CS 160: Lecture 21

Professor John Canny
Fall 2004
Design for Errors

Some favorite error messages:

* 404 not found
* Fatal error 312: aborting
* Syntax error near line 1025
* Internal compiler error
* Segmentation fault: core dumped
* Add your favorite here:
Design for Errors

Some favorite error messages:
* 404 not found
* Fatal error 312: aborting
* Syntax error near line 1025
* Internal compiler error
* Segmentation fault: core dumped

What do these messages communicate?
What is the solution to each?
Naïve users

- **Fatal error 312:**
  * User messed up and someone will die (computer?).
  * Solution: log off and pretend nothing happened.

- **404 not found**
  * The computer misplaced something (the “404”?).
  * Solution: Maybe it will find it if you try later...

- **Segmentation Fault: core dumped**
  * Something bad happened
  * Solution: someone will probably have to clean out the “core” stuff that was dumped inside the computer.
More sophisticated users

- Segmentation fault: core dumped
  * This lazy programmer slept through their programming classes and didn’t learn about array bounds checking, invariants or memory management.
  * Solution: Move them to the user support hotline
More sophisticated users

- 404 + the other messages
  * Why can’t programmers explain what is wrong in normal English?
  * Why do the messages sound like bad movie titles? “fatal error” etc.
  * Why do programmers usually assume that its the user that did something wrong when their silly program crashes?
User Testing

Solution for programmers:
* Test out error messages in person on your boss.
* E.g. walk into office, say “segmentation fault: core dumped”, deposit printout of memory on their desk, and walk out.

Why isn’t this a good idea?

Why would you put it into a program?
Human error recovery

- People make mistakes in communication all the time, and are adept at repairing them.
- Repair need not involve assignment of blame.

- E.g. “I’m sorry, I didn’t understand. Can you rephrase?”
  - Tells the speaker you heard what they said, but were unable to interpret it. They should repeat, but express it differently.
  - There is no assertion that what they said was ungrammatical or incorrect, and you accept some blame by default.
Aside: People attribute personality based on the kind of language others use.

For instance, psychologists have analyzed the personality of writers, politicians etc. from their writings.

They can classify them using standard personality categories, like Myers-Briggs.

People interact better with others when their personalities are compatible.
Human error recovery

 Reeves and Nass studied human interaction with text-based OS'es, found that similar results apply.

 i.e. people have better interactions with text interfaces when the system’s (text) personality is compatible with theirs.

 MS help wizards are a consequence of this work.
Humane error recovery

- In human communication, an error is the beginning of a process of repair. It is “normal” to make errors.

- In human-machine interaction, errors normally lead to the end of the current task. Errors are treated as “abnormal”.

- In other words, application programs often try to escape from the repair process, leaving the user stranded.
Types of errors

**Mistakes**
* User intended to do what they did, and it led to an error. User would probably do the same thing again.

**Slips**
* User did not mean to do what they did. They can recover by doing it differently again.
* Slips are not just for beginners. Experts often make them because they devote less conscious attention to the task (or mere presence or “demo” effects).
Minimizing Error

- **System errors**: Program defensively (assert, bounds check (please!!))

- Estimated loss of productivity due to Windows OS crashes $170 B.$

- Estimate for Windows XP over $20 B.$

- Note: most Windows XP exploits have been standard buffer overruns.
Minimizing Error

- **Note:** Omission of array-bound checking, esp. in I/O code, has no possible justification in a modern computer.
- Its sometimes justified on the basis of impact on performance (speed).
- If you believe that argument, you can make your car “faster” by removing its brake pads (the small air gap slows the car).
- Think about orders of magnitude - between instruction cycles and network/file system speed.
Minimizing Error

User errors:
* Use Intuitive command names.
* Include short explanations as “tool tips”.
* Put longer explanations in help system.
Minimizing Error

- Recognition over recall
  * Easier to select a file icon from a folder than to remember and type in the filename.
  * Auto-completion can help fix this.

- Use appropriate representations
  * E.g. graphical file selector good for choosing individual files
  * Textual file names support automation, richer organization (using command line options).
Typical Errors

From Norman’s 1983 survey:

Mode errors:
  * User forgets what mode they’re in, and does the command appropriate for another mode.
  * Digital watches, VCRs etc.
Mode Errors

Why are mode errors so common in certain devices (watches, VCRs)?

Several attributes:

* There aren’t enough command keys for all the operations – so the mode determines what each button does.
* There isn’t enough display space to provide strong feedback about the mode.
Mode Errors

Solutions:

* Strive for consistent behavior of buttons across modes.
* Provide display feedback about behavior of keys in the current mode.
* Provide an option for scrolling help tips if possible.
* Allow the device to be programmed externally (e.g. from a PC with Bluetooth).
Mode Errors

If you don’t have a tiny screen, then make the context clear!

i.e. use color, tabs, navigation graphics etc. to make clear to the user “where” they are in the interface.
Typical Errors

Description error:
* The action is insufficiently specified by the user.
* User may not know all the command line switches, or all the installation options for a program.
* Solution: Warn the user that the command is ambiguous, or “unusual”. Provide help about options in several standard ways.
Typical Errors

- **Capture error:** (aka the tongue twister error)
  - Command sequences overlap, and one is more common.
  - User reflexively does the common one when trying to do the unusual one.

  - E.g. try typing “soliton” very fast.

- **Solution:** be aware of and test for this error. Try different command names.
Detecting Errors

The earlier the better:
* Check for consistency whenever possible ("asserts" for user input).
* If there’s a high risk of error, check for unusual input, or for common slips (spelling correction).

E.g. google’s “did you mean XX?” response
System Response

1. **Stop** the user from continuing the way they were going (and possibly compounding the error).
2. Take “safe” recovery actions - e.g. auto-save the state with a unique name.
3. Begin the recovery process...
Break
Recovery

- Gag
- Warn
- Do Nothing
- Self Correct
- Teach Me
- Let’s talk about it
Gag Response

- Machine freezes, often not even accepting input.

- Generally a bad idea, but there are good(?) uses:
  * Raskin’s FLOW system, refuses to accept keystrokes that are not legal commands.
  * Intended for naïve users.
  * Even random typing produces legal programs!
Warn Response

- Machine accepts input, even if not legal. Warns user about problem(s).
- Allows user to get into deeper trouble.
- Allow backtracking “undo”, back to start of trouble if possible.
Do Nothing Response

- Machine accepts input, even if not legal. Machine does nothing.
- User has to figure out that something’s wrong.
- Usually a bad idea, but can be useful in automation/DB queries -
  * It allows scripts/queries to continue executing, but e.g. don’t copy a file onto itself.
DWIM (Do What I Mean) was an aggressive self-correcting system. Spell checkers are of this type.

**Generally good but:**
* Sometimes override user intent.
* Can frustrate on frequently-used strings: “naïve” is good but not “Hsi”.
* Solutions:
  + Don’t repeat correction if overridden.
  + Watch user behavior over time.
Let's talk about it

- Inspired by human error recovery.
  - Machine explains its understanding of the situation, gives the user some options.
  - Examples: network diagnostic wizard, modem wizard, ...

- Very difficult to program these, but they're very valuable.
  - Need some analysis of previous user problems.
  - Look at help desk logs.
Teach Me

- System informs user what they can do.
- Common in speech recognizers. Include command "what can I say?", or run speech tutorial.
Explanations

- The first part of the recovery process is to explain what appears to be wrong.
- Remember the user is only supposed to have a functional model of what’s going on. Try to give an explanation at high level.
- People understand basic resources like filesystem, memory etc.
- “I’m afraid the program has an internal fault in the code that <<reads and writes files>>, which was called when trying to <<save your user preferences>>. We regret the inconvenience, and are trying to recover...”
Recovery

- The system should always give the user a reasonable amount of information about the problem.
- Better to err on the side of too much information.
- Some problems are amenable to automatic repair: retry, use redundant information, backup data etc...
- DWIM (Do What I mean) was an editor that attempted to correct common user errors. You need an “undo” key to use this approach.
Recovery

- Run a “diagnostic wizard” for common or unreliable subsystems (e.g. the network).

- These can interact with the user to get extra information needed to fix the problem.
Research Approaches

- **Intentional interfaces**: build models of user intention from their actions.

- Assumes that people have clear goals that the system can understand.

- Can get confused if people multi-task or do sub-optimal things.
Research Approaches

- **Accountable systems**: The system maintains a user-oriented model of internal behavior, somewhere between a structural and a functional model.

- If an error occurs, this “account” can give a sensible explanation, and often recovery suggestions.
Summary

- Error messages are often masterpieces in bad communication. That’s not necessary.
- Error recovery is a “normal” process.
- Types of errors: slips and mistakes.
- Some common error types.
- Five responses to errors.
- Recovery.
- Research approaches.