User Documentation

- Stored information about how to use a system
  - reference manuals, tutorials, online help
  - many systems incorporate a diverse set of online and paper documentation
- Challenge is to support all documentation needs
  - novice users encountering for the first time (not just ‘how’, also need to understand ‘what’ and ‘why’)
  - routine users who need reminder or new task procedure
  - change users who know how to perform routine tasks with related application software, but not with this software
  - expert users who want to find most efficient procedure

Paper or Online?

<table>
<thead>
<tr>
<th>Advantages of Paper</th>
<th>Disadvantages of Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highly portable, can be used anywhere</td>
<td>Finding and turning to a page is an extra task</td>
</tr>
<tr>
<td>Easy to scan at varying levels of detail</td>
<td>Paper is bulky, takes up office or desk space</td>
</tr>
<tr>
<td>Can be annotated with normal writing tools</td>
<td>Large manuals may seem intimidating to novices</td>
</tr>
<tr>
<td>Familiar and well-practiced reading habits</td>
<td>Lack of coordination between paper and software</td>
</tr>
<tr>
<td>Reading is faster from paper than screens</td>
<td>Fixed organization of content</td>
</tr>
<tr>
<td>People like owning books and other manuals</td>
<td>Paper and print deteriorates over time with use</td>
</tr>
</tbody>
</table>

Online info is becoming more ubiquitous, but paper still has a number of advantages
Systematic Documentation

- Comprehensive hierarchical task decomposition
  - recursively analyze each task into constituent subtasks
  - terminate with simple, step-by-step actions
  - can also include custom versions for different users
  - often, designers’ view of what mental model should be
- Each concept introduced, practiced, explained
  - typically presented as structured tutorial, online or paper-based
  - e.g., tell you what you are about to do, tell you how to do it in detail, then tell you what you did

- What are the downsides of this approach?

```
document create/revise
  menu operations initialize workspace input
  main file new name destination default formats
  menu tasks doc
  type navigate/select delete
  copy paste save
```
“Active learning”

- a good thing: people develop “by doing”
  - actively leverage their prior knowledge
  - hypothesize and experiment
  - diagnose and recover from error
  - try to accomplish meaningful tasks
- a bad thing: people are undisciplined users
  - pursue bad inferences and overgeneralizations
  - jump the gun, skip around
  - get tangled in errors within errors
  - don’t follow steps accurately

Similar Issues for Expertise

- With repeated use, action plans are practiced
  - knowledge is converted from declarative (description) to procedural (script)
  - we have already discussed “fast path” techniques that support such chunking
- But proceduralization not the same as optimization
  - many users do not want to improve
  - motivated to generate results, not to learn techniques for doing this efficiently
  - experience ≠ expertise!
The Paradox of the Active User

- The assimilation paradox
  - people interpret new situations in terms of what they already know, but new learning requires going beyond what is already known
- The production paradox
  - people want to get something done, but they must first spend time learning how to get something done

*Design challenge: exploit these tendencies, turn what might be seen as weaknesses to advantage!*

---

Minimalist Instruction

- Embed training in realistic tasks
  - e.g., writing a letter, not learning the menu system
- Allow users to get started fast
  - minimal verbiage, low conceptual overhead
- Rely on users to think and improvise
  - deliberately open-ended tasks, instructions
  - that leverage users’ prior knowledge where relevant
- Support error recognition and recovery
  - careful attention to feedback, just-in-time hints

*Why does this require an iterative design process?*
Examples of Minimalism

- **Minimal Manual**
  - task-oriented, but deliberately brief and incomplete

- **Training Wheels**
  - exotic or expert functionality blocked for novices

- **Guided Exploration Cards**
  - motivating suggestions + error recovery

- **Scaffolded Examples**
  - complex process (e.g., software design) is progressively accomplished through an unfolding example

*These techniques can be combined to create overall training approach*
Information in the Interface

*Help and feedback designed to support longterm use, software intended to be walk-up-and-use*

- Messages: balancing specificity and task relevance with length and complexity
- Consider whether and how to layer help information
  - top level for typical user, but increasing levels of detail to support users who *want* to learn
- Tooltips a useful approach, if done well
  - e.g., should do more than just (re)name a button, icon!
  - combine with layering for task-oriented learning

Socially Mediated Documentation

- Capture, organize, and reuse the “pockets of expertise” in an organization
  - a homegrown user support network
  - sometimes even just a pointer to the right person
  - “knowledge management”
- More globally, network-based FAQs, forums
  - sometimes sponsored by a corporation
  - or people with technical questions, interest
- Key question is who organizes, maintains?
Intelligent Help and Training

- Adaptive instruction: modeling and tracking the knowledge held by individual learners
  - assessing what they know, presenting new problems & activities that will expand the knowledge base
  - some success with algebra, LISP programming, but not for more general applications or users
- Context-sensitive help: recognizing what the user is trying to do, offering suggestions
  - software agents, e.g. Microsoft “Clippy”, not very successful for the general case
  - but “wizards” work well for highly scripted tasks

Designing Documentation

- Develop scenarios and usability specifications that center on learning concerns
  - common metaphors are ‘advice-giving’ people, e.g. a coach, a policeman, a lawyer or judge
  - must consider both novice and longterm use scenarios
- Iterative process, like all user-centered design
  - should parallel other design work as much as possible
  - writing user guides is one way to discover problems
  - e.g., elaborate a scenario to consider ‘what if’ the user did not know what to do, makes an error, ...