CHAPTER 7:
Command and Natural Languages

Designing the User Interface:
Strategies for Effective Human-Computer Interaction

Fifth Edition

Ben Shneiderman & Catherine Plaisant

in collaboration with

Maxine S. Cohen and Steven M. Jacobs

Addison Wesley
is an imprint of
PEARSON

The Basic Goals of Language Design

- Precision
- Compactness
- Ease in writing and reading
- Speed in learning
- Simplicity to reduce errors
- Ease of retention over time

Higher-Level Goals of Language Design

- Close correspondence between reality and the notation
- Convenience in carrying out manipulations relevant to user's tasks
- Compatibility with existing notations
- Flexibility to accommodate novice and expert users
- Expressiveness to encourage creativity
- Visual appeal

Functionality to Support User's Tasks

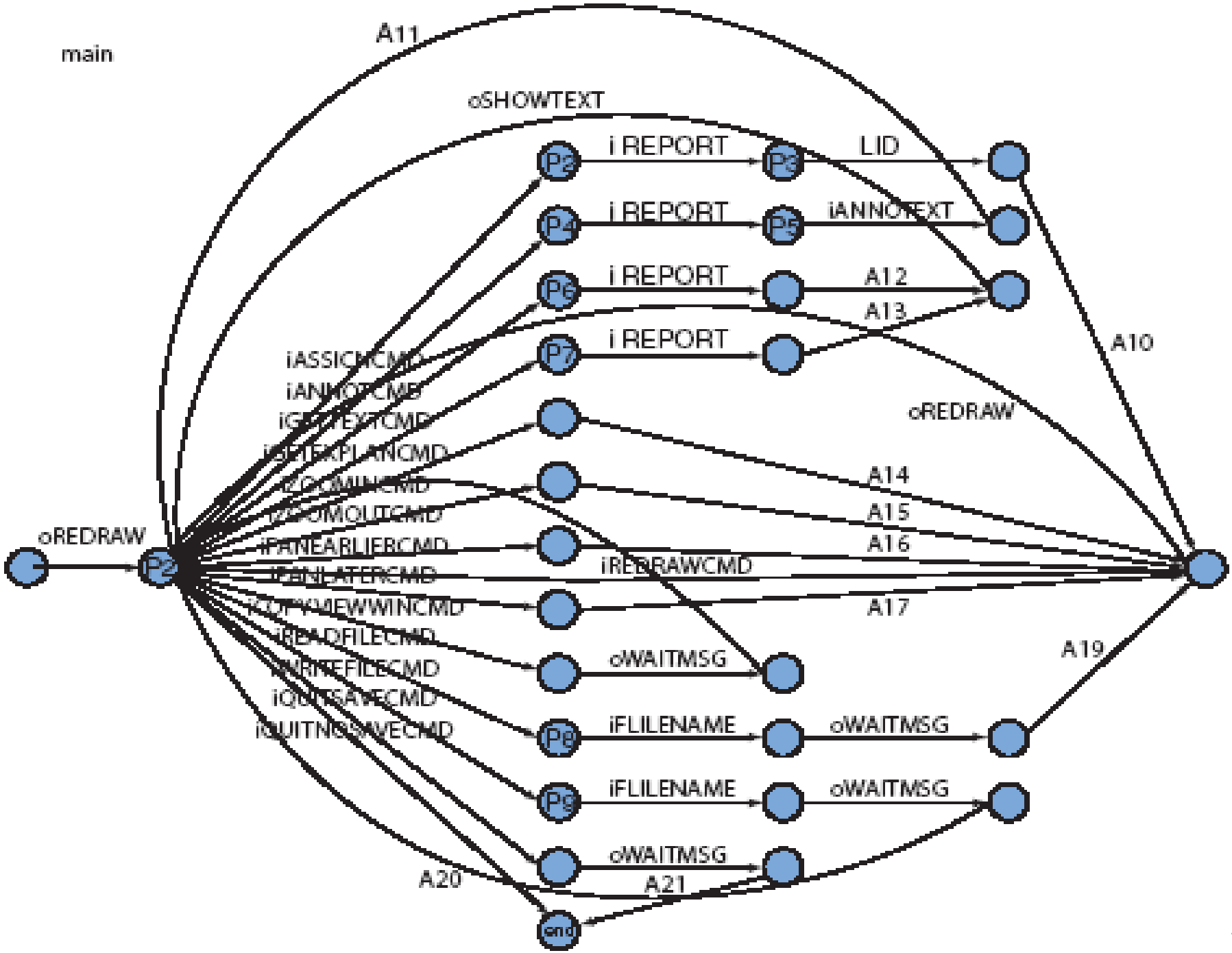
Users do wide range of work:

- text editing
- electronic mail
- financial management
- airline or hotel reservations
- inventory
- manufacturing process control
- gaming

Functionality to Support User's Tasks (cont.)

Designers should

- determine functionality of the system by studying users' task domain
- create a list of task actions and objects
- abstract this list into a set of interface actions and objects
- represent low-level interface syntax
- create a table of user communities and tasks, with expected use frequency
- determine hierarchy of importance of user communities (i.e. prime users)
- evaluate destructive actions (e.g. deleting objects) to ensure reversibility
- identify error conditions and prepare error messages
- allow shortcuts for expert users, such as macros and customizing system parameters


main
A11
oSHOWTEXT
i REPORT
LID
i REPORT
iANNOTEXT
i REPORT
A12
A13
i REPORT
A10
oREDRAW
A14
A15
A16
iREDRAWCMD
A17
A19
oWAITMSG
iFLILENAME
oWAITMSG
iFLILENAME
oWAITMSG
oWAITMSG
A20
A21
oREDRAW
P2
P3
P4
P5
P6
P7
P8
P9
end

Command-Organization Strategies

A unifying interface concept or metaphor aids
- learning
- problem solving
- retention

Designers often err by choosing a metaphor closer to machine domain than to the user's task domain.

Simple command set
- Each command is chosen to carry out a single task. The number of commands match the number of tasks.
- For small number of tasks, this can produce a system easy to learn and use.
- E.g. the vi editor of Unix.

Command plus arguments/options

Command plus arguments

- Follow each command by one or more arguments that indicate objects to be manipulated, e.g.
  - COPY FILEA, FILEB
  - DELETE FILEA
  - PRINT FILEA, FILEB, FILEC
- Keyword labels for arguments are helpful for some users, e.g. COPY FROM=FILEA TO=FILEB.

- Commands may also have options to indicate special cases, e.g.:
  - PRINT/3,HQ FILEA
  - PRINT (3, HQ) FILEA
  - PRINT FILEA -3, HQ

  to produce 3 copies of FILEA on the printer in the headquarters building.
- Error rates and the need for extensive training increase with the number of possible options.

The Benefits of Structure

Human learning, problem solving, and memory are greatly facilitated by meaningful structure.

- Beneficial for
  - task concepts
  - computer concepts
  - syntactic details of command languages

Consistent Argument Ordering

Inconsistent order of arguments		**Consistent order of arguments**	
SEARCH	file no, message id	SEARCH	message id, file no
TRIM	message id, segment size	TRIM	message id, segment size
REPLACE	message id, code no	REPLACE	message id, code no
INVERT	group size, message id	INVERT	message id, group size

Hierarchical command structure

- The full set of commands is organized into a tree structure
- 5x3x4 = 60 tasks with 5 command names and 1 rule of formation

Action	Object	Destination
CREATE	File	File
DISPLAY	Process	Local printer
REMOVE	Directory	Screen
COPY		Remote printer
MOVE		

Symbols versus Keywords

Command structure affects performance

Symbol Editor	Keyword Editor
FIND:/TOOTH/;-1	BACKWARD TO "TOOTH"
LIST;10	LIST 10 LINES
RS:/KO/,/OK/;*	CHANGE ALL "KO" TO "OK"

	Percentage of Task Completed		Percentage of Erroneous Commands	
	Symbol	Keyword	Symbol	Keyword
Inexperienced users	28	42	19.0	11.0
Familiar users	43	62	18.0	6.4
Experienced users	74	84	9.9	5.6

Naming and Abbreviations

There is often a lack of consistency or obvious strategy for construction of command abbreviations.

Specificity Versus Generality

Infrequent, discriminating words	insert	delete
Frequent, discriminating words	add	remove
Infrequent, nondiscriminating words	amble	perceive
Frequent, nondiscriminating words	walk	view
General words (frequent, nondiscriminating)	alter	correct
Nondiscriminating nonwords (nonsense)	GAC	MIK
Discriminating nonwords (icons)	abc-adbc	abc-ab

Six Potential Abbreviation Strategies

1. Simple truncation: The first, second, third, etc. letters of each command.
2. Vowel drop with simple truncation: Eliminate vowels and use some of what remains.
3. First and last letter: Since the first and last letters are highly visible, use them.
4. First letter of each word in a phrase: Use with a hierarchical design plan.
5. Standard abbreviations from other contexts: Use familiar abbreviations.
6. Phonics: Focus attention on the sound.

Guidelines for using abbreviations

Ehrenreich and Porcu (1982) offer this set of guidelines:

- A *simple* primary rule should be used to generate abbreviations for most items; a *simple* secondary rule should be used for those items where there is a conflict.
- Abbreviations generated by the secondary rule should have a marker (for example, an asterisk) incorporated in them.
- The number of words abbreviated by the secondary rule should be kept to a minimum.
- Users should be familiar with the rules used to generate abbreviations.
- Truncation should be used because it is an easy rule for users to comprehend and remember. However, when it produces a large number of identical abbreviations for different words, adjustments must be found.
- Fixed-length abbreviations should be used in preference to variable-length ones.
- Abbreviations should not be designed to incorporate endings (ING, ED, S).
- Unless there is a critical space problem, abbreviations should not be used in messages generated by the computer and read by the user.

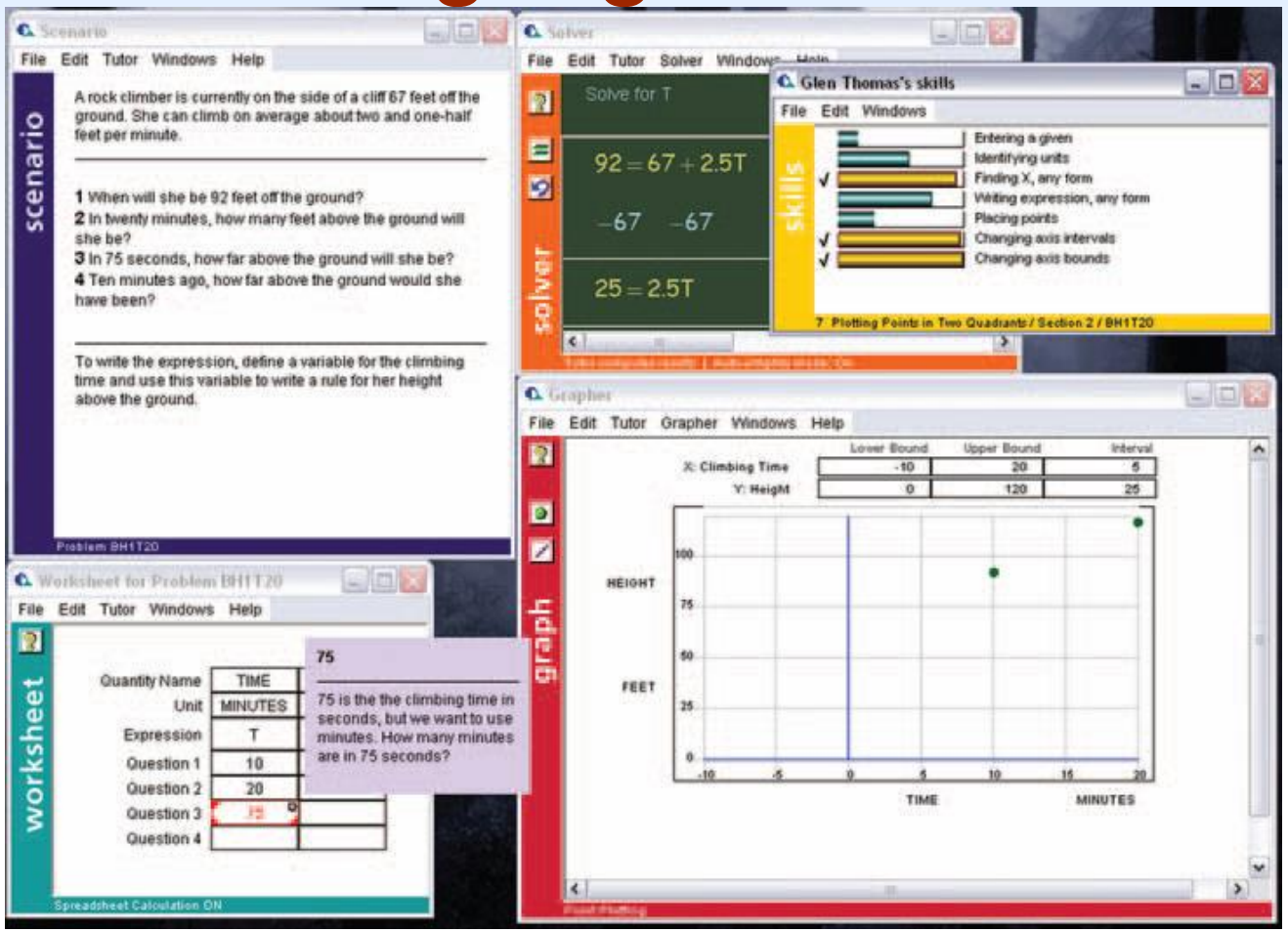
Command-language guidelines

- Create explicit model of objects and actions.
- Choose meaningful, specific, distinctive names.
- Try to achieve hierarchical structure.
- Provide consistent structure (hierarchy, argument order, action-object).
- Support consistent abbreviation rules (prefer truncation to one letter).
- Offer frequent users the ability to create macros.
- Consider command menus on high-speed displays.
- Limit the number of commands and ways of accomplishing a task.

Natural Language in Computing

- Natural-language interaction
- Natural-language queries and question answering
- Text-database searching
- Natural-language text generation
- Adventure games and instructional systems

Natural Language in Education



CognitiveTutor traces student progress in mastering skills and concepts, then assigns individually tuned problems Communicating with students via Natural Language