Technology maturation and research strategy

Testing Techniques

Annie Lu Luo
17-939A Class Presentation
November 19, 2001

Why testing...

- Testing takes 40%–50% of software development effort, maybe much higher in real-time systems. [Beizer, 1988]
- Software quality is becoming the dominant success criterion in industry. [Osterweil, 1996]
- Software testing (arguably) remains the least understood part of the development process. [Whittaker, 2000]

In this talk

- Concept evolution
- Taxonomy
- Technique maturation study
- Research strategies
- Future work

What is software testing?

- The testing of software is a means of measuring or assessing the software to determine its quality. [Marciniak, Encyclopedia of software Engineering, 1994]
- Testing is not a phase, but a lifestyle!
Concept Evolution

1950: Turing Test

1950s: Testing to Reduce Bugs

1970: Software Can Be Tested Exhaustively

1975: Testing Succeded When Error Free

1979: Testing To Find Error

1975: Testing To Build Confidence

1980: Testing To Build Confidence

1983-1987: Evaluation-Oriented

1988~: Prevention-Oriented

1990s: Emphasis on Early Test Design

The testing paradox

- When we test...
  - we are happy if the software has less error.
  - we are happy if the test can find more error.

- This is confusing...
  - The best way to build confidence is to try and destroy it.

Goals of testing

- In practice, a mixture of both debug testing and acceptance testing is used. The goals of testing are to:
  - find faults,
    - "A test is successful if the program fails" (Gleiser and Lefevre, 1985)
  - and provide confidence
    - of reliability
    - of correctness (hopefully)
    - of detection (and therefore the absence) of particular faults

Next...

- Concept evolution
- Taxonomy
  - Strategies
  - Levels
  - Techniques
- Technique maturation study
- Research strategies
- Future work
Testing Strategies

- **Static**
  - Does not require execution of the software

- **Dynamic**
  - Requires execution of the software

For dynamic techniques:

- **White-box**
  - Emphasize on internal structure
  - Based on flow coverage
  - Not scale up
  - Can’t reveal missing path errors

- **Black-box**
  - Emphasize on external behavior
  - Depend on spec. notation
  - Scale up
  - Can’t reveal code based faults

Levels of testing

- **Unit testing**
- **Component integration testing** (a.k.a. link testing or sub-system testing)
- **System testing**
- **Acceptance testing (demo)**
- **Regression testing**
- **Others**
  - Alpha and Beta testing
  - Parallel running

Classic Techniques

<table>
<thead>
<tr>
<th>Year</th>
<th>Unit</th>
<th>Integration</th>
<th>System</th>
<th>Acceptance</th>
<th>Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>1970</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1976</td>
<td>Syn</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1978</td>
<td>Mute</td>
<td>As/Ad</td>
<td>Mute</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1979</td>
<td></td>
<td></td>
<td>C/GB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1981</td>
<td>Run Dst</td>
<td></td>
<td>GB Dst</td>
<td></td>
<td>GB</td>
</tr>
<tr>
<td>1982</td>
<td>DF</td>
<td>DF</td>
<td>FT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td>Mute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Mutation Testing
- Random Testing
- Structural Testing
- Assumption/Adaptive Testing
- Grammar-based Testing
- Data-flow oriented Testing

Symbolic execution Testing
- Domain and input space partitioning Testing
- Functional Testing
- Real-time Testing (thread testing) – very little
- Constraint-based Testing

New technologies

- **Formal method** [since 1978]
- **Object-Oriented Testing** [1994]
- **Concurrent Testing** [1994]
- **Protocol Testing** [1994]
- **Architecture Testing** [1996]
- **GUI Testing** [1998]
- **COTS Testing – Ballista** [1998]
- **Component-based Testing – Microsoft** [2000]
State-of-the-art & State-of-the-practice

• Fundamental research in testing techniques have reached a certain maturation stage.
• Widespread of systematic techniques is not common in industry.
  – Research runs faster than practice?
  – Research can’t meet the requirement of practice?
  – Testing technique research waits for the maturation of other areas (e.g. formal method, architecture)?
  – New/improved areas calls for new/improved testing techniques?

What do we need?
• Further fundamental research
• Research that “bridges the gap”:
  – Demonstrate effectiveness of existing techniques
  – Address the need in new areas
  – Create new adaptive techniques
  – Facilitate transferring technology to industry
  – Consider management issues

The mission
• Refine the current view of taxonomy
  – Is there any need to get a more complete collection of testing techniques?
• Refine the study on research strategies
• Final paper
Research Strategy [Harrold et al., ’92]

• **Question:**
  – **Method/Means** How can we use the hierarchical information to validate each class in the library and reduce testing work?

• **Result:**
  – **Techniques** An incremental class testing technique that take advantage of class inheritance.

• **Validation:**
  – **Implementation** The prototype implementation are given and experimented on existing C++ classes.

Research Strategy [Kroop et al., ’98]

• **Question:**
  – **Method/Means** How can we access component robustness without sacrificing the COTS advantage?

• **Result:**
  – **Techniques** The Ballista methodology to automatically for and harden against software failures caused by exceptional inputs.

• **Validation:**
  – **Implementation and evaluation** The Ballista system, and the automatic testing results on a large scale of systems.

Happy Thanksgiving!

Thanks for all your comments, And I know you have more to say…… Please send email to luluo@cs