CMPT 473
Software Quality Assurance

Managing Bugs

Nick Sumner - Fall 2014

Bugs!

- So far, we've been trying to find & avoid them
  - Test, test, test!

Bugs!

- So far, we've been trying to find & avoid them
  - Test, test, test!
- Eventually, you might actually encounter a bug/error

Bugs!

- So far, we've been trying to find & avoid them
  - Test, test, test!
- Eventually, you might actually encounter a bug/error
- 2 perspectives to consider

Bugs!

- So far, we've been trying to find & avoid them
  - Test, test, test!
- Eventually, you might actually encounter a bug/error
- 2 perspectives to consider
  - *Developer* – how should the program handle errors?
    - *Error Reporting*

Bugs!

- So far, we've been trying to find & avoid them
  - Test, test, test!
- Eventually, you might actually encounter a bug/error
- 2 perspectives to consider
  - *Developer* – how should the program handle errors?
    - *Error Reporting*
  - *Client/Teammate* – how should the bug be reported / prioritized / fixed?
    - *Bug Advocacy*

Bugs!

- So far, we've been trying to find & avoid them
  - Test, test, test!
- Eve
  bug/error
- 2 perspectives to consider
  - *Developer* – how should the program handle errors?
    - *Error Reporting*
  - *Client/Teammate* – how should the bug be reported / prioritized / fixed?
    - *Bug Advocacy*

These perspectives are not independent! Why?

Error Reporting

- What should the program do when it detects an error?

Error Reporting

- What should the program do when it detects an error?
  - Simply ignoring the error is often a poor choice

Error Reporting

- What should the program do when it detects an error?
  - Simply ignoring the error is often a poor choice
  - Log it, print it, or otherwise report it

Error Reporting

- What should the program do when it detects an error?
  - Simply ignoring the error is often a poor choice
  - Log it, print it, or otherwise report it
  - Fail gracefully, continuing if possible

Error Reporting

- What should the program do when it detects an error?
  - Simply ignoring the error is often a poor choice
  - Log it, print it, or otherwise report it
  - Fail gracefully, continuing if possible
- What should error messages contain?

Error Reporting

- What should the program do when it detects an error?
  - Simply ignoring the error is often a poor choice
  - Log it, print it, or otherwise report it
  - Fail gracefully, continuing if possible
- What should error messages contain?
  - **What** specifically is incorrect

Error Reporting

- What should the program do when it detects an error?
  - Simply ignoring the error is often a poor choice
  - Log it, print it, or otherwise report it
  - Fail gracefully, continuing if possible
- What should error messages contain?
  - **What** specifically is incorrect
  - **Why** it is incorrect

Error Reporting

- What should the program do when it detects an error?
  - Simply ignoring the error is often a poor choice
  - Log it, print it, or otherwise report it
  - Fail gracefully, continuing if possible
- What should error messages contain?
  - **What** specifically is incorrect
  - **Why** it is incorrect
  - **Where** / **when** it is incorrect

Error Reporting

- Should match your existing intuition:

Error Reporting

- Should match your existing intuition:
  - **“`try { … } catch (Exception e) {}`”** is hideous!

Error Reporting

- Should match your existing intuition:
  - `“try { … } catch (Exception e) {}”` is hideous!
  - “Segmentation Fault” is frustrating

Error Reporting

- Should match your existing intuition:
  - **`“try { … } catch (Exception e) {}”`** is hideous!
  - “Segmentation Fault” is frustrating
  - “Program Error” is infuriating

Error Reporting

- Should match your existing intuition:
  - **“`try { … } catch (Exception e) {}`”** is hideous!
  - “Segmentation Fault” is frustrating
  - “Program Error” is infuriating
  - “Index out of bounds: index i = 30 > size 15 at line 5 of MyVector.java” is rather pleasant

Error Reporting

- Should match your existing intuition:
  - **`“try { … } catch (Exception e) {}”`** is hideous!
  - “Segmentation Fault” is frustrating
  - “Program Error” is infuriating
  - “Index out of bounds: index i = 30 > size 15 at line 5 of MyVector.java” is rather pleasant
- But not all information should be reported!
  - Why might some values/variables be undesirable to report?

Error Reporting

- Should match your existing intuition:
  - **`"try { … } catch (Exception e) {}"`** is hideous!
  - "Segmentation Fault" is frustrating
  - "Program Error" is infuriating
  - "Index out of bounds: index i = 30 > size 15 at line 5 of MyVector.java" is rather pleasant
- But not all information should be reported!
  - Why might some values/variables be undesirable to report?

Note: Sensitive values should not even be available or possible to report!
This indicates design bugs!

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management
- Help facilitate:

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management
- Help facilitate:
  - Reproducing the failure

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management
- Help facilitate:
  - Reproducing the failure
  - Finding the best initial “owner”

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management
- Help facilitate:
  - Reproducing the failure
  - Finding the best initial “owner”
  - Combining duplicate reports

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management
- Help facilitate:
  - Reproducing the failure
  - Finding the best initial “owner”
  - Combining duplicate reports
  - Identifying possible causes & effects

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management
- Help facilitate:
  - Reproducing the failure
  - Finding the best initial “owner”
  - Combining duplicate reports
  - Identifying possible causes & effects
  - Prioritizing the bug

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management
- Help facilitate:
  - Reproducing the failure
  - Finding the best initial “owner”
  - Combining duplicate reports
  - Identifying possible causes & effects
  - Prioritizing the bug
  - Identifying workarounds & working cases

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management
- Help facilitate:
  - Reproducing the failure
  - Finding the best initial “owner”
  - Combining duplicate reports
  - Identifying possible causes & effects
  - Prioritizing the bug
  - Identifying workarounds & working cases

What have we left out?

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management
- Help facilitate:
  - Reproducing the failure
  - Finding the best initial “owner”
  - Combining duplicate reports
  - Identifying possible causes & effects
  - Prioritizing the bug
  - Identifying workarounds & working cases
  - … and creating a fix

Reporting, Advocacy, & Management

- The reason we need good error messages is to support bug reporting & management
- Help facilitate:
  - Reproducing the failure
  - Finding the best initial “owner”
  - Combining duplicate reports
  - Identifying possible causes & effects
  - Prioritizing the bug
  - Identifying workarounds & working cases
  - … and creating a fix

Bad bug reporting & management is worse that none!
Any ideas why?

Bug Management

- All projects have unfixed bugs
  - How do we keep track of them & decide what to fix?

Bug Management

- All projects have unfixed bugs
  - How do we keep track of them & decide what to fix?
  - Bug Databases
  - e.g. Bugzilla, Mantis, Trac, FogBugz, ...

Bug Management

- All projects have unfixed bugs
  - How do we keep track of them & decide what to fix?
  - Bug Databases
  - e.g. Bugzilla, Mantis, Trac, FogBugz, ...
- Bug Databases
  - Centralize communication (developer & user) to:

Bug Management

- All projects have unfixed bugs
  - How do we keep track of them & decide what to fix?
  - Bug Databases
  - e.g. Bugzilla, Mantis, Trac, FogBugz, ...
- Bug Databases
  - Centralize communication (developer & user) to:
    - Own
    - Prioritize
    - Reproduce, Localize, Explain
    - Patch

Ownership

- Who is responsible for a bug?
  - A very difficult task in general

Ownership

- Who is responsible for a bug?
  - A very difficult task in general
  - “Who knows the most about this module?”
  - “Whose code (if any!) exposed the bug?” (RIP)
  - “Who worked with this code most recently?”

Ownership

- Who is responsible for a bug?
  - A very difficult task in general
  - “Who knows the most about this module?”
  - “Whose code (if any!) exposed the bug?” (RIP)
  - “Who worked with this code most recently?”
  - Is this a client side issue?

Prioritizing Bugs

- All projects have unfixed bugs
  - If a bug doesn't appear important, it won't get fixed

Prioritizing Bugs

- All projects have unfixed bugs
  - If a bug doesn't appear important, it won't get fixed
- Which bugs are important?

What makes bugs a higher priority for you?

Prioritizing Bugs

- All projects have unfixed bugs
  - If a bug doesn't appear important, it won't get fixed
- Which bugs are important?
  - Occur frequently / for most users
  - Have substantial risks / consequences
  - Are new in the latest version

Prioritizing Bugs

- All projects have unfixed bugs
  - If a bug doesn't appear important, it won't get fixed
- Which bugs are important?
  - Occur frequently / for most users
  - Have substantial risks / consequences
  - Are new in the latest version

Why are new bugs important?

Prioritizing Bugs

- All projects have unfixed bugs
  - If a bug doesn't appear important, it won't get fixed
- Which bugs are important?
  - Occur frequently / for most users
  - Have substantial risks / consequences
  - Are new in the latest version
- Identifying the importance of bugs is critical to prioritizing them
  - Usually informally at first until a bug owner is found to estimate the risk

But what should a report contain?

- A concise explanation of anything helpful in evaluating & fixing the bug
  - …?

But what should a report contain?

- A concise explanation of anything helpful in evaluating & fixing the bug
  - Product, version, & relevant feature
  - Platform & environment
  - Potential severity / priority
  - Possible owners
  - Possible duplicates
  - A one line summary
  - An explanation of ***what*** happened, ***when*** it happened, & ***why*** it was unexpected
  - A minimal, self contained test case

Bug Advocacy – An Example

"A colleague of mine have find a hairy bug, here is a simple code to reproduce it."

```
public class FunWithMultiCatch {
   public static void main(String[] args) {
     Runnable r = () -> {
       try {
         Object o = null;
         o.getClass();
         throw new IOException();
       } catch(IOException | IllegalArgumentException e) {
         System.out.println("KO !");
       } catch(RuntimeException e) {
         System.out.println("OK !");
       }
     };
     r.run();
   }
}
```

http://mail.openjdk.java.net/pipermail/lambda-dev/2014-March/011940.html

Bug Advocacy – An Example

"A colleague of mine have find a hairy bug, here is a simple code to reproduce it."

```
public class FunWithMultiCatch {
   public static void main(String[] args) {
     Runnable r = () -> {
       try {
         Object o = null;
         o.getClass();
         throw new IOException();
       } catch(IOException | IllegalArgumentException e) {
         System.out.println("KO !");
       } catch(RuntimeException e) {
         System.out.println("OK !");
       }
     };
     r.run();
   }
}
```

It prints 'KO !' :(

http://mail.openjdk.java.net/pipermail/lambda-dev/2014-March/011940.html

How can we minimize test cases

- For now:
  - How do you already minimize test cases?