Cognitive Dimensions
The Origin: Visual Programming and the Cognitive Dimensions of Notations
The Dimensions

- Abstraction
- Hidden dependencies
- Premature commitment
- Secondary notation
- Viscosity
- Visibility
- Closeness of mapping
- Consistency
- Diffuseness
- Error-proneness
- Hard mental operations
- Progressive evaluation
- Provisionality
- Role-expressiveness
Goals

- A vocabulary for discussion
- Broad brush
- Quick to learn
- Quick to apply
- Applicable at any stage of design
- Differentiates user activity types
- Multi-dimensional
- Vague
- Comprehensible to non-specialists
Compare to...

- A vocabulary for discussion
- Broad brush
- Quick to learn
- Quick to apply
- Applicable at any stage of design
- Differentiates user activity types
- Multi-dimensional
- Vague
- Comprehensible to non-specialists
Components of the CD Approach

- Structure of information artifacts
  - Environments, notations, media: focus is on notations
  - Layers (e.g., editor vs. language)
  - Sub-devices (e.g., MS Word text window vs. MS word prefs dialog)
- Activities
  - Incrementation (adding a formula to a spreadsheet)
  - Transcription (copying a math formula into a spreadsheet formula)
  - Modification (changing layout of spreadsheet)
  - Exploration (exploratory data analysis using a spreadsheet)
- The Dimensions
Why “Dimensions”? Tradeoffs

Figure 20 A line indicates a trade-off relationship identified in the text above. Not all of these trade-offs are available in all cases, and the diagram does not pretend to identify all possible relationships.
Viscosity

- Resistance to change—how hard is it to make small changes?
- Examples:
  - Updating links throughout a website
  - Changing repeated elements on a set of sketches

- Disadvantages: inefficient, causes breaks in flow of thought, fosters resistance to change
- Can be useful to prevent errors (e.g., Copy-Paste errors)

- Types
  - Repetition viscosity: need to repeat actions
  - Knock-on viscosity: need to chase dependencies
Hidden Dependencies

- A property or feature depends on another, but the link is not clear

**Examples**
- Value of another cell in spreadsheet depends on *this one*
  - \((A1+B3)/(C21-F6)\)
- What will break if I change the name of this file?
  - prefs.xml; settings.conf

**Disadvantages:** slow down information search, lead to errors

**Advantages:** easier to implement

**Types**
- One-way (e.g., spreadsheet references, HTML links)
- Local (e.g., file listings that don’t show children’s children)
Abstraction

- Grouping elements to be used as one element

Examples
- Change “Normal” style in Word, applies to all paragraphs in document
- Applying tags to bookmarks for later searching

Advantages: make notation more concise, reusable
Disadvantage: harder to understand and master

Types
- Abstraction hungry (e.g., Java)
- Abstraction tolerant (e.g., CSS, tagging systems)
- Abstraction hating (e.g., spreadsheets—except for “named ranges”)
Premature commitment

- Forcing users to make decisions with incomplete information

Examples
- Tagging systems—choose tags without knowing the full taxonomy, might wish to recategorize later
- Wizards: choose “simple” or “advanced” installation mode

Advantages: None

Disadvantages: Leads to errors, requires work to correct effects of incorrect decision
Secondary notation

- Allow extra information to be carried outside the formal syntax

- Examples:
  - Formatting in programming/HTML/etc.
  - Free-form annotations on paper that are lost in conversion to digital
  - Gestalt principles used in UI design

- Advantages: reinforce structure of information, add to structure of information
- Disadvantages: secondary notation is not “known” to the machine

- Types:
  - Redundant recoding (add indentation to a ToDo list to indicate dependent tasks)
  - Escape from formalism (add arrows between items in notes taken by hand)
Visibility & Juxtaposability

- Ability to view and compare information/controls easily

- Examples:
  - Side-by-side comparisons on shopping sites
  - Information entered in a form-based system that is needed later but not available

- Advantages: recognition vs. recall; allow users to leverage short term memory
- Disadvantages: may require more compact notation and/or more display space
The rest

- **Closeness of mapping:** match between system and domain
- **Consistency:** use of similar syntactic structures
- **Diffuseness:** verbosity of language/notation
- **Error-proneness:** notation invites mistakes
- **Hard mental operations:** high cognitive demand (reasoning, memory)
- **Progressive evaluation:** previous work can be re-checked at any time
- **Provisionality:** degree of commitment to decisions
- **Role-expressiveness:** the purpose of a component (e.g., a button or menu item) is readily understood
When it Matters: Different Activities

<table>
<thead>
<tr>
<th></th>
<th>transcription</th>
<th>incrementation</th>
<th>modification</th>
<th>exploration</th>
</tr>
</thead>
<tbody>
<tr>
<td>viscosity</td>
<td>acceptable</td>
<td>acceptable</td>
<td>harmful</td>
<td>harmful</td>
</tr>
<tr>
<td>hidden dependencies</td>
<td>acceptable</td>
<td>acceptable</td>
<td>harmful</td>
<td>acceptable for small tasks</td>
</tr>
<tr>
<td>premature commitment</td>
<td>harmful</td>
<td>harmful</td>
<td>harmful</td>
<td>harmful</td>
</tr>
<tr>
<td>abstraction barrier</td>
<td>harmful</td>
<td>harmful</td>
<td>harmful</td>
<td>harmful</td>
</tr>
<tr>
<td>abstraction hunger</td>
<td>useful</td>
<td>useful (?)</td>
<td>harmful</td>
<td>harmful</td>
</tr>
<tr>
<td>secondary notation</td>
<td>useful (?)</td>
<td>–</td>
<td>v. useful</td>
<td>v. harmful</td>
</tr>
<tr>
<td>visibility / juxtaposability</td>
<td>not vital</td>
<td>not vital</td>
<td>important</td>
<td>important</td>
</tr>
</tbody>
</table>

Figure 19 Table of user activities
When it Matters: Design Maneuvers

- Reduce viscosity by adding abstraction
  - increases lookahead, hidden dependencies, abstraction barrier
- Improve comprehensibility by adding secondary notation
  - may increase viscosity, hidden dependencies
- Lower cost of premature commitment by reducing viscosity
  - requires increasing abstraction; see above
- Remove dependencies to reduce need for lookahead
  - more complex syntax can be more diffuse, increase errors
- Improve visibility by reducing diffuseness
  - requires increasing abstraction; see above
Today’s Activity

- Go back to week #1 forum posts
- Choose two systems that differ along a pair of tradeoff dimensions (e.g., viscosity and abstraction)
- Analyze impacts of those decisions along other dimensions
- Analyze systems according to which user activities they support well and which they don’t
- Two minute report-backs to group during last 45 minutes of class