Human Computer Interaction Notes

Interaction Design (+Scenarios)

- Interaction Design is about creating user experiences that enhance and augment the way people work, communicate, and interact.1
- Interaction Design has a much wider scope than Human Computer Interaction. ID is concerned with the theory and practice of designing user experiences for any technology or system, whereas HCI has traditionally been focused on/surrounding humans and computers specifically.
- Interaction Design involves understanding the **requirements**.
- Requirements can be **functional** (what should it do) or **non-functional** (what are the constraints). The usability principles (heuristics) fit into the non-functional requirements.
- **User Profiles** are a set of persona's. A persona is a short description of a fictional user.
- Scenarios
  - Activity Scenario (narrative based on user's requirements)
    - Used at the beginning of the design process.
    - Who, When, What, Why
    - Not technical/no specific details (re: does not presuppose the interface)
    - From the users perspective
  - Use Case Scenario (narrative of how the user uses the interface to fulfil their goal)
    - Include the users goals but focus on the user/computer interaction (re: talk about the technology)
    - Basically is a description of the use case diagram.
    - Do these scenarios after you figure out the requirements
    - Different users would have different use cases, we can show this with a use case diagram which shows the actors and the various use case's that they encounter.
  - Task Scenario
    - Used when running a usability test. ie. give the participant a scenario before giving them a task to give them context.
  - When describing a scenario, give the users goals, their context and situation. Use a narrative form. Cooper et al. describe the process of interaction design as,

1. Identifying needs and establishing requirements for the user experience.
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 and the user experience it offers.

Scenarios are narratives about named people with an age. We need some background to understand where they are coming from (for instance their cultural background (eg. the US uses MM/DD/YYYY but Australia uses DD/MM/YYYY)). We try to remove incorrect assumptions about what we think a certain group of people are like. The scenario should explain their motivations and their goals.

Usability

Usability is all about producing things which are usable. Where something is usable when it meets these usability goals, however you should work out which goals are most important for the problem and focus on those first.

Usability Goals

- easy to learn
- easy to remember how to use
- effective to use
- efficient to use
- safe to use

- have good utility (providing the right kind of functionality to allow the user to achieve their goal)

User Experience Goals

- satisfying
- fun
- helpful
- motivating
- universal access (accessibility)

Heuristics (Usability Principles)

- **visibility of system status** (eg. busy mouse icon)
- **match between system and real world** (includes interface metaphors. eg. files and folders concept, "speak the user's language" (avoid gargon that users may not understand))
- **user control and freedom** (includes letting the user cancel/exit. eg. can pause/cancel file transfers)
- **consistency and standards** (eg. consistent terminology, consistent workflows, common look and feel, GUI toolkits. eg. GNOME/GTK+)
- **help and documentation**
- **help users recognise, diagnose and recover from errors** (tell users what the problem was, why it happened and some possible solutions, using plain English. eg. recover from trash)
- **error prevention** (eg. move to trash first)
- **recognition rather than recall** (GUI applications menu as opposed to CLI)
- **flexibility and efficiency of use** (eg. keyboard shortcuts (helpful for experts, but hidden from novices))
- **aesthetic and minimalist (uncluttered) design** (maybe put rarely used info into a help page/manual rather than in the interface)

Design Principles

- visibility
- feedback (can be audio, visual…)
- affordances (clues on how to use. eg. ⁝affords grabable)
- consistency (includes look and feel consistency)
- mapping
  - eg. which light switch controls which light
- constraints
  - logical (eg. grey out menu options that are not allowed)
  - physical (eg. you can't plug a USB cable into a VGA port, c.f. you can put a DVD in a CD player)
  - cultural

When designing a system we need to consider,

- who are the users,
- how will the product be used,
- where will the product be used

Identifying Needs

Requirements

When testing an interface with users/test participants, give them a high level goal and observe how they go about doing it. Don't give them specific instructions.

Use Scenario 1: For each task identified (or major tasks, or particularly special tasks if many tasks are defined), write a description of how that user would accomplish the task*independent* of how they would complete it within the application.

Use Case 1: If a use scenario has been implemented, include a matching use case which describes how the task use scenario can be completed in the application. There may be branching or multiple ways to complete the task, and this is a good way to document it.

To test if something is a requirement just ask, "If I remove this, will the product still fulfil its purpose?"

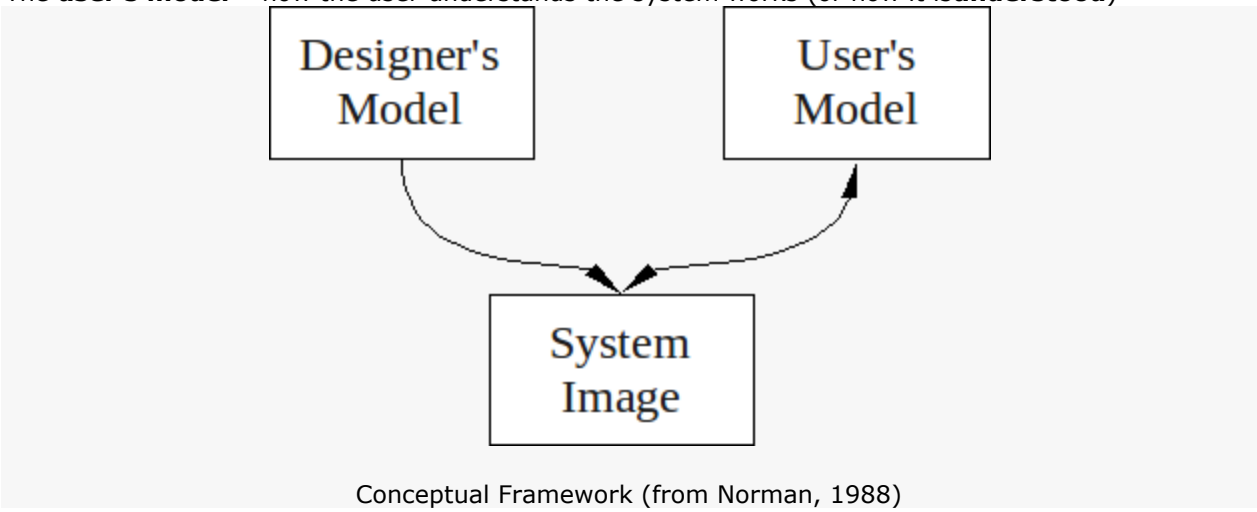
Design Conceptualisation

A conceptual model is a high-level description of how a system is organised and operates. –Johnson and Henderson, 2002, p. 26

I like to think of it as this. The person coding the web browsers understands that when the users types in a URL and presses enter an HTTP GET request is sent and the response is received and the HTML is processed and displayed. There are many technical interactions and details that are happening here. But the conceptual model of this is what the average non-technical uses thinks is happening. They just have some kind of model in their head that they type the URL hit enter and get the web site displayed. Its just an abstraction of what is going on.

Interface metaphors are used as they can help the user understand and determine how an interface works. We try to use them for this purpose but just using the metaphor directly can have some negative affects (eg. if your designing a radio application for desktop PC's, it may not be a good idea to just show an image of a real radio as the UI). We don't want to use the metaphor to an extent that it breaks the design principles.
A classic example of a conceptual framework is that of the relation between the design of a conceptual model and the user's understanding of it. In this framework there are three components, (Sharp et al., 2006)

- The **designer's model** – the model the designer has how how the system works (or how it **should** work)
- The **system image** – how the systems actual workings are portrayed to the users through the interface, manuals, etc. (or how it is **presented**)
- The **user's model** – how the user understands the system works (or how it is**understood**)



Conceptual Framework (from Norman, 1988)

The designers job is to create the system image so that the users will invoke the same conceptual model as the designer's.

- "A good conceptual model allows us to predict the effects of our actions." (Norman, 1988)

The interface could be made more transparent so the user can see the details of how the system works, but this is not always desirable as it may cause confusion. Also many users may not want to know all the gory details, nor should they have to know the actual implementation in order to use the system.

- You can conceptualise how a user interacts with a system in terms of their goals and what they need to do to achieve those goals.
- You can try to give the user a more correct mental model of the system by giving useful feedback based on their input and providing help and documentation.

Prototyping

- Can do low fidelity or high fidelity prototypes.
- Could use paper mockups of screens, storyboards, electronic mockup, electronic prototype…
- Make sure you iterate.
- "the best way to get a good idea is to get lots of ideas" –Linus Pauling

Using A Design Diary

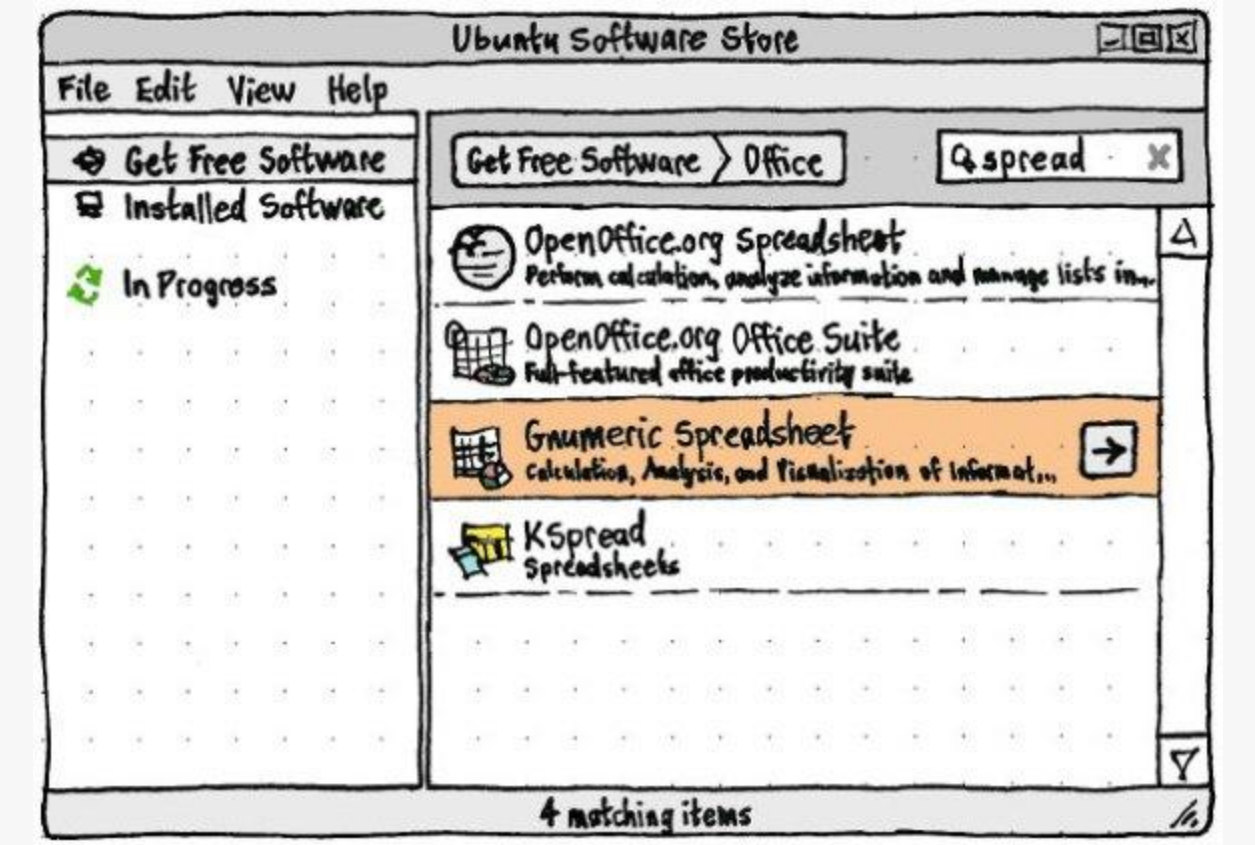
- Brainstorm ideas
- Sketch interface mockups
- Sketch storyboards/work flow diagrams

Wireframes

Here is an example wireframe.

Example Wireframe fromhttps://wiki.ubuntu.com/DesktopExperienceTeam/KarmicBootExperienceDesignSpec

Another paper prototype with a slightly higher fidelity.



An example paper prototype (fromhttps://wiki.ubuntu.com/SoftwareStore).

Issues Table

- In this course we list the issues vertically and the participants horizontally.
- Prioritise and group the issues. (Maybe use affinity diagramming for the grouping)

Usability Testing

- Can do interviews, questionnaires, usability tests (best to run a dry run of these before you start testing on many people), interaction logging…
- The purpose of a usability test is to gather feedback from potential users about usability issues as well as ensuring that an interface can be used and works as expected.
- Testing should be done throughout the whole process during prototyping, beta versions, and deployed applications.
- According to Nielson you only need test an interface with 5 people to find 80% of the issues (see Nielsen, *Usability Engineering*, p173) (however to publish research 5 is statistically too small so you should use at least 10).
- When planning a test you need to define scenarios and tasks as well as deciding what to test and how to collect the results. Its a good idea to have goals that try to measure the success of the usability principles.
- Test the parts of your interface which would be used most, as well as any particularly difficult to design aspects first.
  There are some legal and ethical issues to consider. The participant,
- needs to sign a consent form for you to run a test with them.
- is free to stop participating at any time.

- must be made aware of how you are observing them and what will be done with data collected. eg. is the session been recorded on video, audio, observed by people, screen captured…? Will the data be antonymous, will the anonymous results be published…
- should be made aware that you are testing the software, not them.

During a Usability Test,

- Avoid leading questions. (eg. try to avoid "How much do you like this interface?")
- When running a usability test be careful not to bias your results. For example instead of asking the user "How would do X? when there is a button "Do X", give them a scenario which has a goal and ask them how they would go about achieving this with the interface.
- You want the participant to "think aloud" so that you know what they are thinking when they are using the interface you are testing.
- If users are struggling give them a hint, if that doesn't help explain the expected solution and move on, but note that they needed assistance when recording the test data.
- If a task is difficult for the user, its not the users fault, its the interface's!
- We want to record things like time to complete the task, amount and nature of errors encountered and by who… Things that address the usability principles. You should record both these quantitative measurements and any qualitative things that you observer or the participant mentions.

After the Testing,

- Collate feedback and test data (Use an issues table to record the usability issues that the participants had.)
- Group issues and prioritise them.
- When analysing results consider,
  - If a user is asked to compare two interfaces, the results may bias towards the second as they learn from their first experience.
    - Can try to solve this by getting some participants to do A then B, and others B then A.
  - Observing how a user interacts with an interface may change how they interact with it. (ie. they may have done things differently if they were at home, without you scrutinising their every move).
    - We can try to avoid this by making the participants feel comfortable and reinforcing that we are not testing them we are testing the interface. Assure them that there are no incorrect users and don't avoid doing things just because you know we are taking notes.

Usability Testing

When actually running a usability test you should follow a usability test plan. The test plan just details what the test facilitator should do during the test.

The usability testing procedures we used in this course are:

1. Explain procedures (think aloud, data collection methods in use…)
2. Make sure they agree and sign a consent for before proceeding (you keep one, they keep one)
3. Run a pre-test questionnaire (used to generate a participant profile) (this helps to give you an idea on their experience level, as well as any background they may have in using similar interfaces before, as these affect how the participant performs) (best to get the participant to do this a few days before the test so that you can focus on specific tasks.)
4. Introduce scenario
5. Run through tasks
6. Ask any post test questions
7. Do they have any extra comments/debriefing
8. Thank them for their time

Interviews

- Can be open ended (participant gives longer responses which may include their reasoning) or closed (list of options participant chooses from).
- When running an interview give,
  - An introduction to the interview (what you are doing, purpose, what happens to the responses, how it it being recorded)
  - Warm-up questions
  - Main section. (use a logical sequence)
  - Cool-off questions
  - Closing remarks.

Questionnaires

- Participant Sample (You probably want a sample representative of your users/target users).

User Centred Design Process

The UCD process is all about focusing on the users and tasks. It also means iterate your designs often. The development is driven by users needs rather than technical concerns.

More specifically Gould and Lewis (1985) give three principles,

- Early focus on users and tasks.
- Empirical measurement.
- Iterative design.

Affinity Diagramming

- This is where we collect ideas and then group them.
- Don't use pre-defined groups, make them up as you start sifting through the ideas.
- The idea is to organise a bunch of individual ideas into a hierarchy.

Card Sorting

- Get a bunch of people to sort cards with some idea/concept/whatever into categories and then compare how they were sorted by the different participants.

Software Lifecycles

- Waterfall
- W-model
- Spiral
- Rapid Application Development
- Star Model (evaluation at each step)
- ISO 13407 (the HCI model goes around in a circle and only exits when satisfactory)

Cognitive Load Theory

Cognition is what goes on in our brains. It includes cognitive processes such as,

- attention
  - Some techniques are particularly good at grabbing attention (flashing, moving, colourful or large things). But these should be used sparingly.
- perception and recognition
  - Gestalt principles of perceptual organisation. ie. we group things by proximity, similarity, closure, symmetry and continuity.

Gestalt principles of perceptual grouping. (a. you see two groups the left two and the right two; b. you see the objects in columns rather than rows; c. you group the () braces together; d. you see two curves and most people probably see the same two curves (as opposed to sharp edges that meet at the vertex))

- memory
- learning
- reading, speaking and listening
- problem-solving, planning, reasoning and decision-making.
- automatic processes
  - Stroop effect (trying to say the colour not the words eg. red green blue orangepurple pink) is due to this.

Some Cognitive Load Theory

- Huge long term memory (with the information stored in schemas) and a limited working memory.
  - Schemas allow us to bypass our working memory limitations by chunking information.
  - They allow us to ignore the huge amount of detail coming from our senses and instead integrate with our existing schemas.
  - eg. its much easier to memories SYSTEM than YSMSTE.
  - "Automated Schemas"
- Worked Examples instead of Means-Ends Analysis
  - Its better to give new users a quick (or even not so quick) 'worked example' of how the interface works/how to use it, than just let them work it out for themselves.
- Split Attention Effect
  - e.g. "See figure 16.", "Refer to page 26"… "Requires us to mentally integrate information that is physically split."[2]
  - Solution physically integrate the material.
  - Don't force users to recall information from a previous screen.
- The Redundancy Effect
  - It is better to emit redundant information as it generally just confuses people.
- Expertise Reversal Effect
  - Its better to assume your audience is novice, if you are unsure.
  - Novices need lots of worked out examples
- Reduce Search (reduces cognitive load)
  - By using a consistent layout
  - By reducing visual clutter
- Diagrams can reduce cognitive load
- Modality Effect
  - Separate working memories for audio and visual senses.
  - Therefore presenting information visually and auditory allows for a greater total working memory size than just presenting it visually or auditory. (But we should consider users with disabilities, so taking advantage of this by presenting some information visually and some auditory is not a good idea)

Some HCI Applications

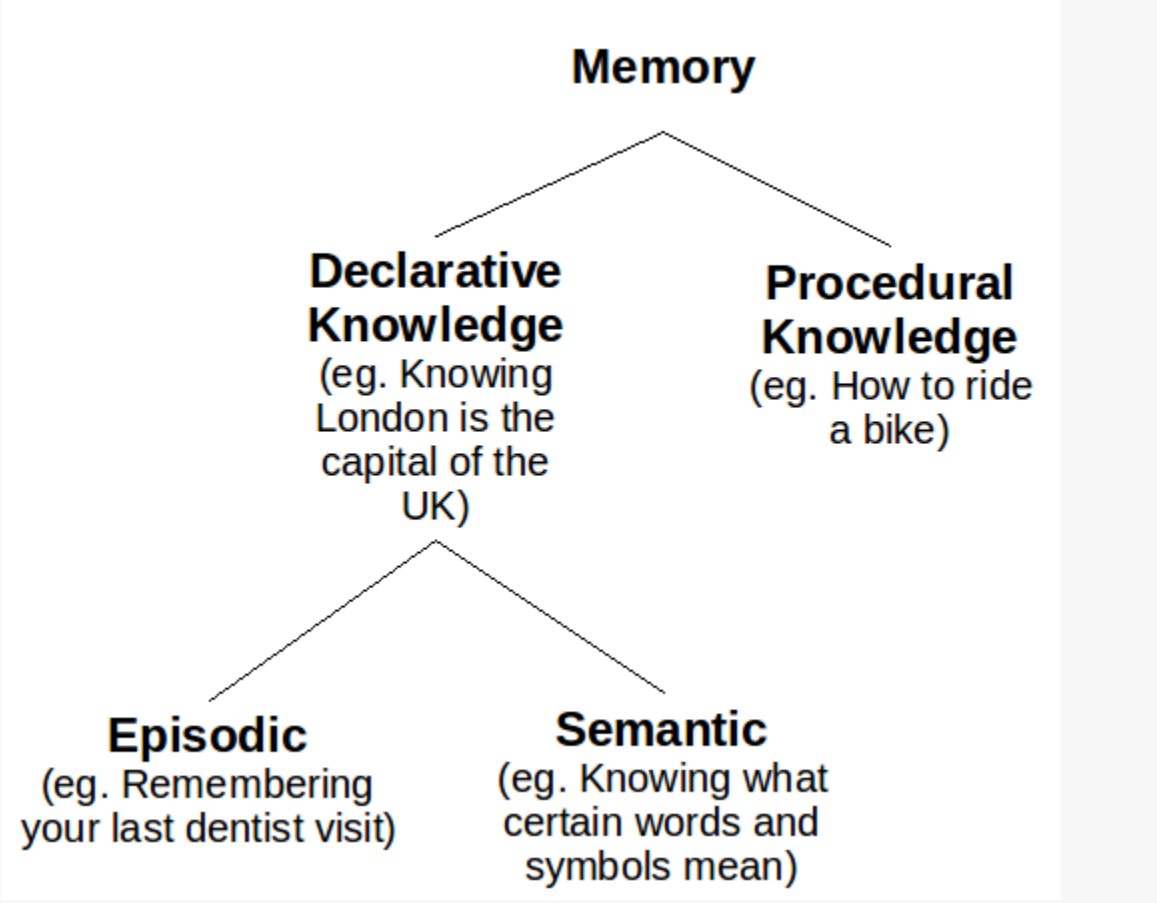
- The Training Wheels approach involves restricting the features of an interface for novices until they become more experienced when advanced features are enabled.

Memory

(From a psychologists perspective).

- Memory is based on your context (eg. night, bed, tired, dark, dream… ask them to recall they will often recall sleep even though it was never mentioned). Give context before this will help them store the information and recall it better.
- Miller's theory is that only 7± 2 chunks of information can be held in short-term memory at any one time. (But this doesn't mean say, only put seven items in any menu or something like that. This is only for short-term memory and when the information comes and goes, not when it can be rescanned.)

Long Term Memory



A Taxonomy of Memory

Explicit and Implicit Memory

"Imagine that you learn a list of items and are then required to recall or recognise them. This memory test would be accompanied by conscious awareness that you were remembering. Imagine that a considerable time later, a test of recall or recognition revealed no memory for the items. However if you were given the original list to relearn there would probably be some savings in learning time (i.e.

> you would take less time to learn the list the second time, oven though you were not aware of your memory of the items). This is the distinction between **explicit memory**, in which remembering is accompanied by either intentional or involuntary awareness of remembering, and **implicit memory**, in which remembering is not accompanied by awareness (Graf & Schacter 1985; Schacter 1987)." — (Walker, "Chapter 9: Memory, Reasoning and Problem Solving". pg. 262 (sorry I don't have the title))

Not really related, but a good thing to hear a text book say,

> "Finally, some long-enduring memories are for passages that our teachers have drulled into us… The Interesting thing about these memories is that they are preserved as they were memorised, in a very literal form, in exact wordings (Rubin, 1982). The memory code is not transferred from literal to semantic. In fact, the words are often remembered mechanically, with almost no attention to their meaning." –(Walker, "Chapter 9: Memory, Reasoning and Problem Solving". pg. 267 (sorry I don't have the title))

- The method of loci.
- The context that a memory is encoded, affects its retrieval. For example you may not initially recognise your neighbour on the train, as you are not used to seeing them there.
- People are much better at recognising things than recalling things.

Obstacles to Problem Solving

- Unwarranted Assumptions
  - Example given in class. "A man married 20 women. yet he broke not laws and never divorced. How? He was a priest."
- Seeing new Relationships
- Functional Fixedness
  - Being an expert at a system, you may not see things that novice would see.
  - You avoid using things in an unconventional way.
  - New users may find new or unintended uses of the system.
- The Set Effect
  - We may not notice that two similar problems actually need to be solved in different ways.

External Cognition

People use external representations to extend or support ones ability to perform cognitive activities. For example, pens and paper, calculators, etc. We do this to,

1. reduce memory load
   - eg. post-it notes, todo lists. But the placement of say post-it notes is also significant.
2. offload computation
   - eg. pen and paper to solve a large arithmetic problem (the mechanical kind).
3. annotate
   - modifying or manipulating the representation to reflect changes

Experts vs. Novices

- What distinguishes an expert is their large knowledge based stored in schemas.
- Declarative (facts)/procedural(how to do) knowledge.
- Skill acquisition.
  - Cognitive stage (learn facts, encode declarative knowledge),
  - Associative stage (procedural knowledge),
  - Autonomous stage.
- Novices tend to use ends-means analysis (uses a lot of trial and error) to solve problems. Experts tend to use their knowledge stored in schemas to apply and solve the problem (ie. past experience).

- In software can have novice (limited functionality) and expert modes. (Could be different applications Photoshop Elements vs. Photoshop Pro, or just hide advanced functionality for novices by default eg. >Advanced Options which is clicked to show more functions.)
- IDEA: Could provide popup hints to intermediate users to explain expert functions (eg. what's going on under the hood), or more advanced options (eg. keyboard shortcuts).

Visual Design

- Alignment of items in an interface makes it easier for users to scan the screen.
- Grouping
- Colour
- Gestalt Principles
- Menu design (see ISO 9241)
- Three types of icons,
  - similar (eg. a file for a file object)
  - analogical (eg. scissors for cut)
  - arbitrary (eg. X for delete or close)
- Can add text near the icon to make it easier for newbie's, but allows expert to ignore and just glance at the icon.

Internationalisation

Differences around the world,

- character set
- keyboard layout
- text direction
- language
- icons
- date, time, currency
- calendars

Internationalisation (i18n) refers to designing and developing a software product to function in multiple locales. Localisation (L10n) refers to modifying or adapting a software product to fit the requirements of a particular locale. This could include translating text, changing icons, modifying layout (eg. of dates).[5]

A locale is a set of conventions affected or determined by human language and customs, as defined within a particular geo-political region. These conventions include (but are not necessarily limited to) the written language, formats for dates, numbers and currency, sorting orders, etc.[5]

Accessibility

- Some clauses relating to requirements for Australian web sites in *Australian Disability Discrimination Act* (1992).

Quantification

- A way to test an interface different to usability testing.

GOMS

- Goals (eg. send an email)
- Operators (eg. double click)
- Methods (recalling what to do/how to do)
- Selection Rules (deciding which method to use to achive the goal)

Keystroke Level Model

- K (keying) – 0.2s – Press (and release) a keyboard key, or click the mouse. (Click and drag is only 1/2 K).
- P (pointing) – 1.1s – Moving the mouse to a location on the screen.
- H (homing) – 0.4s – Moving between the keyboard and mouse.
- M (mentally preparing) – 1.35s – Preparing.
- R (computer responding) – Waiting for the computer to respond.

Fitt's Law

- In the field of ergonomics research.
- Used to predict the time to move the cursor to a target.
- $T = A + B \times log_2\left(\frac{D}{S} + 1\right)$
- A and B are experimentally determined constants. (Raskin used A = 50, B = 150).
- D is distance between start and target
- S is size of target (just dealing with 1 dimension here).
- Lesson: The larger the target the faster one can move the mouse to that location.
- Lesson: Targets at the edge of the screen have an infinite size, so they are fast to navigate to. (Problem, if you use edge bindings a lot your mouse will physically move further and further away, so the user may need to be constantly picking it up moving it)

References

[1] Sharp, Rodgers, Preece. (2006) *Interaction Design: Beyond human computer interaction*. 2nd Ed.
[2] Marcus, Nadine. (2009) *COMP3511 Cognitive Load Theory Lecture Slides*.
Woo, Daniel. (2009) *COMP3511 Lecture Slides*.
Norman, Donald. (1988) *The Design of Everyday Things*.
[5] http://www.mozilla.org/docs/refList/i18n/

Source: http://andrewharvey4.wordpress.com/category/computing/page/2/