Evaluating the Design without Users

- Cognitive Walkthrough
- Action Analysis
- Heuristic Evaluation

Cognitive walkthrough

- A formalized way of imagining people’s thoughts and actions when they use an interface for the first time

Steps in a cognitive walkthrough:
- you have a prototype or a detailed design description of the interface and you know who the users will be
- you select one of the tasks that the design is intended to support
- try to tell a believable story about each action a user has to take to do the task
- to make the story believable you have to motivate each of the user’s actions, relying on the user’s general knowledge and on the prompts and feedback provided by the interface
- if you can’t tell a believable story then you’ve located a problem with the interface
Problems to look for …

• question assumptions about what the users will be thinking
• identify controls that are obvious to the design engineer but may be hidden from the user’s point of view
• suggest difficulties with labels and prompts
• note inadequate feedback
• uncover shortcomings in the current specification (i.e. not in the interface itself but in the way that it is described)
• help insure that the specs are complete
• may require the designer to go back and talk to the user

Who and when?

• Small piece of an interface
  – you can do your own informal “in your head” walkthrough to monitor the design as you work

• larger parts of an interface
  – get together with a group a people, including other designers and users, and do a walkthrough for a complete task

NOTE: the walkthrough is a tool for developing the interface, not validating it. You should go into the walkthrough expecting to find things that can be improved

• Can be performed any time throughout the design process but especially useful in the earlier stages.
Information required

1. A description or prototype of the interface

2. a task description

3. a complete, written list of the actions needed to complete the task with the interface (e.g. scenario)

4. an idea of who the users will be and what kind of experience they’ll bring to the job

Performing the walkthrough ...

- Keep four key questions in mind as you critique the story:
  - Will users be thinking the designers assume they will be thinking?
  - Will users see the control (button, menu, switch, etc.) for the action?
  - Once users find the control, will they recognize that it produces the effect they want?
  - After the action is taken, will users understand the feedback they get, so they can go on to the next action with confidence?
Documenting the walkthrough

- Important to document the cognitive walkthrough to keep a good record of what needs improvement
  - produce standard evaluation forms
  - cover form would list the four pieces of information (description of the prototype, description of the task, written list of the actions, description of the users) as well as identify the date and time of the walkthrough and the names of the evaluators
  - for each action, a separate standard form is filled out
  - document all negative answers on a separate usability problem report sheet
    - should indicate the system being built (the version, if any)
    - the date
    - the evaluators
    - a detailed description of the usability problems, severity of the problems (how often it might occur and the impact on the user)

Example walkthrough

1. A description or prototype of the interface
2. a task description
3. a complete, written list of the actions needed to complete the task with the interface
4. an idea of who the users will be and what kind of experience they’ll bring to the job
#1 Are users thinking what the designers assume they will be thinking?

#2 Can users see the control (button, menu, switch, etc.) for the action?

#3 Once users find the control, can they recognize that it produces the effect they want?

#4 After the action is taken, will users understand the feedback they get, so they can go on to the next action with confidence?

Common mistakes

- the “written list of the actions” is merged into the walkthrough
  - the director makes up the actions as the walkthrough proceeds
  - often ends up stumbling through the interface
  - start with a correct list of the individual actions

- the walkthrough is tested on real users on the system
  - the walkthrough is an evaluation tool that helps you and your co-workers apply your design expertise to the evaluation of the interface (NOT tested with real users)
  - because you have identified the behaviour of the whole class of users, you should identify more problems than you would find a single, unique user to find
What to do with the results?

- Fix the interface!!
  - many of the fixes will be obvious
  - harder problem exists when the user doesn’t have any reason to think that an action needs to be performed

Evaluating the Design without Users

- Cognitive Walkthrough
- Action Analysis
- Heuristic Evaluation
**Action analysis**

- Examine the sequence of actions that a user has to perform to complete a task with an interface

- keystroke-level analysis (formal)
  - extreme detail
  - can be used to make accurate predictions
  - (even before the interface is prototyped)
  - difficult

- back of the envelope
  - less detail
  - can't make accurate predictions of task time
  - can reveal large-scale problems that might otherwise get lost
  - doesn't take a lot of effort

**Two fundamental phases**

1. Decide what physical and mental steps the user will perform to complete one or more tasks with the interface

2. Analyze those steps looking for problems
   - takes too many steps to perform a simple task
   - takes too long to perform a task
   - too much to learn about the interface
   - uncover holes in the interface
Keystroke level analysis

- Sometimes referred to as a “formal action analysis”
- Each task is broken down into unit tasks
  - Start with a basic task and iteratively divide it into subtasks until you reach unit tasks (i.e., a keystroke)
- Estimate the time to perform each unit task (physical or mental)
  - Estimates are found by testing hundreds of individual users, thousands of individual actions, and then calculating averages
  - These values have been determined for most of the common actions that users perform with computer interfaces
- Times are totaled to predict the time for the action

Average times for computer interface actions

<table>
<thead>
<tr>
<th>Physical movements:</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter one keystroke on a standard keyboard</td>
<td>0.28</td>
</tr>
<tr>
<td>Use mouse to point at an object on the screen</td>
<td>1.5</td>
</tr>
<tr>
<td>Move hand to pointing device or function key</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Visual perception:</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Respond to a brief light</td>
<td>0.1</td>
</tr>
<tr>
<td>Recognize a six-letter word</td>
<td>0.34</td>
</tr>
<tr>
<td>Move eyes to a new location on the screen (saccade)</td>
<td>0.23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mental Actions</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieve a simple item from long-term memory</td>
<td>1.2</td>
</tr>
<tr>
<td>Learn a single “step” procedure</td>
<td>25</td>
</tr>
<tr>
<td>Execute a mental “step”</td>
<td>0.075</td>
</tr>
<tr>
<td>Choose among methods</td>
<td>1.2</td>
</tr>
</tbody>
</table>
Back of the envelope analysis

- similar to keystroke analysis:
  - List the actions
  - think about them
- difference:
  - don't need to spend a lot of time developing a detailed hierarchical breakdown of the task
  - instead, use a “natural” series of actions
- every action takes at least two or three seconds
  - selecting something from a menu with the mouse
  - entering a file name
  - deciding whether to save under a new name or the old one
  - remembering your directory name
- or observe a few users and time the actions
- useful for deciding whether or not to add features to an interface

Bank Machine A - withdraw $20

- Put card in slot (5 seconds)
- Select English or French (2 seconds)
- Enter 4-digit PIN number (5 seconds)
- Press Ok (2 seconds)
- Select withdraw (2 seconds)
- Select Account (2 seconds)
- Type in 30.00 (7 seconds)
- Select “Yes” to confirm the transaction (2 seconds)
- Remove money from machine (5 seconds)
- Select “No” when asked if you have more transactions to do (2 seconds)
- Remove card from machine (5 seconds)
- Remove transaction record from machine (5 seconds)

19 Actions ~ 44 seconds
Bank Machine B - withdraw $20

- Put card in slot (5 seconds)
- Enter 4-digit PIN number (5 seconds)
- Select withdraw (2 seconds)
- Select Account (2 seconds)
- Select $20 (2 seconds)
- Remove money from machine (5 seconds)
- Remove card from machine (5 seconds)
- Remove transaction record from machine (5 seconds)

11 Actions 31 seconds

Bank Machine A - withdraw $20 & deposit $34.99

- Put card in slot
- Select English or French
- Enter 4-digit PIN number
- Press Ok
- Select Withdraw
- Select Account
- Type in 3 0 . 0 0
- Select “Yes” to confirm the transaction
- Remove money from machine
- Select “Yes” when asked if you have more transactions to do

- Select Deposit
- Select Account
- Type in 3 4 . 9 9
- Put envelope into machine
- Select “No” when asked if you have more transactions to do
- Remove card from machine
- Remove transaction record from machine
Bank Machine B - withdraw $20 & deposit $34.99

- Put card in slot
- Enter 4-digit PIN number
- Select withdraw
- Select Account
- Select $20
- Remove money from machine
- Remove card from machine
- Remove transaction record from machine

- Put card in slot
- Enter 4-digit PIN number
- Select Deposit
- Select Account
- Type in 3 4 . 9 9
- Put envelope into machine
- Select “No” when asked if you have more transactions to do
- Remove card from machine
- Remove transaction record from machine

GOMS

- A GOMS analysis produces quantitative and qualitative predictions of how people will use a proposed system
  - evaluate alternative systems before buying one
  - evaluate rival designs at the specification stage (before prototype is built)
- predict execution time of tasks that skilled users are likely to perform
  - “cognitive skill”

- GOALS
- OPERATORS
- METHODS
- SELECTION
GOMS Model

• Goals
  – user goals that describe what the user wants to achieve
    • finish each level, increase your standing (points or coins), avoid danger

• Operators
  – actions the user can take to achieve the goal
    • functional level
      – appear in the instructional manuals as direct commands or warnings
      – “Hit blocks … A useful item might pop out!” might be implemented as a
        functional-level operator called search-in-block
    • keystroke level
      – pressing buttons on the controllers

GOMS Model

• Methods
  – sequences of motor operators for accomplishing the functional-level
    operators
    • keystrokes necessary to pick up a shell, run with it, throw it at a block
      on the ground, and thereby break the block to find the treasure within

• Selection
  – prediction of the method to be used (normally based on rules)
    • how to kill an enemy? Jump on its back or hit it with Mario’s tail
    • the jump on its back method gives 200 more points
When to use GOMS

- For tasks involving “cognitive skill”
  - “expert” behaviour
  - users know what to do, all they have to do is act

Why use GOMS?

- Can help predict what a person will do in seemingly unpredictable situations
- To better understand “why” something is slower/faster
- Can find usability problems not found through normal development or other means of analysis

How to do a GOMS analysis

- $K = 0.2$ sec Keying: The time it takes to tap a key on the keyboard
- $P = 1.1$ sec Pointing: The time it takes a user to point to a position on the display
- $H = 0.4$ sec Homing: The time it takes a user’s hand to move from the keyboard to the input device or from the input device to the keyboard
- $M = 1.35$ sec Mentally preparing: The time it takes a user to prepare mentally for the next step
- $R$ Responding: The time a user must wait for a computer to respond to input
GOMS Calculations

- **Step #1**
  - calculate the time it takes to perform a method
  - e.g. move your hand from the input device to the keyboard and type a letter
  - GOMS gestures: HK
- **Step #2**
  - figure out at what points the user will stop to perform an unconscious mental operation (M)
    - can use predefined heuristics
  - H M K
- **Step #3**
  - sum up the timings to get a total time for the method

Example: GOMS meets the phone company

- Small differences in time per call can result in large monetary savings
- NYNEX
  - each second reduction per call in work time for its Toll & Assistance Operators saves three million US dollars per year!
  - New workstations
    - promised up to 2.5 seconds reduction (or $7.5 million US per year)
    - capital cost of 10-15 thousand dollars per workstation
    - 1,000 workstations
- Initially used experimental studies and a GOMS analysis
  - models such as GOMS have the potential to replace empirical trials but had not been validated with large-scale, complex systems
  - compare the empirical data with the GOMS model for accuracy, reliability, and cost
Results

- Experimental data revealed that operators were SLOWER with the new workstations!
  - Significantly slower
  - 4% difference which would translate to about one second loss
  - cost of three million dollars per year

- Obvious possibilities
  - resistance to the new terminals
  - still learning the new system

- The GOMS analysis revealed WHY the old workstation was faster than the new

Hierarchical Task Analysis (HTA)

- This evaluation technique provides a good model of the structure and organization of an interface and its elements.

- It involves writing down the process of proceeding towards a goal, often right down to the keystroke (button selection).
Hierarchical Task Analysis (HTA)

- Define a task to complete.
- In words, break that task down into subtasks, break those subtasks into individual actions.
- Each task has its own set of nested subtasks, each subtask has its own set of actions.
- Following an HTA is just like following a program in pseudocode.

Hierarchical Task Analysis - FIFA 2000

0. Set up Playstation to play a regular game. [Plan 0: 1, 2, 3, 4, 5]
1. Set up Playstation.
2. Load the program.
3. Choose mode to play a regular game.
4. Choose teams. [Plan 4: If any of the leagues of teams are not as desired then repeat 4.1 and/or 4.2 as required, (4.3 or 4.4)] Note that only one player at a time can make selections.
   4.1. Press ↑ buttons until Team 1 or Team 2 League Selection box is highlighted.
       4.1.1. Press ◄ buttons until desired league is displayed.
   4.2. Press ↓ buttons until Team 1 or Team 2 Team Selection box is highlighted.
       4.2.1. Press ► buttons until desired team is displayed.
   4.3. Press <START> button. Screen displays F5-Side Select screen.
   4.4. Press ● button. Control switches to bottom RH icon menu.
       [Plan 4.4: (4.4.1, 4.4.2) or 4.3]
       4.4.1. Press ◄ buttons until “Forward” icon is highlighted.
       4.4.2. Press × button. Screen displays F5-Side Select screen.
5. Choose which team to control.
Action Flow Diagram

- By determining all possible routes between screens, we can get a good idea of the type of navigation system involved in the front-end. Some systems are easier to navigate than others.

- Use the HTA to help construct a diagram of the process.
- Each screen is represented and arrows are labelled with actions required to get to the next (or previous) screen.
GOMS vs. cognitive walkthrough

Evaluating the Design without Users

- Cognitive Walkthrough
- Action Analysis
- Heuristic Evaluation
Heuristic Evaluation

- Discount Usability Engineering -- Jakob Nielson

  - find the usability problems in a user interface design so that they can be attended to as part of an iterative design process
  - systematic inspection of a user interface design for usability
    - look at an interface and try to come up with an opinion about what is good and bad about the interface
    - have a small set of evaluators examine the interface and judge its compliance with recognized usability principles (the heuristics).
    - some usability guidelines have thousands of rules, but Nielson’s discount method involves 10 rules based on broader heuristics

Heuristic evaluation process

- # of evaluators
  - individual evaluators will miss most of the usability problems
    - in a study, single evaluators found only 35% of the usability problems
  - different evaluators tend to find different problems
  - can achieve substantially better performance by aggregating the evaluations from several evaluators
  - cost-benefit analysis
    - use more evaluators when usability is critical or when large payoffs can be expected due to extensive or mission critical use of a system
  - recommendations:
    - around five evaluators
    - minimum of three
Heuristic evaluation process

- each individual evaluator inspects the interface alone
- only after ALL evaluations have been completed are the evaluators allowed to communicate and have their findings aggregated
- ensures independent and unbiased evaluations from each evaluator
- results of an evaluations can be recorded in two ways:
  - written report from the evaluator
  - having an observer take notes as the evaluator vocalizes their comments as they go through the interface
Heuristic evaluation process

- **Evaluation session:**
  - typically lasts 1-2 hours
  - the evaluator goes through the interface several times and inspects various dialogue elements and compares them with a list of recognized usability principles
  - can also consider any additional usability principles or results that come to mind that may be relevant
  - evaluators decide on their own how they want to proceed with evaluating the interface
  - general recommendation: go through the interface at least twice
    - once to get a feel for the flow of the interaction and the general scope of the system
    - a second time to focus on the specific interface elements while knowing how they fit the larger whole

Heuristic evaluation process

- **Domain-dependent systems**
  - evaluators will be naïve with respect to the domain of the system, and therefore must be assisted to enable them to use the interface
  - supply the evaluators with a typical usage scenario, listing the various steps a user would take to perform a few realistic tasks
    - constructed on the basis of a task analysis of the actual users and their work in order to be a representative as possible

- **Output**
  - list of usability problems in the interface, annotate with references to those usability principles that were violated by the design
  - does not provide a systematic way to generate fixes to the usability problems or a way to assess the probably quality of any redesign
    - often easy to correct based on the guidelines for each usability principle violated
    - many are obvious to fix once they have been identified
Heuristic evaluation process

- Debriefing session
  - extending the heuristic evaluation method to provide some design advice
  - participants include the evaluators, any observers used in the evaluation sessions, representative of the design team
  - conducted in brainstorming mode
  - focus on discussions of possible redenigns to address the major usability problems
  - also a good opportunity to discuss the positive aspects of the design

Usability guidelines:

- simple and natural dialogue
- speak the users' language
- minimize the users' memory load
- consistency
- feedback
- clearly marked exits
- shortcuts
- good error messages
- prevent errors
- help and documentation
Simple and natural dialogue

- Ideal:
  - present exactly the information the user needs and no more at exactly the time and place where it is needed
  - information that will be used together should be displayed together
  - information objects and operations should be accessing in a sequence that matches the way users will most effectively and productively do things

- Simplify the user interfaces as much as possible
  - every additional feature or item on the screen is one more thing to learn, one more thing to possibly understand, one more thing to search through, etc.
- Information should appear in a natural and logical order

Simple and natural dialogue (continued)

- Graphic design and color
  - screen layout
    - principles of proximity, alignment, contrast, repetition
    - mumble screens
  - colour
    - don’t over do it
    - make sure the interface can be used without colours
    - use colour to categorize, differentiate and highlight, not to give information, especially quantitative information
    - also applies to features -- one more thing to learn, one more thing to potentially get wrong, manual gets bigger, more intimidating and harder to search
Simple and natural dialogue (continued)

• Less is more
  – limit extraneous information
    • adding information can distract the user from the primary information, can confuse the novice user, and can slow down the expert user
    • better to design a single screen with important information and put less important information on auxiliary screens
  – limit extraneous functions
    • “by providing lots of options and several ways of doing things we can satisfy everybody”
    • also applies to features -- one more thing to learn, one more thing to potentially get wrong, manual gets bigger, more intimidating and harder to search
    • possibly design different complexity of interfaces (novice and expert mode)

Speak the users’ language

• Terminology
  – should be based on users’ language, not on system-orientated terms
  – should be in the users’ native language, including nonverbal elements like icons
  – try not to use words with non-standard meanings
    • might be non-standard in the general population but is standard in the user community
  – if the users population has its own specialized terminology, use those terms rather than more commonly used, but less precise, everyday language
  – view interactions from the users’ perspective
  – don’t force naming conventions or restrictions on objects named by users
  – allow for the use of aliases
Speak the users’ language

• Mappings and metaphors
  – good mapping between the computer display of information and the user’s conceptual model of the information
    • world map
  – user interface metaphors
    • mapping between the computer system and some reference system known to users in the real world
    • can be powerful, misleading, and limiting
    • problems with internationalization

Minimize user memory load

• Easier to recognize something than to recall from memory
  – computer should display dialogue elements and let the user choose
    • menus
  – also easier to modify information rather than generate it all from scratch
    • file completion, default
  – the objects must be visible to the user but too many objects will result in a loss of salience for the objects of interest
    • “less is more”
  – system should describe the required format and provide an example of legal and sensible input (i.e. a default value)
    • date
  – system should supply the range of legal input and unit of measurement
Minimize user memory load

- System should be based on a small number of rules that apply throughout the user interface
  - if there are a large number of rules
    • will have to learn, and remember all those rules
  - if there are no rules
    • will have to learn every single dialogue element on its own
  - generic commands
    • support the transfer of learning from one application to another
    • may not support exactly the same command but the user can “think” of the command as a single unified concept

Consistency

- The same information should be presented in the same location on all screens and all dialog boxes and it should be formatted in the same way to facilitate recognition

- if users know the same command or action will always have the same effect they will feel more confident in using the system

- also includes consideration of task and functionality structure of the system
Feedback

• The system should continuously inform the user about what it is doing and how it is interpreting the user’s input
  – shouldn’t wait until an error occurs
  – provide positive feedback and partial feedback
  – system feedback should not be expressed in abstract and general terms but should restate and rephrase the user’s input to indicate what is being done with it
  – different types of feedback need different degrees of persistence
    • (low, medium and high persistence)

Feedback

• Response times
  – feedback especially important where the system has long response times
  – Guidelines:
    • 0.1 second - limit for having the user feel that the system is reacting instantaneously. No special feedback necessary
    • 1.0 second - limit for the user’s flow of thought to stay uninterrupted. The user will notice the delay but normally, no special feedback is necessary.
    • 10 seconds - limit for keeping the user’s attention focussed on dialogue. For longer delays the user will want to perform other tasks while waiting. Computer should give feedback indicating when it will be done. Especially important when response times are highly variable.
  – Response time TOO fast
Feedback

• Response times (continued)
  – progress indicators
    • reassure the user that the system has not crashed but is working on the problem
    • indicate approximately how long the user can be expected to wait
    • provide something for the user to look at, making the wait less painful
    • for fast operations, the indicator is probably overkill, providing distraction

• System failure
  – informative feedback should be given when the system fails
  – “NO” feedback is the worst possible feedback

Clearly marked exits

• System should offer the user an easy way out of as many situations as possible
  • so the user doesn’t feel trapped
  • increase the user’s feeling of being in control
  – undo facility
    • should be able to undo anything, not restricted to special actions
    • acknowledge that users will make errors and therefore make it easy to recover from these errors
  – high degree of responsiveness
    • new actions get a higher priority than finishing old actions
    • must be visible to the user!
Shortcuts

- Should be possible for the experiences user to perform frequently used operations fast, using shortcuts
  - function keys, command keys, control sequences, double-clicking, right mouse-button clicking, gestures, etc.
  - type-ahead
  - interrupt
  - click-ahead
  - jump directly to the desired location
  - reuse interaction history
    - 35% of all commands were identical to one of the five previous commands
    - 74% of the commands had been issues at least once before
  - system provided defaults

Good error messages

- Critical because ....
  - represent situations where the user is in trouble and potentially will be unable to use the system
  - present opportunities to help the user understand the system better

- Four simple rules:
  - phrased in clear language and avoid obscure codes
  - be precise rather than vague or general (i.e. giving feedback by restating the user’s input)
  - should constructively help the user solve the problem
  - should be polite and not intimidate the user or put the blame on the user
- Should also provide good error recovery
  - undo, edit and reissue previous commands
Good error messages

• Multiple-level messages
  – use shorter messages that will be faster to read
  – give easy access to a more elaborate message
  – can be used to provide detailed, internal system messages to specialized support staff without confusing and intimidating the user

Prevent errors

• Try to design your system to avoid the error completely
  – instead of spelling things out, choose from a list
  – confirmation before executing actions that may have serious consequences
  – be careful of “capture error”
• Avoid modes
  – modes partition the user actions so that not all actions are available at all times which can be frustrating
  – the same user interaction will be interpreted differently depending on the current mode which can lead to errors
  – if you cannot avoid modes, prevent errors but explicitly recognizing the modes in the interface design
• Avoid having too-similar commands - “description error”
  – the descriptions of the two situations are almost identical and therefore likely to be confused.
Help and documentation

• Most users do not read manuals
• if users do want to read the manual, then they are probably in some kind of panic and will need immediate help
  – therefore, need good task-orientated search and lookup tools for manuals and online documentation
• on-line documentation has the potential for getting precise information faster than paper-based documentation
  – Superbook, 4.3 minutes (on-line) vs. 5.6 minutes (printed)
  – initially 7.6 minutes for both, indicated the importance of usability principles for documentation
• on-line documentation adds an extra set of features to a system therefore complicating the interface
  – shouldn’t need “help on how to get help”
  – users tend to scan rather than read long paragraphs of text
Help and documentation

• Quality of writing is important

• 3 stages of interacting with manuals
  – searching - locating information relevant to a specific need
  – understanding the information
  – applying - carry out a procedure as described in the documentation

• Search
  – two main search tools are the index and the overview map of the
    structure of the information space

Help and documentation

• Understand
  – information should be written without jargon and should be written
    in a way that corresponds to the tasks the user wants to do and the
    sequence of actions necessary
    • steps should be numbered
    • include examples

• Apply
  – keep online help systems visible in a separate window while the
    user returns to the main application
  – let the user copy examples directly from the help system to have
    them executed
  – should be stated in step-by-step procedures
  – easy to check that they are understood the instructions and carried
    them out correctly
Usability guidelines:

- simple and natural dialogue
- speak the users' language
- minimize the users' memory load
- consistency
- feedback
- clearly marked exits
- shortcuts
- good error messages
- prevent errors
- help and documentation

Designing for Error
Errors

- The system can’t correctly interpret the information given to it
  - Typically thought of as “a fault generated by the user” (assigns blame)
  - Why not misunderstandings, problems, confusions, ambiguities?
  - both participants assume equal responsibility in understanding

- Dealing with error:
  - eliminate or minimize errors
  - try to make it easier to deal with an error
    - providing a clear indication of the problem, possible causes & remedies
    - providing tools that make the correction easier
  - provide the kind of information that helps the user understand the implications of the actions being followed

Understanding Error

Mistakes
the user did the intended action but the action was not appropriate

Slips
the user performs an action that was not intended
- often occur with experienced users
- usually not as serious as mistakes
Minimizing error

Minimize error by following good user interface design principles

- Avoiding error through appropriate representation
  - every choice of representation provides a set of intrinsic properties, sometimes eliminating whole classes of errors
  - but every design decision is a tradeoff
  - “file doesn’t exist” example
- Avoiding false understandings
  - by giving people more information they may not make false inferences
  - exit vs. quit
- Minimizing errors that result from slips
  - mode errors - better feedback
  - description errors - better consistency
  - capture errors - better feedback

Detecting error

Early detection is extremely important

Slips are easier to detect than mistakes

- How should the system respond
  - find out what was intended so the system can proceed
  - warn when something inappropriate has taken place
- The problem of level
  - where is the problem
- Failure to detect problems
  - the action will accomplish a legal action so it cannot detect an error
  - relevance bias and partial explanation
Six responses to an error

- Gag
  - prevents the user from continuing
- warning
  - presents warning but will not restrict the user from performing the action
- do nothing
  - the action just doesn’t work
- self correct
  - system tries to guess some legal actions that might be what the user wants to do (DWIM)
- teach me
  - queries the user to find out what they meant
- let’s talk about it
  - initiate a dialog with the user

Correcting error

If an error is detected, the user needs to recover from it.

- “undo”
- perform a natural inverse
- provide structured help

- Essential information
  - what the current state of the system is
  - where one came from
  - what possible alternatives there are