Discussion starters

- What types of things can be considered documentation?
- How do you currently use documentation?
  - Do you prefer paper or electronic?
  - When do you look at the documentation?
  - What’s the most frustrating thing about the documentation you’ve used?
- How do you currently write documentation?
  - How much time do you spend?
  - What medium do you use?
  - How do you decide what to document?

User Documentation

- Stored information about how to use a system
  - Reference manuals, tutorials, online help
  - Many systems show up with a diverse set of online and paper documentation
- Challenge is to support all documentation needs
  - Novices encountering for the first time (not just ‘how’, also need to understand ‘what’)
  - Regular users who need reminder or new task procedure
  - Experts 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
  - Analyze each task into constituent subtasks
  - Ultimately end up with step-by-step actions
  - Can also include custom versions for different users
  - Designers’ view of what mental model should hold
- Each concept introduced, practiced, explained
  - Typically presented as online or paper-based tutorial
  - 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?

Active Learning

Many novices attempt to learn by doing, they...

- Jump the gun
- Do not read and follow step-by-step instructions
- Do not carefully plan and analyze their actions
- Use their prior knowledge, even if not helpful
- Make many errors, get into tangles resolving them
- This is especially true when users are expert enough in problem domain to have genuine goals

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 advantages!
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 long term 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
- 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 "Clip-it", not very successful for the general case
  - But "wizards" may 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 long term 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, ...