Preventing Error

SWE 632, Spring 2018
Today

• What causes errors?
• What design choices can help reduce the frequency of errors?
• What design choices can help users resolve errors more effectively?
Human Error
Root cause analysis

• Keep asking **why** to determine causes for erroneous actions, and the causes of these causes

• Example
  • 2010 F-22 crash that killed pilot
  • Official cause: pilot error - pilot failed to take corrective action
  • Why did the pilot not take the action?
  • Pilot was not receiving oxygen and was probably unconscious.
What causes disasters?

• Mechanical malfunction?
• Poor design?
• Human error?
Swiss cheese model

- Accidents must penetrate levels of system defenses
- Reduce accidents by
  - Adding more layers
  - Reduce the size and number of holes
  - Alert users when holes line up
Psychological types of unsafe acts

- **Basic error types**
  - Slip
    - Attentional failures
      - Intrusion
      - Omission
      - Reversal
      - Misordering
      - Mistiming
    - Memory failures
      - Omitting planned items
      - Place-losing
      - Forgetting intentions
  - Lapse
    - Rule-based mistakes
      - Misapplication of good rule
      - Application of bad rule
    - Knowledge-based mistakes
      - Many verbal forms
  - Mistake
  - Violation
    - Routine violation
    - Exceptional violations
    - Acts of sabotage
Violation

• Error occurred because user intended the erroneous output
• Routine violation - user always intends to do it
  • Noncompliance is so frequent it is ignored
  • E.g., running a red light
• Exceptional - only in some cases
• Sabotage - intended destruction
Mistakes

• User **formulated** the wrong goal or plan
  • Executing action will not achieve goal
• Rule based: appropriately diagnosed situation, but chose erroneous course of action
  • Example: Night club attendees blocked from leaving during fire because bouncers thought they
• Knowledge based: does not have correct information
  • Example: Skidding driver feels brake vibrations, believes indicates malfunctioning breaks and takes foot off break, stopping ABS
Memory Lapse

- Failing to do all steps of a procedure, repeating steps, forgetting the outcome of an action, forgetting the goal or plan
- Often caused by interruption
  - Time between when plan was formulated and plan was executed leads to forgetting plan
  - Take a pen out to sign form, get interrupted talking to someone, leave it on desk rather than put it back in bag
Slips

- Attentional failure - user *intended* to do correct action, but did not actually execute action
- Example: forgot to turn off the gas burner on the stove after cooking
Strong habit intrusion

- Performance of some well-practiced activity in familiar surroundings
- Intention to depart from custom
- Failure to make an appropriate check
- Example: start trip to frequent destination, forget going somewhere else
Omissions

• May be interrupted, forgetting intention to act
• “I picked up my coat to go out when the phone rang. I answered it and then went out of the front door without my coat.”
Perceptual confusions

• Take frequent action very often, leading to high System 1 automation
• Don’t perform perceptual check to verify that System 1 action is the correct one to take
• Example: “I began to pour coffee into the sugar bowl”
Mistimed checks

• Highly automated System 1 activity that is interrupted
• Error in resuming activity because usually unconscious.
• Example - interrupted in the middle of tying shoes
Activity

- Think of the last unsafe act you performed.
- What was the underlying cause?
Designing for error
Designing for error

• Humans are not automatons and will never behave like automatons

• Easy to design for the situation in which everything goes well

• But important to think about what might go wrong and how the interaction design can ameliorate issues
IFT perspective

- IFT perspective
  - User exploring patches topology in search of prey
  - Always making a decision about whether a patch is the right place to hunt and changing as new information arrives

- Breaks down when user actions transform the state of the application
  - Patches and topology no longer fixed
  - Visiting a configuration of the system by clicking "Send" on the email editor is a not an undoable action
Some strategies for designing for errors

• Understand the cause, and fix it
• Make it possible to reverse errors
• Offer feedback that enables users to discover and correct errors
• Don't treat actions as errors, but as manipulations
Understand the causes of errors

• What errors occur? What type are they? How can they be prevented?
• Frequent contributing factors
  • Ambiguous or unclear information about the state of the system
  • Lack of an effective conceptual model
  • Inappropriate procedures
• Must design for users as they exist, rather than users as you'd like them to behave
Interruptions

- Interruptions are a frequent cause of error
- User may be using your interface perfectly, with the correct plan to get to their goal
  - What happens if, in the middle of the task, they answer a phone call?
  - Or if they run out of time, and come back the next day?
Designing for interruptions

• Help user resume task, by remembering where they were in task, what steps have been completed, and what steps remain
• Reduce the number of steps
• Use forcing functions to force users to do forgettable action (e.g., take card from before picking up cash)
Example: interruptions

- In groups of two or three
- Imagine a user was interrupted while using one of your project apps
- What errors might this create?
- What challenges might users experience when resuming?
- How could you change your design to address these issues?
Offer feedback for user actions

- Feedback helps keep users on track in accomplishing goals
  - Provide feedback early
  - Provide feedback consistently
- Make feedback visible, noticeable, legible, located w/ in users focus of attention
- Requesting confirmation can be used to prevent costly errors (but use sparingly)
Tone of feedback

- Establishes relationship with user
- Important not to take user feel “stupid”
- Make the system take blame for errors
- Be positive, to encourage
- Provide helpful messages, not cute messages
- Avoid violent, negative, demeaning, threatening terms (e.g., illegal, invalid)
System response times

- 0.1 second - reacting **instantaneously**
  - requiring no special feedback except displaying result
  - limit for direct manipulation of objects in UI
- 1.0 second - **freely** navigating commands
  - noticeable delay, limit for keeping user’s flow of thought uninterrupted
- 10 seconds - keeping users **attention**
  - limit for keeping user’s attention focus in UI
  - longer delays create task breaks
- [Nielsen, Usability Engineering, 1993]
Show users how to fix errors

- Good: detecting user errors
- Better: directly showing how errors can be fixed
- (Best: using constraints to prevent errors from ever occurring)
Direct manipulation
Motivation

- User is trying to do a task, manipulating a model of world
- Hard to plan out long sequence of actions in advance
- Gulf of execution: hard to know if took correct action
- Gulf of evaluation: hard to understand if successfully manipulated world
- Hard to compare hidden world to desired world
Key questions

- What is the cost of an error?
  - Is it low cost or high cost?
  - Is it undoable?
- What feedback is necessary for user to realize the system is not in the desired state?
Direct manipulation

• “Rapid incremental reversible operations whose impact on the objects of interest is immediately visible” (Shneiderman, 1982)
Benefits

• Supports exploration
  • Don’t plan long sequence of actions: pick an action, try it, can change mind if want to do something else instead
• Provides immediate feedback
  • Can quickly see what outcome of actions are in manipulating the world
  • Easy to compare desired state of the world to actual state of the world
## Example - Kayak

<table>
<thead>
<tr>
<th>DCA ↔ CHI</th>
<th>Dec 16 ↔ Dec 19</th>
<th>Economy</th>
<th>1 traveler</th>
</tr>
</thead>
</table>

### Stops
- **nonstop** $127
- **1 stop** $145
- **2+ stops** $303

### Times
- Take-off Washington (DCA)
  - Fri 2:41p – 10:30p

- Take-off Chicago (CHI)
  - Mon 5:30a – 10:00p

### Price
- **$207**
  - JustFly, Experience world-class service
  - Click "View Deal" to find our cheapest flights

### American Airlines
- **$227**
  - 8:12p DCA → 9:26p ORD
  - 2h 14m, nonstop

- **$227**
  - 8:12p DCA → 9:26p ORD
  - 1h 47m, nonstop

---

**Advice:** BUY

**Create a price alert**

**Show landing times**

**Airports**
- DCA: Reagan-National $127
- BWI: Baltimore/Washington $207
Example - Google Maps
Example - GUI builder
Example - Spreadsheets

![Spreadsheet Image]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>O</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>788</td>
<td>355</td>
<td>564</td>
<td>399</td>
<td>413</td>
<td>897</td>
<td>444</td>
<td>523</td>
<td>413</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>800</td>
<td>923</td>
<td>233</td>
<td>307</td>
<td>864</td>
<td>355</td>
<td>90</td>
<td>877</td>
<td>864</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>657</td>
<td>788</td>
<td>755</td>
<td>444</td>
<td>478</td>
<td>432</td>
<td>405</td>
<td>455</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>599</td>
<td>866</td>
<td>233</td>
<td>201</td>
<td>413</td>
<td>361</td>
<td>455</td>
<td>233</td>
<td>413</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>899</td>
<td>755</td>
<td>673</td>
<td>311</td>
<td>780</td>
<td>400</td>
<td>614</td>
<td>754</td>
<td>780</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>334</td>
<td>953</td>
<td>888</td>
<td>214</td>
<td>644</td>
<td>789</td>
<td>361</td>
<td>978</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>233</td>
<td>644</td>
<td>766</td>
<td>444</td>
<td>233</td>
<td>977</td>
<td>776</td>
<td>453</td>
<td>847</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>877</td>
<td>533</td>
<td>968</td>
<td>897</td>
<td>541</td>
<td>977</td>
<td>475</td>
<td>588</td>
<td>975</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>945</td>
<td>365</td>
<td>233</td>
<td>708</td>
<td>564</td>
<td>344</td>
<td>78</td>
<td>359</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>670</td>
<td>607</td>
<td>233</td>
<td>846</td>
<td>980</td>
<td>544</td>
<td>613</td>
<td>529</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>855</td>
<td>792</td>
<td>908</td>
<td>556</td>
<td>352</td>
<td>955</td>
<td>395</td>
<td>413</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>607</td>
<td>224</td>
<td>641</td>
<td>908</td>
<td>556</td>
<td>352</td>
<td>955</td>
<td>413</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>222</td>
<td>645</td>
<td>999</td>
<td>182</td>
<td>388</td>
<td>905</td>
<td>814</td>
<td>444</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>344</td>
<td>756</td>
<td>600</td>
<td>481</td>
<td>399</td>
<td>144</td>
<td>399</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>5455</td>
<td>4280</td>
<td>5058</td>
<td>5002</td>
<td>4521</td>
<td>4866</td>
<td>4084</td>
<td>4042</td>
<td>5687</td>
<td>3718</td>
<td>3990</td>
<td>5477</td>
<td>4796</td>
<td>5913</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>5455</td>
<td>4280</td>
<td>5058</td>
<td>5002</td>
<td>4521</td>
<td>4866</td>
<td>4084</td>
<td>4042</td>
<td>5687</td>
<td>3718</td>
<td>3990</td>
<td>5477</td>
<td>4796</td>
<td>5913</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>5455</td>
<td>4280</td>
<td>5058</td>
<td>5002</td>
<td>4521</td>
<td>4866</td>
<td>4084</td>
<td>4042</td>
<td>5687</td>
<td>3718</td>
<td>3990</td>
<td>5477</td>
<td>4796</td>
<td>5913</td>
<td></td>
</tr>
</tbody>
</table>

**U.K. Factories**

| 21  | 585 | 315 | 908 | 556 | 352 | 556 | 635 | 413 |     |    |    |    |    |    |    |
| 22  | 506 | 605 | 860 | 222 | 459 | 254 | 512 | 897 |     |    |    |    |    |    |    |
| 23  | 670 | 544 | 233 | 846 | 980 | 406 | 613 | 523 |     |    |    |    |    |    |    |
| 24  | 344 | 489 | 600 | 311 | 864 | 414 | 399 | 307 |     |    |    |    |    |    |    |
| 25  | 855 | 315 | 908 | 556 | 352 | 556 | 635 | 413 |     |    |    |    |    |    |    |
| 26  | 607 | 555 | 441 | 908 | 541 | 977 | 453 | 487 |     |    |    |    |    |    |    |
| 27  | 222 | 905 | 999 | 102 | 388 | 182 | 444 | 90  |     |    |    |    |    |    |    |
| 28  | 670 | 544 | 233 | 846 | 980 | 406 | 613 | 523 |     |    |    |    |    |    |    |
| 29  | 585 | 315 | 908 | 556 | 352 | 556 | 635 | 413 |     |    |    |    |    |    |    |
| 30  | 607 | 555 | 441 | 908 | 541 | 977 | 453 | 487 |     |    |    |    |    |    |    |
| 31  | 344 | 489 | 600 | 311 | 864 | 414 | 399 | 307 |     |    |    |    |    |    |    |
| 32  | 674 | 677 | 790 | 650 | 666 | 679 | 677 | 566 |     |    |    |    |    |    |    |
| 33  | 5073 | 4761 | 5982 | 5078 | 4750 | 5078 | 4433 | 4478 | 5441 | 3896 | 3233 | 5086 | 7167 | 7021 |

**Canadian Factories**

| 34  | 344 | 489 | 600 | 600 | 481 | 399 | 521 | 897 |     |    |    |    |    |    |    |
| 35  | 355 | 315 | 908 | 556 | 352 | 556 | 635 | 413 |     |    |    |    |    |    |    |
| 36  | 855 | 315 | 908 | 556 | 352 | 556 | 635 | 413 |     |    |    |    |    |    |    |
| 37  | 855 | 315 | 908 | 556 | 352 | 556 | 635 | 413 |     |    |    |    |    |    |    |
| 38  | 344 | 489 | 600 | 311 | 864 | 414 | 399 | 307 |     |    |    |    |    |    |    |

FlyCalc 1.1 - Copyright Natium 2003-2006
Example: live programming

```js
function helloWorld(x) {
  if (x > 0) {
    console.log('hello world');
  } else {
    var y = x + 1;
  }
}

helloWorld(-1);
```
Example: edit constants by editing output
In-class activity
In Class Activity: Direct Manipulation Programming Interactions

• In groups of 2
  • Design a system for writing code through direct manipulation
    • Create sketches showing key screens
  • Should support
    • Standard programming language features (variables, conditionals, loops, functions)
    • Should make it faster and easier to make code changes
    • Should make it easier to get feedback on if program exhibits intended behavior