Interaction Design: Beyond Human Computer Interaction

by Jennifer Preece

Preece, J., Rogers, Y., & Sharp, H.. (2002). *Interaction Design: Beyond Human-Computer Interaction*.
A typical undergraduate level textbook to introduce you to the field, including both scientific background and usability design methods. One of the few that adequately addresses affective measures. [DS & DN]

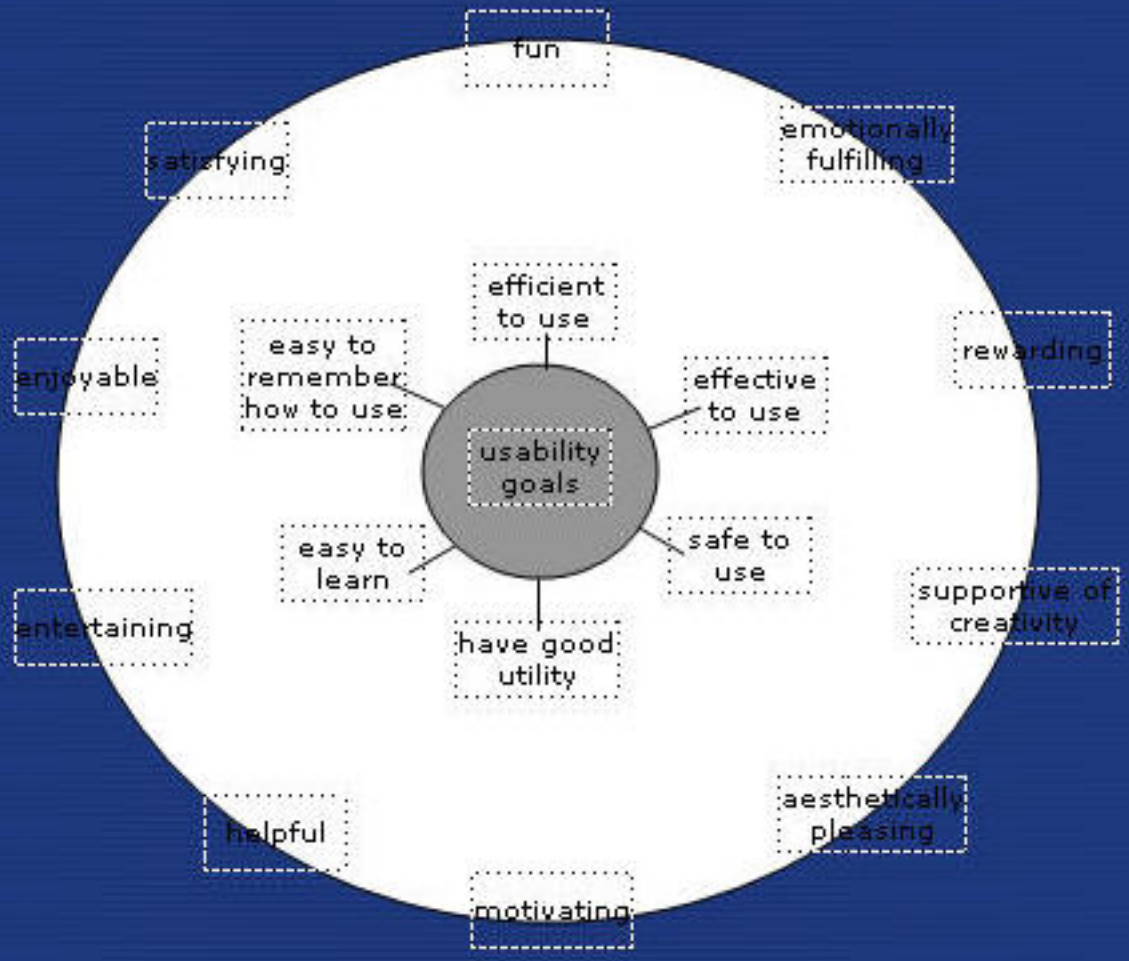
Jump To: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Table of Contents:

Preece's Summary of the Chapters: http://www.id-book.com/chapter_index.htm

Remember, **usability** is of utmost importance to Interaction Design, while **user-experience** follows (see

graph on p. 19 of book)



usability goals: at center of Interaction Design
user-experience goals: outer ring of diagram (secondary to usability goals)

Chapter 1: What is Interaction Design?

Main Goals of this Chapter:

- *Explain the difference between good and poor interaction design*
- *Describe what interaction design is and how it relates to HCI and other fields*
- *Explain what usability is*
- *Describe what is involved in the process of interaction design*
- *Outline the different forms of guidance used in interaction design*
- *Enable you to evaluate an interactive product and explain what is good and bad about it in terms of the goals and principles of interaction design*

Good and Poor Design

- Central concern of interaction design: products that are **USABLE**
    - **Easy to use**
    - **Effective**
    - **Enjoyable**
- Example: Marble interface versus Voicemail: an incoming message signaled by a marble dropping through- you grab it and drop it to play the message (good interface but breaks down if system gets more complex)

What to Design

- Who will use it?
- Where are they going to be used?
- What kinds of activities will it support?
- A key question: How do you optimize the users' interactions with a system, environment or product, so that they match the users activities that are being supported and extended

Match goals to users - get them involved

- Take into account what people are good and bad at
- Consider what might help people with the way they currently do things
- Thinking through what might provide quality user experiences
- Listening to what people might want and getting them involved in design
- Using tried and tested user-based techniques during the design process

Interaction Design:

Definition: "Designing interactive products to support people in their everyday and working lives"

Interaction Design comes from a multidisciplinary background, extends and enhances the way people work, communicate and interact

Interaction Design involves four basic activities:

1. Identifying needs and establishing requirements
2. Developing alternative designs that meet those requirements
3. Building interactive versions of the designs so that they can be communicated and assessed
4. Evaluating what is being built throughout the process

Evaluating what has been built is the heart of Interaction Design

Three characteristics of the Interaction Design Process:

1. Users involved throughout the development of the project
2. Specific usability and user experience goals should be identified, documented and agreed upon at the beginning

3. Iteration through the four activities (above) is inevitable

The Goals of Interaction Design: Usability Goals & User Experience Goals

- Usability Goals: concerned with meeting a usability criteria (e.g. efficiency)

- **Effectiveness** - how good system is at doing what it is supposed to
- **Efficiency** - the way a system supports users in carrying out their tasks
- **Safety** - protecting the users from dangerous conditions / undesirable situations
- **Utility** - extent to which the system provides the right kind of functionality so that users can do what they need or want to do
- **Learnability** - how easy a system is to learn to use
- **Memorability** - how easy a system is to remember how to use, once learned

- User Experience Goals: User experience is what the interaction with the system *feels* like to the users (subjectively)

- ***Satisfying; enjoyable; fun; entertaining; helpful; motivating; aesthetically pleasing; support creativity; rewarding; emotionally fulfilling***

Usability: Design and Usability Goals (generalized abstractions)

- ***Norman's Design Principles:***

- ***Visibility*** - functions can be seen
- ***Feedback*** - necessary part of interaction
- ***Constraints*** - ways of restricting what kinds of interaction can take place

- ***Mapping*** - relationship between controls and what happens
- ***Consistency*** - similar operations / use similar elements for achieving similar goals
- ***Affordance*** - attribute of an object that allows people to know how to use it

Usability Principles / Heuristics (heuristics are design principles used in practice - more prescriptive usability principles that are used as a basis for evaluating a system / prototype)

- ***Nielsen's 10 Usability Principles:***

  - ***Visibility of System Status***
  - ***Match between system and real world***
  - ***User control and freedom***
  - ***Consistency and standards***
  - ***Help users recognize, diagnose and recover from errors***
  - ***Error prevention***
  - ***Recognition rather than recall***
  - ***Flexibility and efficiency of use***
  - ***Aesthetic and minimalist design***
  - ***Help and documentation***

There are always tradeoffs with usability - can't over constrain things, because it limits how much info is displayed

(top)

Chapter 2: Understanding and Conceptualizing Interaction

have a clear understanding of **what, why and how** you are going to design something before writing any code.

Goals of the chapter:

- Explain what is meant by the problem space
- Explain how to conceptualize interaction
- Describe what a conceptual model is and explain the different kinds
- Discuss the pros and cons of using interface metaphors as conceptual models
- Debate the pros and cons of using realism versus abstraction at the interface
- Outline the relationship between conceptual design and physical design

Understanding the Problem Space

- the problem with solving a problem on the nuts and bolts level is that critical usability goals and user needs can be overlooked

- the design of physical aspects are best done AFTER we understand the nature of the problem space

- to understand the problem space: clarify usability and user experience goals. **Make explicit your implicit assumptions and claims.**

Framework for making your implicit assumptions explicit:

- reason through your assumption about why something might be a good idea
- this enables you to see the strengths and weaknesses of the proposed design

Conceptual Models

A conceptual model is **a description of the proposed system in terms of a set of integrated ideas and concepts about what it should do, behave and look like, that will be understandable by the users in the manner intended.**

"The most important thing to design is the users conceptual model." (David Liddle, '96)

To Develop a Conceptual Model:

- Envision the proposed product based on users' needs and requirements
- Do **iterative testing**
- What kind of interaction mode would support this? Which interaction mode to use, and which interaction style to use?
- Concrete solutions to support the above comes last
- Development should be done using: **iteration, using a number of methods, by sketching out ideas, storyboarding, and scenarios, and by making use of prototypes**

Two types of conceptual models are:

- Those based on **activities**

    1. **Instructing:** describes how users carry out their tasks through instructing the system what to do (1-way process: like word processing, CAD, email)

    2. **Conversing:** based on the idea of a person conversing with the system where the system acts as a dialogue performer (2-way process: such as search engines, advisory systems, etc)

    3. **Manipulating and Navigating:** manipulating and navigating through a virtual world by using users' knowledge of the real world (like video games, virtual reality)

    4. **Exploring and Browsing:** allows people to browse / navigate through information

- Those based on **objects**

    based on objects or artifacts, and are more specific than those based on activities (focus on a

particular object in a particular context - for example: a spreadsheet, based off of a ledger sheet)

The best type of conceptual model to use depends on the nature of the activity. Often the best answer is a hybrid (such as shopping on the Internet). However, mixing conceptual models will raise the complexity of the system.

Interface Metaphors

definition: a conceptual model that has been developed to be similar in some ways to aspects of a physical entity, but that also has its own behaviors and properties

Interface metaphors combine the familiar with new concepts

Benefits: a good orientation device
Drawbacks: often the metaphor looks / feels like the physical entity, when they should just map the familiar with the unfamiliar so that users can learn the new (unfamiliar)

There is a growing opposition to metaphors because they can break the rules of the object they represent, they can be too constraining, can conflict with design principles, can cause misunderstanding of system functionality, can limit the designer's imagination, and can have overly literal translation of existing bad design. See Metaphors description for more information.

Interaction Paradigms

- moving away from WIMP interface / paradigm

- new paradigms: ubiquitous computing, pervasive computing, wearable computing, augmented reality, attentive environments

From Conceptual Models to Physical Design

Interaction design is an ITERATIVE PROCESS, involving:

- cycling through various design processes and different levels of detail

- thinking through a design problem
- understanding users needs
- coming up with possible models
- prototyping models
- evaluating them
- thinking about design implications
- making changes
- etc...

The book describes ways of DOING INTERACTION DESIGN

- First pass: thinking about the problem space
- Second pass: more extensive information gathering about users' needs and problems
- Third pass: continue explicating the requirements through models
- Fourth pass: Fleshing out models using variety of user-centered methods. Such as: prototyping, storyboarding, physical objects, informally asking users what they think.

Issues in testing prototypes:

- Way information is to be presented and interacted with
- What combinations of media
- Kinds of feedback
- What combinations of input and output devices to use
- Whether to provide agents and in what format

- Whether to design operations to be hardwired or through physical objects or software
- What kinds of help to provide and in what format

Physical design decisions come out of conceptual decisions (i.e. what information, how to structure graphical objects, what feedback navigation and mechanisms, what kinds of icons…).

- These kinds of design decisions need user testing to ensure usability goals.

(top)

Chapter 3: Understanding Users

This chapter focuses on **USERS and COGNITION. Cognitive aspects of Interaction Design include:**

- what humans are good and bad at
- how this knowledge can be used to *inform* design of technologies that,
- *extend* human capabilities and
- *compensate* for their weaknesses

Main aims of chapter:

- What cognition is and why it is important for I-D
- Main ways cognition has been applied to I-D
- Number of examples from cognitive rese4arch
- Explain what mental models are
- Give examples of conceptual frameworks useful for I-D
- Enable you to try to elicit a mental model and understand what it means

Norman said there are two **modes** of cognition: **Experiential** (real world experiences) and **Reflective**

(thinking, comparing, deciding, etc). Both are necessary for everyday life.

Cognition has been described in SIX KINDS OF PROCESSES:

1. ***Attention*** - selecting things to concentrate on
2. ***Perception / Recognition*** - how information is acquired from the environment via sense organs and translated into experiences (vision is the most dominant)
3. ***Memory*** - recalling various knowledge. We filter what knowledge to process / memorize. (most researched area)
4. ***Learning*** - how to do something (like learning to use a program)
5. ***Reading / Speaking / Writing*** - using language
6. ***Problem Solving / Planning / Reasoning / Decision Making*** - involves reflective cognition

Often designers try to emulate the physical world with designs in the digital world. Sometimes this works well, other times it doesn't.

Conceptual Frameworks for Cognition:

- Mental Models
- Information Processing
- External Cognition

Users' Mental Models

- defined as: when people are using a system, they develop knowledge of how to use the system and to lesser extent how the system works.

- the mental model is used to help people carry out tasks. It can also give suggestions on what to do in unpredictable situations

- in cognitive psychology, mental models are defined as some sort of internal construction of the external

world that are manipulated enabling predictions and inferences to be made

- w/r/t system design: ideally, the users' mental models should match the designer's conceptual model

- to increase transparency- might make system image easier to learn (p. 95 example?)

Information Processing

- another approach to conceptualize how the mind works: through metaphors and analogies

- thinks of the mind as an information processor

- mental representations can be images, mental models, rules, other knowledge forms

the human processor model (Card, et. al 1983) is the best known approach (see p. 96)

- model predicts which cognitive processes are involved when a user interacts with a computer, allowing for calculations to be made on how long it will take a user to complete a task
- this is helpful for comparing different interfaces (efficiency)
- the approach is based on modeling mental **activities that happen exclusively in the head.** There are always external cues in the environment... so how truly representative are these models?

- there has been an increase in people studying cognitive activities 'in the wild' - in the context in which they take place (how can things in the environment aid human cognition and lighten the cognitive load?)

Alternative frameworks have been suggested: **External cognition and Distributed Cognition**

External Cognition

main idea: people interact with or create information through using a variety of external representations (books, etc.)

- an impressive array of technology has been created by humans to aid cognition (calculators, pens, etc)

- these tools have combined with external representations to extend and support our ability to carry out cognitive activities.

- some of the main goals of this:

  - *Externalizing to reduce memory load* (external representations / cues as reminders)
  - *Computational offloading* (using a tool / device to carry out a computation - like a pen / paper to do a math problem) Note: representation of the task is key- imagine the difficulty of multiplication if the numbers were represented as Roman numerals
  - *Annotating / Cognitive tracing* (modifying representations to show changes - like crossing something off a list

Back to Interaction Design: PROVIDE EXTERNAL REPRESENTATIONS AT THE INTERFACE TO REDUCE MEMORY LOAD (visualizations, cues, etc).

Informing Design: From Theory to Practice

Theories, models and frameworks provide abstractions for thinking about phenomena. They provide generalizations, but can be difficult to digest. For this reason researchers have tried to make them more practical by providing design principles / concepts, design rules, analytic methods and design / evaluation methods.

This has helped - for instance - the human processor model (Card, 83) which has been simplified into GOMS.

(top)

Chapter 4: Designing for Collaboration and Communication

humans are SOCIAL beings

Purpose of the chapter: ***look at ways interactive systems could be developed to support and extend communication and collaboration between peoples.***

Social Mechanism in Communication and Collaboration:

Rules, procedures and etiquette have been established to help people know how to behave in social groups, such as:

- **Conversational mechanisms**- to help the flow of talk and to help overcome breakdowns
- **Coordination mechanisms**- to allow people to work / interact together
- **Awareness mechanisms**- to find out what is happening, what others are doing and to let others know what is happening

Conversational Mechanisms:

- "turn-taking" helps coordinate conversation
- Implicit cues and Explicit cues (indirect vs. direct / obvious)
- Turn taking rules: speaker chooses next speaker by asking question / request, etc.
- Back channeling, body orientation, gaze, gestures are used to signal to others the flow of conversation
- Farewell rituals help end a conversation (bye, see ya later)
- Breakdowns in conversation occur when someone is ambiguous and it gets misinterpreted (followed by a re-questioning)
- Conversations can take the form of arguments, discussions, debate, chat, etc.

How to design collaborative technologies to support conversation:

- First, how do technology-mediated conversations compare to FTF? Do the same rules apply? Are there more breakdowns? How do they get repaired?

- Design implications: A key issue has been to determine how to allow for and support people to carry on communicating as if they were in the same place, even thought they are geographically separated.
- Some existing apps: phone, videophone, email, IM, videoconferencing, chatrooms, SMS texting
- How successful are these? Do they mimic or extend existing ways of conversing?

Synchronous CMC: (p.112 table 4.1)

- real-time conversations, like a chatroom
- Pros: more informed of what's going on, can allow shy people to talk more, if video support: can allow nonverbal communication to occur
- Cons: bandwidth issues can cause video to get choppy, very hard to establish eye contact, people can behave badly behind the mask of an avatar

Asynchronous CMC:

- remote communication at various times, such as email, newsgroups
- Pros: read at any time, flexible response, easier to say things, can contact many people easily
- Cons: Flaming, spamming, new message overload, don't know when people will reply

Many new communication technologies combine the above, and try to provide new / novel ways to communicate

- Collaborative virtual environments, media spaces, shared drawing tools, tools for collaborative document creation
- Pros: support talking while doing a task at same time, can be efficient to have multiple people working on the same thing at the same time, and greater awareness of what is going on
- Cons: WYSIWIS: we can't always see what people are referring to in a remote location, and floor control (file conflicts from multiple people working on the same thing at the same time)

Coordination Mechanisms:

Collaborative activities require us to coordinate with each other, so we need to figure out how to work with others to progress through the activities. Examples include:

- Verbal and Nonverbal communication (commands, gestures, nods, shakes, hand raising, to coordinate their communication)
- Schedules, rules and conventions (to organize people who take part in a project- can be formal or informal)
- Shared external representations (allow people to make inferences about the changes / delays on their current project)

How to design collaborative technologies to support coordination:

Shared calendars, schedulers, project management tools, and workflow tools have been developed to support coordination activities.

People tend not to follow conventions, because they are often not socially acceptable. Failure to make them socially acceptable can cause people to not use the system in the way intended or can cause them to abandon it totally.

Awareness Mechanisms:

These provide others with awareness of who is around, what is happening, who they are talking to. This requires knowing when is an appropriate time to interact with others and to get / pass information.

How to design collaborative technologies to support awareness:

Function is to make others aware of the others they are collaborating with. For example:

- Portholes: a series of digitized images showing people in their offices from various locations (led to

increased sense of community)

- Notification systems: users notify others, rather than being monitored, and provide information about shared objects and progress of collaborative tasks (so others can see each other and their progress)

Ethnographic studies of collaboration and communication

A main approach to informing the design of collaborative technologies that takes into account the social concerns is to carry out an ethnographic study.

This can be a home, work, school, public place (setting)

Since Suchman's "Seminal Work" many companies have invested in ethnographic studies to see how work actually gets done in a range of companies (government too)

Conceptual Frameworks: Language / Action Framework and Distributed Cognition

Language / Action Framework:

- people act through language
- goal to inform the design of systems to help people work more effectively by improving the way they communicate with each other
- this framework doesn't take into account the use of artifacts / external representations in everyday work

Distributed Cognition:

- describes what happens in a cognitive system: interactions among people, artifacts they use,

environment they work in

- the way that cognitive activity is described contrasts with others in that it focuses on not only what is happening in the head of individuals, but on what is happening across individuals and artifacts
- Examines: distributed problem solving that is taking place, role of verbal / nonverbal behavior, coordination mechanisms that are used, communicative pathways that take place as collaborative activity progresses, and how knowledge is shared / accessed

Summary:

- **social mechanisms, like turn taking conventions, allow us to collaborate / coordinate our activities**
- **keeping aware of what others are doing and letting others know what you are doing are important aspects of collaborative learning / socializing**
- **many collaborative technologies (CSCW / groupware) systems have been built to support collaboration, especially communication and awareness**

(top)

Chapter 5: Understanding How Interfaces Affect Users

Goals of this chapter: **AFFECTIVE LEARNING - a way to design systems to elicit positive responses from users (feeling at ease, being comfortable, enjoying the experience)**

- How can the appearance of the interface elicit positive responses from the user?
- How can user frustration be caused by an interface?
- How do interface agents (anthropomorphism) and synthetic characters affect us?

Affective Aspects of HCI

- traditionally, HCI has been about designing efficient and effective systems
- recently, HCI has moved towards considering how to design interactive systems to make people respond in a certain way (to be happy, to be trusting, to learn, to be motivated)
- It has been suggested that computers be designed to recognize and express emotions in the same way that humans do
- How can interactive systems be designed (both deliberately and inadvertently) to make people respond in certain ways?

Expressive Interfaces

- Colors, icons, sounds, graphical elements and animations are used to make the "look and feel" of an interface appealing
- *A benefit is that these embellishments provide reassuring feedback to the user that can be both informative and fun- which can affect the usability of the interface*
- People are willing to put up with certain aspects of an interface (slow download rate, etc) if the end result is very appealing and aesthetic
- Aesthetics have been shown to have a positive effect on people's perception of the system's usability
- Some friendly interfaces: Microsoft's 'at home with Bob' interface, 3D metaphors (living rooms, etc), agents in the guise of pets (dog) that talk to the user. These make users feel more at ease and comfortable.
- User-created expressiveness: **emoticons** - these provide non-verbal type expression in interfaces not originally intended to have this :-) Also, icons / shorthand have been used to add emotion to SMS texting (I 12 CU 2NITE)

User Frustration

Can be caused by:

- application doesn't work / crashes
- system won't do what the user wants it to do
- user's expectations aren't met
- when the system doesn't provide enough information to enable the user to know what to do
- vague error messages
- condemning error messages
- when the interface's appearance is patronizing, gimmicky, noisy
- when the user does a long series of steps to complete a task, only to discover an error was made and they have to start over

Things to avoid:

- Gimmicks: "under construction"
- Error messages: "the system has unexpectedly quit" (see Shneiderman's guidelines for error messages: don't use "FATAL" "INVALID" or "BAD", or long hex codes - try to provide context sensitive help)
- Overburdening the user: forcing them to upgrade, install plugins, do housekeeping just to use a product
- Unpleasant Appearance: too much crap on the page, featuritis (too many features), weird sound effects, childish interface, poorly laid out input devices

Dealing with User Frustration:

- people will vent, by beating the hell out of their computers, flaming, etc.
- offer helpful error messages that offer a way to fix the problem, and offer hints, help guides, cartoon agents, etc. that can help point the user in the right direction
- Reeves and Naas (1996) argue that computers should apologize when they mess up

Anthropomorphism in HCI: How much is enough?

Anthropomorphism- assigning human traits to non-human things (dancing butter, talking soda cans, dogs, cars, etc.)
- used heavily in advertising

People debate how much of this to use in system design. They can add a human feel to the system, but can also get annoying.

Which is more preferable?
- "Hello Matt! Welcome back. It's nice to see you again. Now, what were we doing last time? Ah, yes, problem five. Lets get started again."
- "User 24, commence exercise 5."

Or, when doing something wrong: "Now Matt, that's not right, you can do better than that. Try again." vs. "Incorrect, try again."

The answer:
Pros: Reeves and Naas (1996) found it is helpful to use praise in educational settings when people do something right. It increased students willingness to continue working.
Cons: However, others argue this can make you feel stupid, anxious, inferior. People hate when a computer character shakes their finger at them and says "you can do better than that, Matt, try again". In this case, many prefer the impersonal message "Incorrect, try again".

Virtual Characters

virtual characters are becoming more common. They can be used on the web, in video games, as learning companions, wizards, newsreaders, etc. However, they can be misleading (people confide in them), they can be very annoying and frustrating ("Clippy" from MS Office 97, etc).

- categorized by degree of anthropomorphism: synthetic characters, animated agents, emotional agents, embodied conversational agents

Design Implications: which one to use?
- believability: the extent to which users come to believe an agent's intentions and personality
- appearance: better to use cartoon-like characters, or those resembling humans? it depends on the situation- often the cartoon character is more believable / acceptable
- behavior: how does the agent move, gesture, refer to things? facial expressions can show emotion (remember we want constructive feedback rather than conveying inferiority / stupidity / patronizing effect

on users)

Summary:

- affective aspects are concerned with how interactive systems make people respond in emotional ways
- well designed interfaces can elicit good feelings in users
- expressive interfaces can provide reassuring feedback
- badly designed interfaces make people angry and frustrated
- anthropomorphism is increasingly used at the interface, through the use of agents and virtual screen characters

(top)

Chapter 6: The Process of Interaction Design

The ultimate goal of design is to develop a product that helps its users achieve their goals. Developing a product must begin with gaining understanding of what is required of it.

The goals of this chapter are to:

- Consider what 'doing' interaction design involves
- Ask and provide answers for some important questions about the interaction design process
- Introduce the idea of a lifecycle model to represent a set of activities and how they are related
- Describe some lifecycle models from software engineering and HCI and discuss how they relate to the process of interaction design
- Present a lifecycle model of interaction design

What is Interaction Design?

dictionary: "design is a plan or scheme conceived in the mind and intended for subsequent execution"

The plan or scheme must be informed with knowledge about its use and the target domain, together with practical constraints such as materials, cost and feasibility.

In Interaction Design, we investigate the artifact's use and target domain by taking a user-centered approach to development. The users' concerns direct the development rather than technical concerns.

Design is also about trade-offs and about balancing conflicting requirements. Generating alternatives is a key principle and one that should be encouraged in interaction. "To get a good idea, get lots of ideas." (Mark Rettig)

Typically there is a group of designers. Therefore, plans should be captured and expressed in a way that allows for review, such as sketches, descriptions in natural language, a series of diagrams, and building prototypes.

Four Basic Activities:

1. Identify needs and establish requirements
- Who is the target user?
- What kind of support will the interactive product provide?

2. Develop alternative designs that meet those requirements
- suggest ideas to meet the requirements
- Conceptual design: produce the conceptual model for the product
- Physical design: consider the details of the product (colors, sounds, images, menu design, icons, etc.)

Alternatives are considered at every point.

3. Build interactive versions (so that they can be communicated and assessed)
- a software version is not required- paper based prototypes are quick and cheap to build
- through role-playing, users can get a real sense of what it is like to interact with the product

4. Evaluate the designs (measure their acceptability)

- determine the usability of the product or design. Criteria are: how appealing is it? how well does it match the requirements? Is the product fit for the purpose?
- Evaluation results are fed back into further design (FEEDBACK / ITERATIVE DESIGN PROCESS)

Three Characteristics of Interaction Design:

1. Focus on the USERS
- involve users in the interactive design process, provide opportunities for evaluation and user feedback

2. Specific usability and user experience goals
- identify and clearly document these at the beginning of the project. They help designers to choose between different alternative designs

3. Iteration
- allows for designs to be refined. It is always necessary to revise ideas in light of feedback, several times. Innovation rarely emerges whole and ready to go. Iteration is inevitable because designers never get the solution right the first time

Practical Issues in Interaction Design:

1. Who are the users?
- those who directly interact with the system. However, can be any stakeholder: purchaser, testers, people receiving products from the system
- primary user: directly use it
- secondary user: occasionally use it or use through intermediary
- tertiary user: affected by the system, or will influence its purchase
- stakeholders: people or organizations affected by the system who influence the system requirements

2. What do we mean by "needs"?
- we need to understand the characteristics / capabilities of users, what they are trying to achieve, how they currently achieve it, and whether they would achieve their goals more effectively if they were supported differently
- characteristics that impact a product's design: users physical characteristics (ergonomics: size of

hands, height, etc), strength of product (so a child can't break it), cultural diversity of intended users
- representative users MUST be consulted!
- users rarely know what is possible. Therefore, users cannot tell us what they "need" to do achieve their goals.
- We need to examine existing task (Activity Theory?) and what the tasks' context, requirements, collaborative nature, and procedure is. Then we can envision the task being done in a new way (scenarios, etc.)

3. How do you generate alternative designs?
- it is easy to stick with something that is "good enough". Humans stick to what they know works.
- innovations arise from cross-fertilization from different applications- allows us to "break out of the box"
- often browsing a collection of designs will inspire designers to consider alternative perspectives and solutions. Designers are trained to consider alternatives, software people are not.
- design is a process of balancing constraints and constantly trading off one set of requirements with another, and the constraints may be such that there are few viable alternatives available.
- alternatives come from looking at other, similar designs, and the process of inspiration and creativity can be enhanced by prompting a designer's own experience and by looking at others' ideas and solutions. ("Flair and creativity" : research and synthesis)

4. How do you choose among alternative designs?
- there are factors that are externally visible and measurable and those that are hidden from the users' view. *Focus on the external / visible*.
- prototypes can be used to evaluate with peers and users
- fundamental user-centered design: choose between alternative designs by letting users and stakeholders interact with them and by discussing their experiences, preferences and suggestions for improvement.
- technical feasibility: some are just not possible
- quality thresholds: usability goals lead to criteria. This USABILITY CRITERIA need to be set early on and checked frequently.
- usability engineering: the process of writing down formal, verifiable and measurable usability criteria.
Some suggestions:

- Effectiveness: Appropriate support? Task coverage, information available

- Efficiency: response time? Performance measurements?
- Safety: How safe? How often does it crash / loose data?
- Utility: Which functions are superfluous?
- Learnability: How long does a novice take to learn? High learning curve?
- Memorability: How log to remember how to perform common tasks?

Lifecycle Models: Showing how the activities are related

- lifecycle models: represent a set of activities and how they are used; management tools; simplified versions of reality

- some models from software engineering: waterfall, spiral, RAD, etc

- Waterfall model: a linear process where each step must be completed before moving to the next. This is bad because there is no iteration, and modifications cannot be made to the design. Users cannot evaluate prototypes

- Spiral model: two features: risk analysis and prototyping. Alternatives are considered and encouraged.

- RAD (Rapid Applications Development): takes a user centered view and tries to minimize the risk of changing requirements through the project. A system or partial system must be delivered on a set of intervals.

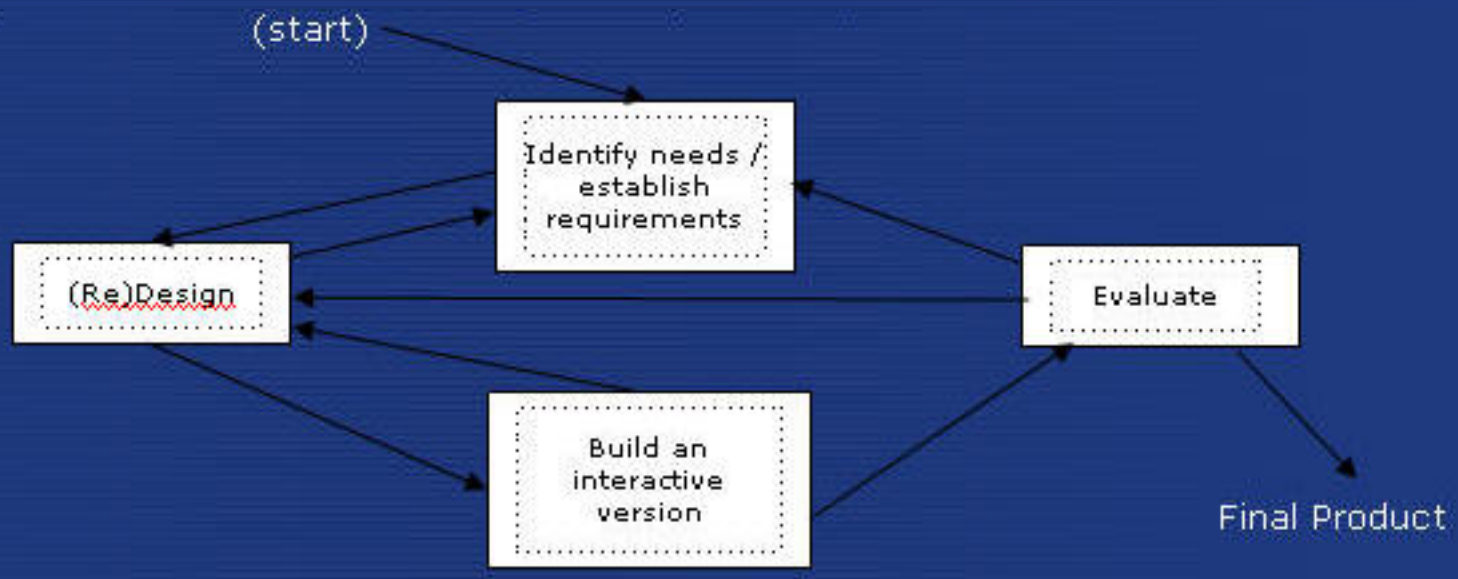
- some models from HCI: Star, usability engineering

- Star Lifecycle model: does not specify order of activity. All activities are highly interconnected. You can move from one activity to another easily, but you MUST go through the evaluation activity (in the center). Evaluation is central to this model

- Usability Engineering Lifecycle model: provides a holistic view of usability engineering and a detailed description of how to perform usability tasks. This is helpful for those with little experience. Three

phases: requirements analysis; designing / testing / development; installation

- a simple lifecycle model for interaction design: see p. 186, Fig. 6.7



(note how the above principles apply to this model)

Summary:

There are four basic activities in the interactive design process:

1. Identify needs and establish requirements
2. Develop ([re]design) alternative designs that meet those requirements
3. Build interactive versions of the designs so that they can be communicated and assessed
4. Evaluate them

These are permeated with three principles:

1. Involve users early in the design process and evaluation of the artifact
2. Define quantifiable & measurable usability criteria
3. Iteration is inevitable

Key characteristics of the interaction design process are explicit incorporation of user involvement,

iteration and specific usability criteria.

Before you can begin to establish requirements, you must understand who the users are and what their goals are in using the device.

Looking at others' designs provides useful inspiration and encourages designers to consider alternative design solutions, which is key to effective design.

Usability criteria, technical feasibility, and users' feedback on prototypes can all be used to choose among alternatives.

Prototyping is a useful technique for facilitating user feedback on designs at all stages.

Lifecycle models show how development activities relate to one another.

The interaction design process is complementary to lifecycle models from other fields.

(top)

Chapter 7: Identifying Needs and Establishing Requirements

This chapter talks about different ways to gather requirements by introducing:

- **Types of Requirements**
- **Data Gathering Techniques**
- **Task Descriptions**
  - **Scenarios**
  - **Use Cases**
  - **Essential Use Cases**

- **Task Analysis**
  - **Hierarchical Task Analysis (HTA) (task analysis)**

What are we trying to achieve in this design activity?

1. Understand as much as possible about users, task, and context
2. Produce a stable set of requirements

How can we do this?

- Data gathering activities
- Data analysis activities
- Expression as 'requirements'
- All of this is **iterative**

Why bother getting it right?

- Typically, the requirements definition stage is the most common place for failure
- Getting the requirements right is crucial, because unclear objectives will cause a project to FAIL
- A 'user-centered approach' to development is the way to solve this problem (What do users want? What do users 'need'?)

Requirements need clarification, refinement, completion and re-scoping.

Input: requirements document (maybe)
Output: stable requirements

Why 'establish' requirements?

- 'Establish' requirements: we *establish* requirements because -they arise from the data-gathering and

interpretation activities and have been *established* from a sound understanding of the users' needs.

- Because of this, requirements can be *justified by* and *related back to* the data collected.

Types of requirements:

- **Functional requirements:** what the system should do (historically this was the main function of requirements activity)
- **Non-functional requirements:** memory size, response time, date product must be finished by, etc.
- **Data requirements:** What kind of data needs to be stored, how will they be stored (database)?
- **Environment / Context of use requirements:** Circumstances in which product will be used / expected to operate
  - What are the *physical* (dusty, noisy, vibration, light, heat humidity, etc) requirements?
  - What are the *social* (sharing of files, displays, paper across distances, working individually, privacy of clients, etc.) requirements?
  - What are the *organizational* (hierarchy, IT department's attitude, user support, communication structure / infrastructure, training ability, etc.) requirements?
- **User requirements:** Who are they? - captures the characteristics of the intended user group
  - *Characteristics* - ability, background, attitude to computers
  - *System use* - novice, expert, casual, frequent
    - Novice: step by step (prompted), constrained, clear information
    - Expert: flexibility, access / power
    - Frequent: short-cuts

    - Casual / infrequent: clear instructions (such as menu paths)
- **Usability requirements:** (note: different than user requirements) - these capture the usability goals and associated measures for a particular project.

    - learnability
    - throughput
    - flexibility
    - attitude

Data Gathering Techniques: (see p.214 / Table 7.1 for excellent graph)

- **Questionnaires**

    - elicit specific information
    - can be YES / NO, multiple choice, comment
    - often used with other techniques
    - can give quantitative / qualitative data
    - good for answering specific questions from a large, dispersed group of people

- **Interviews**

    - forum for talking to people
    - can be structured, unstructured, or semi-structured
    - props, scenarios of use, prototypes can be used
    - good for exploring issues
    - can be time consuming, infeasible to visit everyone

- **Workshops / Focus Groups**

    - group interviews

- good at gaining a consensus view and / or highlighting areas of conflict

- **Naturalistic Observation**
  - spend time with stakeholders in their day-to-day tasks, observing work as it happens
  - gain insight into stakeholders' tasks
  - good for understanding the nature / context of tasks
  - Requires time and commitment from a member of the design team, and can result in huge amounts of data
  - *ethnography* is one form of this
- **Studying Documentation**
  - procedures and rules are often written down in manuals
  - good source of data about the steps involved in an activity, and any regulations governing a task
  - not to be used in isolation
  - good for understanding legislation and getting background information
  - no stakeholder time, which is a limiting factor on the other techniques

Which of the above data gathering techniques to use? The above techniques differ in the *amount of time, level of detail and risk associated with the findings*, and *the knowledge the analyst requires*

The choice of technique is also affected by the kind of task to be studied:

- sequential steps or overlapping series of subtasks?;
- high or low, complex or simple information?;
- task for a layman or skilled practitioner?

Problems with data-gathering:

- identifying and involving stakeholders: users, managers, developers, customer reps?, union reps?, shareholders?
- involving stakeholders: workshops, interviews, workplace studies, co-opt stakeholders onto the development team
- 'Real' users, not managers: traditionally a problem in software engineering, not so bad now
- requirements management: version control, ownership
- communication between parties: within development team, with customer / user, between users
- domain knowledge distributed / implicit: difficult to dig up and understand
- availability of key people
- political problems
- dominance of certain stakeholders
- economic / business environment changes
- balancing functional and usability demands

Guidelines:

- focus on identifying the stakeholders' needs
- involve all the stakeholder groups
- involve more than one stakeholder from each group
- use a combination of data gathering techniques
- start data interpretation soon after the data gathering session
- do an initial interpretation before deeper analysis
- use different approaches to different problems, such as *class diagrams* for object-oriented systems, and *entity-relationship (E-R) diagrams* for data intensive systems

Task Descriptions: (more on p. 226-231)

- **Scenarios** - an *informal narrative story*, simple, 'natural', personal, not generalizable

- **Use Cases** - assume interaction with a system, assume detailed understanding of the interaction
- **Essential Use Cases** - abstract away from the details, does not have the same assumptions as use cases

Scenario for a shared calendar:

"The user types in all the names of the meeting participants together with some constraints such as the length of the meeting, roughly when the meeting needs to take place, and possibly where it needs to take place. The system then checks against the individuals' calendars and the central departmental calendar and presents the user with a series of dates on which everyone is free all at the same time. Then the meeting could be confirmed and written into people's calendars. Some people, though, will want to be asked before the calendar entry is made. Perhaps the system could email them automatically and ask that it be confirmed before it is written in."

Use case for a shared calendar:

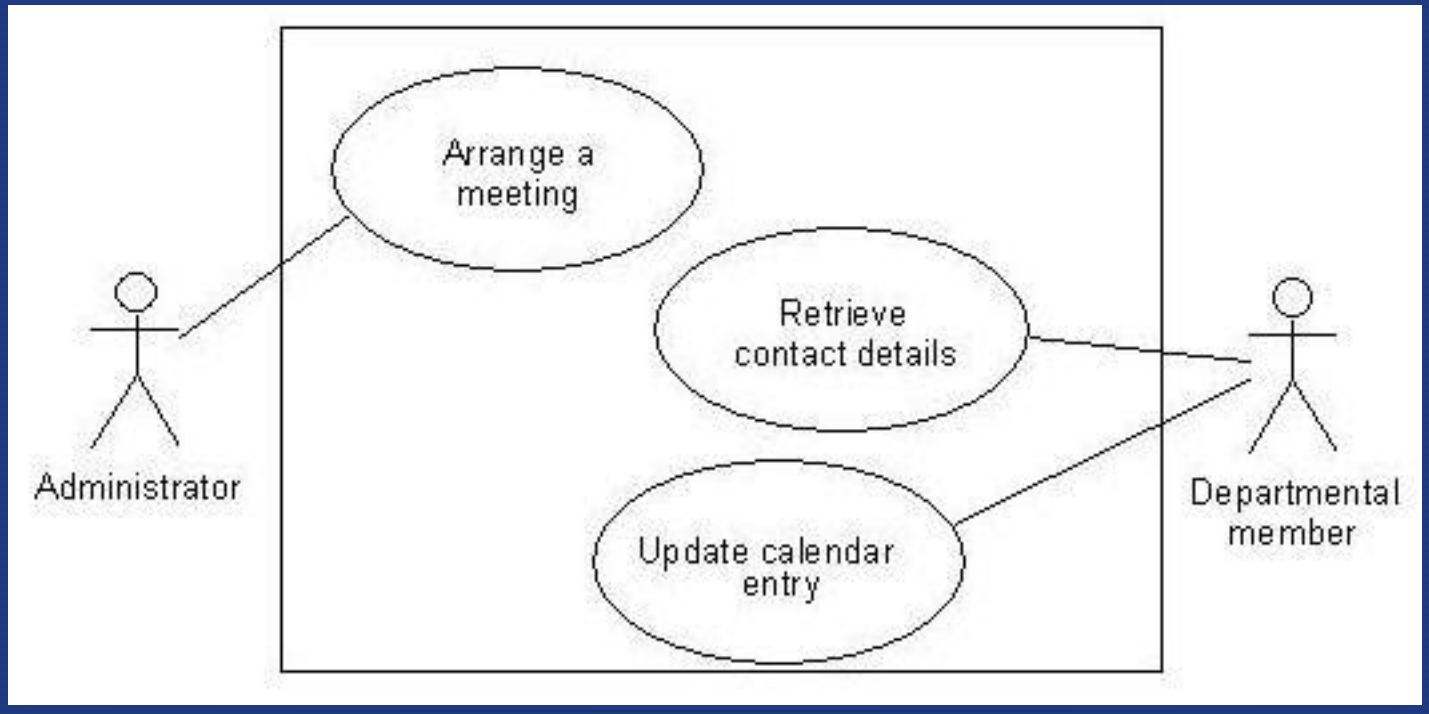
1. The user chooses the option to arrange a meeting.
2. The system prompts user for the names of attendees.
3. The user types in a list of names.
4. The system checks that the list is valid.
5. The system prompts the user for meeting constraints.
6. The user types in meeting constraints.
7. The system searches the calendars for a date that satisfies the constraints.
8. The system displays a list of potential dates.
9. The user chooses one of the dates.
10. The system writes the meeting into the calendar.
11. The system emails all the meeting participants informing them of them appointment

Alternative courses for a shared calendar:

Some alternative courses:

5. If the list of people is invalid,
    5.1 The system displays an error message.
    5.2 The system returns to step 2.
8. If no potential dates are found,
    8.1 The system displays a suitable message.
    8.2 The system returns to step 5.

Example Use Case Diagram for a shared calendar:



Example *Essential Use Case* for a shared calendar:

arrangeMeeting USER INTENTION	SYSTEM RESPONSIBILITY
arrange a meeting	
	request meeting attendees & constraints
identify meeting attendees & constraints	
	search calendars for suitable dates
	suggest potential dates
choose preferred date	
	book meeting

Task Analysis: (p.231-234)

Task analysis is an umbrella term that covers techniques for investigating cognitive processes and physical actions, at a high level of abstraction and in minute detail.

Hierarchical Task Analysis - involves breaking down a task into subtasks, then sub-sub-tasks and so on. These are grouped as plans which specify how the tasks might be performed in practice

- HTA focuses on physical and observable actions, and includes looking at actions not related to software or an interaction device
- HTA starts with a user goal which is examined and the main tasks for achieving it are identified
- These tasks are then divided into sub-tasks

Example Hierarchical Task Analysis: Borrowing a book from the library

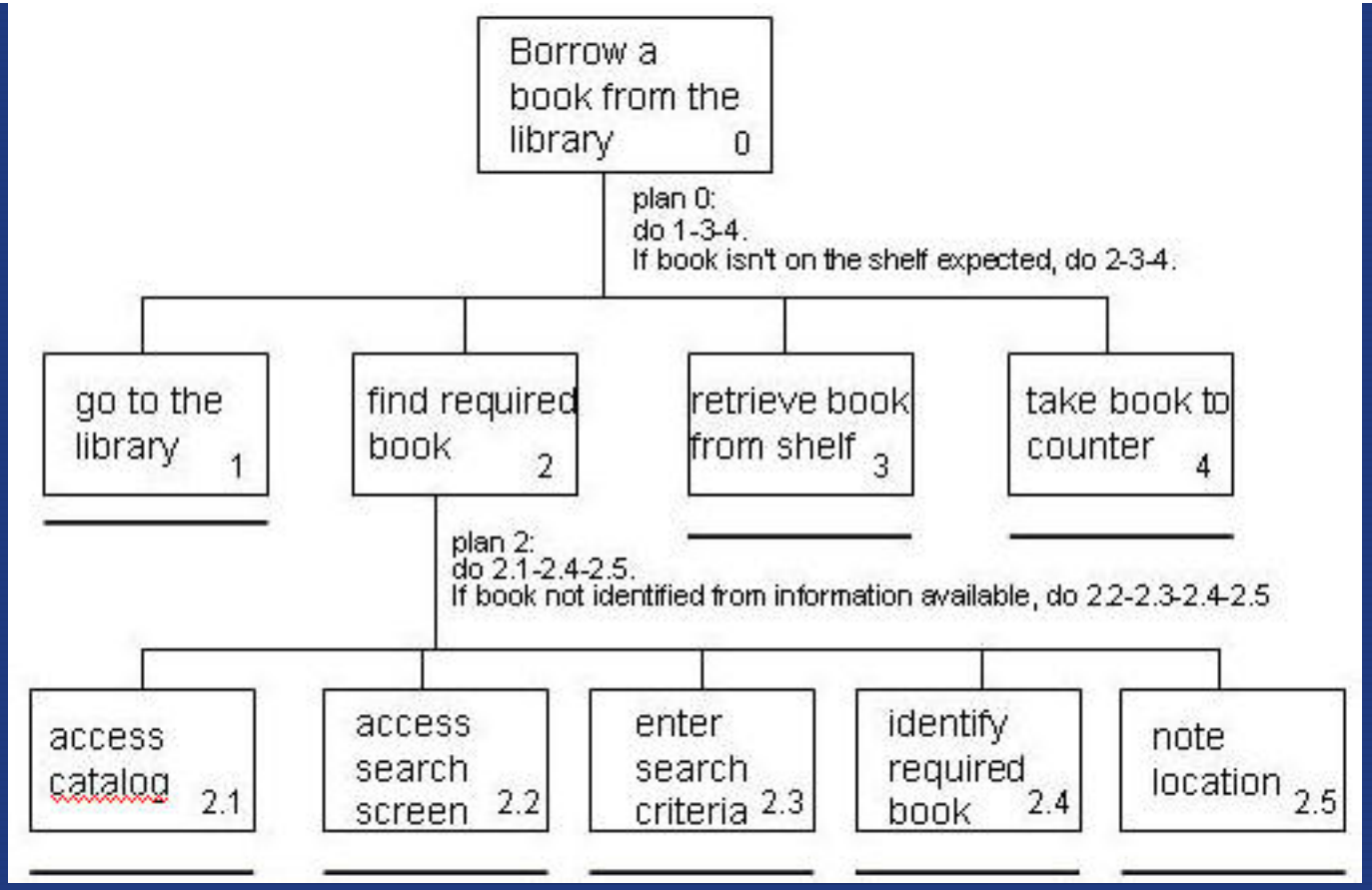
0. In order to borrow a book from the library
    1. go to the library

2. find the required book
    2.1 access library catalogue
    2.2 access the search screen
    2.3 enter search criteria
    2.4 identify required book
    2.5 note location
3. go to correct shelf and retrieve book
4. take book to checkout counter

Example HTA (Plans):

- plan 0: do 1-3-4. If book isn't on the shelf expected, do 2-3-4.
- plan 2: do 2.1-2.4-2.5. If book not identified do 2.2-2.3-2.4.

Example HTA (Graphical):



Summary:

- Getting requirements right is crucial
- There are different kinds of requirements, each is significant for interaction design
- The most commonly-used techniques for data gathering are: questionnaires, interviews, focus groups and workshops, naturalistic observation, studying documentation
- Scenarios, use cases and essential use cases can be used to articulate existing and envisioned work practices.
- Task analysis techniques such as HTA help to investigate existing systems and practices

Chapter 8: Design, Prototyping and Construction

This chapter will cover: prototyping and construction (low and high fidelity prototyping, vertical and horizontal compromises); conceptual design (conceptual model, using scenarios and prototypes in conceptual design); and physical design (guidelines and widgets).

Flow of Interaction Design:
Identify Needs / Requirements (Ch. 7) --> Prototype cycles / Design (Ch. 8) --> Construction

What is a Prototype?

- in other fields, it's a small scale model (miniature car, building, etc)
- in Interaction Design it can be a series of screen sketches, a storyboard, a PowerPoint, a video simulating use of the system, a lump of wood (e.g. PalmPilot), a cardboard mock-up, or a piece of software with limited functionality

Why Prototype?

- Evaluation and feedback are central to interaction design
- stakeholders can see, hold, interact with a prototype more easily than documents / drawings
- team members can communicate effectively
- can test out ideas immediately
- it encourages reflection, a very important aspect of design
- prototypes answer questions and support designers in choosing between alternatives

What gets prototyped: technical issues, work flow, task design, screen layouts / information displays, and any difficult / controversial / critical areas

Low-fidelity Prototyping:

- uses a medium unlike the final product (paper, cardboard)
- quick, cheap and easily changed
- can be screen sketches (drawing ability not important, practices simple symbols), task sequences, post-it notes, storyboards (often used with scenarios, and consists of a series of sketches [such as 3x5 index cards] showing how a user might progress through a task using the device)
- low fidelity prototypes have limited functionality / utility, but are helpful for identifying requirements and evaluating multiple design concepts

High-fidelity Prototyping:

- uses materials you would expect in the final product
- prototype looks more like final version than a low-fidelity version
- include software environments like Macromedia Director, Visual Basic, Smalltalk, etc.
- one drawback / compromise is that users might think they have a full system. High fidelity prototypes are time consuming / expensive to make, and are not effective in requirements gathering

Compromises associated with Prototyping:

- every prototype has a compromise - for software this may be slow response time, sketchy icons, limited functionality, etc.
- Two types of compromise: horizontal and vertical
  - 'horizontal' compromise: provide a wide range of functions, but with little detail
  - 'vertical' compromise: provide a lot of detail for only a few functions
- Compromises must not be ignored: *products need to be engineered*

Construction: taking a prototype and making it whole by engineering a complete product (focus on quality: usability, reliability, robustness, maintainability, integrity, portability, efficiency, etc.)

Conceptual Design: From requirements to design

- transforms user requirements / needs into a conceptual model, "a description of the proposed system in terms of a set of integrated ideas and concepts about what it should do, behave and look like, that will be understandable by the users in the manner intended"
- should ITERATE, ITERATE, ITERATE: don't move to a solution too quickly
- should consider alternatives (prototyping helps this) Fudd's first law of creativity: "To get a good idea, get lots of ideas" (Rettig, 1994)

Three Perspectives for a Conceptual Model:

1. *Which interaction mode?*

   - how user invokes actions
   - Activity-based: instructing, conversing, manipulating and navigating, exploring and browsing
   - Object-based: structured around real-world objects

2. *Which interaction paradigm?*

   - desktop paradigm with WIMP interface
   - ubiquitous computing
   - pervasive computing
   - wearable computing
   - mobile devices, etc.

3. *Is there a suitable metaphor?*

- Interface metaphors combine familiar knowledge with new knowledge in a way that will help the user understand the product
- 3 steps: understand functionality, identify potential problem areas, generate metaphors
- evaluate metaphors: How much structure does it provide? How much is relevant to the problem? Is it easy to represent? Will the audience understand it? How extensible is it?

Expanding the Conceptual Model:

- What functions will the product perform? What will the product do and what will the human do? (task allocation)
- How are the functions related to each other? Sequential or Parallel? How are they categorized?
- What information needs to be available? What data is required to perform the task? How is this data to be transformed by the system?

Scenarios: Using them for Conceptual Design

- scenarios can be used to explicate existing work situations, but are more commonly used for expressing proposed or imagined situations to help in conceptual design.

- they can be used through design in various ways: as scripts for user evaluation of prototypes, as a means of co-operation across professional boundaries

- in extreme cases, *plus and minus* scenarios can be used, which attempt to capture the most positive and the most negative consequences of a proposed design solution

Prototypes: Using them for Conceptual Design

- prototypes allow for evaluation of emerging ideas

- low-fidelity prototypes are used early on, while high-fidelity prototypes are used later

Physical Design: Getting Concrete

- physical design considers more concrete & detailed issues of designing the interface
- can be things like screen or keypad design, which icons to use, how to structure menus
- Some guidelines for physical design:
  - Nielsen's heuristics (see Chapter 1)
  - **Shneiderman's eight golden rules:**

    1. Be consistent
    2. Enable frequent users to use shortcuts
    3. Offer informative feedback (meaningful error messages)
    4. Design dialogs to yield closure (like when you complete a task)
    5. Offer error prevention and simple error handling (to err is human, so figure that in to your design)
    6. Permit easy reversal of actions ('undo' button)
    7. Support internal locus of control (user feels in control)
    8. Reduce short-term memory load (less info to remember between screens)

- style guides (commercial, corporate, etc. - decide the 'look and feel', along with widgets [icons, menus, toolbars, dialog boxes, etc])

  - **menu design:** How long will menu be? In what order? How will they be structured (sub-menus / dialog boxes)? What categories will group menu items? How will division of items be denoted? How many menus? What terminology will be used? What physical constraints (mobile phone) must be accommodated?
  - **icon design:** can be difficult, as icons can be cultural / context sensitive - so draw on traditions / standard, use concrete objects
  - **screen design:** Split screen? How much white space? How to group things (boxes / lines / colors)? Draw attention to the focus point, using color, motion, possibly animation, and use

good organization. Balance the tradeoff between overcrowded / sparse displays

- **Information display:** show only relevant information, make different mediums (computer / paper) consistent

There is no rigid border between conceptual and physical design... they are all iterative processes. Often in conceptual design some detailed issues come up in the iterations. The important part is that in the conceptual design that we don't get tied to physical constraints early as they will inhibit creativity and limit our options.

Summary:

- Different kinds of prototyping are used for different purposes and at different stages
- Prototypes answer questions, so prototype appropriately
- Construction: the final product must be engineered appropriately
- Conceptual design (the first step of design)
- Physical design: e.g. menus, icons, screen design, information display
- Prototypes and scenarios are used throughout design

(top)

Chapter 9: User-Centered Approaches to Interaction Design

The main aims of this chapter are to:

- *Explain some advantages of involving users in development.*
- *Explain the main principles of a usercentered approach.*
- *Describe some ethnographicbased methods aimed at understanding users' work.*
- *Describe some participative design techniques that help users take an active part in design decisions. (users as co-designers will raise acceptance of product)*

Why involve users at all?

1. To manage their *expectations:* no surprises, communicate their expectations, get a common set of realistic expectations

2. So users feel *ownership:* by making users active stakeholders, they are more likely to forgive / accept problems, and are more likely to accept the final product

Degrees of user involvement:

- **member of the design team** (*part time vs. full time:* degree of input / time and contact; *short term vs. long term:* degree of consistency across project life. Long term members might loose contact with users)
- **newsletters / e-mail / etc** (disseminate information to a large selection of users, but requires 2-way communication, not 1-way)
- Actual user involvement may be a combination of the above two ways.
- Microsoft involves users by 'activity based planning' (studying users doing tasks), usability tests, internal developer usage of products, and customer support lines.

What is a user-centered approach? User-centered approach is based on:

- *Early focus on users and tasks:* directly studying cognitive, behavioral, anthropomorphic & attitudinal characteristics
  - users' tasks and goals are the driving force behind the development
  - users' behavior and context of use are studied and the product is designed to support them
  - users' characteristics are captured and designed for
  - users are consulted throughout development, from earliest phases to the latest, and their input is seriously taken into account
  - all design decisions are taken within the context of the user, their work, and their environment

- *Empirical measurement:* users' reactions and performance to scenarios, manuals, simulations & prototypes are observed, recorded and analyzed

- *Iterative design:* when problems are found in user testing, fix them and carry out more tests

Understanding Users' Work: Ethnography

- ***Ethnography*** stems from anthropology, and literally means 'writing the culture' - a form of participant observation. However, it is difficult to use the output of ethnography in design. Design is concerned with abstraction and rationalization, while ethnography is concerned with minute details, so it is difficult to harness the data gathered from ethnography so that it can be used in design.
- *Framework for using ethnography in design:*
  - *distributed coordination:* distributed nature of the tasks / activities and the means / mechanisms by which they are coordinated
  - *plans and procedures:* organizational support for the work, such as workflow models and organizational charts, and how these are used to support the work
  - *awareness of work:* how people keep themselves aware of others' work

- ***Coherence:*** a method offering questions to address these dimensions (above) by presenting the ethnographic study data as a set of *"viewpoints"* and *"concerns"*

  Examples:
  Distributed coordination: How is the division of labor manifested through the work of individuals and its coordination with others?
  Plans and procedures: How do plans and procedures function in the workplace?
  Awareness of work: How does the spatial organization of the workplace facilitate interaction between workers and with the objects they use?

Contextual Design: developed to handle data collection and analysis from fieldwork for developing a software-based product (used commercially quite widely) There are seven parts to Contextual Design:

1. **Contextual Inquiry**
2. **Work Modeling**
3. **Consolidation**
4. **Work Re-design**
5. **User Environment Design**
6. **Mock-up and test with customers**
7. **Putting it into Practice**

1. Contextual Inquiry: an approach to ethnographic study where the user is an expert, and the designer is an apprentice. It is a form of interviewing, but takes place at the users' workplace / workstation, and is often 2-3 hours long. Four main principles of contextual inquiry are:

1. *Context:* see workplace and what happens
2. *Partnership:* user and developer collaborate
3. *Interpretation:* observations interpreted by user and developer together
4. *Focus:* project focus to help understand what to look for

2. Work Modeling: In interpretation sessions, models are drawn from the observations. Five models are:

- **Work flow model:** the people, communication and coordination
- **Sequence model:** detailed work steps to achieve a goal
- **Artifact model:** the physical 'things' created to do the work
- **Cultural model:** constraints on the system from organizational culture
- **Physical model:** physical structure of the work, e.g. office layout

3. Consolidation: each contextual inquiry (one for each user / developer pair) results in a set of models, which need to be consolidated into one view of 'the work'

- **Affinity Diagram:** organizes interpretation session notes into common structures and themes

- Categories arise from the data
- Diagram is built through induction

- Work models consolidated into one of each type

Participatory Design: (Scandinavian background) emphasizes social and organizational aspects - based on study, model-building and analysis of new and potential future systems. Aspects to user involvement include:

- Who will represent the user community? Interaction may need to be assisted by a facilitator
- Shared representations
- Co-design using simple tools such as paper or video scenarios
- Designers and users communicate about proposed designs
- Cooperative evaluation such as assessment of prototypes

Benefits of Participatory Design: "Computer-based systems that are poorly suited to how people actually work impose cost not only on the organization in terms of low productivity but also on the people who work with them. Studies of work in computer-intensive workplaces have pointed to a host of serious problems that can be caused by job design that is insensitive to the nature of the work being performed, or to the needs of human beings in an automated workplace." [Kuhn, S. in Bringing Design to Software, 1996]

PICTIVE: Plastic Interface for Collaborative Technology Initiatives through Video Exploration: Intended to empower users to act as full participants in design

Materials used are:

- Low-fidelity office items such as pens, paper, sticky notes
- Collection of (plastic) design objects for screen and window layouts

Equipment required:

- Shared design surface, e.g. table
- Video recording equipment

Before a PICTIVE session:

- Users generate scenarios of use
- Developers produce design elements for the design session

A PICTIVE session has four parts:

- Stakeholders all introduce themselves
- Brief tutorials about areas represented in the session (optional
- Brainstorming of ideas for the design
- Walkthrough of the design and summary of decisions made

CARD: Collaborative Analysis of Requirements and Design - Similar to PICTIVE but at a higher level of abstraction: explores work flow not detailed screen design

- Uses playing cards with pictures of computers and screen dumps
  Similar structure to the session as for PICTIVE
- PICTIVE and CARD can be used together to give complementary views of a design

Summary:

- User involvement helps manage users' expectations & feelings of ownership
- A user-centered approach has three main elements: early focus on users, empirical measurement and iterative design
- Ethnography is useful for understanding work, but can be difficult to use in design
- Coherence and Contextual Design support the use of ethnographic data in design
- Participative design involves users taking an active part in design decisions

CARD and PICTIVE are example techniques

Exercise:

This exercise is to be done in pairs.

Consider a website application for booking theatre or cinema tickets online

(a) Think about how you would design such a site, and sketch out some ideas
(b) Run a CARD session with a colleague acting as a 'user' to map out the functional flow of the website
(c) Ask your colleague to produce some scenarios of how the system may be used. Meanwhile, prepare some 'empty' templates for a PICTIVE session for this system, using paper, sticky notes and pens
(d) Run a PICTIVE session to develop the online booking system collaboratively, using PICTIVE.

(top)

Chapter 10: Introducing Evaluation

What, Why and When to Evaluate...

Goals of this chapter:

- discuss how developers cope with real-world constraints
- explain the concepts and terms used to discuss evaluation
- examine how different techniques are used at different stages of development

What to evaluate:

iterative design and evaluation is a continuous process that examines: early ideas for a conceptual model; early prototypes of the new system; later, more complete prototypes

Designers need to check that they understand users' requirements

Why evaluate:

- because user experience can be extremely important for a product's success

- because the cycle of **design** and **testing** is the only validated methodology in existence that will consistently produce successful results

Five good reasons for investing in user testing (Tognazzini):
- fixed problems before the product is shipped
- the team can concentrate on real problems
- Engineers code instead of debating
- Reduced time to market
- sell without bothering to release patches

When to evaluate:

- when it's brand new: develop markups of the product to elicit reactions from the potential users
- upgrades: compare user performance & attitude w/ previous versions

Two main types of Evaluation: ***Formative Evaluation and Summative Evaluation***

1. ***Formative Evaluation:*** *done at different stages of development to check that the product meets users needs*
2. ***Summative Evaluation:*** *assess the quality of a finished product*

it is best to evaluate throughout design- from the first descriptions / sketches, all the way to the final product

Design proceeds through iterative cycles of design / test / re-design - where evaluation is a key ingredient for a successful design

(top)

Chapter 11: An Evaluation Framework

Goals of this chapter:

- Explain key evaluation concepts and terms
- Describe the evaluation paradigms & techniques used in interaction design
- Discuss the conceptual, practical and ethical issues that must be considered when planning evaluations
- Introduce the DECIDE framework

User studies focus on how people behave, *in their natural environments*, or *in the laboratory,* with *old technologies and with new ones*.

Evaluation Paradigms:

- any kind of evaluation is guided by a set of beliefs, implicitly or explicitly. These beliefs are often supported by a theory. The beliefs and the methods associated with them are called ***evaluation paradigms***.
- There are four main evaluation paradigms discussed:
  - **Quick & Dirty:**
    - this is the common practice by which designers get feedback from users, to confirm that they are in line with users' needs and are liked. Emphasis is on fast input to the design process rather than carefully documented findings.
  - **Usability Testing:**

- records typical users' performance on typical tasks in controlled settings. As the users perform the tasks they are watched and recorded on video, with keypresses logged. Data is used to calculate performance times, identify errors and help explain why the users did what they did.
- User satisfaction questionnaires & interviews are used to gather users' opinions.

- **Field Studies:**
  - done in natural settings- aims to understand what users do naturally and how technology impacts them. This can help to:
    - identify opportunities for new technology
    - determine design requirements
    - decide how best to introduce new technology
    - evaluate technology in use
- **Predictive Evaluation:**
  - experts apply their knowledge of typical users to predict usability problems
  - can involve heuristics, and theoretically based models
  - users do not need to be present, and this method is cheap and quick to do
  - GOMS / KLM / Fitt's Law???

Evaluation Techniques:

Observing users (ch. 12)

- can help identify needs, which can lead to new products
- can help evaluate prototypes
- ways in which to record observations: notes, audio, video, interaction logs
- challenges: how can we observe others without disturbing them? how will we analyze the data gathered?

Asking users their opinions (ch. 13)

- Techniques: Interviews and Questionnaires
- Asking experts their opinions is inexpensive and quick

Testing users' performance (ch. 14)

- ways to measure user performance to compare 2 or more designs
- modeling users' task performance to predict the efficacy and problems of a user interface
- some of these techniques: GOMS and the keystroke model

DECIDE: A framework to guide evaluation

Determine the goals the evaluation addresses (what, who, why...)

Explore the specific questions to be addressed (break down into subquestions- e.g. consumers' attitudes, security, interface, reputation of system, trust, adequate access)

Choose the evaluation paradigm and techniques to answer the questions (paradigm selected determines the techniques used. Practical issues, ethical issues, and tradeoffs must be considered)

Identify the practical issues (select users, stay on budget, stay on schedule, find evaluators, select equipment)

Decide how to deal with the ethical issues (informed consent form, privacy / confidentiality, and let the participants know the goals of the study, what will happen to the findings, privacy of information, etc..)

Evaluate, interpret, and present the data (what data to collect, how to analyze and present depends on the paradigm used. Need to consider reliability, validity, biases, scope and ecological validity)

Pilot Studies:

- these are a small trial of the main study. They can help make sure the study is viable.
- pilot studies check that you can conduct the procedure, and that your interview skills, questionnaire questions, and experiment procedure works properly
- they can identify potential problems and is useful for ironing out problems before doing the main study
- can use colleagues if you can't spare real users

(top)

Chapter 12: Observing Users

The goals of this chapter:

- how to select appropriate observation techniques
- how to do observation
- how to collect, analyze the data, and present findings from it

Goals and questions: provide a focus for observation

Paradigms: guide all evaluation studies

What & When to observe:

observing is useful at any time during product development
- early on in design: helps designers understand users' needs
- later on: to examine whether the developing prototype meets users' needs

evaluators can be an onlooker, a participant or an ethnographer

Approaches to Observation:

- ***Quick & Dirty observation:*** can occur anywhere, anytime. Good for immediate feedback. Evaluators can temporarily join a group to observe.
- ***Observation in Usability Testing:*** video and interaction logs (keystrokes, mouse clicks, conversations). Observers can watch through one-way mirror or on remote TV screen.
- ***Observation in Field Studies:*** can observe on any of four levels: complete participation, marginal participation, observers who also participate, observers who do not participate

How to observe:

- *In controlled environments:* decide where to set up equipment, test the equipment, provide an informed consent form. Cons: observer doesn't know what users are thinking
  - to overcome this: the ***think-aloud technique*** allows us to observe the users' problem solving strategies by requiring them to say out loud everything they are thinking and trying to do
- *In the field:* different framework to structure and focus observation and data collection activity (the person, the place, the thing; who, what, where, why, how; space, actors, activities, objectives, acts, events, goals, feelings)
- *Participant observation and ethnography:* follows the same guidelines as above, but the observer must be accepted into the group (gain their trust by offering them the results, etc). ***Ethnographic study allows multiple interpretations of reality: it is interpretivist.*** *Often data collection and analysis happen simultaneously in ethnographic study, through participant observation and interviews by immersing the observer in the users' culture.*

Data collection:

- Notes + still camera
- Audio recording + still camera
  - helps provide a visual record, but transcribing the data can be a pain in the butt
- Video

- gets audio and visual data, but can be intrusive. Also, it's easy to miss stuff (outside the camera's viewpoint). This too can be very time consuming to process the data

Indirect Observation: Tracking users' activities

Techniques include:

- Diaries: good when users are scattered and unreachable in person
  - pros: inexpensive, require no special equipment or expertise
  - are good for long term studies
- Interaction logging (key presses, mouse / device movements)
  - used in usability testing
  - typically synchronized with video and audio logs
  - good for being unobtrusive and for logging large amounts of data automatically

Analyzing, Interpreting and Presenting the Data

There are three types of data:

1. **qualitative analysis to tell a story**
   1. review data and identify key themes
   2. record themes in a coherent / flexible form
   3. recode the data and time of data analysis session
   4. check your understanding with the people you observe
   5. iterate process until your story represents what you observed
   6. report findings to development team (oral or written report)

   analyzing and reporting ethnographic data (from participant observation, interviews, artifacts:

   1. look for key events

2. look for patterns of behavior (in various situations and players)
3. compare sources of data
4. report findings

2. **qualitative analysis for categorization**
    1. look for incident patterns
    2. analyze data into categories (content analysis)
    3. determine the content categories reliability and inter-research reliability ratings
    4. analyze discourse (conversation analysis)

3. **quantitative data analysis**
    1. video data collected in usability labs
    2. annotated
    3. recording to calculate performance times
    4. analyze statistically

Finally, feed the findings back into design!

Summary

- it is very valuable to be able to observe users in the field to see how technology is used in context
- this observation can confirm ideas and offer possibilities to explore new design ideas
- the way that observational data is collected and analyzed depends on the paradigm in which it is used: quick & dirty, user testing, or field studies.
- various amounts of control, intervention, and involvement are possible when observing:
  lab studies / usability testing <----------> participant observation / ethnography
  - on one end, lab studies offer a strongly controlled environment with little evaluator involvement
  - on the other end participant observation and ethnography require deeper involvement with users

and understanding of context
- Diaries and data-logging techniques provide a way to track user activity without intruding

(top)

Chapter 13: Asking Users and Experts

Interviews and Questionnaires are used in "quick and dirty" evaluation, in usability testing, and in field studies to ask about *facts, behavior, beliefs and attitudes.*

Interviews

- 4 types: can be **open-ended (unstructured), structured, semi-structured, or group interviews**
  - ***unstructured:*** no script, not replicable
  - ***structured:*** tightly scripted like a questionnaire
  - ***semi-structured:*** guided by script but open for deeper exploration if desired
  - ***group:*** often a 'focus group' of 3-10 people, consensus reached on questions
  - *data analysis on structured interviews is like a questionnaire, while unstructured like participant observation*

- 2 types of questions: open and closed. Closed require interviewee to choose between options, open allow for a free-range response.

Questionnaires

- can use: yes / no; Likert scale; semantic scale; open-ended responses on questions
- to make a good questionnaire, provide a clear purpose statement, plan the questions, decide if phrases have a positive or negative connotation, pilot test your questions, and decide how the data will be analyzed

- to reach a large amount of people: guarantee anonymity, offer online (large base / instant results & often instant data analysis)
- Be careful- online questionnaires don't prevent people from answering multiple times, and can have a low response rate
- *data analysis includes identifying trends, using simple statistics, making use of percentages & bar graphs*

Heuristic Evaluation

- when cost of accessing users is too much, use expert inspections or heuristics to analyze usability, etc.
- For example: ***Nielsen's heuristics***
  - *Visibility of system status*
  - *Match between system and real world*
  - *User control and freedom*
  - *Consistency and standards*
  - *Help users recognize, diagnose and recover from errors*
  - *Error prevention*
  - *Recognition rather than recall*
  - *Flexibility and efficiency of use*
  - *Aesthetic an minimalist design*
  - *Help and documentation*
- ***Heuristic evaluation is referred to as 'discount evaluation' because evidence supports the fact that 5 evaluators can detect 75% of the usability problems! It is inexpensive and quick.***

- Three stages of heuristic evaluation:
  1. briefing session telling experts what to do
  2. evaluation period of 1-2 hours where each expert works separately to get a feel for the product,

and then to focus on specific features

3. a debriefing session where the experts work together to prioritize problems

- Pros: quick and cheap evaluation, diagnosis of problems
- Cons: can be hard to find experts, important problems can get missed, often trivial problems get identified

- Note: different combinations and types of heuristics are needed to evaluate different types of applications and products.

Walkthroughs

- walkthroughs are an alternative to heuristic evaluation and involve 'walking through' a task with the system an noting problematic usability features. often these don't involve users.
- Two main types: **Cognitive Walkthroughs** and **Pluralistic Walkthroughs**

- **Cognitive Walkthroughs** simulate a users' problem solving process at each step in the human-computer dialog, checking to see if the users' goals and memory for actions can be assumed to lead to the next correct action.
  - focus on ease of learning
  - designer presents an aspect of the design and usage scenarios
  - one of many experts walk through the design prototype with the scenario
  - expert is told the assumptions about user population, context of use, task details
  - Expert is guided by 3 questions:
    - Will the correct action be sufficiently evident to the user?
    - Will the user notice that the correct action is available?
    - Will the user associate and interpret the response from the action correctly?
  - As the experts work through the scenario they note problems (focus on users' problem in

details)

- **Pluralistic Walkthroughs** are where developers and usability experts work together to step through scenarios and discussing usability issues associated with dialog elements involved in the scenario steps. Each group of experts are asked to assume the role of typical users.

    1. Scenarios are developed as a series of hard-copy screens representing a single path through the interface
    2. The scenarios are presented to the group of evaluators
    3. When everyone writes down actions, the panelists discuss the actions for the review
    4. The panel moves on to the next round of screens

    - Pluralistic Walkthroughs produce a set of quantitative data which lends itself well to participatory design practices by involving a multidisciplinary team in which users / users' tasks play a key role

- Walkthroughs are very focused and are therefore suitable for evaluating small parts of systems

Summary

- Interviews can be structured, semi-structured, or unstructured. Structured and semi-structured are designed to be replicated. Questions can be open or closed (format)

- Focus groups are a type of group interview.

- Questionnaires are a cheap and easy way to reach large numbers of people.

- typically 5 experts can find 75% of the usability problems.

- heuristic evaluation is cheaper and more flexible than user testing.

- User testing and heuristic evaluation often reveal different usability problems.

- pluralistic and cognitive walkthroughs are focused and good for evaluating a small part of the interface.

(top)

Chapter 14: Testing and Modeling Users

A central part of Interaction design is user testing.

Usability testing uses a combination of techniques, including user testing and user satisfaction questionnaires. User testing is of central concern.

The end of this chapter talks about GOMS (Goals, Operators, Methods, Selection rules), KLM (Keystroke Level Model) and Fitt's Law.

User Testing

- User testing is applied experimentation in which developers check that the system being developed is usable by the intended user population for their tasks.
- User testing tests typical users, measuring their typical task time, and the number and type of errors are recorded
- Can consist of completion time, observational data, answers to questionnaires, answers from Interviews, and keystroke logs.
- User testing is a systematic approach to evaluating user performance to inform or improve usability design
- Usually there are few participants (5-10), but can be 1-2 in "quick and dirty" for quick feedback
- Typically we record: task completion time; task completion after being away from the product; number / types of errors; errors per unit of time; number of navigations to help manuals; number of users making a particular error; number of users completing a task successfully

Using the DECIDE Framework for User Testing

- Determine the goals & Explore the questions
- Choose the paradigm and techniques
- Identify the practical issues - design typical tasks (typical tasks, typical users, testing conditions, how to run the test, contingency plans if users take too long) and record what's mentioned above
- Deal with ethical issues (make an informed consent form)
- Evaluate, analyze and present the data

Using Experiments for Usability Evaluation

- can have one (control condition vs. experimental condition) or 2+ (test multiple conditions, so break users into groups) independent variables.
- Random participant allocation not always best. Can attempt to group users by expertise, then balance them across conditions. However, this can cause problems if users are not assessed properly / exactly equal (ordering effects)

Design	**Advantages**	**Disadvantages**
Different Participants	No order effects	Many participants needed. Individual differences between participants is a problem. Can be offset to some extent by randomly assigning to groups.
Same participants	Eliminates individual differences between experimental conditions.	Need to counterbalance to avoid ordering effects.
Matched participants	Same as different participants, but the effects of individual differences are reduced.	Can never be sure that subjects are matched across variables.

Predictive Models

- Predictive models provide a way of evaluating products or designs **without directly involving users.**
- The usefulness of predictive models is limited to **systems with predictable tasks** (answering machines, etc.)
- Predictive models are based on **expert behavior**

1. GOMS: Goals, Operators, Methods, Selection rules (Card, et. al 1983)

- see Carroll Ch. 4 for more information

- GOMS aims to model knowledge and cognitive processes when users interact with a system.
- It is good for evaluating efficiency between two ways of doing things.
- This is the most well-known predictive modeling technique
- See example, p. 450

Goals: the state the user wants to achieve
Operators: the cognitive processes and physical actions performed to attain those goals (decide which search engine to use, think up a keyword. The goal is obtained; an OPERATOR is executed)
Methods: learned procedures for accomplishing the goals, steps to do so (drag mouse over field, type in keywords, press 'Go' button)
Selection rules: determine which method to select when there is more than one available

2. The Keystroke Level Model (KLM) (Card et. al, 1983)

- differs from GOMS in that it **provides actual numerical predictions of user performance**
- **KLM** is good for **comparing task times between two different strategies**
- can come up with average times to do something from empirical studies of actual user performance
- this model allows for predictions to be made about how long it takes an expert user to perform a task. The predicted time is computed by describing the sequence of actions involved in the task and summing their approximate times (looked up from empirical data):

  T(execute) = Tk + Tp + + Th + Td + Tm + Tr

Operators: K (keystroke); P (pointing); H (homing); D (drawing); M (mental preparation); R (system response time)

- for more explanation, see summary from Carroll book (Ch. 4)

Pros and Cons of GOMS:

Pros: allows for comparative analysis for different interfaces or computer systems relatively easily
outcome: counter-intuitive, help make decisions about the effectiveness of new products

Cons: not often used for evaluation purposes because of its **highly limited scope**
only good for predicting **expert performance**, and **error is not modeled** (average users not predicted)
many unpredictable factors come into play

A Con of Predictive Models: they can make predictions about predictable behavior, but it is difficult to use them as a way of evaluating how systems will be used in the real world. They are only useful for comparing the efficiency of different methods in completing a short, simple task.

3. Fitt's Law (Paul Fitts, 1954)

- used for *evaluating systems* where the **time to physically locate an object** is critical to the task at hand
- the law predicts that the time to point to an object is a function of the distance from the target object and the object's size (derived from Shannon: amplitude and noise)
- originally the law spoke of the speed and accuracy when moving towards an object on a display. In Interaction Design, it is used to describe the **time it takes to point at a target**, based on the **size of the object and the distance to the object**.
- The further away the object, and the smaller the size, the longer it takes to locate it and point to it.
- Fitt's law predicts that **the most quickly accessed targets on any computer display are the four corners of the screen**
- Fitt's law is ***useful for evaluating systems for which the time to locate an object is critical*** -

such as handheld devices like mobile phones.

Summary

- user testing is at the CORE of usability testing
- GOMS, Keystroke level model & Fitt's law are **used to predict expert, error free performance**
- ***Predictive models are used to evaluate systems with limited, clearly-defined functionality & predictable tasks***

(top)

Chapter 15: Design and Evaluation in the Real World: Communicators and Advisory Systems

Key Issues:

- user-centered approaches to interaction design involve iterative cycles of **design-evaluate-redesign** as development progresses from initial ideas through various prototypes to the final product
- a multitude of questions, concerns and decisions come up throughout system design projects (to be good at system design: must be good at working through the cycles and be good at multitasking, decision-making, team work and firefighting)

From Requirements to Design:

- which design cycle to use?
- which combination of methods to use when designing and evaluating your product?
- what happens when the product being developed is confidential and there are no users available to test it?
- how many users should be involved in tests?

- what should we do with the evaluation findings?
- how much should we expect from users?

Case study: Nokia's mobile communicator

- used ethnographic research and did scenarios and task models to get requirements
- (method) followed participatory design- involved users throughout
- (method) used interface metaphors
- (method) followed with frequent low fidelity prototypes based on alternative designs / immediate evaluation (yielded invaluable insight to designers)
- wrote usage scenarios (high level descriptions of device in use)
- user testing (usability tests): did summative testing before release (at end, not throughout [formative]) and questionnaires after release

Case study: redesign of a telephone response information system (TRIS)

- current system was very hard to use- a deep menu system over the phone- users can't remember it without cues
- GOMS / KLM used to show how interface supported users' tasks
- heuristic evaluation used as an alternative method for showing usability problems (expert review of system)
- methods complemented each other and showed benefit of doing a re-design

Key points:

- **Design involves trade-offs that can limit choices but can also result in exciting design challenges**
- Prototypes can be used for a variety of purposes throughout development, including for marketing presentations and evaluations
- The design space for making upgrades to existing systems is limited by the design decisions previous system. The design space for new products is much greater.

- Cycles of rapid prototyping and evaluation allow designers to examine alternatives in a short time
- Simulations are useful when evaluating systems used by large numbers of people when it is not feasible for them to work on the system directly
- **Piecing together evidence from data from a variety of sources can provide a rich picture of usability problems, why they occur, and possible ways of fixing them.**

(top)
