INF320 notes

Chapter 1 – What is interaction design?

The differences between good and poor interaction design

Bad design:

- Can require you to carry out a number of steps for basic tasks

Good design:

- Aesthetically pleasing and enjoyable to use

Interaction design - how it relates to HCI and other fields

Interaction design:

- Designing interactive products to support people in their everyday and working lives

HCI:

- An interdisciplinary field concerned with design, evaluation, and implementation of interactive computing systems for humans

Disciplines involved in interaction design:

- Psychologists & sociologists (to understand how users act and react to events and communicate together)
- Graphic designers, artists, film experts (for designing different kinds of interactive media effectively)
- Educational technologists & training experts (for interactive learning environments and educational software)

Fields related to interaction design:

- HCI
- Human factors
- Cognitive ergonomics
- Cognitive engineering
- Information systems
- CSCW (computer-supported cooperative work)
- (fields concerned with designing systems to match users' goals)

Advantage of multiple disciplines:

- More ideas are generated so more creative designs are produced

Disadvantage of multiple disciplines:

- The more people with different backgrounds in a team, the more difficult it can be to communicate and progress forward

What is involved in the process of interaction design

Four basic activities of design:

1. Identify needs and establish <u>requirements</u>
2. Develop alternative <u>designs</u> that meet those requirements
3. <u>Build</u> interactive versions of the designs so that they can be communicated and assessed
4. <u>Evaluate</u> what is being built throughout the process

Three key characteristics of design:

1. Users should be involved through 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 is inevitable

Usability and the goals of interaction design

Usability goals:

1. Effectiveness
How good a system is at doing what it's supposed to do
2. Efficiency
The way a system supports users in carrying out their tasks
E.g. Common tasks should need a single key press
3. Safety
Protecting the user from dangerous conditions and bad situations
E.g. Remote interaction with x-ray / chemical plant conditions
E.g. Help users avoid carrying out unwanted actions accidentally:
    - Prevent the user from making serious errors by reducing the risk of wrong keys being mistakenly activated
    - Provide users with various means of recovery
4. Utility
The extent to which the system provides the right kind of functionality so that users can do what they need to do
5. Learnability
How easy a system is to learn to use
People want to learn everyday & infrequently-used products quickly
People spend longer learning complex things with more functionality
The 10-minute rule: Novices should be able to learn how to use a non-complex system in under 10 minutes
6. Memorability
How easy a system is to remember how to use, once learned
Important for systems that are used infrequently
Use meaningful icons, command names, and menu options to help users remember the sequence of operations at different stages of a task
Structure options and icons so they are placed in relevant categories of options so the user remembers where to find them

Usability criteria:

- *Specific* objectives that enable the usability of a product to be assessed in terms of how it can improve a user's performance
- E.g. Time to complete a task (efficiency)
- E.g. Time to learn a task (learnability)
- E.g. The number of errors when performing a task (memorability)

User experience goals:

- Concerned with how the user experiences an interactive product from their perspective (rather than assessing how productive / useful a system is from its own perspective: *usability* goals)

- Interactive products can be designed to be fun, satisfying, entertaining, motivating, rewarding, helpful…
- Some tasks may require more effort, but still be more fun
- Designers must understand the trade-offs between usability and user experience goals and find a good combination

The different forms of guidance used in interaction design

Design and usability principles:

- Help designers explain and improve the design (but don't specify how to design an actual interface)

Visibility

- If functions are visible, users will know what to do

Feedback

- Information is sent back about what has been accomplished, allowing the person to continue with the activity

Constraints

- Restrict what the user can do at a given moment
    - *Physical constraints*
      E.g. A disk can go into a drive in only one way
    - *Logical constraints*
      E.g. Disabling menu options when not appropriate
    - *Cultural constraints*
      E.g. Windows for information and icons for operations

Mapping

- The relationship between controls and their effects
- E.g. the up & down arrows on the keyboard

Consistency

- Interfaces must have similar operations for similar tasks
- Designers must decide what to be consistent with

Affordance

- An attribute of an object that lets people know how to use it
    - *Real affordances*
      You can see without learning what to do to real objects
    - *Perceived affordances*
      Screen-based interfaces are learned conventions
- Don't try to design for real affordances at the interface

Heuristics and usability principles:

- Heuristics = design principles used in practice, interpreted in the design context and drawn from past experience
- Usability principles = like design principles, but they are more rule-based
- Design principles are used mainly for informing a design
- Usability principles are used mainly for evaluating prototypes

Nielsen's 10 usability principles:

1. Visibility of system status
   Users must always know what's going on, through feedback

2. Match between system and the real world
   Speak the users’ language – don’t use system-oriented terms
3. User control and freedom
   Users must be able to easily escape from places
4. Consistency and standards
   Don’t make users wonder if different words mean the same thing
5. Help users recognize, diagnose, and recover from errors
   Use plain language to describe the problem and how to solve it
6. Error prevention
   Where possible, prevent errors occurring in the first place
7. Recognition rather than recall
   Make objects, actions, and options visible
8. Flexibility and efficiency of use
   Provide accelerators for experts that are invisible to novices
9. Aesthetic and minimalist design
   Avoid using information that is irrelevant or rarely needed
10. Help and documentation
   Provide help in a set of steps that can easily be followed

- **Rules** = guidelines that should be followed
- They are derived from design & usability principles, but are more specific
- E.g. “Always place the exit button at the bottom of the 1st menu”

Ch 2 – Understanding & conceptualizing interaction

The problem space

- First understand the problem space, *then* make design decisions
- Clarify your usability & user experience goals
- This involves making explicit your implicit *assumptions* & claims
- Reason through your assumptions about why something might be a good idea to see the pros and cons of your proposed design
- Questions that provide a useful framework for the problem space:
  - Are there problems with the existing product?
  - Why do you think your proposed ideas might be useful?
  - How will your design support people in their activities?
- Working through your assumptions for a problem space before building anything can highlight problematic concerns

Conceptualizing interaction design before building

- Having a good understanding of the problem space can also help greatly in formulating what it is you want to design
- Think about the overall structure of what will be built and how this will be conveyed to the users (In particular, this involves developing a conceptual model)

Conceptual models

Definition: A description of the system in terms of ideas and concepts about what it should do, behave and look like, that will be understandable by the users in the manner intended.

A. Conceptual models based on activities

1. Instructing

- Users type in / speak commands, select menu options…

2. Conversing

- Users speak / type questions and the system replies
- E.g. Search engines and help facilities…
- Advantage: Novices can interact with a system in a way they are already familiar with
- Disadvantages: Misunderstandings can arise, and some tasks can be cumbersome (e.g. if you have to listen to all the options)
- Animated agents are better than hidden ones because they look simple so users won't expect them to be too intelligent

3. Manipulating and navigating

- Users navigate through an environment of virtual objects
- The virtual environment shares properties of the physical world
- Virtual objects can be manipulated by moving, opening, zooming…
- Properties of direct manipulation interfaces:
  - Continuous representation of the objects and actions
  - Rapid reversible incremental actions with quick feedback

  - Physical actions & button pressing, instead of commands
- Benefits of direct manipulation interfaces:
  - Novices learn basic functionality rapidly
  - Expert users can work rapidly on a wide range of tasks
  - Infrequent users can remember how to do things over time
  - No need for error messages, except very rarely
  - Users can immediately see if their actions are correct
  - Users experience less anxiety
  - Users gain confidence and mastery and feel in control
- Disadvantages of direct manipulation:
  - Some people may take the conceptual model too literally
  - Not all tasks can be described by objects
  - Not all actions can be done directly

4. Exploring and browsing

- Information is structured to allow users to find out / learn

<u>B. Conceptual models based on objects</u>

- More specific than conceptual models based on activities
- Focus on the way a certain object is used in a certain context
- Often based on an analogy with something in the physical world
- E.g. the spreadsheet is based on the ledger sheet

<u>C. Mixing A and B</u>

- Which kind of conceptual model is best depends on the activity
- E.g. manipulation & navigation for a flight simulator
- Often a form of hybrid conceptual model is appropriate
- E.g. Using both the conversing and exploring models
- Disadvantage of mixing interaction modes: the underlying conceptual model can be more difficult to understand & learn

Interface metaphors

- Metaphor = “A thing considered representative of another thing”
- E.g. “The *evening* of one’s life”; “*Food* for thought”
- Interface metaphor = a conceptual model that has been developed to be similar in some way to *aspects* of a physical entity, but that also has its own behaviors and properties
- Such models can be based on an activity / object / both
- E.g. The desktop, spreadsheet, and search engine
- Interface metaphors are based on conceptual models that combine familiar knowledge with new concepts

<u>Advantages:</u>

- Provide users with a familiar orienting device, helping them understand and learn how to use a new system
- People find it easier to talk about what they are doing at the computer in terms familiar to them

<u>Disadvantages:</u>

- Breaking the rules

  - Cultural and logical contradictions can arise when the metaphor is instantiated as a GUI (E.g. the recycle bin should be under the desk, not on the desktop!)
- Too constraining
  - E.g. it is inefficient to make users scan through many files to find something; rather use the file's name
- Conflicts with design principles
  - If you make everything match the real world, you can end up with bad design solutions
- Not being able to understand the functions beyond the metaphor
  - Users who understand the interface metaphor well may find it hard to see what else can be done with the system
- Overly literal translation of existing bad designs
  - E.g. A virtual calculator designed to be like a physical one is even harder to use
- Prevents the designer from finding new paradigms and models
  - Designers may fixate on ideas based on well known technologies, that they know people are familiar with

From conceptual models to physical design

- Throughout the iterations of design,
  - consider whether the conceptual model being developed is working in the way intended, and
  - ensure that it is supporting the user's tasks
- With each iteration, *progress through* the design in more depth:
  - Pass 1: Identify initial user requirements
  - Pass 2: More extensive info gathering about users' needs
  - Pass 3: Think of possible appropriate conceptual models
  - Pass 4: Flesh out some conceptual models (Prototypes)
- Issues to address when developing & testing initial prototypes of conceptual models:
  - The way information is to be presented and interacted with
  - What combinations of media to use (E.g. sound + animation)
  - The kind of feedback that will be provided
  - What combinations of input & output devices to use
  - Whether to provide agents and in what format
  - Whether operations should be hardwired (activated through physical buttons) or represented on the screen
  - What kinds of help to provide and in what format
- Some issues to address concerning the actual physical design:
  - Information presentation
    - Which dialogs and interaction styles to use
      - E.g. form fill-ins, speech input, menus
    - How to structure items in graphical objects
      - E.g. Windows, dialog boxes and menus
  - Feedback
    - What navigation mechanisms to provide
  - Media combination

  - Which kinds of icons to use

The pros & cons of using realism versus abstraction at the interface

- Realism = giving the illusion of behaving & looking like real
  - Pro: Enables novices to feel comfortable with something new
  - Con: Expert users may feel less comfortable
- Abstraction = depicting only a few salient features
  - Pro: Often more efficient to use
  - Con: Can appear too computer-like & off-putting to novices

Ch 5 – Understanding how interfaces affect users

Affective aspects

- 'Affective' refers to producing an emotional response
- 'Affective computing' refers to computers recognizing and expressing emotions in the same way humans do

Expressive interfaces

- Expressive icons etc. can indicate the current computer state
- Ways of conveying the status of a system:
  - Dynamic icons (e.g. recycle bin expands when you fill it)
  - Animations (e.g. a flying bee to show the computer's busy)
  - Spoken messages (e.g. telling the user what to do)
  - Sounds indicating actions & events (e.g. email arriving)
- Benefit: provides reassuring feedback to the user and can be fun
- Interface agents are also a popular kind of expressive interface
- Emoticons convey emotional states & elicit emotional responses

User frustration

1. Gimmicks
   - Cause: Gimmicky display instead of meeting user expectations
   - Level of frustration: Mild
   - How to avoid / help reduce the frustration
     - E.g. Don't say your site is 'under construction'
2. Error messages
   - Cause: When a system / application crashes
   - Level of frustration: High
     - The use of cryptic language & jargon is a major factor
   - How to avoid / help reduce the frustration
     - State the cause of the problem and how to fix it
3. Overburdening the user
   - Cause: Upgrading software so that users are required to carry out excessive house-keeping tasks
   - Level of frustration: Medium to high
   - How to avoid / help reduce the frustration
     - Designers should consider the trade-offs incurred when introducing upgrades, especially the amount of relearning

4. Appearance
    - Cause: When the appearance of an interface is unpleasant
    - Level of frustration: Medium
    - How to avoid / help reduce the frustration
        - Interfaces should be simple, elegant, adhere to usability & graphic design principles, and ergonomic guidelines

- Dealing with user frustration
    - Error messages should be provided that explain what to do
    - Online help (tips, contextual advice) can also be provided

Applying anthropomorphism to interaction design

- Anthropomorphism = attributing human qualities to objects

<u>Pros:</u>

- Can be fun and motivate people to carry out tasks
- Being addressed in the first person ("Hello X") is endearing and makes you feel more at ease and reduces anxiety
- Screen characters are pleasanter than cold dialog boxes
- Typing a question in plain English is more natural than keywords
- Computers that flatter & praise users have a positive affect
- People may interact more with a talking-face display

<u>Cons:</u>

- Anthropomorphic interfaces can be deceptive
- They can make you feel anxious, and feel inferior / stupid
- Computers with human qualities can be annoying & frustrating
- People tire of artificial screen characters and ignore them
- Increasing the 'humanness' can be counterproductive because people are mislead to believe that the computer is like a human

Agents

<u>Kinds of agents</u>

1. Synthetic characters

- 3D characters in video games, and appear as a $1^{st}/3^{rd}$ person agent
- Designed to be lifelike, exhibiting realistic human movements
- Appearance, facial expressions etc are important design concerns

2. Animated agents

- Similar to synthetic characters, but play a collaborating role
- Typically appear at the screen's side as tutors / wizards
- Most are designed to be cartoon-like, rather than human-like

3. Emotional agents

- Have a predefined personality with emotions you can manipulate
- Aim: to allow people to change the agent's mood & see the effect

4. Embodied conversational interface agents

- The most anthropomorphic because they emulate human conversation

<u>General design concerns</u>

- Believability of virtual characters

  - o Key aspect: match the mood of the character to its actions
- Appearance
  - o People prefer simple characters, rather than detailed ones
- Behavior
  - o Characters should be able to point out relevant objects on the screen, preferably by leading with their eyes
  - o Exaggerate the behavior to draw attention to the emotions
- Mode of interaction
  - o An artificial mode of interaction can be more believable
  - o E.g. Play prerecorded speech at certain choice points and limit the user's responses to selecting menu options

Ch 6 – The process of interaction design

What interaction design is about

- User-centered approach: users' concerns direct the development rather than technical concerns
- The simplest way to communicate the plan is with sketches
- Other approaches: natural language description, prototypes…

Four basic activities of interaction design

1. Identifying needs and establishing requirements
2. Developing alternative designs
   Two activities: Conceptual design & physical design
3. Building interactive versions of the designs
4. Evaluating designs

Three key characteristics of the interaction design process

1. User focus
2. Specific usability criteria
3. Iteration

Some practical issues

Questions that must be answered to be able to do interaction design:

- Who are the users?
  - o E.g. People who manage direct users, those who receive products from the system, those who test the system…
  - o Primary users: frequent hands-on users
  - o Secondary users: Occasional users / through an intermediary
  - o Tertiary users: affected by the introduction of the system
  - o Be aware of the wide net of stakeholders
- What do we mean by needs?
  - o You can't just ask people what they need because they don't necessarily know what is possible
  - o Rather understand the users, what they are trying to achieve, and how they achieve it currently
  - o If a product is a new invention, it can be difficult to identify the users and representative tasks for them
  - o In these circumstances, a good indication of future behavior is current / past behavior

- How do you generate alternative designs?
    - Innovations can arise from ideas from other applications
- How do you choose among alternative designs?
    - Use info gathered about users and their tasks
    - Externally visible features:
        - Measurable
        - E.g. Physical size & speed of a photocopier
    - Internal features:
        - Can’t be measured without dissecting it
        - E.g. Friction rating of a photocopier
    - In interaction design, we concentrate on the externally visible and measurable behavior
    - Choose between alternative designs by letting users interact with them and by discussing their experiences
    - Descriptions & diagrams were traditionally used, but they can’t capture the dynamics of behavior
    - Prototyping can test technical feasibility of a design
    - You can use quality as a basis on which to choose between two alternatives, but first understand what ‘quality’ means
    - E.g. ‘fast’ can mean different things to different people
    - Usability engineering = the process of writing down formal, verifiable and measurable usability criteria

Ch 7 – Identifying needs & establishing requirements

What, how, and why

1. What are we trying to achieve in this design activity?

- Two aims:
    - Understand as much as possible about the users, their work, and the context of that work. (“Identifying needs”)
    - Produce, from the needs identified, a set of stable requirements that form a sound basis for design

2. How can we achieve this?

- Gather data, interpret it, extract requirements from it…
- (The activities influence one another as the process iterates)

3. Why bother? The importance of getting it right

- Unclear requirements are the main cause of project failure

4. Why *establish* requirements?

- Requirements ‘elicitation’: Others must tell us the requirements
- Requirements ‘analysis’: Investigating the gathered requirements
- Requirements ‘engineering’: Develop requirements iteratively
- ‘Establishing’ requirements: Requirements arise from the data-gathering and interpretation activities and have been established from a sound understanding of the users’ needs

What are requirements?

- Definition: Statements about an intended product that specify what it should do / how it should perform

- One aim of the requirements activity is to make the requirements as specific, unambiguous, and clear as possible

Different kinds of requirements

- Functional requirements
  - Capture what the product should do
- Data requirements
  - Capture the type, volatility, size, persistence, accuracy, and value of the amounts of the required data
- Environmental requirements / context of use
  - The circumstances in which the product will operate:
  - Physical environment
    - How much lighting, noise, dust… is expected
    - Will users need protective clothing?
  - Social environment
    - Will data need to be shared?
  - Organizational environment
    - How good is user support likely to be?
    - Are there facilities for training?
    - How efficient is the communications infrastructure?
  - Technical environment
    - What technologies will the product run on?
    - What technological limitations might be relevant?
- User requirements
  - Novice
    - Requires step-by-step instructions
  - Expert
    - Requires flexible interaction with more control
  - Casual user
    - Like a novice, requires clear instructions and easily understood prompts & commands, like a series of menus
  - Frequent user
    - Important to provide short cuts like function keys
  - 'User profile' = the collection of attributes for a 'typical user'
  - Any one device may have a number of different user profiles
- Usability requirements
  - The usability goals & associated measures for a product

Data gathering

Data-gathering techniques

Technique	**Good for**	**Kind of data**	**Advantages**	**Dis-advantages**
Question-naires	Answering specific questions	Quantita-tive & qualita-tive data	Can reach many people with low resource	Design is crucial. Few responses. Responses may

				not be what you want
Inter-views	Exploring issues	Some quantita-tive but mostly qualita-tive data	Interviewer can guide interviewee. Encourages user & developer contact	Time consuming. Artificial environment intimidates interviewee
Focus groups & workshops	Collecting multiple viewpoints	Some quantita-tive but mostly qualita-tive data	Highlights conflict & consensus areas Encourages user & developer contact	Possibility of dominant characters
Natural-istic Obser-vation	Under-standing context of user activity	Qualita-tive	Observing work gives insights that other techniques can't give	Very time consuming. Huge amounts of data
Studying Document-tation	Learning about procedures regulations & standards	Quantita-tive	No time commitment from users required	Everyday working differs from documented procedures

<u>Choosing between techniques</u>

- Data-gathering techniques differ in two main respects:
  - o The amount of time the data gathering takes and the level of detail & risk associated with the findings
  - o The knowledge the analyst must have about basic cognitive processes
- Tasks can be classified along three scales:
  - o Is the task a set of sequential steps or is it a rapidly overlapping series of subtasks?
  - o Does the task involve high information content with complex visual displays / low info content with simple signals?
  - o Is the task intended to be performed by a layman without much training / by a skilled practitioner?
- When choosing between techniques you need to consider:
  - o The nature of the technique
  - o The knowledge required of the analyst
  - o The nature of the task to be studied
  - o The availability of stakeholders & other resources
  - o The kind of information you need

<u>Basic data-gathering guidelines</u>

- Focus on identifying the stakeholders' needs
  - o Study their existing behavior & support tools
  - o Look at other products, such as earlier releases

- Involve all the stakeholder groups
  - Make sure you get all the views of the right people
- Involving only one representative from each stakeholder group is not enough, especially if the group is large
- Use a combination of data gathering techniques
  - Different techniques yield different kinds of information
- Support the data-gathering sessions with suitable props
  - E.g. Task descriptions and prototypes
  - Props jog people's memories and can focus the discussion
- Run a pilot session if possible
  - This is particularly important for questionnaires
- Know what you're looking for and what you'd *really* like
- How you record the data is as important as the technique
  - Main options: video & audio recording, and note taking

Task description

Scenarios

- Scenario = an informal narrative description
- It describes human activities / tasks in a story that allows exploration and discussion of contexts, needs, and requirements
- It doesn't explicitly describe the use of software / technology
- Vocab of stakeholders is used, so they can participate fully
- Stakeholders can easily relate to telling stories
- The focus of stories is be about what the users want to achieve
- The level of detail varies
- Scenarios are often generated during workshop / interview sessions to help explain / discuss an aspect of the user's goals
- They can be used to imagine potential uses of a device as well as to capture existing behavior
- Scenarios are not intended to capture a full set of requirements, but are a very personalized 1-perspective account
- Capturing scenarios of existing behavior and goals helps in determining new scenarios & gathering data for new requirements

Ch 8 – Design, prototyping and construction

Prototyping and construction

What is a prototype?

- A limited representation of a design for users to interact with
- Can be anything from a paper-based storyboard to software

Why prototype?

- Test out technical feasibility of ideas
- Help designers choose between alternatives
- Clarify vague requirements
- Do user testing and evaluation

- Check that a certain design direction is compatible with the rest of the system development

Low-fidelity prototypes

- Don't look much like the final product
- Simple, cheap, and quick to produce and modify
- Good for early stages (conceptual design) when exploring ideas
- Never intended to be kept and integrated into the final product

Storyboarding
- A series of sketches shows how a user can perform a task
- Can bring more detail to the written scenario
- Offers stakeholders a chance to role-play with the prototype, interacting by stepping through the scenario

Sketching
- Devise symbols and icons for storyboard elements

Prototyping with index cards
- Commonly used when developing websites
- Each card represents one screen / one element of a task
- Users pretend to perform the task while going through cards

Wizard of Oz
- Assumes you have a software-based prototype
- The user interacts with the software on a computer connected to one where a human operator simulates responses

High-fidelity prototypes

- Use materials that you would expect to be in the final product
- Good for selling ideas to people and testing technical issues
- Problems:
  - Take long to build
  - Testers tend to comment on superficial aspects
  - Developers are reluctant to change something they have crafted for hours
  - A software prototype can set expectations too high
  - Just one bug can bring the testing to a halt

Type	**Advantages**	**Disadvantages**
Low-fidelity	Lower development cost	Limited error checking
	Evaluate multiple design concepts	Poor detailed specification to code to
	Useful communication device	Facilitator-driven
	Address screen layout issues	Limited utility after requirements established
	Useful for identifying market requirements	Limited usefulness for usability tests
	Proof-of-concept	Navigational and flow limitations
High-fidelity	Complete functionality	More expensive to develop
	Fully interactive	Time-consuming to create
	User-driven	Inefficient for proof-of-concept designs
	Clearly defines navigational	Not effective for

	scheme	requirements gathering
	Use for exploration and test	
	Look and feel of final product	
	Serves as a living specification	
	Marketing and sales tool	

<u>Compromises in prototyping</u>

- Common compromises: breadth of functionality versus depth
- Horizontal prototyping: many functions but with little detail
- Vertical prototyping: a lot of detail for only a few functions

<u>Construction</u>

- Prototypes are not necessarily rigorously tested
- Throwaway prototypes: the final product is built from scratch
- Evolutionary prototypes: they should be subjected to testing along the way because they evolve into the final product

Conceptual design

- = a description of the proposed system about what it should do
- The set of user tasks is the basis for designing this model
- Steep yourself in the data and try to empathize with users
- Key guiding principles of conceptual design:
  - Keep an open mind but never forget the users & context
  - Discuss ideas with other stakeholders as much as possible
  - Use low-fidelity prototyping to get rapid feedback
  - Iterate: To get a good idea, get lots of ideas

<u>Three perspectives for developing a conceptual model</u>

Which interaction mode?

- The interaction mode refers to how the user invokes actions
- It depends on the activities the user will engage in
- Most conceptual models will be modes based on a combination of activities and objects
- Conceptual models can be
  - Process-oriented
    - Support a process; no identifiable work products
  - Product-oriented
    - Clear, identifiable work products that users individually create, modify and maintain

Is there a suitable interface metaphor?

- Interface metaphors combine familiar knowledge with new knowledge in a way that will help users understand the system
- Three-step process for choosing a good interface metaphor:
  - Understand what the system will do
    - Identify functional requirements
    - Develop partial conceptual models and try them out
  - Understand which bits of the system would cause problems
    - Look at problematic / critical tasks & subtasks
  - Generate metaphors

- Look for metaphors in the users description of tasks

- When metaphors have been generated, they need to be evaluated:
    - How much structure does the metaphor provide?
    - How much of the metaphor is relevant to the problem?
    - Is the interface metaphor easy to represent?
    - Will your audience understand the metaphor?
    - How extensible is the metaphor?

Which interaction paradigm?

- Interaction paradigms include: the WIMP interface, ubiquitous computing, pervasive computing, wearable computing, tangible bits, attentive environments, and the Workaday World

Expanding the conceptual model

What functions will the product perform?

- How will the task be divided up between human & machine?
- Develop scenarios & (essential) use cases to clarify answers
- Task allocation = deciding what the system will do and what must be left for the user
- The trade-off between what to hand over to the device and what to keep in control of the user has cognitive implications and is linked to social aspects of collaboration
- What functions will the hardware perform and what will be left under software control?

How are the functions related to each other?

- Functions may be related temporally / performed in parallel
- They may also be related through categorizations
- The relationships between tasks may constrain use / indicate suitable task structures within the device

What information needs to be available?

- What data is required to perform a task and how is this data to be transformed by the system?

Using scenarios in conceptual design

- Scenarios can be used to explicate existing work situations, but are more commonly used for expressing imagined ones to help in conceptual design
- Four roles for scenarios:
    - A basis for the overall design
    - For technical implementation
    - A means of cooperation within design teams
    - A means of cooperation across professional boundaries
- Plus and minus scenarios capture the most positive and most negative consequences of a particular proposed solution

Using prototypes in conceptual design

- Prototypes allow evaluation of emerging ideas to take place

Ch 9: User-centered approaches to interaction design

Why it is important to involve users

- Expectation management
  - The process of making sure that the users' views and expectations of the new product are realistic
  - Purpose: to ensure that there are no surprises for users
  - Marketing of the arrival must not misrepresent the product
  - It is better to exceed users' expectations
  - Involving users helps with expectation management because they can see from an early stage what the capabilities are
  - Adequate and timely training is another technique for managing expectations
- Ownership
  - Users who are involved and feel they have contributed to development are more likely to feel a sense of ownership
  - Ownership means that users will be more receptive to it

Degrees of involvement

- User co-opted full time on the design team
  - Advantage: The user will become very familiar with the system and give consistent input
  - Disadvantage: If a project takes years they may lose touch with the rest of the user group → less valuable input
- User co-opted part time on the design team
  - Advantage: Consistent input to development, while remaining in touch with other users
  - Disadvantage: Can become stressful for the individuals
- Users may be kept informed through regular newsletters
  - They must be given a chance to feed into the development process through workshops or similar events
  - With thousands of users this may be the only option
- Compromise situation
  - Representatives from each group are co-opted on the team
  - Other users are involved through design workshops etc.

A user-centered approach

- Principles that lead to a useful and easy to use system:
  - Early focus on users and tasks
    - First understand who the users will be by directly studying their cognitive, behavioral, anthropomorphic, and attitudinal characteristics
    - This requires observing users doing their normal tasks, studying the nature of those tasks, and then involving users in the design process
  - Empirical measurement
    - Early in development, observe and measure the reactions & performance of intended users to printed scenarios, manuals, etc.
    - Later on, analyze their reactions & performance when they interact with simulations and prototypes
  - Iterative design

    - When problems are found in user testing, they are fixed and then more tests & observations are carried out to see the effects of the fixes
    - Design → test → measure → redesign
- 5 principles that expand on 'Early focus on users and tasks'
  1. Users' tasks and goals are the driving force behind the development
  2. Users' behavior and context of use are studied and then the system is designed to support them
  3. Users' characteristics are captured and designed for
  4. Users are consulted throughout, from earliest to latest phases, and their input is seriously taken into account
  5. All design decisions are taken within the context of the users, their work, and their environment

Chapter 10 – Introducing evaluation

What evaluation is

- The process of systematically collecting data that informs us about what it is like for a particular user / group to use a product for a particular task in a certain environment

Why evaluate

1. Problems are fixed before the product is shipped, not after
2. The team can concentrate on real problems, not imaginary ones
3. Engineers code instead of debating
4. Time to market is sharply reduced
5. Upon first release, your sales department has a rock-solid design it can sell without having to pepper their pitches with how it will all actually work in the release

- Usability testing involves measuring the performance of typical users on typical tasks

When to evaluate

- Formative evaluations are done during design to check that the product continues to meet users' needs
- Summative evaluations are done to assess the success of a finished product

Chapter 11 – An evaluation framework

Evaluation paradigms

- Definition: Beliefs & practices associated with evaluation
- Evaluation paradigms are often related to a particular discipline in that they strongly influence how people from the discipline think about evaluation
- Each paradigm has its own particular methods and techniques associated with it

1. Quick & dirty evaluation

- Designers informally get feedback from users / consultants to confirm that their ideas are in line with users' needs
- Can be done at any stage
- Emphasis on fast input rather than carefully documented findings
- Data collected is fed back into the design process as verbal / written notes, sketches, and anecdotes…
- Consultants can also use their knowledge to review software quickly and provide suggestions for improvement

2. Usability testing

- Involves measuring typical users' performance on carefully prepared tasks that are typical
- Users' performance is measured in terms of number of errors and time to complete the task
- User satisfaction questionnaires and interviews are also used
- Usability testing is strongly controlled by the evaluator
- Typically tests take place in laboratory-like conditions
- Quantifying users' performance is a dominant theme

3. Field studies

- Done in natural settings
- Can be used to:
  - Help identify opportunities for new technology
  - Determine requirements for design
  - Facilitate the introduction of technology
  - Evaluate technology
- Qualitative techniques like interviews, observation, and ethnography can be used
- Two overall approaches to field studies:
  - An outsider looking on
    - Qualitative techniques are used to collect the data, which may be analyzed qualitatively or quantitatively
  - An insider
    - Ethnography is a particular type of insider evaluation in which the aim is to explore the details of what happens in a particular social setting

4. Predictive evaluation

- Experts apply their knowledge of typical users, often guided by heuristics, to predict usability problems
- Users needn't be present, which makes the process quick & cheap

Evaluation techniques

1. Observing users

- Notes, audio, video, and interaction logs are well-known ways of recording observations

2. Asking users

- Interviews & questionnaires

3. Asking experts

- Experts step through tasks, role-playing users

<u>4. User testing</u>

- Tests are conducted in controlled settings

<u>5. Modeling users' task performance</u>

- GOMS and the keystroke model are the best known techniques

The DECIDE framework to guide evaluation

Determine the <u>goals</u>

- Examples of goals:
    - Identify the metaphor on which to base the design
    - Check to ensure that the final interface is consistent, etc

Explore the <u>questions</u>

- Identify questions to make the goals operational

Choose the <u>evaluation paradigm</u> and <u>techniques</u>

- The evaluation paradigm determines the techniques used
- Consider practical and ethical issues and make trade-offs

Identify the <u>practical issues</u>

- Users
    - They must be appropriate users
    - Consider how the users will be involved
- Facilities and equipment
    - E.g. spare film and batteries may be needed
- Schedule and budget constraints
    - Make the most of the available time and resources
- Expertise
    - Make sure you have the expertise needed

Decide how to deal with the <u>ethical issues</u>

- Tell participants the goals of the study and what to expect
- Explain that sensitive info that they disclose is confidential
- Make sure users know they can stop the evaluation at any time
- Pay users to create a formal relationship
- Don't use quotes that inadvertently reveal a person's identity
- Ask users' permission in advance to quote them

Evaluate, interpret, and present the <u>data</u>

- Reliability
    - The reliability of a technique is how well it produces the <u>same</u> results on separate occasions with the <u>same</u> conditions
- Validity
    - Validity is concerned with whether the evaluation technique measures what it is supposed to measure
- Biases
    - Bias occurs when the results are distorted
- Scope
    - Scope refers to how much the findings can be generalized
- Ecological validity
    - Concerns how the environment in which an evaluation is conducted influences / distorts the results

Pilot studies

- Aim: to make sure that the plan is viable before embarking on the real study