Usable Privacy and Security
Introduction to HCI Methods

January 19, 2006
Jason Hong
Notes By: Kami Vaniea

Due Today: List of preferred lectures to present
Due Next Week: IRB training completion certificate (http://www.cmu.edu/provost/spon-res/compliance/hs.htm)

What is HCI?

Human-computer interaction *" (HCI) is the study of interaction between people (users) and computers. It is an interdisciplinary subject, relating computer science with many other fields of study and research. Interaction between users and computers occurs at the user interface (or simply interface), which includes both hardware (i.e. peripherals and other hardware) and software (for example determining which, and how, information is presented to the user on a screen)."*[1]

Why is HCI Important?

HCI is a very important field. When humans and computers fail to interact properly the results can be everything from annoying to deadly.

- Real programs spend 50% of their time interacting with people
- WiFi Alliance estimates 30% of boxes are returned because people can't set them up correctly resulting in financial loss.
- In extreme instances lives have been lost due to poor design. The "Therac-25 was a radiation therapy machine . . . It was involved with at least six known accidents between 1985 and 1987, in which patients were given massive overdoses of radiation, which were in some cases on the order of tens of thousands of rads. At least five patients died of the overdoses."[2]

How does HCI Work?

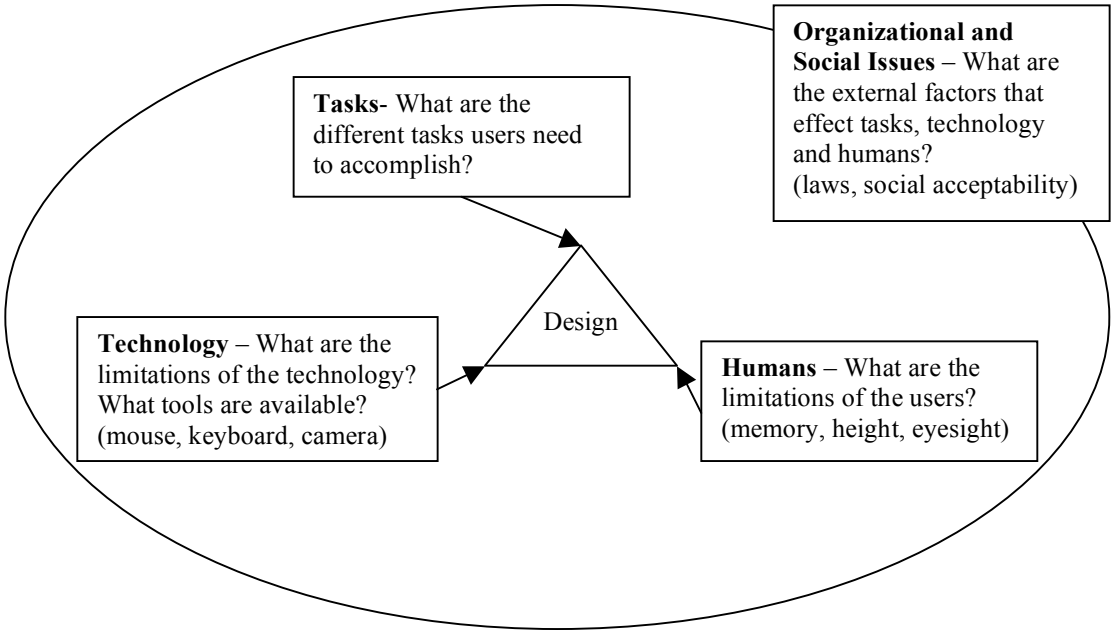
Human computer interaction is made up of three components:

- **Human** – The people who interact with the system. This may include groups of people or specific individuals.
- **Computer** – The computer or technological device that is being interacted with. This category includes devices beyond the standard desktop computer. For example: PDA, cell phone, car, door lock and ATM.
- **Interaction** – The sharing of data between the computer and the human. The human gives the computer input such as giving a cell phone the number to call.

[1] http://en.wikipedia.org/wiki/Human_computer_interaction
[2] http://en.wikipedia.org/wiki/Therac-25

The computer returns output such as displaying that the person has been called and activating the speaker and microphone.

The following model shows how these factors and others influence the design of a system. This particular diagram offers a very goal oriented view of design. It makes the assumption that the user understands what they want. Latter we will learn about dealing with users who may not know or understand what they need.



Tasks - When looking at the design of a system we first look at the tasks that need to be accomplished
Humans - We also look at the humans themselves. What are their limitations and abilities? These limitations could be anything from physical limitations such as eyesight or height to mental abilities such as level of education or how many things they can remember at once.
Technology - The limitations of technology also need to be considered. What tools do we have available to us. When designing for a PC we have a mouse and keyboard but when designing for an ATM or cell phone we have more limited tools.
Organizational and Social Issues - Last but not least, we need to look at the organizational and social issues surrounding the system. Interfaces are not designed in isolation they are affected by such things as laws and social acceptability. For example users may not use security software because it makes them look paranoid.

4 Myths of Good Design

1. Only experts can make a good design.
    a. Every day non-expert designers create good designs.

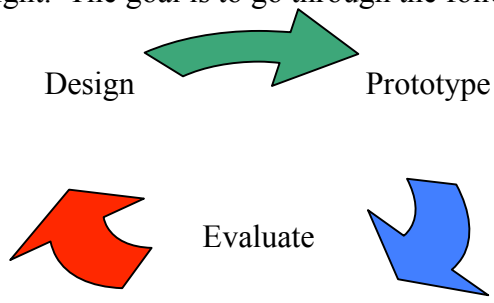
    b. Simple design strategies can be taught in only a couple of hours.
2. Build system then build an interface at the end.
    a. HCI is about more than slapping on the graphics or deciding which buttons should go where. It is about discovering what features the users need.
    b. Waiting till the end of a project to look at users' needs could result in a product that fails to accomplish the tasks of its users.
3. Good design takes too long and costs too much.
    a. There are very simple, quick design strategies that can vastly improve the quality of your product.
4. Good design is just making cool graphics.
    a. Making a system look good is important but there is more to usability than making the system look nice.

Who builds Interfaces?

Interfaces are built by all sorts of people from software designers to mechanical engineers but the most important ones are the **users themselves**. Users know more about their environment and needs than any designer. Putting the users on the design team helps keep their needs in prospective and may lead to important incites. In Scandinavia it is required by law to have end users on the design teams so they are represented in the design process.

Iterative Design

Designing interfaces isn't easy and getting it right the first time is nearly impossible. In fact getting it right the second time is also fairly difficult. Enter the iterative design process. The basic idea of iterative design is fairly simple; keep doing it till you get it right. The goal is to go through the following cycle as often and as fast as possible.



- **Design** – Learn about your users and based on what you learn design a system. How should it work? What should it look like? What does it need to do?
- **Prototype** – Take your design and build it. This doesn't need to be a fully functional implementation of the system it just needs to accurately represent what the system will look and act like. In the early stages of iterative design prototypes are often made by drawing them on paper.

- **Evaluate** – Test your design on real users. See how well users understand the design. Make notes on what you learn and use them when you return to the design stage.

Design

The first part of iterative design is, well, design. As mentioned earlier designing a system is not an easy task but there are a few things that might help.

Goals

Deciding on and clarifying goals early in the design process can help you focus on the important aspects of a system. Goals also provide a good measure of progress.

Some potential goals are:

- Learnable - faster second time round
- Memorable - easy to remember between sessions
- Flexible – multiple ways to accomplish tasks
- Efficient – perform tasks quickly
- Robust – minimal errors, good feedback to recover from errors
- Pleasing – high user satisfaction
- Fun – users think it is fun

For example consider an intrusion detection system. Which of the above goals would be most important?

- Robustness is an obvious one since the system needs to withstand attacks.
- Efficiency is also important since the system may deal with many thousands of packets hourly.

What about a user interface for encrypting email?

- Learnability is important since users must be able to understand the interface on their own, potentially without instruction.
- Robustness is also important since failing to encrypt sensitive email could be dangerous to the user.

Know thy User

This is one of the cardinal rules of interface design. If you don't know your users' limitations and needs you can't properly design an interface that meets those requirements.

So make an effort to learn about your user population.

- Age
- Sex
- Education level
- Specific limitations (color blindness, height, weight, injury)

You are not the user

This is another cardinal rule of interface design. As a designer you know way too much about your system and you can never un-learn what you know. As a result you can never completely see the world from the user's point of view.

- Keep users involved in the design process
- Do your best to think about the world from the user's point of view. (User-centered design)
- Avoid technology centered or feature driven design. Just because you have the coolest feature ever created doesn't make it the most useful feature for this situation.

Task Analysis

When creating a new system it is a good idea to start by understanding your users and what tasks they do. By analyzing these tasks early in the design process we can get rid of potential problems before they ever appear. To do a task analysis you need to answer some questions:

- Who are your users?
- What tasks do they need to perform?
- Where are those tasks performed?
- What is the relationship between users and data?
    - Are you dealing with private data, public data, does the information on the system need to be protected?
- What other tools does the user have and use?
    - A system isn't designed in isolation, other tools used by the user may impact the usability of your design
- How often do users perform each task?
- What happens when things go wrong?

Design from Data

Yet another cardinal rule of interface design. When in doubt about how something should be designed go back to your users. Build a prototype and test it. Or ask your users for clarification. However keep in mind that users don't always know what they want or need.

Contextual Inquiry

Contextual inquiry is an important tool for determining what users actually do as opposed to what they say they do. It is an important fact that users don't always know what they want or need. Sometimes social pressure can cause them to provide more social acceptable answers as opposed to the actual truth. A contextual inquiry allows you to view the user doing normal tasks in their work environment.

A contextual inquiry looks something like this:

- Conventional interview
    - Introduce focus and get some basic facts

- Transition
    - Tell them that you are going to watch them work and explain how a contextual interview works
- Contextual interview
    - Watch them work, occasionally interrupt and ask for clarification
- Wrap-up
    - Summarize your notes and confirm what is important

Personas and Tasks

Use the data you gather from contextual inquiries to create tasks and personas. A persona is a "typical" user of the system. They don't need to be real person, just based on real details. Personas help the design team think in terms of the user.

Tasks are also created from the contextual inquiry data. These tasks should be representative of the way users will use the system. You will need to create two types of tasks:

- Specific – A specific task details the exact situation the user is in complete with details and a goal. This type of task is good for understanding usability.
- Free form – A free form task is more open ended stating only the big picture goal. This type of task is good for understanding the usefulness of a system.

Prototype

Now that we have a design for our system we need to prototype it. Prototyping is a useful tool in that it allows us to quickly build systems to be tested. When prototyping systems it is useful to design them in both high and low fidelity. Fidelity refers to the level of detail in the prototype. A high fidelity prototype looks like the finished product while a low fidelity prototype looks like a quick sketch.

Low fidelity prototypes

Low fidelity prototypes have some very important advantages.

- Quick to implement (many prototypes are made from index cards and post-it notes)
- Cheap to implement
- Easy to make adjustments
- Encourages creativity
- Promotes high-level feedback from users

High fidelity prototypes

- More difficult and costly to make adjustments
- More difficult and costly to build prototype
- More low-level detailed feedback from users (I don't like that color)

Storyboards

Storyboards are another very useful prototyping tool. A storyboard maps out how the interface will look and work. Storyboards show the connections between different windows and the actions that occur at each one. A storyboard lets developers visualize how the users will use the system.

Evaluate

Evaluation is the last step in the iterative design process. This is when you finally get to try your design out on real users and see what happens.

Selecting Users

Be careful how you select your users. Make sure to select users who are representatives of the target population. Also try to select unbiased users. Family, friends or other developers are biased towards telling you good things. You need users who will tell you what is wrong with the system.

Types of Tests

There are two main types of evaluations that could be run.

- Within group
    - All members of the group try all the prototypes
    - Make sure the users don't experience the prototypes in the same order. The first prototype they see will bias them.
- Between groups
    - Two groups of test users
    - Each group tries a different prototype

Conducting a test

- Describe the purpose of the test
- Demonstrate any equipment
- Explain how to think aloud so you know what they are thinking
- Explain anything else needed
- Describe the task
- Observe how they interact with the system
- Debrief them at the end
- Be considerate of your users, try and make the experience as non-stressful as possible

Data

There are two types of data you can collect. While both types are important they each give you different types of information.

- Process data
    - observations of what the users are doing or thinking

    - gives a good overview of where the problems are in the interface
- Bottom-line data
    - summery of what happened
    - independent variables – factors you controlled (which prototype users tested)
    - dependent variables – factors you didn't control (how long it took users to complete a specific task)
    - doesn't tell you what to fix just that something is wrong.

Using Results

Now that you have your results you can use them in your design process. Analyzing the data will tell you where the problems lie and what to do about them.

- What is important?
- Lot of problems in one area?
- List critical incidents, positive and negative. Did your interface make the user exclaim "That's cool!"?
- Did the UI work the way you thought it would?

Conceptual Models

A conceptual model is a mental representation of how an object works and how interface controls affect it. A good interface communicates a good mental model to the user. It may use affordances, clues as to the correct operation. The user then uses their mental model to give the interface appropriate input.

A poor user interface provides no clues or misleading clues. The result is an inaccurate conceptual model of the system and a very frustrated user.

You can help the user construct good conceptual models by:

- Making controls visible
- Have the number of controls match the number of functions
- Label and group controls
- Have controls mirror real world