Lecture Notes on CS 443/543

By Anthony Hornof - last updated November 21, 2023

These are Anthony Hornof's notes from:

Annett, J. (2003). Hierarchical task analysis. In Hollnagel (Ed.), *Handbook of Cognitive Task Design*. Erlbaum.
Cooper, Reimann, Cronin, & Noessel. (2014). *About Face: The Essentials of Interaction Design.* Wiley.
Rosson & Carroll (2002) Usability Engineering
Sharp, Rogers, & Preece. (2019). *Interaction Design: Beyond Human-Computer Interaction* (5th ed.). Wiley.

The notes were taken to (a) learn and organize an understanding of the material and (b) prepare lectures. The notes are not complete. All chapters and sections are not included here. You must do the reading itself to learn the required course material.

Some of the notes are copied directly from the textbooks.

The content in boxes, such as this, is *not* from the book.

These notes are organized around major topics..

Table of Contents

Introduction

The class is organized around the following main topics:

1. Understanding the user's task.

This includes understanding (a) the human situation and (b) the procedural knowledge needed to do the task.

2. Designing to support the user's cognitive strategy.

This includes (a) proposing multiple different ideas for systems that will assist the user in their human situation, and with their tasks, and (b) evaluating those ideas with low-cost "formative" evaluation, including paper prototypes.

4. Implementing design ideas

5. User Observation Studies

The gold standard of usability evaluation is real users using your actual system to try to do real tasks.

A design process for digital products. (Cooper et al. Ch. 1)

Cooper et al. Start Chapter 1 on page 6:

Why Digital Products Fail

The chapter discusses the many complex influences on the development of computer systems, and notes that the main reason that systems fail is because....

The design of digital products rarely take into account the users' goals, needs, or motivations.

So what are user goals? How can we identify them?

Goal-Directed Design combines techniques of ethnography, stakeholder interviews, market research, detailed user models, scenario-based design, and a core set of interaction principles and patterns. It provides solutions that meet users' needs and goals while also addressing business/organizational and technical imperatives.

Most digital products, while technically advanced, are difficult to use. There are four main reasons why this is the case:

- Misplaced priorities on the part of the management and development teams.
- Ignorance about real users and what they need.
- Conflicts of interest when a development team is charged with both designing *and building* the user experience.
- Lack of a design process that permits knowledge about user needs to be gathered, analyzed, and used to drive the development of the end experience.

The UO CS web pages were dramatically changed on December 1, 2022. It got a lot harder to use the departmental website for the kinds of things that students, faculty, and staff might need to do. For example:

On the old web page, students could download a "Prerequisite Override Request Form" with three clicks and, along the way, receive lots of feedback that they were making progress towards their goal.

On the new web page, students need to make four clicks but, after the first click, they need scroll through page after page of photos and stories before they arrive at the needed link at the bottom of the page. The long page, it could be argued, does not communicate to the user that they are getting closer to their goal..

See the Lecture Examples file on the course website....

The new web page does a worse job supporting student learning. According to Cooper, how is it possible that this could happen?

Goals, not features, are the key to product success. (p.29)

Developers and marketers often use the language of features and functions to discuss products. But reducing a product's definition to a list of features and functions ignores the real opportunity—orchestrating technological capability to serve human needs and goals.

Goal-Directed Design is a powerful tool for answering the most important questions that crop up during the definition and design of a digital product:

- Who are my users?
- What are my users trying to accomplish?
- How do my users think about what they're trying to accomplish?
- What kind of experiences do my users find appealing and rewarding?
- How should my product behave?
- What form should my product take?
- How will users interact with my product?
- How can my product's functions be most effectively organized?
- How will my product introduce itself to first-time users?
- How can my product put an understandable, appealing, and controllable face on technology?
- How can my product deal with problems that users encounter?
- How will my product help infrequent and inexperienced users understand how to accomplish their goals?
- How can my product provide sufficient depth and power for expert users?

Someone noticed that first-year students walk around campus the day before classes start, with a map in one hand and their schedule in the other (though now students just look at their phones, which could be anything). And so someone built this system. (With fun yelling like “Find your class tour!” and “Make a new friend!” And music playing.) (Observe your world. Detective.)

Sharp 1.7 - Usability and User Experience Goals

Notes in part from: Sharp, Rogers, & Preece. (2019). *Interaction Design: Beyond Human-Computer Interaction* (5th ed.). Wiley.

1.7.3 Design Principles

The system needs to provide the following:

Visibility: Display the functions or controls that the user needs to do their task.

Feedback: Communicate to the user that they are making progress towards their goal.

Constraints: Limit the kinds of user interaction that can take place at a given moment, to what the user needs to accomplish their tasks.

Consistency: Using the same operations across interface objects, and systems.

Affordance: Provide visible clues as to how controls work.

Sharp 2 - The Design Process

Notes in part from: Sharp, Rogers, & Preece. (2019). *Interaction Design: Beyond Human-Computer Interaction* (5th ed.). Wiley.

2.2.5 The Four Basic Activities of Interaction Design are as follows:

1. **Discovering requirements for the interactive product.** This includes understanding the target users and the support an interactive product could usefully provide. It generally includes interviewing users, and observing users doing the task that you intend to support. It forms the basis of the product's requirements and underpins subsequent design and development.
2. **Designing alternatives that meet those requirements.** Propose ideas to meet the requirements. This includes conceptual design and concrete design. Conceptual design provides an abstract description of what people would be able to do with a product, and how they would interact with it. Concrete design provides specifics such as the information that would be displayed, and the commands that would be available. Alternatives are considered at every point.
3. **Prototyping the alternative designs so that they can be communicated and assessed.** The most effective way to evaluate interactive designs is to have users interact with them while attempting to do real tasks, and this can be achieved through prototyping. Not all prototypes are working pieces of software. For example, paper-based prototypes are quick and cheap to build and are effective for identifying problems in the early stages of design.
4. **Evaluate the product and the user experience it offers.** Evaluating is the process of determining the usability and acceptability of the product or design measured in terms of a variety of usability and user-experience criteria.

The "Double Diamond of Design":

- **Discover:** Designers try to gather insights about the problem.
- **Define:** Designers develop a clear brief that frames the design challenge.
- **Develop:** Solutions or concepts are created, prototyped, tested, and iterated.
- **Deliver:** The resulting project is finalized, produced, and launched.

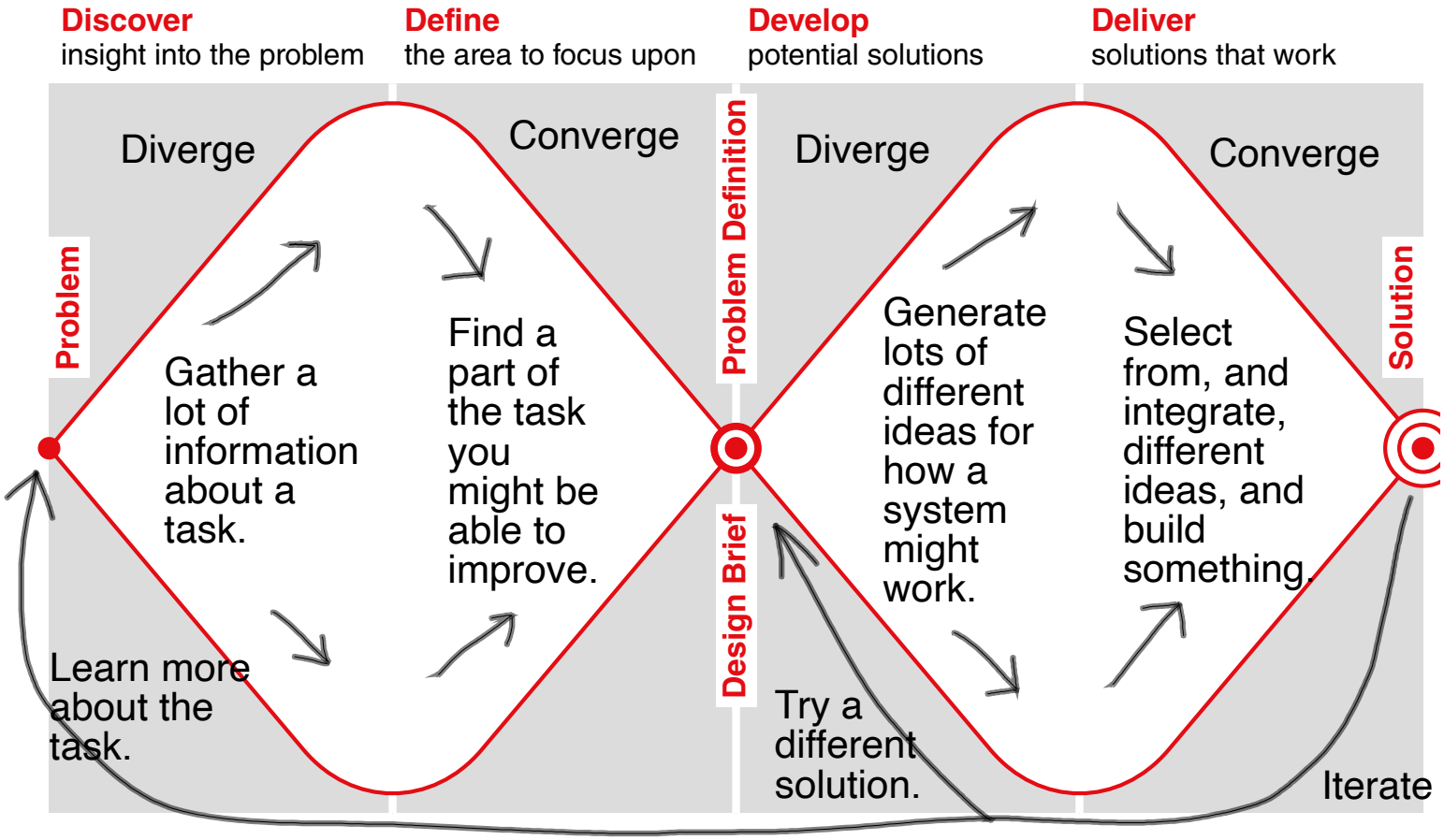


Figure 2.1 The "double diamond of design" from Sharp et al. (2019). Annotated to show how the diamond shapes represent the process of exploring solutions, and selecting one. This expansion and contraction occurs throughout a design process. And often, after you have a "solution", you need to go back and start again.

The "double diamond of design" provides a nice summary of the design process that has existed for ages: Explore and select, explore and select, as you move from abstract ideas to concrete decisions. It is not as if the creators of the "double diamond of design" invented the design process.

This process not typically taught in computer science programs. It is also somewhat hard to teach.

Hierarchical Task Analysis

Notes in part from: Shepherd, A. (2001). *Hierarchical Task Analysis*.
Annett, J. (2003). *Hierarchical Task Analysis*.
The reading is compiled in a document entitled "HTA_Materials.pdf".

"Any effort to improve human performance in a work or recreational setting must start by some understanding of what people are required to do and how they achieve tHierarchical Task Analysis

Notes in part from: Shepherd, A. (2001). *Hierarchical Task Analysis*.
Annett, J. (2003). *Hierarchical Task Analysis*.
The reading is compiled in a document entitled "HTA_Materials.pdf".

"Any effort to improve human performance in a work or recreational setting must start by some understanding of what people are required to do and how they achieve their goals." (Shepherd, p.1)

Task analysis is the process of gaining this understanding, and writing it down.

Hierarchical Task Analysis (HTA) is a methodology for describing the procedural knowledge that a person needs to do a task, and showing that knowledge in a tree-like structure.

In HTA, tasks are represented in terms of hierarchies of *goals* and *subgoals*, using the idea of *plans* to show when subgoals need to be carried out.

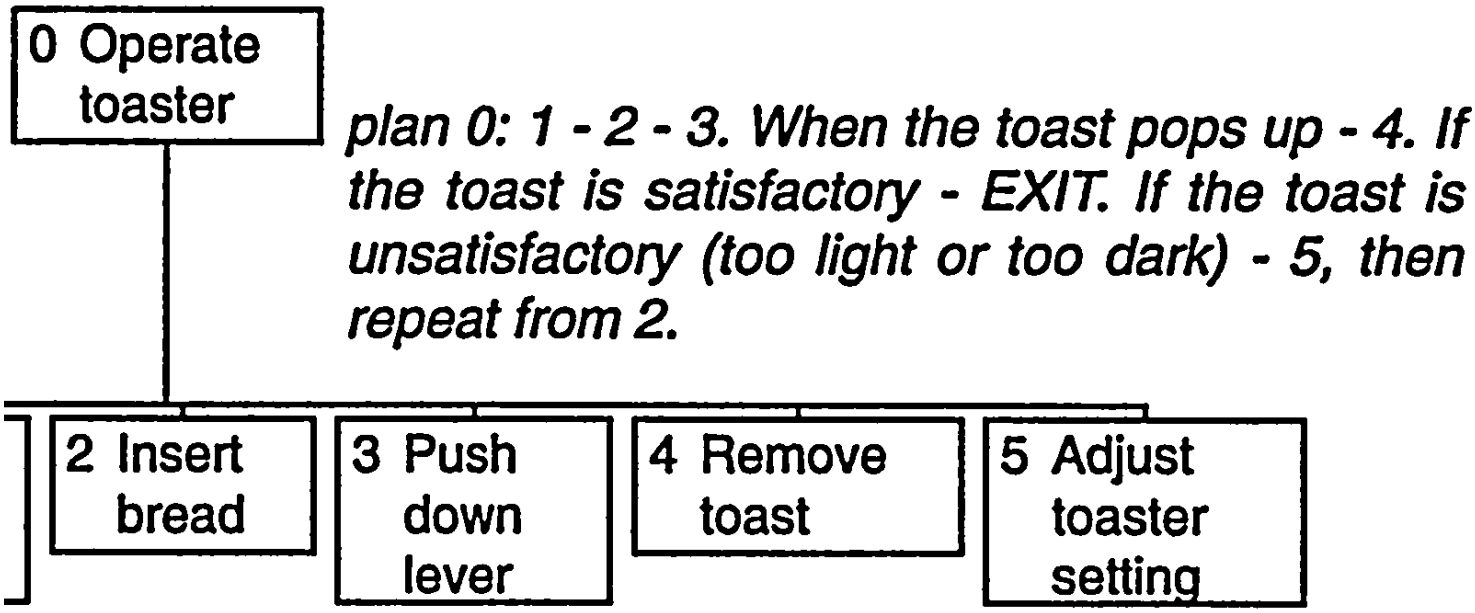
'In task analysis, it is always important to think of the reason why the task is carried out…. Thus, the task has a *purpose* or *goal* and *criteria* against which the outcome can be judged to be satisfactory or otherwise.'

Just as a task has a purpose, **the task analyst** (the person doing the analysis) also **has a purpose in doing the task analysis**. For example, the analyst might aim to understand the basic structure of a task, to design or improve a user interface, to create training materials, or to determine how people can

coordinate activities. Knowing why you are carrying out the analysis affects how you should do the analysis.

For example, if you are trying to **use HTA to assist in the design of a user interface to support a specific task**, your analysis should perhaps (a) identify mid-level subtasks that reveal potential groupings of functionality and (b) take the analysis down to the level at which individual display and control decisions could be made, such as to specify exactly what information and inputs needs to be visible for a specific subtask. (Such as a thumb keyboard, and the text that you are typing.)

The *plan* is a very important part of the procedural knowledge. The boxes (?) tell you the actions to take, but the plans tell you the conditions under which you should take each action. Such as, if and when to take the action, and for how long.



A plan for the steps required to make toast. (Figure 0.1 from Shepherd.) (The dash should be read as “then”.) Note how the plan captures the conditions under which each step should be taken.

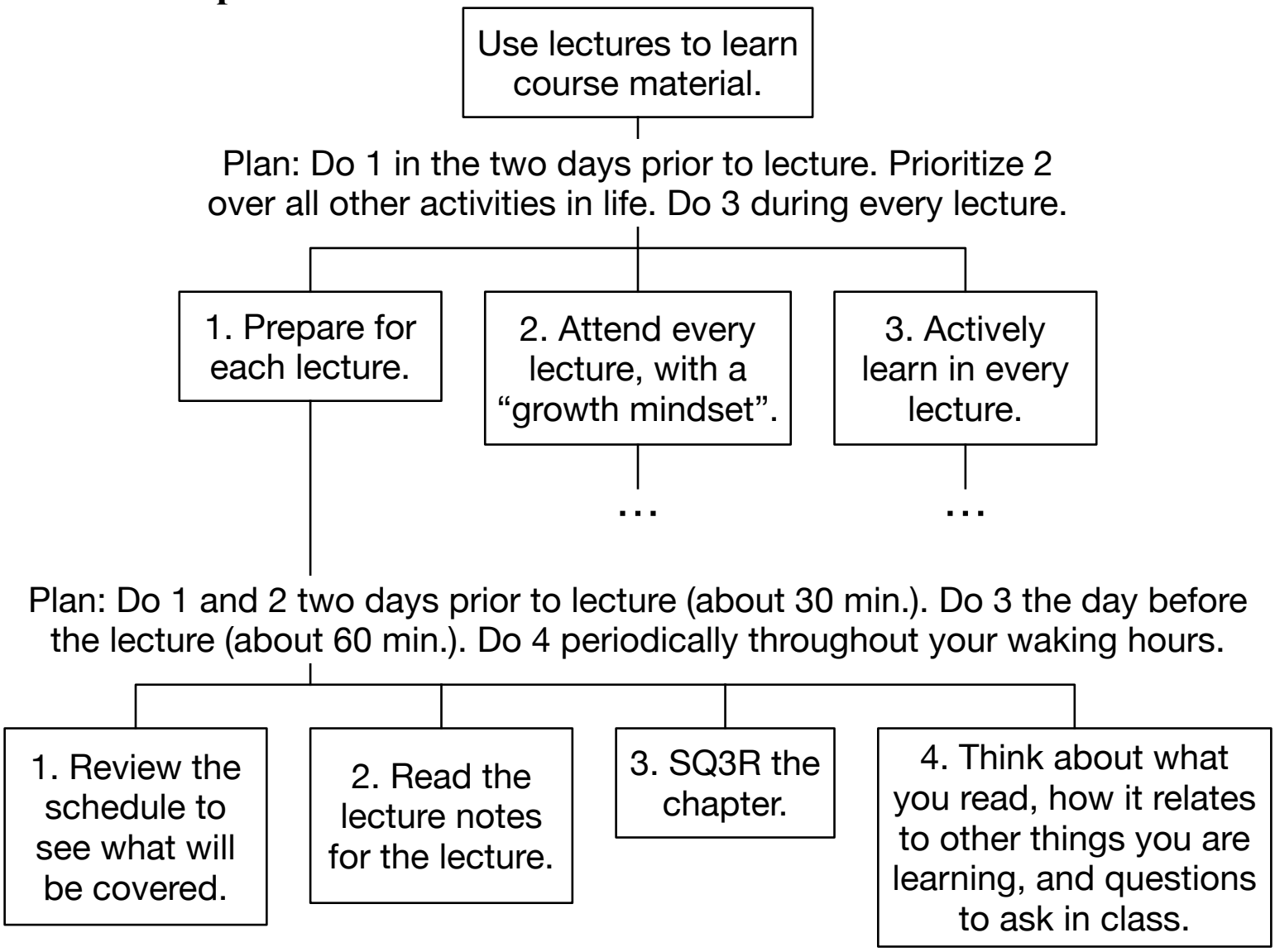
More elaborate versions of this procedural knowledge might include additional steps such as to scrape the burnt part off and serve it, or to monitor the toast while it is toasting.

Use the plan to capture the true expertise, the decisions of what to do next.

Three common types of plans in HTA:

1. *Fixed sequence* or *routine procedure*, such as "do this, then this, then this".
2. *Selective rule* or *decision*. "If *x* is the case, do this; if *y* is the case, do that. These two types of plan are significant because they imply knowledge on the part of the operator. It may be simple *procedural knowledge*, or the plan may require extensive declarative knowledge of the environment….
3. *Time-sharing* or *dual task plan*. Two or more operations ared pursued in parallel. That is, the superordinate goal cannot be attained unless two or more subordinate goals are attained at the same time. Not well-defined.

A final example:



A hierarchical task analysis (HTA) that shows an approximation of an expert college student's procedural knowledge for how to use lectures (and in-class activities) to learn the material covered in a course.

Chapter 5 - Interaction Design

Notes from Rosson & Carroll (2002) by A. Hornof in 2023

Information design focuses on figuring out what task objects and actions to show, and how to represent them. The goal of *interaction design* is to specify the mechanisms for accessing and manipulating task information.

(Don Norman's example of a wall of doors with identical handles.)

Interaction design tries to make sure that people can do the right things at the right time.

The interaction design that you build into a system will determine the activities that your users can engage in.

The human-computer interaction cycle: Establish a human goal, translate it into a system goal, develop an action plan, execute the plan, perceive the results of the execution, interpret the results, and decide whether the goal has been accomplished.

The plan-execute-perceive cycle of human-computer interaction

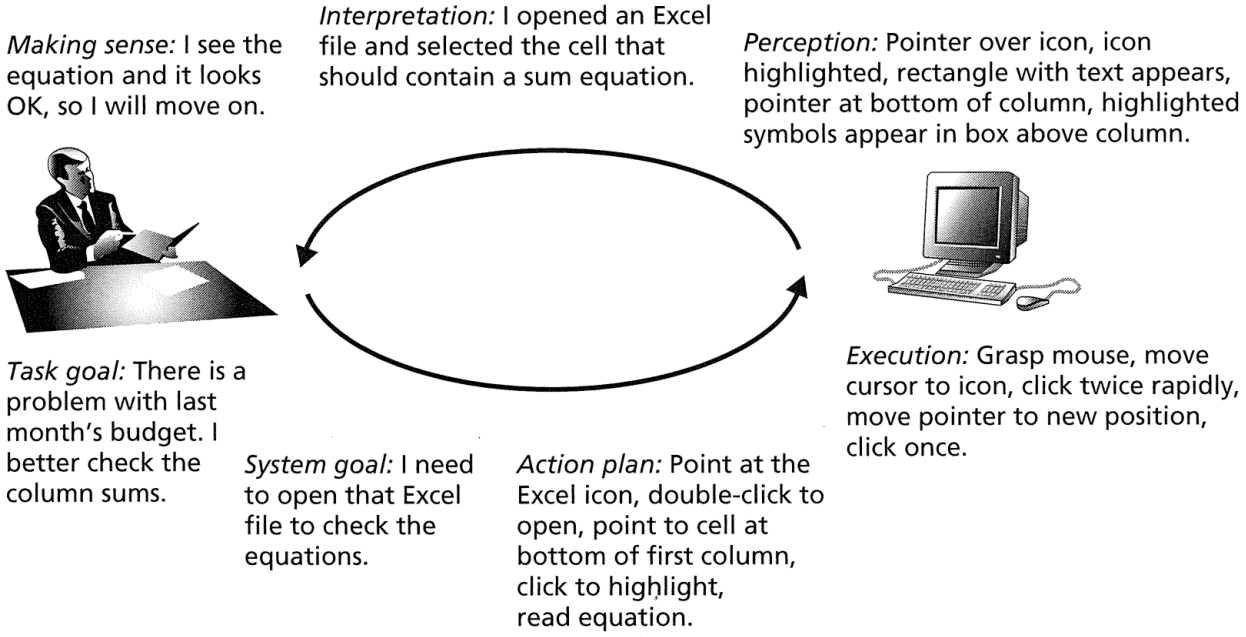


Figure 5.1 Stages of action in a budget problem: choosing, planning, and executing an action, and then perceiving, interpreting, and making sense of the computer's response.

Interaction design relates to how easy it is for a user to (a) translate his or her goals into the procedures for using a system to accomplish those goals, (b) carry out those procedures, and (c) determine that he or she is making progress towards his or her goals.

(Example: Installing Keynote on iPad.)

5.1 Selecting a System Goal

People approach a computer with a human goal. They translate it into a **system goal** and determine and execute an appropriate **task strategy** to accomplish the human goal using the system.

An **affordance** refers to perceivable characteristics of an object that helps a person to know (not "that makes it obvious" as the book defines) what the object can do, and how it can be manipulated. It relates to "stimulus-response compatibility", which is a measure of the speed and accuracy with which a person can learn, execute, and retain knowledge of the mappings between stimuli and responses. Such as, the mappings between four lights and four buttons.

When one of these lights turn on ⟶ ① ② ③ ④

Press the button it is mapped to ⟶ Ⓙ Ⓚ Ⓛ ;

Stimulus-response compatible mappings: 1J 2K 3L 4;

Stimulus-response *incompatible* mappings: 1L 2J 3; 4K

Figure 5.1 in the book also shows the "gulf of execution" and the "gulf of evaluation". The **gulf of execution** refers to the difficulty that people have in determining the physical actions needed to accomplish a task with an interface. The **gulf of evaluation** refers to the difficulty that people have in determining whether they are making progress towards those goals after executing an action.

Direct manipulation is thought to make computers easy to use by introducing graphical user interfaces (GUIs) rather than command-line interfaces because GUIs perhaps reduce the gulf of execution by making screen objects look and sort-of behave like things in the world. And because it makes it difficult for programmers to get away with assigning radically different functions to the same actions. Though they sometimes do, such as how dragging a file or a folder to the trash deletes it, but dragging a floppy disk to the trash ejects it.

Direct manipulation started with WIMP interfaces: Windows, Icons, Menus, Pointers. Touchscreen displays, such as with tablets and smartphones, take direct manipulation to a greater extreme. But all kinds of inconsistencies are introduced. For example, what is “clickable” still needs to be made very clear, and often is not. Direct manipulation is not a magical way to make interfaces easier to use. For example, on a touchscreen, there is no “right click” to see a number of potential commands for an object. And you cannot rest your finger on a button while deciding whether to press it, or touch type. And it introduces many, many modes.

5.2 Planning an action sequence: People develop and execute task strategies. When interacting with computers, these typically include perceptual and motor. They can also be purely cognitive. They can be planned ahead, prepared. So consistency matters a lot, because they permit a user to plan a few steps in advance based on how they expect the functionality to be accessed, and how the computer will behave. (Such as, when you encounter a couple fields that say “username” and “password,” to be able to type your username, tab, your password, and enter. This was not the case on DuckWeb a few years ago.)

Action sequences, or cognitive strategies, are planned and executed on the micro level (tasks that last a few seconds, such as above) as well as the macro level (tasks that last minutes, such as connecting to a network and sending a print job to a printer).

The UI designer’s challenge is to support the user at every step in their action plan, and to make it clear to them what functionality is available so that the users can map that functionality to their tasks and goals. Such as, if a user wants to print double-sided, make it clear whether that functionality is available, and if it is how to access it.

Consistency is important. People can **chunk** interaction sequences such as typing in a username and password, copying and pasting, opening applications. To “chunk” is to join several interrelated pieces of data into a single piece of

data. Such as how encoding LBT WCP ULO may require more than just three chunks, but other arrangements should take just three chunks. You can also chunk procedural knowledge, such as how scrolling down in a document should be consistent across all applications, and the same actions should always accomplish it (whether it be two fingers up—or down—on a trackpad, moving your hand to and rolling the scroll wheel on the mouse in a manner that can be prepared before your hand arrives.

A expert-user command sequence for....

Opening a program: Command-Space and the first few chars of the application name.

Googling something: Command-Space, “Fire”, Enter, Command-L Tab.

Turning off unwanted “help” in PyCharm: See “+Notes on Using PyCharm IDE.pages”

Action sequences—or task strategies—should be consistent across applications, and should not conflict. This permits the user to plan and prepare the execution of the strategy before initiating the task. When the system fails to support the prepared and executed action sequence, not only does the user have to diagnose, troubleshoot, and experiment to figure out how to do it; but all of the preparation for the initial execution is also wasted. And the interaction with the device becomes the primary task, not the human-centered goal that initiated the interaction. For example: You go to print or scan a document, and it doesn’t work.

Mistakes: An inappropriate intention is established and pursued. More common among novice users. Buying a copy of “Garage Band” because you want to start a band in your garage.

Slip: The correct goal is attempted, but a problem arises along the way. More common among experts. Example: The goal is to get cash from an ATM; you do it but you leave your ATM card. Can often be avoided by improving the interaction design, such as by giving back the card before the cash.

More examples on page 169, with design approaches to avoid the problems.

Modes should, in general, be avoided in UI design. Modes are restricted interaction states in which only certain actions are possible. Such as a "modal" dialog box that requires a response before you can do anything else with your computer; some reminders software work this way, such as to alert you of a scheduled event. A pop-up window on a web page asking you to take a survey is a modal dialog box within the context of that web page. Smartphones use modes extensively; it contribute to their reconfigurable flexibility, but it also requires lots and lots of extra button presses and swipes to switch from one mode to another.

Articulatory directness—how directly a device maps to its input requirements—is interesting to think about in terms of touch-displays. Spreading two fingers is surely like stretching something, to zoom, but a four-finger versus a three-finger swipe does not seem to have articulatory directness with anything in particular.

Interpreting System Feedback

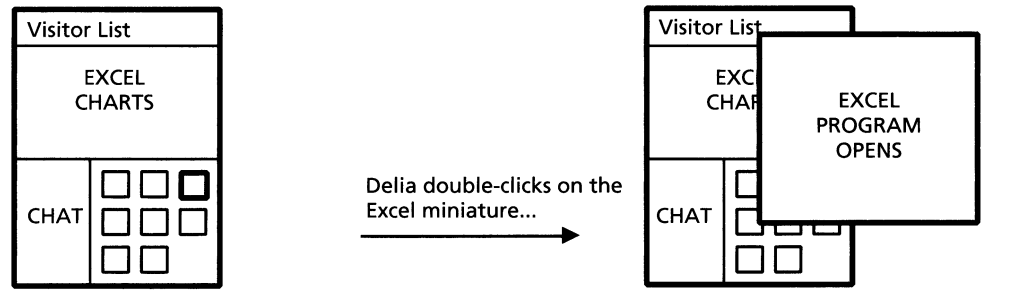
Give the user feedback with regards to how they are progressing towards their goals, at multiple time scales, including responding to any input within 100 ms, just to show that the system received your command, but also on the time scale of seconds, showing progress towards the goal. (Unix does not give great feedback. Many direct manipulation interfaces do.)

Storyboards

A **storyboard** is an event-by-event description of a sequence of interactions between a user and a device. They are named after the comic-book-like sequences that are used to plan movie shots.

A storyboard of the start of the bank robbery in *Batman - The Dark Knight* (2008)

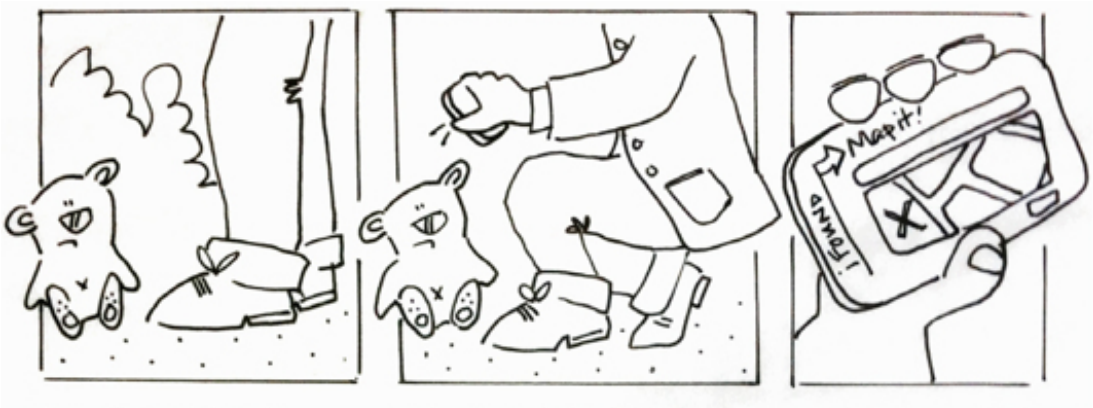
http://s3images.coroflot.com/user_files/individual_files/152129_WOEkopMM6ezXqt52G9vrtE758.jpg



1. Alicia and Delia look at the Excel charts Sally has prepared.

2. The Excel application is launched on the data files Sally has provided; Delia works with Excel independently of the exhibit.

From Figure 5.7 in Rosson & Carroll



On a walk through the park, Marco stumbles across a teddy bear fallen on the side of the path.

Realizing it must be lost, he uses his mobile phone to photograph it where he found it, and takes the bear home.

Once home, Marco uploads the photo to iFound®. The MapIt! function uses the GPS from the photo to record where the bear was found.

Part of a storyboard for an "iFound" app, which would interact with "iLost".

http://web.mit.edu/2.744/www/Project/Assignments/userExperienceDesign/ifound.jpg

Storyboards are not interfaces but they capture, in a static representation, the time-based element of the interface, which makes it easier to consider alternative designs side-by-side.
(Perhaps show the EyeMusic storyboards, annotating sound file, and the NIME promo video.)

How can you represent interaction sequences? Remember, a screenshot is not an interface. You must show how an interface evolves over time, such as with a storyboard. "Here is what the user sees. If they click here, then they see this...." The challenge is to represent a dynamic artifact.

Action sequences can be studied, and improved, at different time scales, including the fractions of a second needed to move the mouse to click on a target, or press keystrokes.

Fitts' law predicts pointing time as a function of distance (d) and width (w). There is a logarithmic relationship between d/w and pointing time. MT = a + b log (d/w) + 1. The main point is that tiny targets are very slow and difficult to click on, and the edges of the screen have certain advantages. But overall pointing-and-clicking is quite slow for time-pressured practiced tasks. You should learn keyboard shortcuts, even for responding to dialog boxes. (It is sort of foolish not to.) A good interface design should support keyboard shortcuts. One of the big differences between software for the masses like iPhoto and software for the pros such as Lightroom is that the pro versions support *lots* of keyboard shortcuts, such as to rate a photo *and* advance to the next photo with a single keystroke. (My friend Mark in NYC took my advice.)

Sharp 14 - Usability Testing

Notes in part from Sharp et al. (2019). *Interaction Design*, Chapter 14.
Much of the following text are direct quotes from the chapter.

14.3.1 Controlled Settings Involving Users

Usability testing is a fundamental, essential HCI process. User observation studies are "gold standard" of usability testing.

User observation studies collect data using a combination of methods in a controlled setting, for example, experiments that follow basic experimental design, observation, and interviews.

The studies control what users do, when they do it, and for how long.

User responses are observed and recorded. The observed data can be *quantitative* (such as speed and accuracy) and *qualitative* (such as user comments).

The primary goal is to determine whether an interface is usable by the intended user population to carry out the tasks for which it was designed. *The first question is: Could people use the device to do the task?*

A system design process produces a *software specification* (such as the Python functions that need to be written) as well as a *usability specification* (such as the time required to do representative tasks, and number of errors permitted).

A user observation study can determine if the usability specification is met.

A User Observation Study Looks for Causal Relationships.

Changes in Independent Variables
==== CAUSE ====>
Changes in Dependent Variables.

For example:
A UI design *causes* a system to be usable to do a set of tasks.
A button arrangement *causes* buttons to be clickable quickly and accurately

Experimental design is the creative process of trying to isolate and show causal relationships. This requires you to demonstrate that your findings are *valid*.

Validity

In experimental design, *validity* is the extent to which an experiment successfully isolates a causal relationship. Validity captures the extent to which an evaluation method measures what it is intended to measure.

Internal validity is the extent to which the experiment truly measures what it tries to measure; that is, within the context of this particular experiment.

External validity is the extent to which the experiment measures and shows something that is true about the world, beyond this one experiment.

Ecological validity is related to external validity. It is the extent to which the environment in which an evaluation is conducted influences or even distorts the results.

An example of validity: If you want to measure how long it takes someone to solve a Rubik's cube, you can measure their time by (a) using a stopwatch or (b) having someone count "one one-thousand, two one-thousand, ..." You would feel more confident in the the stopwatch. It is more *valid*.

A challenge when learning how to run user-observation studies is developing your ability to make decisions that contribute to validity.This is not easy.

There are many potential "threats to validity". You need to guard against them when designing and executing of a study; and when analyzing and drawing conclusions from the data.

Every aspect of a user observation study should be done with validity in mind, even if the word is never mentioned.

Some things to improve validity:

- Get your users into the "mental set" of a real user doing a real task.
- Present them with a real task, and motivate them to really do it.
- Vary the order of presentation of test conditions.
- Remove extraneous variables that may interfere with performance.
- Providing the same instructions to all of the participants.

• Train your users so they have the same expertise needed to start a task.
• Don't help the user if they get stuck.
• Use many participants.
• Recruit your participants based on the kind of users you need.
• Screen your participants to make sure they have the qualities you need.
• Run your study in an appropriate setting (such as a quiet room, a loud room, or in everyday contexts "in the wild").
• Much of Apple's Guidelines for Conducting a User Observation Study.

How do you "prove" that your study has good validity?

You cannot. Also, you cannot *measure* validity.

What you *can* do is provide information that explains everything that you did to try to ensure that your study is valid.

That is one of the main goals of the Methodology section of a usability report.
Participants - Describe who participated in your study.
Setting - Describe the physical and social environment.
Materials - Describe the devices, instructions, and scripts.
Experimental Design - Describe the different conditions, and how presented.
Procedure - Describe the steps that you took to administer the study.

The information provided in these sections can helps to convince the reader of the validity of the study. Or can leave the reader unable to conclude whether the study was valid.

Informed consent

Participants must be told what they will be asked to do, the conditions under which data will be collected, and what will happen to their data when they finish the task. Participants must also be told their rights, for instance, that they may withdraw from the study at any time, for any reason, and with no penalty.

The goal of a user observation study is to get as close as possible to real users doing real tasks in a real-world task environment.