Help models

- What kind of help works best for you?
- Do you ever “read the manual”?
- Is help usually “where you need it?”
- What are some differences between help you get from people and from systems?

Types of Help

Quick Reference:
- Reminders of common command names or options.
- Good to place on a card, or for small devices, on the device itself.
- Use a few main categories to avoid long search.
- E.g. for an editor, categories like “basic”, “search/replace”, “load/save”, etc.

Task-specific help
- User needs help on how to apply a command, or to complete a task.
- Can be part of a “how-to” system for common tasks.
- Should be easily accessible from the command line (if text).
- Make “options” windows obvious and easy to find!
- E.g. add “advanced” button in the dialogue to apply any command.

Full explanation
- User wants complete understanding, to get best value out of the application.
- This part explains the “why” more than the “how”.
- E.g. How do compiler options affect performance?
- What are various installation components used for? What are the uncommon commands?
- Probably need a chapter in the help system for this. More system-centric than task-centric.

Tutorial
- The tutorial leads the user through a task, scaffolding their actions.
- Should allow users to act as well as watch (sandboxing).
- The “best” way to teach!
- More work to build into the system, but you should leverage your company’s other effort:
  - E.g. most software houses conduct regular training sessions for customers - these are ideal tutorial content.
More advanced ideas

Help is a kind of ongoing learning environment.

Minimalist instruction (Carroll '92) is a learning approach
* It shows users what to do,
* then gives them realistic tasks to solve.
* It eliminates conventional exercises, tedium and repetition, and encourages users to explore.
* It also has extensive coverage of error recovery.
  * users feel confident exploring.

Help could be enjoyable? - at least it's a special case of computer-supported learning.

"Training wheels" (Carroll)
* Advanced commands are removed until user gains some experience with the system.
* Also some "dangerous" commands.
* Users explore freely in this sandbox.
* Users gained better understanding (IBM trial).

The "scenario machine" uses this idea and adds more help:
* Explanations of why certain commands are missing.
* Alternative command sequences for missing commands.

Desiderata for help

Availability
* Should be accessible anywhere (always include a help key on each major window).

Accuracy and Completeness (hard!)
* Make sure it matches program version, and that it covers all the commands.
* As well as commands, common tasks should be described.

Consistency
* Content, terminology, style.
* These days, online and printed manuals are often the same.

Robustness
* Help shouldn't crash if the program does (need another thread).
* Program exceptions can bring up the help system.

Flexibility
* Includes adaptation to context or user skill. Multi-level help is a good idea.

Unobtrusiveness
* Shouldn't disrupt users work (like the annoying help characters in MS Office). A separate help screen is often good - supports rapid switching.
Help in the days of yore..

Text command languages (Unix, DOS) had explicit "help" commands ("man", "help").
- Many design problems:
  - Shell kernel commands in one (90-page!) file.
  - Multiple OS and shell versions didn't match help!
- Some commands self-document with
  command -? Command --help command -h
  unfortunately, this doesn't always happen.
- No state: to see the options again, start over.
- Emacs info was a better system (early hypertext).

Context-sensitive help

Help depends on where it is used:
- Tool tips ↓ or the windows ? symbol: ❋?
  Groups Images
  Document Bed-Fill
  Quit the program Quit

Save the user the burden of synchronizing program state with help system state.
Almost always a good idea to do this.
Just make sure the user can easily find the main help contents and index.

Online tutorials

Can be useful, BUT:
- Users are not the same, some need minimal help.
- Forcing the user to execute a particular command is boring and annoying, and doesn’t help learning.

So..
- Make sure users can skip steps.
- Show users multiple ways of doing things.
- Give partial information on what to do, with more information available if the user requests it.

On-line docs

Online docs differ from paper manuals in the same way web sites differ from books.
Information organization is the key -
- Some pages contain content, others are primarily for navigation.
- Use best practices for web site design:
  - Intuitive names, careful clustering of information, intuitive link names, consistency...
  - Need a good search feature and an index.

Adaptive Help Systems

Adaptation is a good idea because:
- It avoids information that is too detailed or not detailed enough for a particular user.
- It avoids repetition of info the user has already seen.
- Can make suggestions of new ways to do tasks that the user may not know.

Weaknesses:
- Information can disappear (bad if the user forgot it too!).
- System needs to know user identity and user must use the system for some time.

Break
User Models

Beware:
* Linear scales (Novice - competent - expert), people don't advance on the same path.
* Stereotypes - same as above, plus users may have different kinds of problems in using the system.

Problematic:
* Overlay model - ideal behavior that the user should follow (e.g. in tutorials). But doesn't allow the user to do things their own way or explore.
* Task modeling: automatic task modeling is hard, and doesn't model bottom-up user behavior or "distributed cognition" (e.g. desk as a blackboard)

User Models

Rule-based techniques
* Limited success in AI, but scales well.
Frame-based techniques
* better organized version of RBT.
Semantic nets...
Decision trees...
Other AI stuff...

Generally, the most successful KR techniques today use probabilistic models.

Particularly in HCI, the system can't "observe" the user's expertise and can only guess what it is.

Knowledge representation

Probabilistic models provide a way to:
* Allow several alternative interpretations at once.
* Incorporate new evidence.
* Model and explain the system's certainty in its own decisions.

Used in MS' help system.

The trick is to figure out the appropriate "traits" of the user:

i.e. clusters of knowledge that users typically have.

You can mine help logs from user studies to identify these clusters, or do it by inspection.

Knowledge representation
Initiative

A Help system works with the user, and ideally should allow a spectrum of control:

- "Help me", "tell me what to do", "show me what to do", "OK, I'll take over now..."
- This is called "mixed initiative".

A good mixed-initiative help system requires links between all parts of the system including a tutorial.

User should be able to "take over" at any time, then give back control.

Effect

Don't overdo user modeling:

* The idea is to figure out just enough to adapt the help system.
* Figure out the effects you need (novice/expert pages, or specific task support), and then the user data need to do adaption.

Scope

How much should help cover:

* Common questions.
* Functional model description.
* Structural model info?

Design issues

Help system design is like other parts of the interface:

* Start with task analysis.
* Do paper prototypes.
* Do user tests at informal and formal stages - look for errors.
* Use errors as the "objects" to guide the design of the help system.

User modeling:

* The error list can be used to derive user models.
* Run pattern recognition on the error list to find the dimensions of the user profile.
Help presentation

- Make sure that help presentation is appropriate:
  - Help shouldn’t obscure the working areas of the screen.
  - Make sure it’s easy to jump to the help index/contents page.
  - Use animation for tutorial or “show me” tasks.
  - Keep help system context - i.e. for each help section, show where it fits in the help contents.

Summary

- Types of help.
- Help as a learning system or sandbox.
- Adaptive help - user modeling - knowledge representation.
- Initiative/Effect/Scope.
- Design/implementation issues.