT has released a series of MOOCs targeting user experiences in mobile app development.



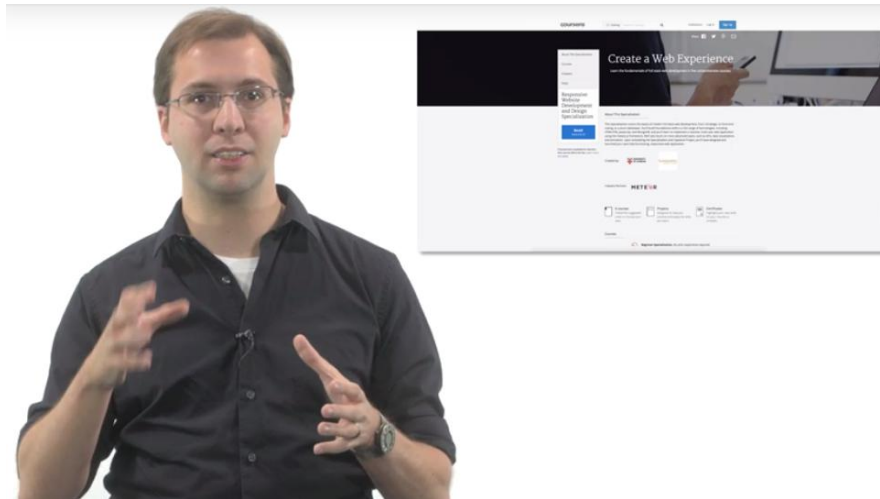
The University of Michigan has an Xseries on UX Research.



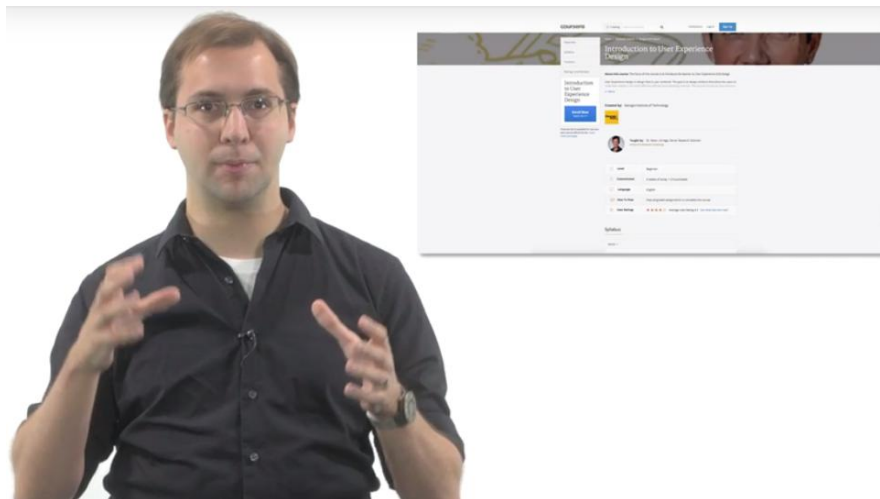
And Tsinghua University has a course on user experience design with a special emphasis on human factors and culture.



On Coursera, Scott Klemmer, one of the most prominent HCI researchers, has a specialization entitled Interaction Design. The University of Minnesota also has a specialization on UI design that covers a lot of the same topics that we've covered here, developed in part by another Georgia Tech alum, Lana Yarosh.



The University of London has a specialization on responsive web design.



Georgia Tech is planning a specialization on human computer interaction as well that might be live by the time you see this. And that's all just core HCI courses—all of these providers and more have courses on applied areas of HCI like video game design, educational technology, virtual reality, and more. Most of these courses are available for free to watch. Note also that new MOOCs are coming online all the time, so there are quite likely some that I haven't mentioned here. So, check out Udacity, check out edX, check out Coursera, check out FutureLearn. Also check out Class-Central.com for a list of MOOCs across several different platforms. Or, just Google HCI MOOC. This space is so new and fast-paced that by the time you view this video, half the MOOCs I've mentioned might be gone and twice as many new ones may have been created. We'll try to also keep an updated list of available courses in the notes below.

MSCS-HCI

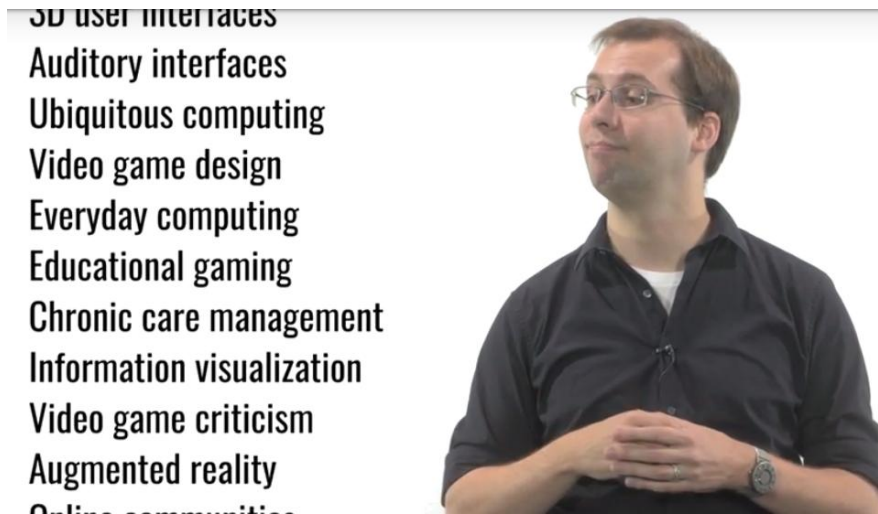
If you want to take it a step further, though, you might get an actual Master's in Computer Science specializing in HCI. If you're a Georgia Tech student watching this, you might already be working towards a Master's in CS, and you might be wondering if the HCI specialization is available online. Right now while I'm recording this, it's not, but I'll pause for a second so Amanda can let us know if that's changed. If you're an on-campus student watching this, or if you're watching just an open MOOC then you might want to look into an MS in CS with a focus on HCI. Most MSCS programs I've seen have an HCI specialization or at least an HCI focus. The specialization lets you go far deeper into the field, taking several classes on topics like core HCI, user interface design, educational technology, mixed reality design, information visualization, and more. We'll gather a list of schools with MSCS programs with HCI specializations and provide it in the notes below.

MS-HCI

If you already have a strong background in CS, you might want to go all the way to getting a Master's specifically in HCI. These programs aren't as common as MSCS programs with HCI specializations, but many universities do have them, including Georgia Tech, Carnegie Mellon, the University of Washington, the University of Maryland, Iowa State University, and Rochester Institute of Technology. I myself completed the MSHCI program here at Georgia Tech before starting my PhD. In focusing an entire Master's degree on HCI, you'll find you have even more time to spend getting into the relationship between HCI and other fields. At Georgia Tech, for example, the MS-HCI program has specializations in interactive computing, psychology, industrial design, and digital media. That allows the flexibility to focus on different areas of HCI, like how it integrates with physical devices in industrial design or how it helps us understand human cognition in psychology. Most Master's programs in HCI that I've seen are also heavily project-focused. Carnegie Mellon provides sponsored capstone projects from industry, and every student in Georgia Tech's MSHCI program completes a 6-credit hour independent project. So, if you're really set on moving forward with a career in HCI, a dedicated Master's in the field is a great way to move forward.

PhD-HCI

If research really is your calling, though, then you're going to want to move on to a PhD, which can have a specialization in HCI as well. A PhD isn't for everyone. There's a tendency to view it as the next logical step after a Master's degree, but a PhD program is far more of an apprenticeship program. At Georgia Tech at least, the PhD program actually requires fewer classes to complete than a Master's, but that's because 90% of your time is spent working on research closely with a faculty advisor. But if you're very interested in research, the PhD may very well be the way to go. As a reminder, here were some of the project areas that have ongoing research here in HCI at Georgia Tech.



A PhD program is a huge commitment—it's your full-time job for typically five years. It's absolutely not for everyone... but it absolutely is for some people. So, if you're really passionate about what you've learned here, a PhD focusing on HCI may be the way to go.

PhD-HCC

Finally, the highest level of educational achievement in HCI is likely a PhD specifically in HCI or similar fields. Here at Georgia Tech and at schools like Clemson and Maryland, that's a PhD in human-centered computing, which is actually what my PhD was in. Other schools have PhDs in similar fields—Carnegie Mellon, Iowa State, IUPUI, and others have a PhD programs in HCI as well. Pursuing a PhD in HCI or HCC lets you take a deep dive into how humans and computers interact in our modern world. You might dive deep into artificial intelligence and cognitive science, using AI agents to reflect on how humans think. You might delve into learning sciences and technology, studying the intersection between computers and education in depth. You might focus on social computing and how online communities function or how social media is changing our society. Or you might stick closer to the core of HCI and focus on how people make sense of new technologies. You might answer questions like, how do we give immediate feedback on motion controls? Or how do we adapt a user interface to the user's mood? There are enormous questions to answer, and I'm excited to see some of you move on to change the world in very profound ways.

Thank you, and good luck!



No matter what you do next, I hope you've enjoyed this foray into the world of human-computer interaction. If you end this course feeling like you actually know less than you started, then that's perfectly fine: my goal was not only to teach you about HCI, but also to help you understand how big this community is. I look forward to seeing y'all go further in the community and make a difference in the world. To close, I have to give a special thank you to Georgia Tech for creating the fantastic online Master's program in which I'm developing this course. And I'd also like to thank the HCI faculty at Georgia Tech for letting me be the one to record this and bring it to you. And most of all, I'd like to thank Amanda and Morgan, my partners in creating this course, for being totally responsible for how amazing it's looked. I like to think this isn't just a course about human-computer interaction, but it's also an example of human-computer interactions: humans using computers to teach about HCI in new and engaging ways. Thank you for watching.