

Usability Heuristics

Avoid common design pitfalls by following 9 design principles

Inspect an interface for usability problems with these principles

Design principles

broad usability statements that guide a developer's design efforts

- use the users language
- provide feedback...

derived from common design problems across many systems

© Saul Greenberg

Heuristic evaluation

Systematic inspection to see if interface complies to guidelines

Method

- 3-5 inspectors
- usability engineers, end users, double experts...
- inspect interface in isolation (~1–2 hours for simple interfaces)
- compare notes afterwards
 - single evaluator only catches ~35% of usability problems
 - 5 evaluators catch 75%

Works for paper, prototypes, and working systems

Saul Greenberg

Heuristic evaluation

Advantages

- “minimalist” approach
 - a few guidelines identify many common usability problems
 - easily remembered, easily applied with modest effort
- discount usability engineering
 - end users not required
 - cheap and fast way to inspect a system
 - can be done by usability experts, double experts, and end users

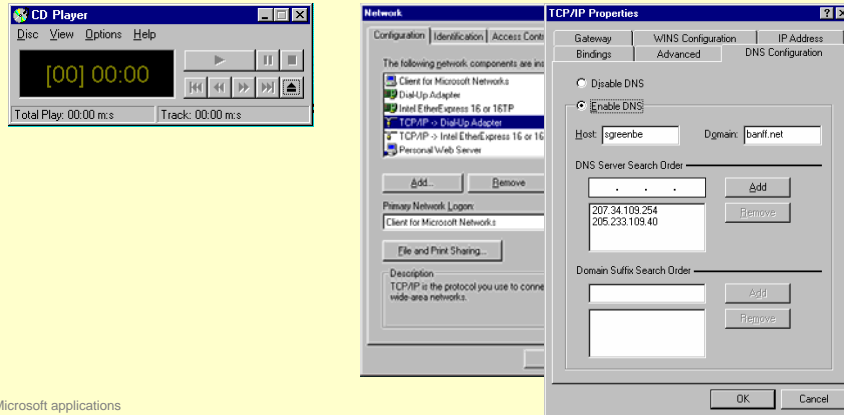
Problems:

- principles are more or less at the motherhood level
 - can't be treated as a simple checklist
 - subtleties involved in their use

Saul Greenberg

1 Simple and natural dialogue

- use the user's conceptual model
- match the users' task sequence
- minimize mapping between interface and task semantics

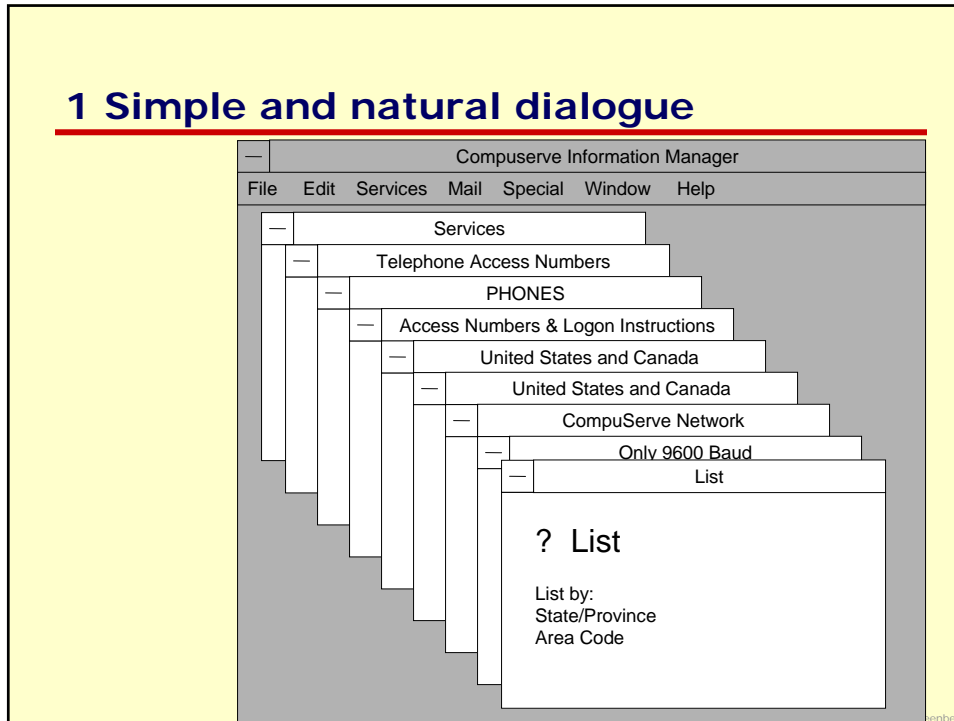


1 Simple and natural dialogue

Present exactly the information the user needs

- less is more
 - less to learn, to get wrong, to distract...
- information should appear in natural order
 - related information is graphically clustered
 - order of accessing information matches user's expectations
- remove or hide irrelevant or rarely needed information
 - competes with important information on screen
- remove modes
- use windows frugally
 - don't add unneeded navigation and window management

1 Simple and natural dialogue



The screenshot shows a 'Dr. Drugs Pharmacy Master' application window. The patient profile for Thomas Davison is displayed, with the 'Edit Customer' button highlighted by a red circle. The profile includes the following information:

- First Name:** Thomas
- Last Name:** Davison
- Address:** 4 BRADEN CRS Nw
- City:** Prudhoe
- Province:** Alberta
- Postal Code:** F4T6H8
- Phone Number:** 403 282-9759
- Alberta Health Care #:** 45688-3357
- Doctor:** Jane Yang
- Insurer:** FARMERS INSURANCE AGE
- Policy #:** 4215487-56

The 'Prescriptions' section shows a list of medications and a corresponding bar chart for the year 2001:

Medication	Start Date	End Date
Alprazolam	Jan 2001	Feb 2001
Citalopram	Feb 2001	Mar 2001
Ranitidine	Apr 2001	May 2001
Folic Acid	Jun 2001	Jul 2001
Levothyroxine	Aug 2001	Aug 2001
Celecoxib	Oct 2001	Oct 2001
Omeprazole	Nov 2001	Nov 2001

Good: information all in the same place

By previous 481 students Brant LeClerc, Lloyd Yoon, Amy Yang (with permission)

Dr. Drugs Pharmacy Master

First Name: Thomas
Last Name: Davison
Address: 4 BRADEN CRS NW
City: Prudhoe
Province: Alberta
Postal Code: F4T6H8
Phone Number: 403

Alberta Health Care #: 45688-3357
Doctor: Jane Yang
Insurer: FARMERS INSURANCE AG
Policy #: 4215487-56
Notes:

Save Cancel

Prescriptions

See List View Edit Prescription New Prescription

Alprazolam	Jan	Feb
Citalopram	Feb	Mar
Ranitidine	Mar	Apr
Folic Acid	Jun	Jul
Levothyroxine	Jul	Aug
Celecoxib	Oct	Nov
Omeprazole	Nov	Dec

1999 <Year> 2001 Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

Good: information all in the same place
Bad: special edit mode

By previous 481 students Brant LeClercq, Lloyd Yoon, Amy Yang (with permission)

Dr. Drugs Pharmacy Master

Davison, Thomas Edit Customer New Customer

First Name: Thomas
Last Name: Davison
Address: 4 BRADEN CRS NW
City: Prudhoe
Province: Alberta
Postal Code: F4T6H8
Phone Number: 403 282-9759

Alberta Health Care #: 45688-3357
Doctor: Jane Yang
Insurer: FARMERS INSURANCE AGE
Policy #: 4215487-56
Notes:

Prescriptions

See List View Edit Prescription New Prescription

Alprazolam	Jan	Feb
Citalopram	Feb	Mar
Ranitidine	Mar	Apr
Folic Acid	Jun	Jul
Levothyroxine	Jul	Aug
Celecoxib	Oct	Nov
Omeprazole	Nov	Dec

1999 <Year> 2001 Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

By previous 481 students Brant LeClercq, Lloyd Yoon, Amy Yang (with permission)

Dr. Drugs Pharmacy Master

Davison, Thomas Edit Customer New Customer

First Name: Thomas **Alberta Health Care #:** 45688-3357
Last Name: Davison **Doctor:** Jane Yang
Address: 4 BRADEN CRS NW **Insurer:** FARMERS INSURANCE AGE
City: Prudhoe **Policy #:** 4215487-56
Province: Alberta **Notes:**
Postal Code: F4T6H8
Phone Number: 403 282-9759

Prescriptions See List View Edit Prescription New Prescription

Drug Name:
 Manufacturer: Doctor:
 Quantity: Units:
 Price/Unit: Refill Quantity:

Save Cancel

Good: Stable parts of the window
Bad: Prescriptions separate from graphics

By previous 481 students Brant LeClerc, Lloyd Yoon, Amy Yang (with permission)

Dr. Drugs Pharmacy Master

Davison, Thomas New Customer

First Name: Thomas **Alberta Health Care #:** 45688-3357
Last Name: Davison **Doctor:** Jane Yang
Address: 4 BRADEN CRS NW **Insurer:** FARMERS INSURANCE AGE
City: Prudhoe **Policy #:** 4215487-56
Province: Alberta **Notes:**
Postal Code: F4T6H8
Phone Number: 403 282-9759

Prescriptions See List View Add Undo New Prescription

Alprazolam (Jan - Feb)
Citalopram (Feb - Mar)
Ranitidine (Mar - Apr)
Folic Acid (Jun - Jul)
Levothyroxine (Jul - Aug)
Celecoxib (Oct - Nov)
Omeprazole (Nov - Dec)

1999 < Year > 2001 | Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

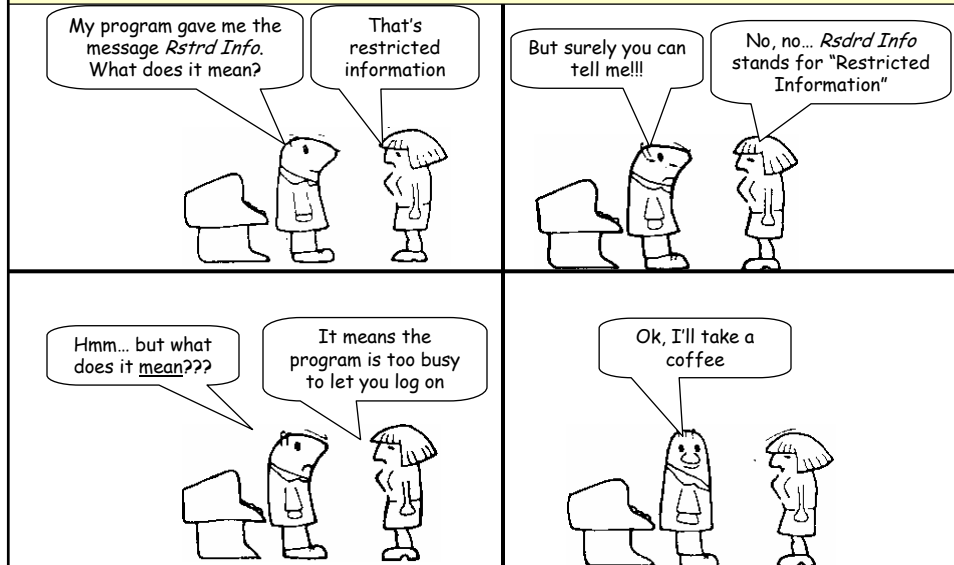
collapsed onto one screen (needs formatting)

Click to get info

Double click to edit (mode buttons gone)

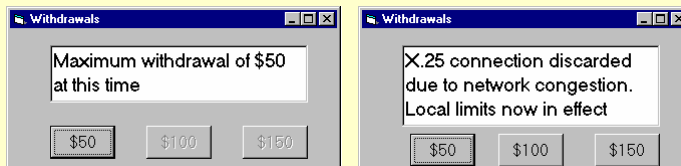
By previous 481 students Brant LeClerc, Lloyd Yoon, Amy Yang (with permission)

2 Speak the users' language




2 Speak the users' language

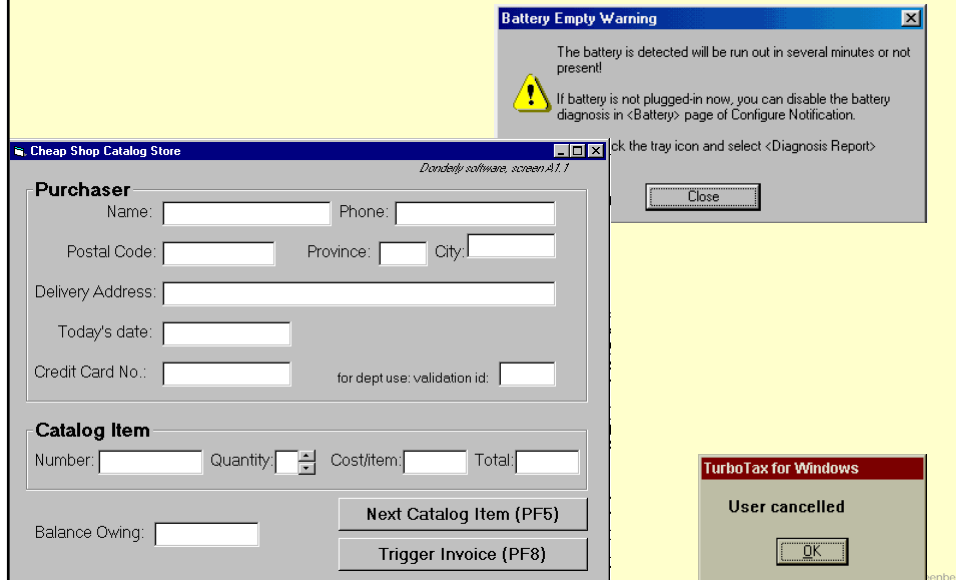
Terminology based on users' language for task
– e.g. withdrawing money from a bank machine



Use meaningful mnemonics, icons & abbreviations

- eg File / Save
 - Ctrl + S (abbreviation)
 - Alt FS (mnemonic for menu action)
 -  (tooltip icon)

2 Speak the users' language



3 Minimize user's memory load

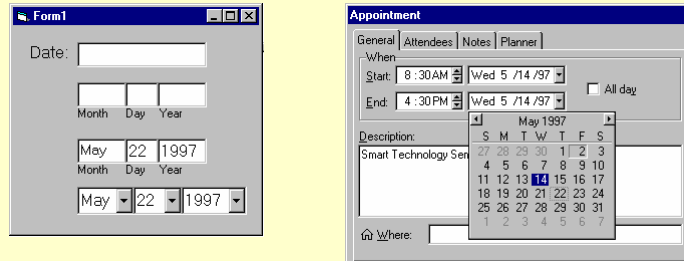
Computers good at remembering, people are not!
Promote recognition over recall

- menus, icons, choice dialog boxes vs commands, field formats
- relies on visibility of objects to the user (but less is more!)



3: Minimize user's memory load

Gives input format, example and default

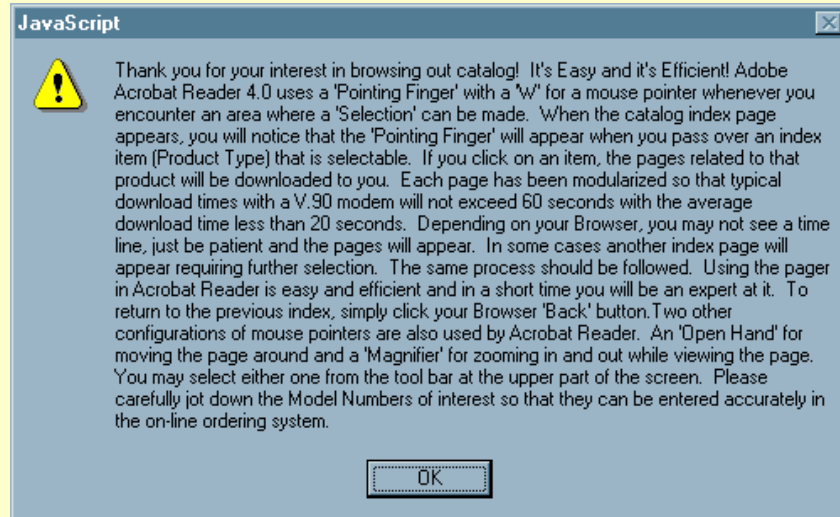


3: Minimize user's memory load

Small number of rules applied universally

- generic commands
 - same command can be applied to all interface objects
 - interpreted in context of interface object
 - copy, cut, paste, drag 'n drop, ... for characters, words, paragraphs, circles, files
 - context menus

3: Minimize user's memory load

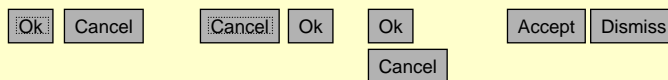


4: Be consistent

Consistent syntax of input

Consistent language and graphics

- same visual appearance across the system (e.g. widgets)
- same information/controls in same location on all windows



Consistent effects

- commands, actions have same effect in equivalent situations
 - predictability

4. Be Consistent

These are labels with a raised appearance.

Is it any surprise that people try and click on them?

Subscriber

Name: Tech. Re

Account #: Status:

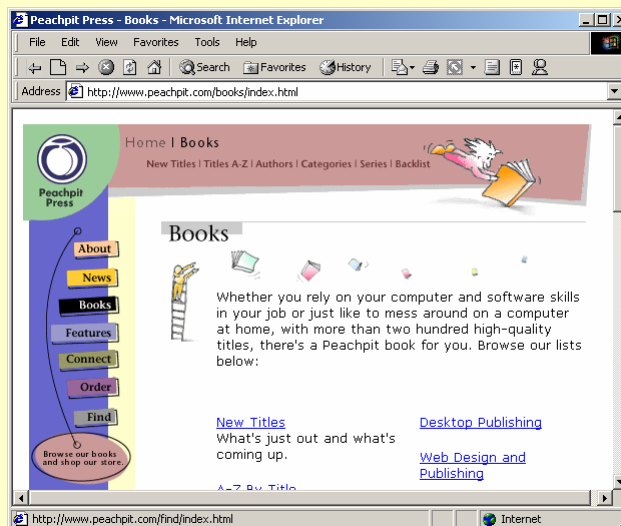
Contact

Telephone: E-Mail:

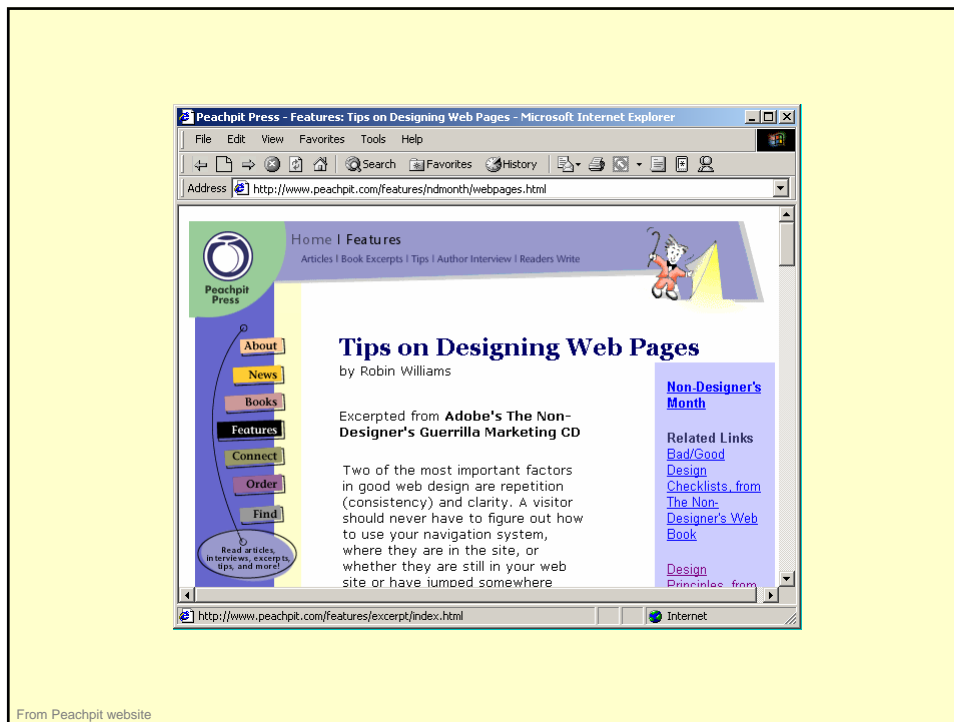
Address: St

Save Cancel

Saul Greenberg



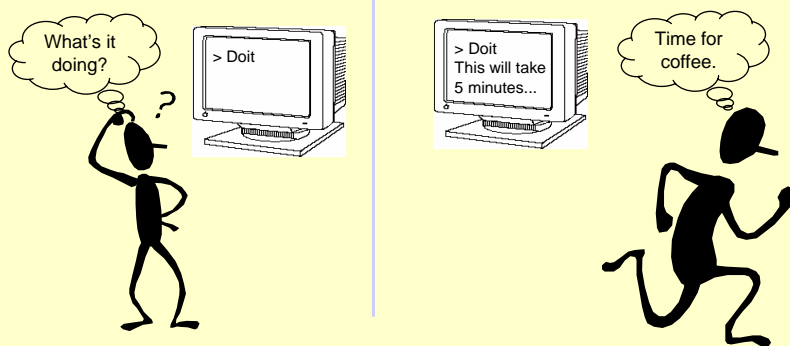
From Peachpit website



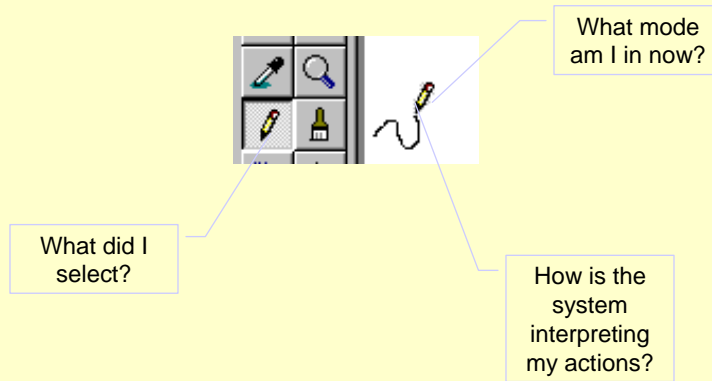
5: Provide feedback

Continuously inform the user about

- what it is doing
- how it is interpreting the user's input
- user should always be aware of what is going on



5. Provide feedback

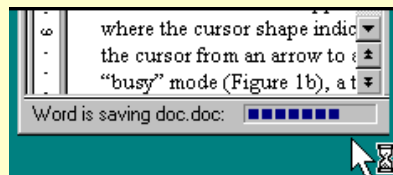
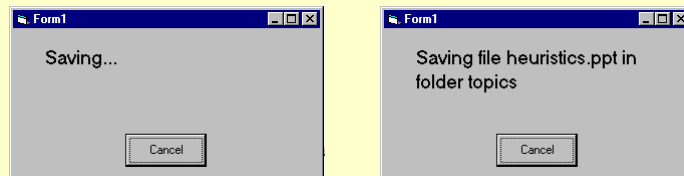


Microsoft Paint

Saul Greenberg

5. Provide feedback

Be as specific as possible, based on user's input

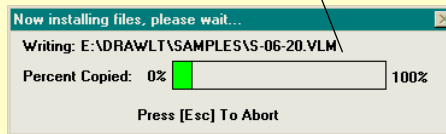


Best within the context of the action

Saul Greenberg

Provide feedback

Multiple files being copied,
but feedback is file by file.



Drawing Board LT

Saul Greenberg

5. Provide feedback

Response time

– how users perceive delays

- <0.1s perceived as “instantaneous”
- 1s user’s flow of thought stays uninterrupted, but delay noticed
- 10s limit for keeping user’s attention focused on the dialog
- > 10s user will want to perform other tasks while waiting

Saul Greenberg

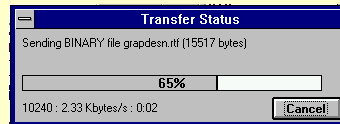
5. Provide feedback

Dealing with long delays

- Cursors
 - for short transactions



- Percent done dialogs
 - time left
 - estimated time



- Random
 - for unknown times



Saul Greenberg

6. Provide clearly marked exits



How do I get out of this?

Saul Greenberg

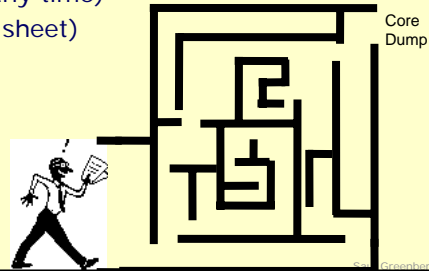
6. Provide clearly marked exits

Users don't like to feel trapped by the computer!

- should offer an easy way out of as many situations as possible

Strategies:

- Cancel button (for dialogs waiting for user input)
- Universal Undo (can get back to previous state)
- Interrupt (especially for lengthy operations)
- Quit (for leaving the program at any time)
- Defaults (for restoring a property sheet)

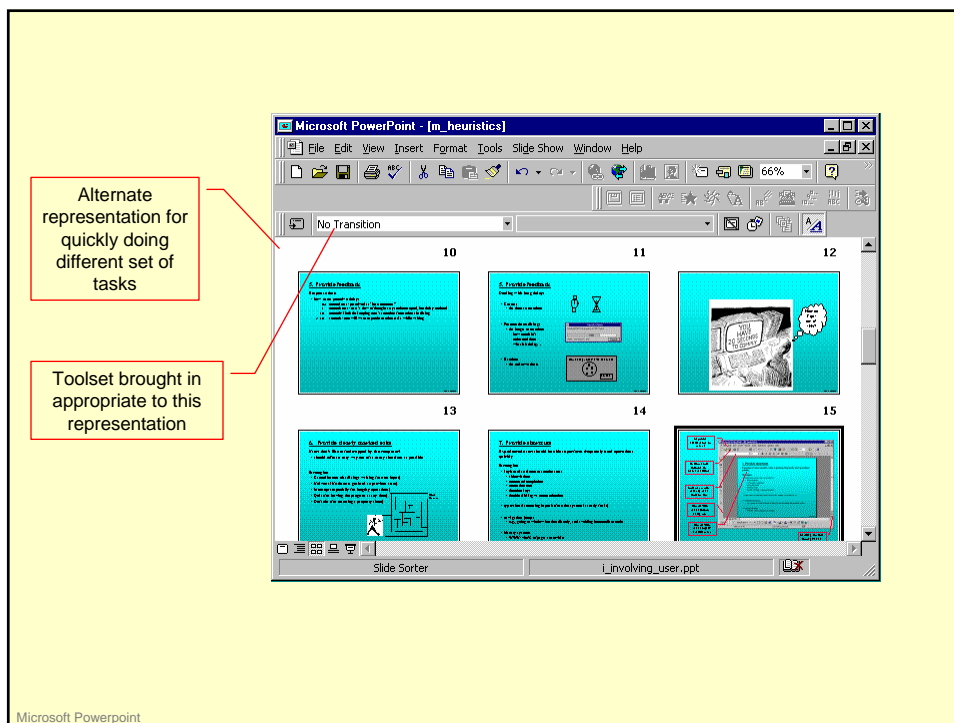
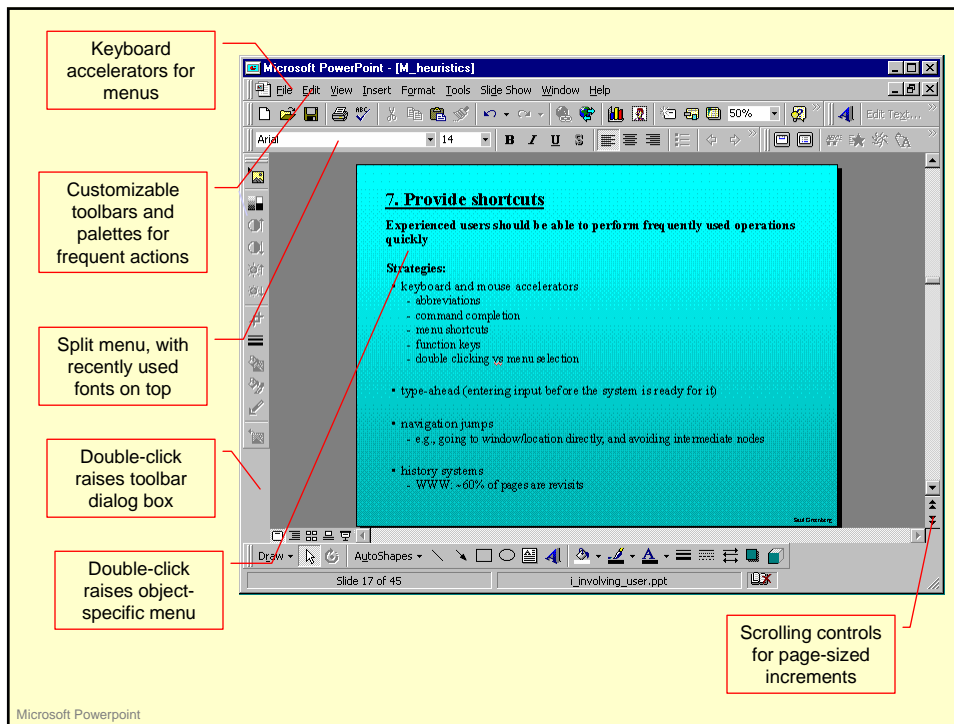


7. Provide shortcuts

Experienced users - perform frequent operations quickly

Strategies:

- keyboard and mouse accelerators
 - abbreviations
 - command completion
 - context menus
 - function keys
 - double clicking vs menu selection
- type-ahead (entering input before the system is ready for it)
- navigation jumps
 - e.g., going to window/location directly, and avoiding intermediate nodes
- history systems
 - WWW: ~60% of pages are revisits



8: Deal with errors in a positive manner

People will make errors!

Errors we make

- Mistakes
 - conscious deliberations lead to an error instead of correct solution
- Slips
 - unconscious behaviour gets misdirected en route to satisfying goal
 - e.g. drive to store, end up in the office
 - shows up frequently in skilled behaviour
 - usually due to inattention
 - often arises from similar actions



© Saul Greenberg

Designing for slips

General rules

- prevent slips before they occur
- detect and correct slips when they do occur
- user correction through feedback and undo

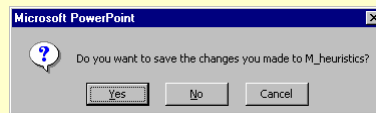


© Saul Greenberg

Types of slips

Capture error

- frequently done activity takes charge instead of one intended
- occurs when common & rarer actions have same initial sequence
 - change clothes for dinner and find oneself in bed (William James, 1890)
 - confirm saving of a file when you don't want to delete it
- minimize by
 - make actions undoable instead of confirmation
 - allows reconsideration of action by user
 - e.g. open trash to undelete a file



Types of slips

Description error

- intended action similar to others that are possible
 - usually occurs when right & wrong objects physically near each other
 - pour juice into bowl instead of glass
 - throw sweaty shirt in toilet instead of laundry basket
 - move file to wrong folder with similar name
- minimize by
 - rich feedback
 - check for reasonable input, etc.
 - undo

Types of slips

Loss of activation

- forget what the goal is while undergoing the sequence of actions
 - start going to room and forget why you are going there
 - navigating menus/dialogs & can't remember what you are looking for
 - but continue action to remember (or go back to beginning)!
- minimize by
 - if system knows goal, make it explicit
 - if not, allow person to see path taken

Saul Greenberg

Types of slips

Mode errors

- people do actions in one mode thinking they are in another
 - refer to file that's in a different directory
 - look for commands / menu options that are not relevant
- minimize by
 - have as few modes as possible (preferably none)
 - make modes highly visible

Saul Greenberg

Generic system responses for errors

General idea: Forcing functions

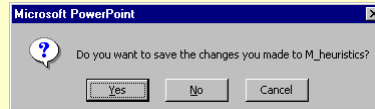
- prevent / mitigate continuation of wrongful action

Gag

- deals with errors by preventing the user from continuing
 - eg cannot get past login screen until correct password entered

Warn

- warn people that an unusual situation is occurring
- when overused, becomes an irritant
 - e.g.,
 - audible bell
 - alert box



Saul Greenberg

Generic system responses for errors

Do nothing

- illegal action just doesn't do anything
- user must infer what happened
 - enter letter into a numeric-only field (key clicks ignored)
 - put a file icon on top of another file icon (returns it to original position)

Self-correct

- system guesses legal action and does it instead
- but leads to a problem of trust
 - spelling corrector

Saul Greenberg

Generic system responses for errors

Lets talk about it

- system initiates dialog with user to come up with solution to the problem
 - compile error brings up offending line in source code

Teach me

- system asks user what the action was supposed to have meant
- action then becomes a legal one

Saul Greenberg

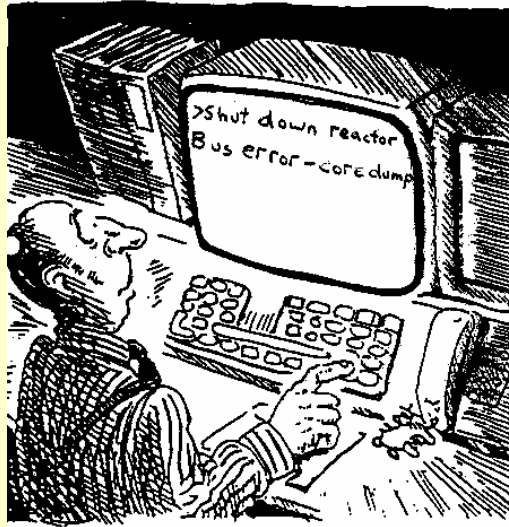
8: Deal with errors in a positive manner



What is "error 15762"?

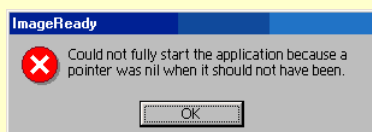
Saul Greenberg

8: Deal with errors in a positive manner

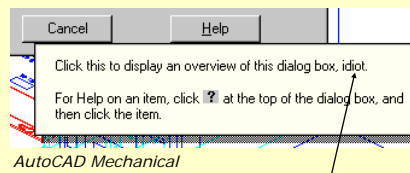


A problematic message to a nuclear power plant operator

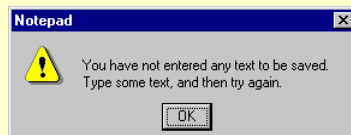
8: Deal with errors in a positive manner



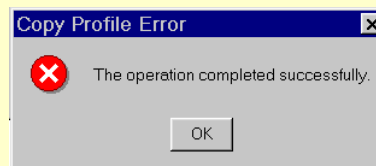
Adobe's *ImageReady*



AutoCAD Mechanical



Windows Notepad



Microsoft's NT Operating System

8: Deal with errors in a positive manner

Provide meaningful error messages

- error messages should be in the user's task language
- don't make people feel stupid

Try again, bonehead!

Error 25

Cannot open this document

Cannot open "chapter 5" because the application "Microsoft Word" is not on your system

Cannot open "chapter 5" because the application "Microsoft Word" is not on your system. Open it with "Teachtext" instead?

Saul Greenberg

8: Deal with errors in a positive manner

Prevent errors

- try to make errors impossible
- modern widgets: can only enter legal data

Form1

Date: [] [] []

Month Day Year

May 22 1997

Month Day Year

May 22 1997

Appointment

General Attendees Notes Planner

When

Start: 8:30AM Wed 5 /14 /97

End: 4:30PM Wed 5 /14 /97

All day

Description:

Smart Technology Ser

May 1997

S	M	T	W	T	F	S
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Where: []

Provide reasonableness checks on input data

- on entering order for office supplies
 - 5000 pencils is an unusually large order. Do you really want to order that many?

Saul Greenberg

Manuals...



9. Provide help

Help is not a replacement for bad design!

Simple systems:

- walk up and use; minimal instructions

Most other systems

- feature rich
- simple things should be simple
- learning path for advanced features



Documentation and how it is used

Many users do not read manuals

- prefer to spend their time pursuing their task

Usually used when users are in some kind of panic

- paper manuals unavailable in many businesses!
 - e.g. single copy locked away in system administrator's office
- online documentation better
- good search/lookup tools
- online help specific to current context

Sometimes used for quick reference

- syntax of actions, possibilities...
- list of shortcuts ...

Saul Greenberg

Types of help

Tutorial and/or getting started manuals

- short guides that people are likely to read when first obtaining their systems
 - encourages exploration and getting to know the system
 - tries to get conceptual material across and essential syntax
- on-line "tours", exercises, and demos
 - demonstrates very basic principles through working examples

Saul Greenberg

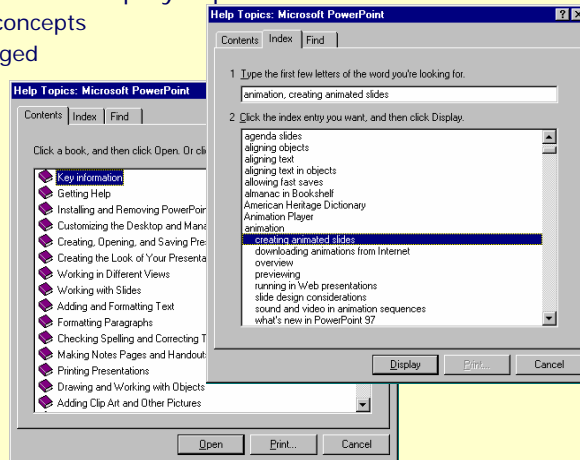
Types of help

Reference manuals

- used mostly for detailed lookup by experts
 - rarely introduces concepts
 - thematically arranged
- on-line hypertext
 - search / find
 - table of contents
 - index
 - cross-index



Microsoft Help



Saul Greenberg

Types of help

Reminders

- short reference cards
 - expert user who just wants to check facts
 - novice who wants to get overview of system's capabilities
- keyboard templates
 - shortcuts/syntactic meanings of keys; recognition vs. recall; capabilities
- tooltips and other context-sensitive help
 - text over graphical items indicates their meaning or purpose



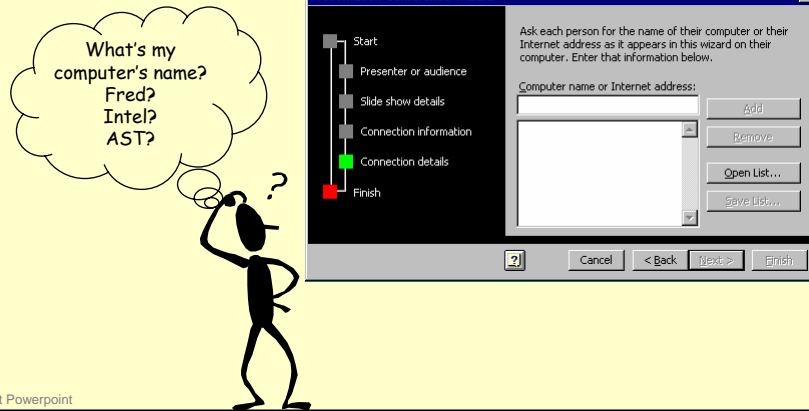
Microsoft Word

Saul Greenberg

Types of help

Wizards

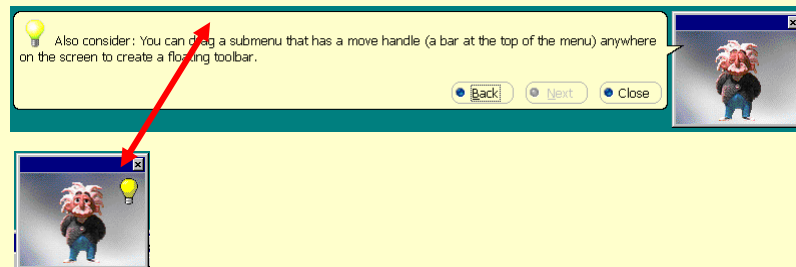
- walks user through typical tasks
- *but* dangerous if user gets stuck



Types of help

Tips

- migration path to learning system features
- also context-specific tips on being more efficient
- must be "smart", otherwise boring and tedious



Other Guidelines: Style guides

Guidelines published by producers of graphical user interfaces (GUIs)

- examples:
 - Open Software Foundation MOTIF
 - Open Look
 - MS Windows
 - Apple

Describes the “look and feel” of the GUI

- e.g. Open Look
 - grouping items in the same menu:
 - Use white space between long groups of controls on menus or in short groups when screen real estate is not an issue

Good, but hard to follow

- GUI and widget specific
- vast number of guidelines
- may miss fundamental design principles

Saul Greenberg

Example Motif Style Guide, Release 1.1

Message Dialogs

Description

MessageDialogs should be used to convey a message to the user. They must not interrupt the user's interaction with the application. They should include a message, and one of the following button arrangements.

- OK
- OK Help
- OK Cancel
- OK Cancel Help
- Yes No
- Yes No Help
- Yes No Cancel
- Yes No Cancel Help
- Cancel
- Cancel Help
- Retry Cancel
- Retry Cancel Help

Related Information

For more information, see the reference pages for DialogBox, ErrorDialog, InformationDialog, QuestionDialog, WorkingDialog, and WarningDialog

Information Dialog

Description

An InformationDialog should be used to convey information to the user. It must not interrupt the user's interaction with the application. It should include an information symbol, a message, and one of the following button arrangements.

- OK
- OK Help

Illustration



Related Information

For more information, see the reference page for DialogBox

Saul Greenberg

Other Guidelines: Widget-level “guides”

Toolkit “hard-wires” guidelines

- repertoire of widgets
- look & feel of particular widgets
- grouping behaviour of widgets

Outside of “normal” programmer’s control

- easier to use defaults than to re-invent the wheel!

Some toolkits

- look & feel is programmer-settable or platform-dependent

Advantages:

- easy to be consistent
- widgets developed by experts (graphical designers, etc.)

Disadvantages

- can be hacked around
- interfaces “assembled” by non-interface designers can still be terrible

Saul Greenberg

You know now

Nine principles of design

- Simple and natural dialog
- Speak the user’s language
- Minimize user’s memory load
- Be consistent
- Provide feedback
- Provide clearly marked exits
- Provide shortcuts
- Deal with errors in a positive manner
- Provide help

Heuristic evaluation

- Principles can be used to systematically inspect the interface for usability problems

Saul Greenberg

Evaluating Heuristic evaluation

- Problems found by a single inspector
- Problems found by multiple inspectors
- Individuals vs. teams
- Self guided or scenarios?

Saul Greenberg

Problems found by a single inspector

Average over six case studies

- 35% of all usability problems;
- 42% of the major problems
- 32% of the minor problems

Not great, but

- finding some problems with one evaluator is *much* better than finding no problems with no evaluators!



Saul Greenberg

Problems found by a single inspector

Varies according to

- difficulty of the interface being evaluated
- the expertise of the inspectors

Average problems found by:

- novice evaluators - 22%
 - no usability expertise
- regular specialists - 41%
 - expertise in usability
- double specialists - 60%
 - experience in both usability and the particular kind of interface being evaluated
 - also find domain-related problems

Tradeoff

- novices poorer, but cheaper!

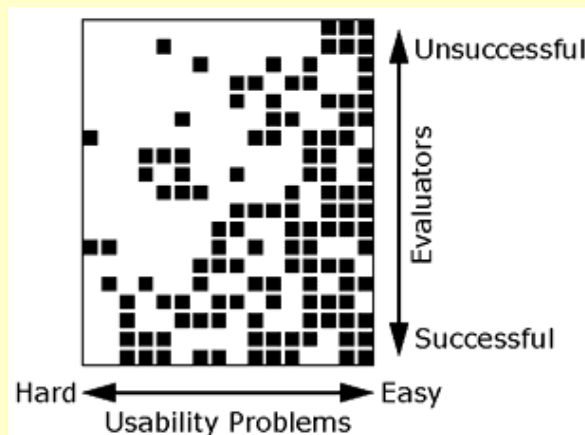


Saul Greenberg

Problems found by a single inspector

Evaluators miss both easy and hard problems

- 'best' evaluators can miss easy problems
- 'worse' evaluators can discover hard problems

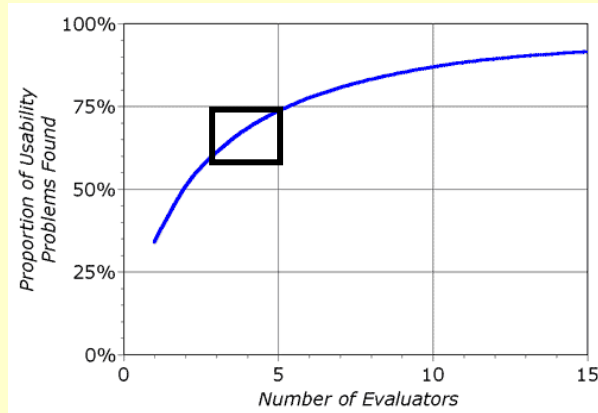


Saul Greenberg

Problems found by multiple evaluators

3-5 evaluators find 66-75% of usability problems

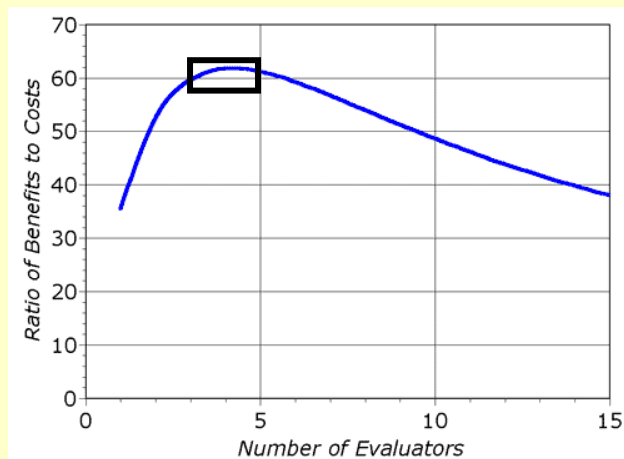
- different people find different usability problems
- only modest overlap between the sets of problems found



Saul Greenberg

Problems found by multiple evaluators

Where is the best cost/benefit?



Saul Greenberg

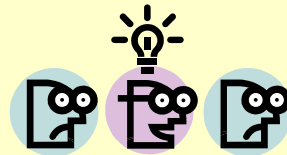
Individuals vs teams

Nielsen

- recommends individual evaluators inspect the interface alone

Why?

- evaluation is not influenced by others
- independent and unbiased
- greater variability in the kinds of errors found
- no overhead required to organize group meetings



Saul Greenberg

Self Guided vs Scenario Exploration

Self-guided

- open-ended exploration
- Not necessarily task-directed
- good for exploring diverse aspects of the interface, and to follow potential pitfalls

Scenarios

- step through the interface using representative end user tasks
- ensures problems identified in relevant portions of the interface
- ensures that specific features of interest are evaluated
- but limits the scope of the evaluation - problems can be missed

Saul Greenberg