TODAY

• Introduction to evaluation studies
• Expert studies
  • Heuristic evaluations
  • Guidelines reviews
  • Cognitive walkthroughs
  • Cognitive dimensions

• Next time: user studies (usability and comparative), study design, planning.
EVALUATION STUDIES

• Software testing ≠ HCI evaluations.
• Some types of study:
  • **Qualitative:** studying meaning, text, conversation
  • **Quantitative:** numerical studies, statistics
  • **Case study:** small number of highly detailed subjects
  • **Ethnographic:** studying within a ‘natural’ environment
• No one size fits all.
USER AND EXPERT STUDIES

• **Expert studies:** one or more HCI experts evaluates a system and finds issues and weak points.

• **User studies:** users try out the system, and observations are made to determine the system’s issues and weak points.

• No one size fits all!
EXPERT STUDIES

• Advantages:
  • Cheap (if you do them yourself) and easy.
  • Members of the development team struggle do them because they already know how the system works!
  • Can be done at any time.
  • No ethics review required (usually).
EXPERT STUDIES

Disadvantages:

- You need an expert.
- How valid are they?
  - Some estimates say that around 50% of errors found by experts are not really a problem.
  - Can also miss problems that non-expert users may have.
- How reliable are they?
  - Will two experts give the same review?
- They have little status for publication
  - You can only use them when there is no alternative
HEURISTIC EVALUATION

- Use standard usability heuristics to structure an analysis.

**Schneiderman’s Golden Rules**
1. Strive for consistency.
2. Enable frequent users to use shortcuts.
3. Offer informative feedback.
4. Design dialogs to yield closure.
5. Offer error prevention and simple error handling.
6. Permit easy reversal of actions.
7. Support internal locus of control.

**Nielsen’s Heuristics**
1. Visibility of system status.
2. Match between system and the real world.
3. User control and freedom.
5. Error prevention.
7. Flexibility and efficiency of use.
8. Aesthetic and minimalist design.
9. Help users recognize, diagnose, and recover from errors.
GUIDELINES REVIEW

• Similar to a heuristic evaluation, but uses a set of user interface guidelines instead of a set of heuristics.
• Guidelines are often more specific and measurable.
GUIDELINES REVIEW

- Sets of guidelines may be used:
  - Organisation’s own guidelines
  - Governmental guidelines
  - W3C Web Accessibility Guidelines
  - Operating system guidelines
- Can sometimes be tested automatically.
W3C Accessibility Guidelines

1. Provide equivalent alternatives to auditory and visual content.
2. Don't rely on color alone.
3. Use markup and style sheets and do so properly.
5. Create tables that transform gracefully.
7. Ensure user control of time-sensitive content changes.
8. Ensure direct accessibility of embedded user interfaces.
10. Use interim solutions.
11. Use W3C technologies and guidelines.
12. Provide context and orientation information.
13. Provide clear navigation mechanisms.
14. Ensure that documents are clear and simple.
5. Create tables that transform gracefully.
5.1 For data tables, identify row and column headers. [Priority 1]
   For example, in HTML, use TD to identify data cells and TH to identify headers.

5.2 For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells. [Priority 1]
   For example, in HTML, use THEAD, TFOOT, and TBODY to group rows, COL and COLGROUP to group columns, and the "axis", "scope", and "headers" attributes, to describe more complex relationships among data.

5.3 Do not use tables for layout unless the table makes sense when linearized. Otherwise, if the table does not make sense, provide an alternative equivalent (which may be a linearized version). [Priority 2]
   Note. Once user agents support style sheet positioning, tables should not be used for layout. Refer also to checkpoint 3.3.
COGNITIVE WALKTHROUGHS

• More formal analysis, based on code walkthroughs.
• Evaluators step (fairly painstakingly) through a task.
• Can be done with a prototype.
• It helps to have scenarios and personas.
• Look at an action sequence to complete a task.
COGNITIVE WALKTHROUGH

Questions to focus on:

1. **Does the user want this effect to happen?**
   - Will the user expect this based on their interaction history and the system’s interface?

2. **Will the user notice that the action is available?**
   - Are the controls actually visible?

3. **Will the user know that this is the correct action?**
   - Is the user able to tell what this control does?

4. **Will the user understand the feedback?**
   - And is feedback even given?
• Can be documented as a hierarchical task analysis diagram (or similar)

• Paying for parking
COGNITIVE DIMENSIONS

• Reasonably comprehensive expert analysis, focused on psychological aspects of interface.
• More difficult to do than some of the other analyses.
• Can be done on your own software as long as you’re thorough and prepared to be critical.

• [Website Link](http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDtutorial.pdf)
<table>
<thead>
<tr>
<th>Cognitive Dimensions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstraction</strong></td>
<td>types and availability of abstraction mechanisms</td>
</tr>
<tr>
<td><strong>Hidden dependencies</strong></td>
<td>important links between entities are not visible</td>
</tr>
<tr>
<td><strong>Premature commitment</strong></td>
<td>constraints on the order of doing things</td>
</tr>
<tr>
<td><strong>Secondary notation</strong></td>
<td>extra information in means other than formal syntax</td>
</tr>
<tr>
<td><strong>Viscosity</strong></td>
<td>resistance to change</td>
</tr>
<tr>
<td><strong>Visibility</strong></td>
<td>ability to view components easily</td>
</tr>
<tr>
<td><strong>Closeness of mapping</strong></td>
<td>closeness of representation to domain</td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
<td>similar semantics are expressed in similar syntactic forms</td>
</tr>
<tr>
<td><strong>Diffuseness</strong></td>
<td>verbosity of language</td>
</tr>
<tr>
<td><strong>Error-proneness</strong></td>
<td>notation invites mistakes</td>
</tr>
<tr>
<td><strong>Hard mental operations</strong></td>
<td>high demand on cognitive resources</td>
</tr>
<tr>
<td><strong>Progressive evaluation</strong></td>
<td>work-to-date can be checked at any time</td>
</tr>
<tr>
<td><strong>Provisionality</strong></td>
<td>degree of commitment to actions or marks</td>
</tr>
<tr>
<td><strong>Role-expressiveness</strong></td>
<td>the purpose of a component is readily inferred</td>
</tr>
</tbody>
</table>
A hidden dependency is a relationship between two components such that one of them is dependent on the other, but that the dependency is not fully visible. In particular, the one-way pointer, where A points to B but B does not contain a back-pointer to A.

The search cost of a hidden dependency structure is a measure of the effort required to expose a typical dependency. Search cost is a function of the length of the trail, the amount of branching, and the effort required to follow each link.

HTML links: if your page is linked to someone else's... how will you know if and when that page is moved, changed, or deleted?

Spreadsheets are a mass of one-way, local dependencies.
CD APPLIED TO MCSIG

The Task

Learning to write with only non-visual modalities for feedback is a hard mental operation. By providing feedback with good closeness of mapping between the feedback and the real world, and suitably using abstraction mechanisms we seek to minimise the cognitive load required.
Audio

- Earcons are used as an abstraction to indicate the start and finish of a stroke. Error-proneness is reduced by using two distinctive sounds for the start and finish.

- The pan and pitch feedback simplify the idea of position in space into two single dimensions, creating an abstraction. Consistency is ensured by providing the same sound feedback in freedraw mode.

- The speech output of recognised letters that was tested in the McSig 1.0 usability study can only be carried out in a discrete manner, on a fully formed and accurate letter. Pitch and pan feedback offer better progressive evaluation.

- The viscosity of the stereo pan is low as stereo panning is readily changed between signature and single letter modes. However, presetting the stereo width requires premature commitment.

- The stereo pan is a spatial movement – moving between left and right extremes demonstrates closeness of mapping to the x movement of the pen.

- There is a problem with consistency because in single letter mode the starting point of the stroke is the centre of the panning, whereas with signature mode the starting point of the stroke is close to the left extreme of the panning. This is a trade-off between consistency and “visibility”.

- Even a small change in the pitch is noticeably discernable, giving this modality good “visibility”. But this has low closeness of mapping and role-expressiveness as change in pitch has no inherent relevance to the concept of movement in space along a y axis; the changing pitch is an abstraction.
Haptic

- Haptic guidance demonstrates juxtaposition. The movement of the student’s pen is juxtaposed with respect to the movement of the teacher’s pen in order to mitigate hidden dependency of the student’s pen movement upon the teacher’s pen movement. The system shows resistance to change, or viscosity, because the device has physical constraints; the writing area size is defined by limits of the device.

Tactile

- The tactile surface increases a tactile equivalent of visibility. Because the raised letters are persistent, and not erasable, the tactile surface has high viscosity, and provisionally is low. As a stroke is being carried out, the student can touch the tactile representation. This allows a measure of progressive evaluation. Two types of rubber band of different thickness can be used to show two types of line, demonstrating diffuseness.

USER STUDIES (NEXT TIME)

• **Usability Studies**
  • Evaluating a single piece of software in isolation.
  • Usually you ask users to complete specific tasks.
  • You can then calculate metrics like:
    • Time
    • Success rate
    • Number of attempts needed to succeed
    • Enjoyability

• **Comparative Studies**
  • Comparing two (or more) pieces of software.
  • Considerably more challenging!
  • Needs to be a fair test.
  • How can you be sure that an effect isn’t just due to the task ordering, or a user’s experience with doing the task?