Code Inspections and Heuristic Evaluation

Objectives

• Today
  – The inspection process
    • Practice inspection
  – Heuristic evaluation process
    • Practice evaluation

• Next time
  – Rationale behind why inspections and heuristic evaluation is so great
  – Normally would do this in the opposite order, but this way you should be able to better prepare any materials over the weekend
Presentation Order

• Email me your materials (code to review or if doing the heuristic evaluation, software to run if possible) by the Friday prior to your presentation if going on Monday, or on the Monday prior to your presentation if going on Wednesday

• Tuesday 3/25: Curtis Conner, Devin Lyons, Matt Grimm, Scott Mahar
• Thursday 3/27: Devin Homan, Dean Sawyer, Hector Sanchez, Joshua Tester
• Tuesday 4/1: Richard To, Britny Herzog, Bobby Porter, Robert Bailey

Code Inspection / Fagan Inspection

• Definition
  – A formal review of a work product by peers. A standard process is followed with the purpose of detecting defects early in the development lifecycle.
• Can inspect many different kinds of documents
  – We will focus on just the code
Defects

- Inspections are used to find defects
- A defect is a deviation from specific or expected behavior
  - Something wrong
  - Missing information
  - Common error
  - Standards violation
  - Ambiguity
  - Inconsistency
  - Perception error
  - Design error

A defect is a defect

- A defect is based on the opinion of the person doing the review
  - This means that any defect that is found is a defect
  - Not open to debate
  - Not all defects are necessarily bugs
  - Many defects may not be “fixed” in the end
- No voting or consensus process on what is a defect
- How to fix a defect should be debated later, not when the defects are logged
What should be inspected?

• For existing code or documentation, select
  – The most critical piece to the program’s operation
  – Most used section
  – Most costly if defects were to exist
  – Most error-prone
  – Least well-known
  – Most frequently changed

• For new code or documentation
  – 20% <= inspect <= 100%

Our Inspection Exercise

<table>
<thead>
<tr>
<th>Individual Work</th>
<th>Group in Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner Planning</td>
<td>Introduction</td>
</tr>
<tr>
<td>30-60 mins</td>
<td>2-5 mins</td>
</tr>
<tr>
<td>Inspectors</td>
<td>Inspect and</td>
</tr>
<tr>
<td>review code</td>
<td>Log Defects</td>
</tr>
<tr>
<td>20-60 mins</td>
<td>10-15 mins</td>
</tr>
<tr>
<td>Owner rework</td>
<td>Owner rework</td>
</tr>
<tr>
<td>? mins</td>
<td>? mins</td>
</tr>
</tbody>
</table>
Owner Planning

• Owner decides what code/documents to review

• Copy of code listing for everyone
  – Send me code by the prior class before the inspection date and I’ll post it on the calendar page for everyone to get
  – **Code should be numbered by line**
  – Not all code, just the selected code (see previous slide on “What should be inspected?”)
  – Up to owner’s discretion as to what/how much, but we will stop after 20 minutes
    • Probably about 2-3 pages

Preparation

• Each inspector should have the materials to inspect in advance
  – Identify defects on their own to ensure independent thought
  – Note defects and questions
  – Complete a defect log
    • High/Medium/Low
  – Without this preparation, group review might find only 10% of defects that could otherwise be found (Fagan)

• Rules of thumb
  – 2 hours for 10 full pages of text
Common Defects

- Mistakes you’ve made in the past
- Anything we discussed in class
  - Code techniques
    - E.g. variable names, location, initialization, refactoring, defensive programming, error checking, magic numbers, loop length, etc.
  - Security
  - Usability
  - Etc.

- Similar issues apply to other languages

Inspection Day

- Prior to inspection
  - Code has already been posted
  - Inspectors have prepared by inspecting the code and noting their defects
- Inspection process
  - Owner provides brief introduction for code
  - Round-robin where each inspector describes a defect found or passes if no defects noted
    - Might find new defects during the inspection exercise
  - Total of 10-20 minutes in our exercise
  - Scribe writes down defects in the defect log
Defect Logging

• High, Medium, Low, or Question
• Brief description should be ~7 words or less, or until the owner understands
• If possible, resolve questions: defect or not
• Also log defects found in
  – Parent document, e.g. requirements
  – Common errors list
  – Work product guidelines
• Will be up to the work owner whether or not to fix a defect

Inspection Example

• Requirement: Support authentication based upon user@host using regular expressions

```c
1  /*******************************************************************
2   * Returns a 1 if the user is on the op's list, and
3   * returns a 0 if the user is not on the op's list.
4   ************************************************************/
5  int Authorized(char *user)
6  {
7   FILE *f;
8   f=fopen(OPSPATH,"r"); /* open authorized file */
9   while (fgets(tempstr,80,f)!=NULL)
10  {
11   tempstr[strlen(tempstr)-1]='\0'; /* annoying \r at end */
12   if (!fnmatch(tempstr,user,FNM_CASEFOLD)) { fclose(f); return(1); }
13  }
14  fclose(f);
15  return(0);
16 }
```

Open file Containing operators

Returns true if wildcards match
Defect Log

<table>
<thead>
<tr>
<th>Severity:</th>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>H M L Q</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In-Class Exercise

• Take 5-10 minutes to find defects in the code posted online
  – This is C# code that highlights the location of my pen on the tablet screen
• We will then do a short round-robin to note defects
### Defect Log

<table>
<thead>
<tr>
<th>Severity: H M L Q</th>
<th>Location</th>
<th>Description</th>
</tr>
</thead>
</table>

### Heuristic Evaluation

“Discount” Usability Testing
Heuristic Evaluation

- Developed by Jakob Nielsen
- Helps find usability problems in a UI design
- Small set (3-5) of evaluators examine UI
  - independently check for compliance with usability principles (“heuristics”)
    - Can also refer to any of the GUI Bloopers we covered
  - different evaluators will find different problems
  - evaluators only communicate afterwards during meeting
    - findings are then aggregated
- Can perform on working UI or on sketches

Jakob Nielsen’s Heuristics

1. Aesthetic and minimalist design
2. Match between system and real world
3. Recognition rather than recall
4. Consistency and standards
5. Visibility of system status
6. User control and freedom
7. Flexibility and efficiency of use
8. Help users recognize, diagnose, and recover from errors
9. Error prevention
10. Help and documentation
Evaluation Day

• Similar to code inspection
• Prior to evaluation
  – Ideally, the program has already been posted (by class prior to the inspection) and inspectors have prepared by running the program and noting issues
    • This may not be possible depending upon the nature of your project. If so, you may give an in-class “demo” and do evaluation on the fly
• Evaluation process
  – Owner provides brief introduction for the program
  – Round-robin where each evaluator describes an issue found or passes if no defects noted
    • Might find new issues during the exercise
  – Total of 15-20 minutes in our exercise
  – Scribe writes down issues in the issue log

Example Problem Descriptions

• Have to remember command codes
  – Violates “Minimize the users’ memory load” (H3)
  – Fix: add drop down box with selectable codes
• Typography uses mix of upper/lower case formats and fonts
  – Violates “Consistency and standards” (H4)
  – Slows users down
  – Probably wouldn’t be found by user testing
  – Fix: pick a single format for entire interface

Adapted from slide by James Landay
Severity ratings

- Used to allocate resources to fix problems
- Should be calculated after all evaluations are done
- Should be done independently by all evaluators
- Based on
  - **Frequency** the problem will occur
  - **Impact** of problem (hard or easy to overcome)
  - **Persistence** (will users learn a work around or will they be bothered every time?)
- **1** – cosmetic problem
- **2** – minor usability problem
- **3** – major usability problem; important to fix
- **4** – usability catastrophe – must fix

Heuristic Evaluation Issue Log

<table>
<thead>
<tr>
<th>Heuristic Issue</th>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consistency</td>
<td>3</td>
<td>The interface used the string &quot;Save&quot; on the first screen for saving the user’s file, but used the string &quot;Write file&quot; on the second screen. Users may be confused by this different terminology for the same function</td>
</tr>
<tr>
<td>Error Message</td>
<td>4</td>
<td>Entering invalid input into dialog box on first form results in &quot;Error 103&quot;</td>
</tr>
</tbody>
</table>

...
Exercise

• Evaluate an application using heuristic evaluation
  – Use your computer or share with a neighbor
    • Java Required: http://www.math.uaa.alaska.edu/~afkj/GeoPixelCounter/
    • Program to compute % of a mineral in a thin section of rock
      – Refer back to slide with the 10 heuristics

• Fill out issues found on paper

• We will discuss and debrief after you are done

Heuristic Evaluation Issue Log

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Issue</th>
<th>Severity</th>
<th>Description</th>
</tr>
</thead>
</table>

---
Theory – Code Inspections

Why inspections?

• Inspections can be applied to many different things by many different groups
• Inspections are a “Best Known Method” (BKM) for increasing quality
  – Developed by Michael Fagan at IBM, paper published 1976
  – Estimates: Inspections of design and code usually remove 50-90% of defects before testing
  – Very economical compared to testing
• Formal inspections are more productive than informal reviews
Formal Inspections

• By formalizing the process, inspections become systematic and repeatable
  – Each person in the inspection process must understand their role
  – Use of checklists focus concentration on detection of defects that have been problematic
• Metrics
  – Feedback and data collection metrics are quantifiable
  – Feed into future inspections to improve them
• Designers and developers learn to improve their work through inspection participation

More reasons to use inspections

• Inspections are measurable
• Ability to track progress
• Reduces rework and debug time
• Cannot guarantee that a deadline will be met but can give early warning of impending problems
• Information sharing with other developers, testers
Definition

• What is an inspection?
  – A formal review of a work product by peers. A standard process is followed with the purpose of detecting defects early in the development lifecycle.

• Examples of work products
  – Code, Specs, Web Pages
  – Presentations, Guides, Requirements,
  – Specifications, Documentation

When are inspections used?

• Possible anytime code or documents are complete
  – Requirements: Inspect specs, plans, schedules
  – Design: Inspect architecture, design doc
  – Implementation: Inspect technical code
  – Test: Inspect test procedure, test report
Defects

- Inspections are used to find defects

- A defect is a deviation from specific or expected behavior

  - Something wrong
  - Missing information
  - Common error
  - Standards violation
  - Ambiguity
  - Inconsistency
  - Perception error
  - Design error

Other Review Methods

<table>
<thead>
<tr>
<th>What</th>
<th>Presentation</th>
<th>Walkthrough</th>
<th>Inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audience</td>
<td>Mgmt/Tech</td>
<td>Tech</td>
<td>Tech</td>
</tr>
<tr>
<td>Objective</td>
<td>Provide Info, Evaluate specs or plan – Give Status</td>
<td>Explain work, may find design or logic defect - Give context</td>
<td>Find defects early - Find defects</td>
</tr>
</tbody>
</table>
Other Defect Detection Methods

<table>
<thead>
<tr>
<th></th>
<th>Buddy</th>
<th>Testing</th>
<th>Inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What</strong></td>
<td>Developers work in pairs</td>
<td>Formal testing</td>
<td>Formal review by peers</td>
</tr>
<tr>
<td><strong>Audience</strong></td>
<td>Tech</td>
<td>Tech</td>
<td>Tech</td>
</tr>
<tr>
<td><strong>Objective</strong></td>
<td>Develop, explain work, find defects</td>
<td>Find defects by symptom, usability, performance</td>
<td>Find defects where they occur</td>
</tr>
</tbody>
</table>

**Why a formal review?**

- Provides a well-defined process
  - Repeatability, measurement
  - Avoids some scenarios with less formal processes
    - “My work is perfect”
      - Point is not to criticize the author
    - “I don’t have time”
      - Formal process proceeds only when all are prepared, have inspected code in advance
# Walkthrough vs. Inspection

<table>
<thead>
<tr>
<th></th>
<th>Walkthrough</th>
<th>Inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Focus</strong></td>
<td>Improve product</td>
<td>Find defects</td>
</tr>
<tr>
<td><strong>Activities</strong></td>
<td>Find defects</td>
<td>Find defects</td>
</tr>
<tr>
<td></td>
<td>Examine alternatives</td>
<td>Only defect explanation allowed</td>
</tr>
<tr>
<td></td>
<td><strong>Forum for learning</strong></td>
<td>Learning through defects and inspection</td>
</tr>
<tr>
<td></td>
<td><strong>Discussion</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Process</strong></td>
<td>Informal</td>
<td><strong>Formal</strong></td>
</tr>
<tr>
<td><strong>Quality</strong></td>
<td>Variable; personalities can modify outcome</td>
<td>Repeatable with fixed process</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td><strong>Preparation ad-hoc, less formal</strong></td>
<td>Preparation required, efficient use of time</td>
</tr>
</tbody>
</table>

---

"The only valid measurement of code quality: WTFs/minute"
Typical Inspection Process

We are using a shortened process in class, but it is essentially the same as the “normal” process.

Roles

Moderator

Inspectors

Scribe

Work Owner
Causal Analysis Meeting

• Purpose – Brainstorming session on the root cause of specific defects
  – This takes place sometime after the inspection has been completed
  – This meeting supports the continuous improvement
  – Initiate thinking and action about most common or severe defects
  – Can help prevent future defects from occurring
    • Specific action items may be achieve this goal

Rework

• Purpose: Address defects found during the logging process
• Rules
  – Performed by product owner
  – All defects must be addressed
    • Does not mean they are fixed, but that sufficient analysis/action has taken place
  – All defects found in any other documents should be recorded
  – Owner should keep work log
Follow-Up

• Purpose: Verify resolution of defects
  – Work product redistributed for review
  – Inspection team can re-inspect or assign a few inspectors to review
  – Unfixed defects are reported to the team and discussed to resolution

• We’re skipping these last few phases for the class

• I would like to see how you addressed defects in your final writeup

Theory - Heuristic Evaluation

Adapted from material by Marti Hearst, Loren Terveen
Evaluating UI Designs

• Usability testing is a major technique
  – Formal techniques require users, rigid control experiments, statistical analysis
• “Discount” methods don’t require users
  – Heuristic Evaluation
  – Cognitive Walkthrough

Heuristic Evaluation

• Developed by Jakob Nielsen
• Helps find usability problems in a UI design
• Small set (3-5) of evaluators examine UI
  – independently check for compliance with usability principles (“heuristics”)
  – different evaluators will find different problems
  – evaluators only communicate afterwards
    • findings are then aggregated
• Can perform on working UI or on sketches
Phases of Heuristic Evaluation

1) Pre-evaluation training
   – give evaluators needed domain knowledge and information on the scenarios

2) Evaluation
   – individuals evaluate and then aggregate results

3) Severity rating
   – determine how severe each problem is (priority)

4) Debriefing
   – discuss the outcome with design team

Jakob Nielsen’s heuristics

<table>
<thead>
<tr>
<th>1.0</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1. Simple and natural dialog</td>
<td>Aesthetic and minimalist design</td>
</tr>
<tr>
<td>H2. Speak the user’s language</td>
<td>Match between system and real world</td>
</tr>
<tr>
<td>H3. Minimize user memory load</td>
<td>Recognition rather than recall</td>
</tr>
<tr>
<td>H4. Be consistent</td>
<td>Consistency and standards</td>
</tr>
<tr>
<td>H5. Provide feedback</td>
<td>Visibility of system status</td>
</tr>
<tr>
<td>H6. Provide clearly marked exits</td>
<td>User control and freedom</td>
</tr>
<tr>
<td>H7. Provide shortcuts</td>
<td>Flexibility and efficiency of use</td>
</tr>
<tr>
<td>H8. Provide good error messages</td>
<td>Help users recognize, diagnose, and recover from errors</td>
</tr>
<tr>
<td>H9. Prevent errors</td>
<td>Error prevention</td>
</tr>
<tr>
<td>H10. Help and documentation</td>
<td>Help and documentation</td>
</tr>
</tbody>
</table>
Pros / Cons

- + Cheap (no special lab or equipment)
- + Easy
- + Fast (about 1 day)
- + Cost-effective
- + Detects many problems without users
- + Complementary to task-centered approaches
- + Coverage
- + Catches cross-task interactions
- - Requires subjective interpretation
- - Does not specify how to fix problems
- - Performance improves as evaluator knowledge increases

Procedure

- A set of evaluators (3-5 is about optimal) evaluate a UI (some training may be needed)
- Each one independently checks for compliance with the heuristics
  – Different evaluators find different problems
- Individually rate severity of the problems
- Evaluators then get together and merge their findings
- Debriefing/brainstorming → how to fix the problems (and point out what’s really good)
How to Perform H.E.

• At least two passes for each evaluator
  – first to get feel for flow and scope of system
  – second to focus on specific elements
• Assistance from implementors/domain experts
  – If system is walk-up-and-use or evaluators are domain experts, then no assistance needed
  – Otherwise might supply evaluators with scenarios and have implementors standing by

How to Perform Evaluation

• Where problems may be found
  – single location in UI
  – two or more locations that need to be compared
  – problem with overall structure of UI
  – something that is missing
Severity ratings

• Used to allocate resources to fix problems
• Should be calculated after all evaluations are done
• Should be done independently by each evaluators
• Based on
  – Frequency the problem will occur
  – Impact of problem (hard or easy to overcome)
  – Persistence (will users learn a work around or will they be bothered every time?)
• 1 – cosmetic problem
• 2 – minor usability problem
• 3 – major usability problem; important to fix
• 4 – usability catastrophe – must fix

Debriefing

• Conduct with evaluators, observers, and development team members
• Discuss general characteristics of UI
• Suggest potential improvements to address major usability problems
• Developer team rates how hard things are to fix
• Make it a brainstorming session

Adapted from slide by James Landay
Results of Using HE (cont.)

• Single evaluator achieves poor results
  – only finds 35% of usability problems
  – 5 evaluators find ~ 75% of usability problems
  – why not more evaluators? 10? 20?
    • adding evaluators costs more
    • adding more evaluators doesn’t increase the number of unique problems found

Decreasing Returns

problems found

benefits / cost

• (from Nielsen)
• Caveat: these graphs are for a specific example
• This is a controversial point.
Why Multiple Evaluators?

- Every evaluator doesn’t find every problem
- Good evaluators find both easy & hard ones

Summary

- Inspections and Heuristic Evaluation are considered Best Known Methods to improve software quality
  - Relatively cheap to perform
  - Finds errors directly
  - Developers share their knowledge with one another, leading to quality improvements on future projects