Testing Code

CSC 116 – Section 002
April 13, 2005

Compile Errors

• Caused by coding errors in the source code
• Generate messages when the code is compiled
• Generate one or more lines of errors
• Correct the first error first, and then attempt to recompile
Runtime Errors

- Logic errors in the design of the code
- Generate errors in the terminal window or may cause abnormal behavior

Verification

- Verification: “The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of the phase.” [IEEE Standard Terminology 1990]
- Are we building the product right?
- Testing and reviews
Validation

- Validation: “The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.” [IEEE Standard Terminology 1990]
- Are we building the right product?

Testing

- Test: “an activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component.” [IEEE Standard Terminology 1990]
Software Testing Techniques

• Black Box Testing (functional testing)
  – “Testing that ignores the internal mechanism of the system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions” [IEEE Standard Terminology 1990]
  – Treats the program like a black box
  – Testing GUIs

Software Testing Techniques (2)

• White Box Testing (structural testing)
  – “Testing that takes into account the internal mechanism of a system or component” [IEEE Standard Terminology 1990]
  – You test the code based on knowledge of the code
  – Testing to make sure that conditional statements are handled properly.
Equivalence Partitioning

- Divides the domain of input into classes (not like Java classes) based on different types of input
- Example: You’re playing Monopoly, and you’re in Jail. You must pay $50 to get out of Jail.
  - Player has $50 or more
  - Player had less than $50

Equivalence Partitioning

- Once you have determined your classes you choose values that are in the middle of each class (or somewhere above the boundary) and test them
  - Player with $100
  - Player with $25
Equivalence Partitioning

- Ways to create equivalence classes
  - Input is a range of values, create a valid and invalid equivalence class
  - Input requires specific values, create an equivalence class for each valid value and an invalid equivalence class
  - Input is a member of a set, create a valid and invalid equivalence class
  - Input is a Boolean, create a valid and invalid equivalence class

Boundary Value

- Programmers tend to make mistakes at boundaries
- Boundary: “A data value that corresponds to a minimum or maximum input, internal, or output value specified for a system or component.” [IEEE Standard Terminology 1990]
- You want to test the exact boundary, boundary +/- 1
Six Types of Testing

- Unit Testing
- Integration Testing
- Functional and System Testing
- Acceptance Testing
- Regression Testing
- Beta Testing

Unit Testing

- White Box Testing
- Testing is done on small units of code, like a method
- Goal: Test the full range of inputs and all paths through the code.
  - Equivalence Partitioning
  - Boundary Values
  - Diabolical
- Where ever you use objects, test for null pointer exceptions.
Integration Testing

• White and Black Box Testing
• Tests how different components of a program work together (class interaction)
• Any class that relies on another class must be tested to ensure that it properly interacts with that class.

Test Case Planning

• Test cases require:
  – An ID
  – Description of the test case
    • Exact values (repeatability)
  – Expected Results
  – Actual Results
Example

- Lets write test cases for the getHtmlLine() method from the JavaHtmlLine class from Program 3
- Equivalence Classes:
  - Starting spaces/tabs
  - &
  - <
  - >

Example (2)

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Description</th>
<th>Expected Results</th>
<th>Actual Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab</td>
<td>line = &quot;\t&quot;</td>
<td>&quot; ,&quot;</td>
<td></td>
</tr>
<tr>
<td>Space</td>
<td>line = &quot; &quot;</td>
<td>&quot; ,&quot;</td>
<td></td>
</tr>
<tr>
<td>Amp</td>
<td>line = &quot;&amp;&quot;</td>
<td>&quot;&amp;,&quot;</td>
<td></td>
</tr>
<tr>
<td>Greater</td>
<td>line = &quot;&gt;&quot;</td>
<td>&quot;&gt;&quot;</td>
<td></td>
</tr>
<tr>
<td>LessThan</td>
<td>line = &quot;&lt;&quot;</td>
<td>&quot;&lt;&quot;</td>
<td></td>
</tr>
</tbody>
</table>

© 2005 Sarah E. Smith
References

- IEEE Standard Terminology 1990:
  http://www.ieee.org/
- Laurie Williams’ Software Engineering Text Book Online Chapters:
  http://openseminar.org/se/
- Jason Schwarz’s Lecture 26 slides:
  http://courses.ncsu.edu/csc116/

© 2005 Sarah E. Smith