L3: Interviews and Observations

EECS 330: Winter 2017

So today we're going to talk about interviews and observations, which are really important techniques for learning about your users that you'll be employing in P2, your second project assignment.

Announcements

- P1 is due today!
- If you don't have a project team yet, or you need more people, please see us after class.
- We will email studio assignments tomorrow
- **HW1 deadline extended to next Wednesday, Jan 18**
- P2 is due on Monday Jan 16

But first, a couple of announcements. P1 is due today. If you don't have a project team yet, or if you're still looking for more people, please come up and see us after class.

We're going to be working on your studio assignments right after class, so please get your Google form in as soon as possible if you haven't already, or we won't be able to prioritize your studio times. We'll be emailing your studio assignments out tomorrow.

We've decided to extend the deadline for HW1 until next Wednesday, Jan 18. Some of the TAs pointed out that they hadn't had time to cover all the CSS you need to know to complete the homework assignment last week, so we want to make sure you can cover that content and ask questions in studio this week. I'm sorry for this change, but hopefully this will help ease the pressure of HW1.

Also, a reminder that P2 is due on Monday, a week from today. We'll be talking about all the details later in lecture.

Today's Topics

- Needfinding
- Interviews
- Observations
- Overview of P2

Okay, so today we're going to be talking about the process of needfinding, or learning about your users needs, and then we're going to be going through two different methods of collecting information about your users, interviews and observations. Finally, we'll go over P2 and talk about how you can incorporate these principles of interviewing and observing in that assignment.

Today's Topics

- Needfinding
- Interviews
- Observations
- Overview of P2

What is Needfinding?

A qualitative research approach for studying people to identify their unmet needs.

Why focus on needs?

- Needs last longer than any specific solution
- Needs are opportunities waiting to be exploited
- Needs spur action

[For more, see http://www.jumpassociates.com/learning-posts/needfinding-uncovering-peoples-needs/]

Okay, so what is needfinding? It's a qualitative research approach for studying people to identify their unmet needs. So in other words, it's a method for studying your target user population to uncover needs that you can try to address through software.

So why this focus on needs? There are a couple of reasons why needs are important. First, needs last longer than any specific solution. So for example, technologies like punch cards, magnetic tape, floppy disks have successively been introduced and become obsolete. But the underlying need to store computer data has existed ever since the beginning of computers, and it continues to exist today, so we will always use some solution to address this need.

As designers, you can look at needs as opportunities waiting to be exploited. If your target user group has a set of needs, they are always going to be looking for solutions to address that need, and your software has great potential to be successful if it addresses a real user need.

Finally, needs spur action. If you can identify a user need through interviews and

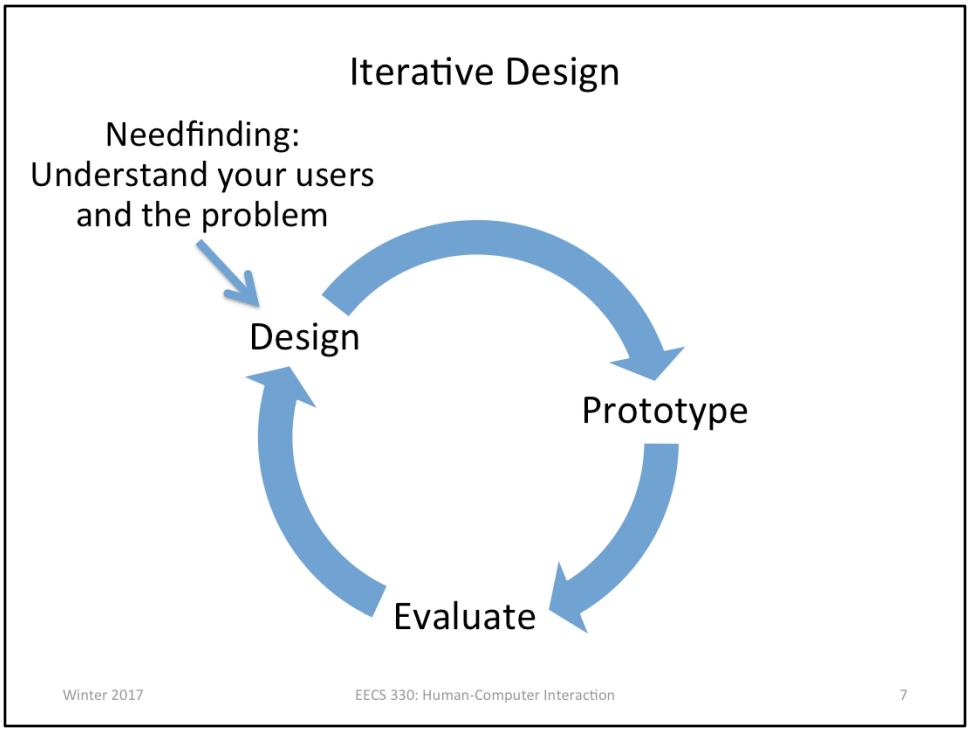
User-Centered Design

A design framework in which the needs, wants, and limitations of the end users of system are given extensive attention at each stage of the design process

Key Features:

- Iterative design
- **Early focus on users and tasks**
- Constant evaluation

So recall that in our last lecture, we talked about the user-centered design process. One of the key features of user centered design is an **early focus on users and tasks.** This is exactly where need finding fits in.



Recall the iterative design process, where we start out by imagining our software through design, then physically realizing our design through a prototype, and finally evaluating through user testing.

Needfinding is an optimization for the iterative design process. If you go through a needfinding process before you begin your initial design, you'll have a better understanding of the problem or need that your software solution should address, and that will help you make a better initial design. Without needfinding, you might not realize what your target users really need until you evaluate your first prototype, and discover that it isn't actually useful for your users. So in the long run, needfinding will help you reduce the number of iterations you need to make before developing a useful and usable software solution.

What if I already know what my users need?

Remember: **you are not the user!**

Seems obvious but…

- You have different experiences
- You have a different level of comfort with tech
- You use different terminology
- You have different ways of looking at the world

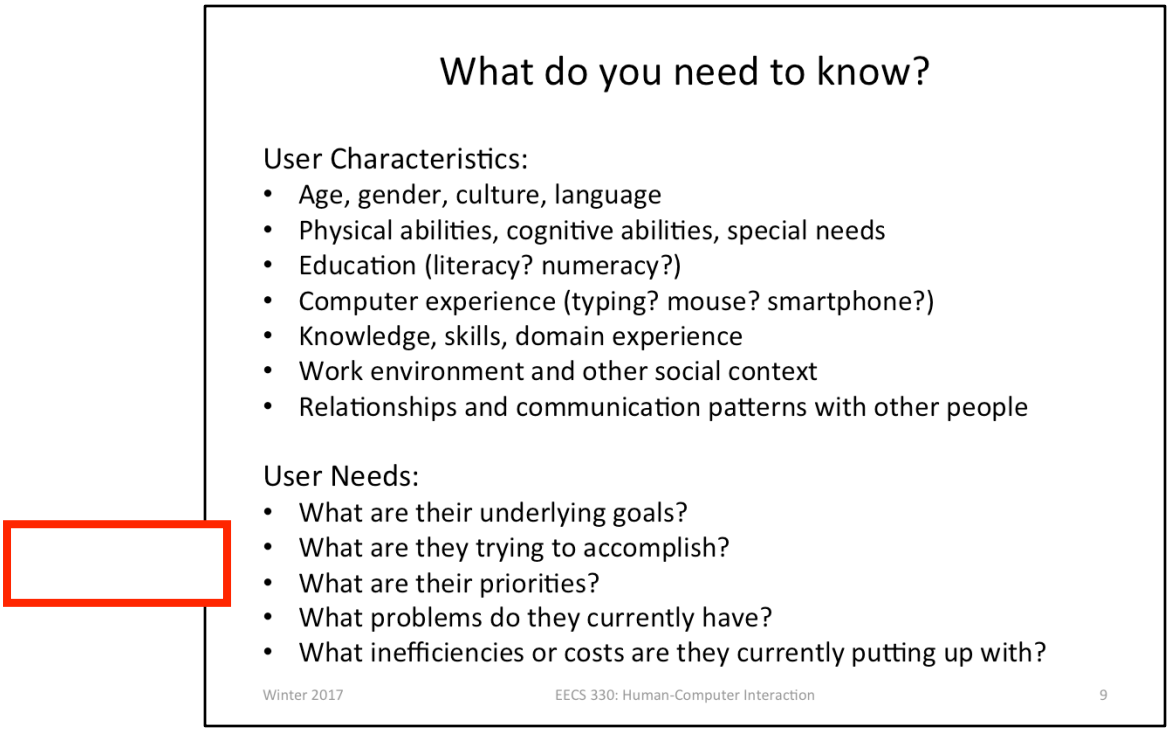
Easy to think of yourself as a typical user

Easy to make mistaken assumptions

Okay, so what if you feel like you already know exactly what your users need? My response to that is if you haven't talked to any users yet, you're wrong. You don't know what they need.

One of the easiest mistakes to make in user-centered design is forgetting that you are not the user. I know this seems really obvious, but it's really important to keep this in the back of your mind all the time. Maybe you're designing an app that you would want to use. Still, as a computer scientists and a design, you have different experiences than the traditional user. You have a different level of comfort and familiarity with technology. You use different terminology when you talk about apps. And you have a different way of looking at the world because of your experiences with technology.

It's really easy to think of yourself as a typical user, and assume that you know what your users are going to need. And as a result, it's really easy to make mistaken assumptions about what your users want and need. But as a good designer of software, it's really **really** important to recognize that not all of your users are going to be like you, and you need to design something that will work for all of the people who you want to benefit from your system.



So what do you need to know about your user?

First you want to know some basic characteristics about your users. Things like age, gender, culture, language. Physical and cognitive abilities, any special needs. Level of education and computer experience. Any knowledge or skills they're bringing to your system, or any specific expertise in your domain.

It's also important to learn about your user's context. For example the work environment or context in which they'll be using your application. And also any relationships and communications they have with other people related to your problem area.

For needfinding, it's also clearly important to learn about your users' needs. What are their underlying goals? What are they trying to accomplish? What are their priorities, and what do they care about? What problems do they currently have while trying to accomplish these goals? What inefficiencies or costs are they currently putting up with? You should be thinking about these needs as opportunities for new designs and solutions.

Common Pitfalls

- Assume all users are alike
- Assume all users are like you
- Describe what you ***want*** your users to be, instead of what they actually ***are***

"Users should be literate in English and be able to use a smartphone with only one hand"

There are some common pitfalls that new designers encounter when trying to learn about their users. First, we sometimes assume that all users are alike. But unless you're targeting a very narrow user group, it's likely that you're going to have some diversity among your users, and they may have different needs and limitations. Ideally, you want a solution that will support all of your target users.

As I already mentioned, another pitfall is assuming that all users are like you, the designer. Remember that you are not the user.

Finally, it's easy to fall into the trap of describing what you **want** your users to be, instead of what they actually **are**. For example, maybe you decide that your users should be literate in English and be able to use a smartphone with only one hand. It would certainly be easier to design for this narrow group of users, but maybe you're cutting out a lot of people who could really benefit from your system.

How can we learn about users?

Collect and analyze data.

We will talk about two common techniques:
- Interviews
- Observations

There are many other techniques
- Ethnography
- Participatory Design
- Experience Sampling
- Task analysis

The only way we can really learn about our users is by collecting and analyzing data from a representative sample of users.

Today we're going to talk about two common techniques for gathering information about users: interviews and observations. These are the techniques you're going to use in P2, your next project assignment. But there are many other techniques out there for learning about users, things like ethnography, participatory design, experience sampling, and task analysis. We could easily teach a whole class on methods for learning about users and how they interact with technology, but for now we're just going to focus on two of the most common and useful methods of learning about users.

Today's Topics

- Needfinding
- Interviews
- Observations
- Overview of P2

Interviews

Method of asking questions and listening

Format can be:
- Structured
- Semi-Structured
- Unstructured

Helpful to use:
- Props
- Competitor products
- Sample scenarios
- Prototypes

Winter 2017 EECS 330: Human-Computer Interaction 13

Interviews are a method of asking your users questions and listening to their responses to learn about their backgrounds, experiences, and needs. Interviews allow users to tell you what they know about themselves, which can provide very useful information for user interface design. Interview skills are also a basis for many other more advanced methods of collecting information about users, such as survey design and task analysis. So interviewing is a very valuable skillset for human-computer interaction practitioners and researchers.

Interviews can take a few different formats: they can be completely structured, meaning that a set of questions is determined in advance, and all participants are asked the exact same set of questions. A more common format is the semi-structured interview, in which the interviewers prepare questions in advance that they ask all participants, but they also add on some additional questions to get more details or probe areas of interest that arise during the interview. Finally, unstructured interviews are just conversations around a topic, without any pre-prepared questions. Most of the time, unless you have a real reason for structure or lack of structure, we recommend that you use semi-structured interviews.

Sometimes, it can be helpful to get users into the mindset of thinking about your problem area by using props, competitor products, sample scenarios or prototypes. This can sometimes inspire your participants to talk in more depth about how they

Interviews

Interviewing isn't as straightforward as is sounds

In the interviewer/interviewee relationship:

- The interviewer asks a question, the interviewee responds
- At a pause, the interview asks another question from a list
- When all the questions are answered, the interview is over

Potential Pitfalls:

- Traditionally the interviewer has too much power
- You don't know what will turn out to be important
- You need to make sure to ask the "right" questions

Okay, so how do you conduct an interview? It might seem like interview is really straightforward and simple, but it isn't always as easy as it sounds.

One thing to be aware of is the interviewer/interviewee relationship. Typically, the interviewer will ask a question, and the interviewee will respond to the question. When the interviewee finishes the response, there's a pause, and then the interviewer asks another question from the list. Then when all the questions are answered, the interview is over. How does this sound? What can go wrong?

The biggest potential pitfall in interviewing is that traditionally, the interview has too much power. As the interviewer you are the leader, and at the end of the day, the user who you're interviewing is doing you a favor. So the user is mostly trying to keep you happy. The problem is that you might not know yet what information is going to be important, so you may ask the user questions that don't uncover the information you need. And since you're the leader, the user typically isn't going to jump in and provide you with lots of information that you didn't ask for. So one of the most important parts of interviewing is making sure that you're asking the "right" questions, the questions that are going to give you as much information as possible about the user.

Interviewing Activity

You're doing a study to understand the quality of the Northwestern Computer Science program.

Write down five interview questions to help you explore this topic (3 mins)

Try your questions on the person sitting next to you and write down their responses (3 mins each).

What were questions that worked? Why?
What were questions that didn't work? Why?

Interview Question Types

Open-ended: *What do you think about your job at the hospital?*
Close-ended: *Do you like your job at the hospital?*

Behavioral: *Can you describe a recent occasion when a patient alert sounded, and tell me what you did?*

Feeling: *What do you like most about your job?*

Knowledge: *If a patient says she is in pain, what do you look for?*

Illustrative: *Some nurses hate working at night, but others like the flexibility. What's your experience?*

Role-playing: *If I were a new nurse coming to this hospital, and I asked you what I should do to succeed, what would you tell me?*

Preparatory: *We've been talking about your job. Now I want to ask you about how you got to be where you are today.*

There are a lot of different types of questions you can use while interviewing, and choosing the right type of question often depends on the information you're trying to learn. So let's say that we want to design a new app that can help nurses keep track of patient information while working at a hospital.

First, in terms of format, you questions can be open-ended or close-ended. Open-ended questions tend to get more information and detail, because they allow the user to answer in her on words. Close-ended questions can be useful if you're taking a survey or want more structured impressions of how your users feel. So for example, *what do you think about your job at the hospital?* leaves the user open to add detail and opinions, while *do you like your job at the hospital?* encourages the user to just answer yes or no.

Beyond format, your questions can be designed to elicit different types of responses from your users. Behavioral questions can help you figure out how your user handles different situations, for example, *can you describe a recent occasion when a patient alert sounded, and tell me what you did?* You can also ask questions that elicit feelings from your users, for example *what do you like most about your job?* To learn about the specific knowledge that the user applies in different situations, you can ask knowledge-focused questions, for example *if a patient says she's in pain, what do you look for?*

Be cautions about asking…

- How they would act in hypothetical situations
  - They can't accurately predict how they would act
- What features they would like
  - They will focus only on features they know are possible
- What other people would like
  - They can only speak to their own experiences
- How to design a user interface
  - They aren't design experts, they don't know what's possible

Now there are a couple of things that you should be cautious about asking during an interview.

First, it isn't very useful to ask users how they would act in hypothetical situations. For example, asking something like "if you had a tool to track patient information, how would you use it", isn't going to get you much useful information. This is because users can't accurately predict how they would act in a situation they've never been in before.

It's also not very helpful to ask users what features they would like, because they're only going to focus on the features they already know are possible. But maybe you can add a feature that they never would have though of, and maybe that feature will actually be really useful.

You also shouldn't ask users what they think other people would like, because they can only speak from their own experiences. And you shouldn't ask them how to design a user interface, because they aren't design experts, and again, they don't know what's possible.

It's better to use interview questions to learn about your users and their needs, instead of asking about specific designs or features.

Interviewing Guidelines

- Be flexible (semi-structured interviews help)
- Adjust your questions to their previous answers
- Ask follow-up questions, ask for examples

- Your ultimate goal: learn about the user's needs

Finally, here are a couple of high-level guidelines for interviewing. First, be flexible. This is why semi-structured interviews are so common: because they allow you to adapt your questions to match the situation. It can be really useful to adapt your questions to the participant's previous answers, and ask follow up questions, and ask for examples. Remember that your goal is to learn about the user's needs, not to perfectly follow a pre-planned interview protocol. So it's okay to change things up as needed.

Today's Topics

- Needfinding
- Interviews
- Observations
- Overview of P2

Why would you want to observe users?

- Interviews can only capture what users **say** and **think**, not what they actually do
- Observations can capture what users **do** to solve problems currently, and what solutions they **use.**
- Discovering problems and inefficiencies with users' current solutions can help you identify needs.

Okay, so why would you want to observe your users? We've already talked about all of the information that you can learn through interviews, so why are observations necessary?

Interviews can only really capture what your users say and think, not what they actually do in real-world situations. On the other hand, if you observe your users, you can capture what they actually do to solve problems right now, and what types of solutions they use.

Discovering problems and inefficiencies with users current solutions can help you to identify their needs.

Contextual Inquiry

A design-oriented, ethnographically inspired method for finding out what users currently do, and the problems they encounter.

"The core premise of Contextual Inquiry is very simple: go where the customer works, observe the customer as he or she works, and talk to the customer about the work. Do that, and you can't help but gain a better understanding of your customer."

– Hugh Beyer and Karen Holtzblatt, "Contextual Design"

One of the most effective methods for observing users is something called **contextual inquiry**. Contextual inquiry is a design-oriented, ethnographically inspired method for finding out what users currently do, and the problems they encounter.

In their book "Contextual Design" Hugh Beyer and Karen Holtzblatt say that "*the core premise of Contextual Inquiry is very simple: go where the customer works, observe the customer as he or she works, and talk to the customer about the work. Do that, and you can't help but gain a better understanding of your customer".*

So this makes it seem really simple. But there are a few tricks to observing your users using contextual inquiry.

What is the relationship?

The "master/apprentice" relationship is at the heart of contextual inquiry

In a master/apprentice relationship:

- The master is doing stuff
- The master explains what they're doing
- The apprentice asks clarifying questions
- The master answers

The participant is the master and you are the apprentice!

So one of the first questions you might have, is what is the relationship between the the user and the observer during contextual inquiry? How does it work?

You can think of the relationship as a master/apprentice relationship. And this relationship is really at the heart of contextual inquiry In this relationship, the master is doing stuff, the master explains what she's doing, the apprentice asks clarifying questions, and the master answers.

In case it isn't clear already, the participant is the master, and you are the apprentice! The participant is the one who knows their own needs and process, and you are trying to learn about their current process to identify areas where your software solution might be able to help.

It's not quite master/apprentice

Your goal is not to learn how to do the task

Instead, your goal is to learn how the participant does the task in order to learn how to support it

To achieve this goal, you to enlist the participant's active assistance in understanding the task

No while the master/apprentice metaphor is useful, it isn't quite perfect for contextual inquiry, because your goal isn't to learn how to do the task that the participant is showing you. Instead, your goal is to learn how the participant does the task to help you figure out how to support it through your new software solution. And to achieve this goal, you're enlisting the participant's active assistance to help you understand the task. But the key thing to remember here is that **the participant is the expert**, and you are trying to learn from the participant through observation.

Principles of Contextual Inquiry

Context
- Must be done in the setting of the participant.

Partnership
- Master/apprentice model; observer is humble.

Interpretation
- Observed facts must be considered for their design implications. Raw facts without interpretation are not very useful.

There are a few key principles of contextual inquiry that you should keep in mind.

First, context. In order for your observations to truly be contextual inquiry, they must be done in the setting of the participant. It may seem unnecessary to go into the real context, but you can learn tons of unexpected information that could affect your design by going to the real context. For example, maybe your user needs to multitask while using your app, or maybe they're going to be interrupted by other people in their environment. Those are all factors that are going to impact how people use your system.

The partnership between the participant and the observer is also fundamental to contextual inquiry. The observer is the apprentice, and should be humble.

Interpretation is another key component of contextual inquiry. The facts that you observe have to considered for their design implications. Raw facts without any interpretation are not going to be very useful in helping you come up with an initial design.

Context

Go to the context of your users and see them work

People summarize, but we want details

Tips:

- Avoid summary data by watching the work unfold
- Have your users think aloud
- Keep it concrete when your users start to abstract
- "*Can you walk through the process for me?*"

So first, context. You want to go to the context of your users, wherever they typically deal with the problem area that you are going to target in your project, and watch them work. One important thing to note is that people tend to summarize what they do, but you really want to know all the details. So a few tips to help you get more information.

The most useful way to learn is to watch the work unfold. Watch how your participant uses a competitor's product, or watch whatever non-technical process your participant uses to address the problem right now. It's also really useful to have your users think aloud as they're going through actions, so that you can understand what they're doing and why.

If your user starts to make an abstract, ask follow-up questions to try to keep things concrete. For example, you can say something like "can you walk through the process for me?" when a user gives you a summary of what they do. Your goal is to watch them and understand all the nuance and details.

Context

We once asked a secretary how she started her day. Her answer was, "I guess I just come in and check my messages and get started." She wasn't able to go beyond this brief summary overview. It was the first thing in the morning and she had just arrived at the office, so we asked her to go ahead and do as she would any other morning. She unhesitatingly started her morning routine, telling us about it as she went: "First I hang up my coat, then I start my computer. Actually, even before that I'll see if my boss has left something on my chair. If he has, that's first priority. While the computer's coming up, I check the answering machine for urgent messages. There aren't any. Then I look to see if there's a fax that has to be handled right away. Nope, none today. If there were, I'd take it right in and put it on the desk of whoever was responsible. Then I go in the back room and start coffee. Now I'll check the counters on the copier and postage meter. I'm only doing that because today's the first of the month. . . ."

– Hugh Beyer and Karen Holtzblatt, "Contextual Design"

Take this example from Beyer and Holtzblatt's book. They say…

We once asked a secretary how she started her day. Her answer was "I guess I just come in and check my messages and get started." She wasn't able to go beyond this brief summary overview. It was the first thin in the morning and she had just arrived at the office, so we asked her to go ahead and do as she would any other morning. She unhesitatingly started her morning routine, telling us as she went "First I hang up my coat, then I start my computer. Actually, even before that I'll see if my boss has left something on my chair. If he has, that's first priority.

So this shows that by watching users in context and asking them to think aloud, you can get a lot more information that you would from their own summary of their process.

Partnership

In one interview with a user of page layout software, the user was positioning text on the page, entering the text and moving it around. Then he created a box around a line of text, moved it down until the top of the box butted the bottom of the line of text, and moved another line of text up until it butted the bottom of the box. Then he deleted the box.

Interviewer: *Could I see that again?*

Customer: *What?*

I: *What you just did with the box.*

C: *Oh, I'm just using it to position this text here. The box doesn't matter.*

I: *But why are you using a box?*

C: *See, I want the white space to be exactly the same height as a line of text. So I draw the box to get the height.* (He repeats the actions to illustrate, going more slowly.) *Then I drag it down, and it shows where the next line of text should go.*

I: *Why do you want to get the spacing exact?*

C: *It's to make the appearance of the page more even. You want all the lines to have some regular relationship to the other things on the page.*

The designer should create a partnership

Alternate between watching and probing

Withdraw and return:

- The designer observes an action that indicates something meaningful
- The designer asks about this, and the pair withdraw from the task
- Discuss the question
- Return to the task

– Hugh Beyer and Karen Holtzblatt, "Contextual Design"

The partnership between the designer and the participant is also really important, and the designer needs to create this partnership. This can be done by alternating between watching the participant and probing into her behavior with questions.

This is another example from the contextual design book, where a designer is watching a user interact with page layout software. I'm not going to read through the whole example, but the key point is that the user does something odd with a layout box, and the interviewer probes into this behavior, saying *could I see that again?*

This inquiry behavior involves withdrawing from the task and then returning to it. The designer observes an action that indicates something meaningful, and the designer asks about it, so the pair withdraws from the task and discusses the question. Then the participant returns to the task. This can be a really useful method of drawing more information from your participant and really understanding why they're doing what they're doing.

Partnership

Avoid Other Relationship Models

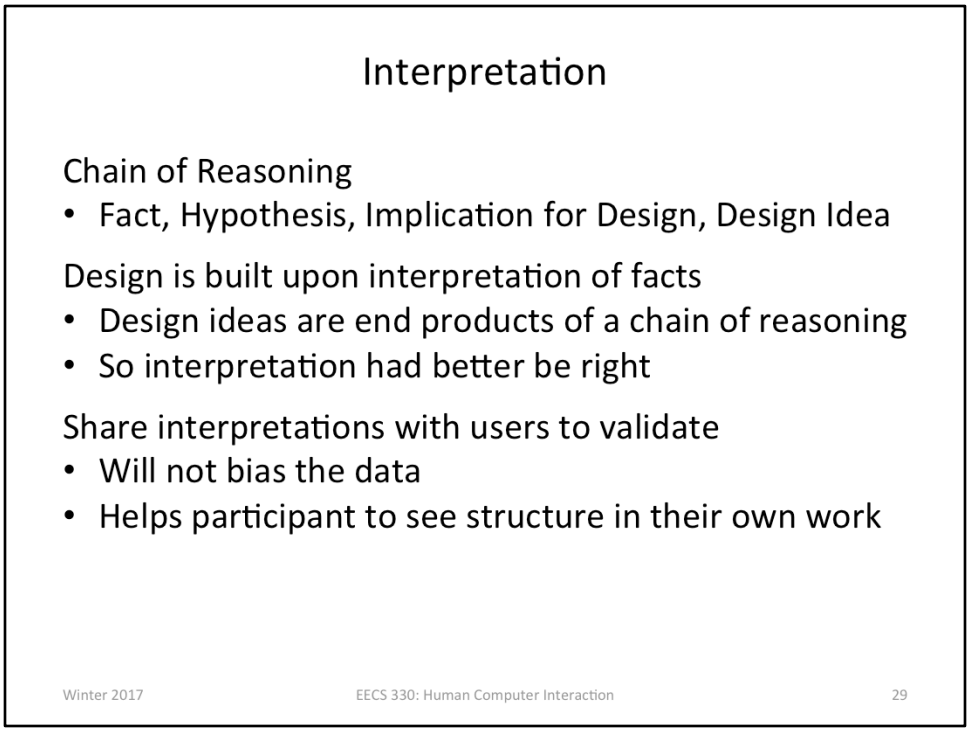
Interviewer / Interviewee
- You aren't there to get a list of questions answered

Expert / Novice
- You aren't there to answer questions

Guest / Host
- Move closer, ask questions, be nosy, fill in holes

When building this partnership as the designer, it's important to avoid other relationship models.

This isn't supposed to be an interviewer/interviewee relationship. You aren't trying to get a list of questions answered, you're trying to learn about the user's process. This also isn't an expert/novice relationship – you aren't there to answer the user's questions about technology or your problem domain area. Finally, this isn't a guest/ host relationship. It's okay for you to be nosy, ask questions, and move closer to help you fill in holes. If you're too polite, it will be hard to get the information you need.



So how do you go about interpreting the actions that you observe during contextual inquiry? There's a chain of reasoning that can get you from the facts that you observe to a design idea. The fact inspires a hypothesis about the user's needs, and that hypothesis has an implication for design, which in turn impacts your design idea.

Let's think back to the example about the secretary that I just read. Let's say as the designer you observed that the secretary first checks her chair to see if her boss left anything for her. This might lead to a hypothesis that notes from her boss are top priority first thing in the morning, and a design implication that these notes need to be visible and easy to access in your new messaging app for the workplace. This in turn leads to a design idea, for how you will display messages from the boss.

At a base level, design is built on interpretation of facts. Your design ideas are the products of this chain of reasoning, so your interpretation of the facts had better be right. One way to make check your interpretations is to share them with users to validate them. For example, you could ask the secretary something like *so messages from your boss are the first thing you need to see when you get in, correct?* This won't bias your data, because you're just asking a clarifying question, and it will also help participants to see the inherent structure in their own work.

Interpretation

Instead of asking open ended questions
- "Do you have a strategy to start the day?"
- "Not particularly."

Give participants a starting point
- "Do you check urgent messages first, no matter where they are from?
- "Actually, things from my boss are important, because they are for me to do. Messages or faxes may be for anybody."

Participants fine-tune interpretations
- Probe contradictions, don't make assumptions

So instead of asking open-ended questions like *do you have a strategy to start your day?* You may want to try giving your participants a starting point, by saying something like *do you check urgent messages first, no matter who they are from?* Your participants can help you fine-tune your interpretations, so use them. Ask questions to probe contradictions rather than making assumptions.

Contextual Inquiry Pitfalls

Forgetting to explain "the rules" of how you'll be interacting
- If this isn't clear, may devolve into a traditional interview (since an interview relationship is more familiar to people)

Slipping into abstraction
- Keep it concrete, in the work, in the details

Not being inquisitive or nosy enough
- If you have the impulse to ask, do it right away

Overly disrupting the task
- Don't ask so many questions that participants stop doing their tasks

So what pitfalls might you encounter during contextual inquiry?

One big one is forgetting to explain the rules of how you'll be interacting. If it isn't clear that you'll be shadowing the user as they go through their typical process for completing a task, the interaction might devolve into a traditional interview. So let them know how contextual inquiry works, and how you want them to think aloud as they're going through a process.

It can also be a problem when users slip into abstraction, so make sure to notice when this happens, and ask them follow-up questions to keep things concrete.

Another big pitfall is not being inquisitive or nosy enough. Part of the role of the apprentice is to ask clarifying questions, so if you have the impulse to ask, do it right away. In contrast, overly disrupting the task can also be a problem, because it can distract your participant from doing the task as they normally would. So try to keep a good balance.

Contextual Inquiry

A design-oriented, ethnographically inspired method for finding out what users currently do, and the problems they encounter.

"The core premise of Contextual Inquiry is very simple: go where the customer works, observe the customer as he or she works, and talk to the customer about the work. ***Do that, and you can't help but gain a better understanding of your customer****."*

– Hugh Beyer and Karen Holtzblatt, "Contextual Design"

The most important thing to remember is that no matter what, you're going to learn something about your user by going through the contextual inquiry process. Try to follow these tips as best as you can, but it's okay if you make some mistakes. Just try to learn as much as you can about your users and their needs, and you'll wind up with valuable information that will help shape your design.

Today's Topics

- Needfinding
- Interviews
- Observations
- Overview of P2

P2: Interviews and Observations

1. **Describe your target user population**. Write a paragraph describing your target users, including age range, background, occupation, computer experience, etc. You may find that you have multiple classes of users. If so, identify each class.
2. **Talk to and observe at least four representative users**. Every member of your project team should participate in at least one session. Observe users in settings where they would typically use your design.

Okay, so in P2, you're going to be doing interviews and observations of representative users for your own problem area. There are two central parts to this assignment. First, you need to describe your target user population. This is going to involve writing a paragraph that describes the target users, including things like age range, background, and computer experience. You may find that you have multiple different types, or classes, of users. If this is the case, identify each class separately.

Then in the second part of the assignment, you will interview and observe at least four users. Each member of your project team needs to participate in at least one session. You should use a contextual inquiry approach, so make sure you observe your users in a setting where they would typically use your design.

P2: Interviews and Observations

Example For Part 1:

We are designing a product for **working moms with newborn babies** who want to include regular low-to-medium intensity workouts in their already busy schedules. These are women from many different backgrounds who work at least part-time and have an infant to care for at home. Typical ages can be anywhere from 20 to 45 years old. Most of these women will own a smartphone and will be active on one or more social media platforms (such as Facebook, Pinterest, or Twitter). Most of our users will probably not be computer experts, but they will be savvy enough to easily navigate online.

So here's an example for part 1.

We are designing a product for **working moms with newborn babies** who want to include regular low-to-medium intensity workouts in their already busy schedules. These are women from many different backgrounds who work at least part-time and have an infant to care for at home. Typical ages can be anywhere from 20 to 45 years old. Most of these women will own a smartphone and will be active on one or more social media platforms (such as Facebook, Pinterest, or Twitter). Most of our users will probably not be computer experts, but they will be savvy enough to easily navigate online.

P2: Interviews and Observations

Steps for Part 2:

1. Introduce your problem
2. Make sure to tell them that you are not testing them!
3. Ask them to perform a task related to your problem. Ask them to **talk aloud** while performing the task.
4. Take notes:
   - Pain points, mistakes, or false assumptions
   - Things that were fun
   - Things that were confusing
   - Anything else that was surprising or unexpected
5. Ask them to reflect on the task they just performed
6. What else should we know about your users that will inform your design?

For part 2, we suggest a set of steps that you should go through with each participant. First you should introduce your problem area, and give them some context for why you're interviewing them. You should make sure to tell your users that you aren't testing them, and that all you want to do is learn how the perform a certain task.

Then, you should ask them to perform a task related to your problem. If they use a competitor's software to address your problem right now, ask them to go through the process of using that software to complete a common task. If there isn't currently a software system that addresses your problem, ask them to go through whatever process they currently use to address your problem. Maybe they keep track of their exercise regimen in a notebook, or maybe they track the behavior of their favorite sports team on twitter. Choose whatever task makes sense for your problem. And remember to ask your users to talk aloud while performing the task.

While your user is working, make sure to take notes about any pain points or mistakes, as well as things that were fun, things that were confusing, or anything else that you found surprising or unexpected.

After your user completes the task, ask them to reflect on the task that they just performed. This is where you can ask some interview questions about different

P2: Interviews and Observations

Step 7: Write up your report. This should include the following for each representative user:

- Tell us who you talked to (age, gender, occupation, life details) and why they were good representative users for your project. Don't use people's real names.
- Tell us what task you observed them doing.
- Write up your interview and observation notes in a way that other people in your team will find helpful
- Tell us what you learned about your problem area. What was surprising or unexpected? Are there pain points that you should be thinking about for your design? What are your users' needs? What opportunities for innovation did you uncover?

After you've collected information from a user, write up a report. For each of the users, you should tell us who you talked to and why they were a good representative user for your project. Tell us what you observed them doing, and write up your interview and observation notes in a way that your team members will find helpful. Finally, tell us what you learned about your problem area. We want you to reflect on things that were surprising or unexpected, identify what you learned about your users' needs, and tell us about any opportunities for innovation that you uncovered.

Readings for Wednesday

- Don Norman (2002). *Design of Everyday Things*
  - Chapter 1