Analysis and Testing of Deployed Software

Alessandro (Alex) Orso

College of Computing
Georgia Tech
http://www.cc.gatech.edu/~orso

Supported in part by National Science Foundation, the State of Georgia under the Yamacraw Mission, and Boeing Commercial Airplane Group.
A Bug Can Be More Than a Nuisance

- 10 years of development
- $7 billion cost
- $500 million loss
- (no humans onboard)

“No test was performed to verify that the SRI would behave correctly when being subjected to […] the trajectory of Ariane 5. Had such a test been performed, the failure […] would have been exposed.”

(from the report of the Inquiry Board)

Software bugs are costing the U.S. economy an estimated $59.5 billion each year. Improvements in testing and maintenance could reduce this cost by about a third, or $22.5 billion.

(from NIST Estimated Planning Report 02-3)
Goal of My Research

Improve effectiveness and efficiency of testing and maintenance

- Develop new testing techniques (and required analyses)
- Develop systems and infrastructure
- Perform empirical evaluation in realistic settings
Outline

Background
  • Testing and maintenance
  • My research focus and contributions

The Gamma Project
  • Overview
  • Impact analysis
  • Replay of executions

Conclusion and Future Work
Testing and Maintenance

1. Develop P
2. Develop T for P
3. Modify P
4. Select subset of T to rerun
5. Identify faults
6. Release P
7. P correct for T?
8. T adequate?
9. Augment T for untested parts of P
10. Modify P
11. Select subset of T to rerun
12. Identify faults
13. Release P
14. P correct for T?
15. T adequate?
Contributions

- 2005: Scalable, safe regression testing of OO software [OOPSLA01, GT-TR04]
- 2004: Pointer-aware DF testing and program comprehension [TOSEM04]
- 2003: Unit and integration testing of OO SW [TCS99, ISSTA00]
Contributions

- Gammatella: visualization of runs
- InsECT: dynamic analysis framework
- DUSC: Java dynamic updating
- DejaVOO: Java regression testing
- Aristotle, JABA: analysis frontends
Contributions

Select/collect subjects

Design experiments or case studies

Analyze results and draw conclusions

Testing Techniques

Systems

Empirical evaluation

orso@cc.gatech.edu, March 03

9
Outline

Background

- Testing and maintenance
- My research focus and contributions

The Gamma Project

- Overview
- Impact analysis
- Replay of executions

Current and Future Work
Motivation for Gamma

Fundamental shift in software development
  • Software virtually everywhere
  • Most computers interconnected
  • Large amount of user resources

Problems
  • Short development cycles
  • Increasingly dynamic interactions
  • Increased complexity (environments, SW structure, …)

Opportunity to use field data/resources for SE tasks
  • In-house activities limited by the use of synthetic workloads and configurations
  • The Gamma approach aims to overcome these limitations
Gamma: Overall Picture

Program P

SE Tasks

User

Field Data

[ISSTA02]
Challenges

- Applications
  - How to augment existing analyses
  - How to enable new analyses

- Data collection
  - Effectiveness
  - Efficiency
  - Security and privacy
Applications

- Coverage analysis for deployed software [PASTE02]
- Impact analysis [FSE03, ICSE04]
- Regression testing [FSE03]
- Visualization of program executions [InfoVis04]
- Partial replay of users’ executions [Dagstuhl03]

Data collection

- Software Tomography [ISSTA02, PASTE02]
- Dynamic SW Update [ICSM02]
Outline

Background
  • Testing and maintenance
  • My research focus and contributions

The Gamma Project
  • Overview
  • Impact analysis
  • Replay of executions

Current and Future Work
Testing and Maintenance

Develop P

Modify P

Select subset of T to rerun

Identify faults

Develop T for P

P correct for T?

T adequate?

Augment T for untested parts of P

Release P'

T

F
Call-Graph-Based Impact Analysis

```
m1() {
    if (...) m2();
    m3();
}
m2() {
    for(...) m4();
    ...
    exit;
}
m3() {...}
m4() {
    if(...) m5();
    if(...) m6();
}
m5() {
    if(...) m4();
}
m6() {...}
```

Impact set = \{ m4, m5, m6 \}
Slicing-Based Impact Analysis

```
m1() {
    if (...) m2();
    m3();
}
m2() {
    for(...) m4();
    ...
    exit;
}
m3() {...}
m4() {
    if(...) m5();
    if(...) m6();
}
m5() {
    if(...) m4();
}
m6() {...}
```

Impact set = { m2, m4, m5, m6 }
m1() {
    if (...) m2();
    m3();
}
m2() {
    for(...) m4();
    ...
    exit;
}
m3() {...}
m4() {
    if(...) m5();
    if(...) m6();
}
m5() {
    if(...) m4();
}
m6() {...}

**Execution trace1:** m₁ m₂ m₄ m₅ r r x

**Execution trace2:** m₁ m₂ m₄ m₆ r r x

Impact set = { m₂, m₄, m₅, m₆ }
Dynamic Impact Analysis

```c
m1() {
    if (...) m2();
    m3();
}

m2() {
    for(...) m4();
    ...
    exit;
}

m3() {...}

m4() {
    if(...) m5();
    if(...) m6();
}

m5() {
    if(...) m4();
}

m6() {...}
```

Execution trace: m₁ m₂ m₄ m₅ m₄ r r r m₄ m₅ m₄ r r r m₄ m₅ m₄ r r m₄ m₅ r r m₄ m₅ m₄ r r m₄ m₅ m₄ r r m₄ m₅ m₄ r r m₄ m₅ m₄ r r m₄ m₅ m₄ r ...

![Diagram]
Impact Analysis: Goal

Develop an impact-analysis technique that

• Reflects actual usage in the field
• Conservative (safe) w.r.t. considered profile
• Lightweight

Lightweight dynamic forward slicing

• Coverage analysis
• Forward reachability

[FSE03]
Coverage Analysis

Program P

User A

User B

{A,m3,m6}

Field Data

<table>
<thead>
<tr>
<th></th>
<th>m₁</th>
<th>m₂</th>
<th>m₃</th>
<th>m₄</th>
<th>m₅</th>
<th>m₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>B1</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>B2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>C1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Forward Reachability

m1() {  
  if (...) m2();  
  m3();  
}
m2() {  
  for(...) m4();  
  ...  
  exit;  
}
m3() {...}
m4() {  
  if(...) m5();  
  if(...) m6();  
}
m5() {  
  if(...) m4();  
}
m6() {...}

<table>
<thead>
<tr>
<th></th>
<th>m1</th>
<th>m2</th>
<th>m3</th>
<th>m4</th>
<th>m5</th>
<th>m6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>B1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Impact set = { m2, m4, m5 }
Forward Reachability

m1() {
    if () m2();
    A1
    m1 m2
    ...
} m6() {...}

Input:
- program P
- set of changes C
- users’ execution data

Output:
- impact set for C

1. Build augmented call graph
2. For each changed method c in C
   a. Identify user executions through c
   b. Mark methods covered by such executions
   c. Perform reachability considering only marked nodes
3. Return the set of reached methods

set = { m2,m4,m5 }
Sources of Imprecision

\[
\begin{align*}
m1() \{ \\
  \text{if (...) m2();} \\
  \text{m3();} \\
\} \\
m2() \{ \\
  \text{for(...) m4();} \\
  \ldots \\
  \text{exit;} \\
\} \\
m3() \{\ldots\} \\
m4() \{ \\
  \text{if(...) m5();} \\
  \text{if(...) m6();} \\
\} \\
m5() \{ \\
  \text{if(...) m4();} \\
\} \\
m6() \{\ldots\}
\end{align*}
\]

\[
\begin{array}{cccccc}
\text{A1: } & \text{m}_1 & \text{m}_2 & \text{m}_4 & \text{m}_6 & \text{r} & \text{m}_5 & \text{r} & \text{r} & \text{x} \\
\text{B1: } & \text{m}_1 & \text{m}_2 & \text{m}_4 & \text{m}_6 & \text{r} & \text{r} & \text{x} \\
\hline
\text{A1} & X & X & X & X & X & X \\
\text{B1} & X & X & X & X & X & X \\
\end{array}
\]

Impact set = \{ m2, m4, m5, m6 \}
Study 1

How does the technique compare with alternative approaches?

- Dynamic technique based on execution traces
- Static techniques
  - Based on transitive closure on the call graph
  - Based on slicing

Subjects

<table>
<thead>
<tr>
<th>Program</th>
<th>Versions</th>
<th>Classes</th>
<th>Methods</th>
<th>LOC</th>
<th>Test Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>NanoXML</td>
<td>6</td>
<td>19</td>
<td>251</td>
<td>3,279</td>
<td>216</td>
</tr>
<tr>
<td>Siena</td>
<td>8</td>
<td>24</td>
<td>219</td>
<td>3,674</td>
<td>564</td>
</tr>
<tr>
<td>Jaba</td>
<td>11</td>
<td>550</td>
<td>2,695</td>
<td>60,000</td>
<td>125</td>
</tr>
</tbody>
</table>

[ICSE04]
Results: Relative Precision

For 13 out of 21 cases considered (62%) precision is not statistically different

Difference in precision from 0 to 13.4, with avg. 3.8
### Results: Time Overhead

<table>
<thead>
<tr>
<th>Program</th>
<th>Build 1</th>
<th>Build 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>NanoXML v0</td>
<td>0</td>
<td>2,334.08</td>
</tr>
<tr>
<td>NanoXML v1</td>
<td>20.19</td>
<td>862.39</td>
</tr>
<tr>
<td>NanoXML v2</td>
<td>21.24</td>
<td>1,199.63</td>
</tr>
<tr>
<td>Siena v0</td>
<td>391.58</td>
<td>1,807.54</td>
</tr>
<tr>
<td>Siena v1</td>
<td>391.12</td>
<td>1,807.19</td>
</tr>
<tr>
<td>Siena v2</td>
<td>391.23</td>
<td>1,786.14</td>
</tr>
<tr>
<td>Jaba v0</td>
<td>53.34</td>
<td>57,749.81</td>
</tr>
<tr>
<td>Jaba v1</td>
<td>53.69</td>
<td>58,487.19</td>
</tr>
<tr>
<td>Jaba v2</td>
<td>53.71</td>
<td>60,012.22</td>
</tr>
<tr>
<td>Jaba v3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **14min (4,171%)**
- **16hrs (111,634%)**
Study 2

Does the use of field data actually yield different results than the use of in-house data?

Subject

<table>
<thead>
<tr>
<th>Program</th>
<th>Versions</th>
<th>Classes</th>
<th>Methods</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jaba</td>
<td>21</td>
<td>550</td>
<td>2,695</td>
<td>60,000</td>
</tr>
</tbody>
</table>

Data

- In-house data: 707 test cases, 63% coverage
- Field data: 1,100 executions (14 users, 12 weeks)

[FSE03]
Results: Differences in Impact Sets

- Field - InHouse
- InHouse - Field

# Methods

Jaba version

Field

InHouse
Outline

Background
  • Testing and maintenance
  • My research focus and contributions

The Gamma Project
  • Overview
  • Impact analysis
  • Replay of executions

Current and Future Work
Practicality
- High volume of data
- Hard to capture (custom)

Privacy
- Sensitive information

Safety
- Side-effects

Partial replay

[Dagstuhl03]
Replaying Executions of Subsystems

Capture

- Select subsystem of interest and its boundaries
- Collect information flowing in and from the subsystem

Replay

- Automatically generate driver
  - Perform incoming calls
  - Consume outgoing calls
Replaying Executions: Scenarios

Scenarios
- Run your favorite dynamic analysis
- Extract subsystem/unit regression tests
Outline

Background
- Testing and maintenance
- My research focus and contributions

The Gamma Project
- Overview
- Impact analysis
- Replay of executions

Conclusion and Future Work
Related Work

Gamma Project

- Perpetual/Residual testing (Clarke, Osterweil, Richardson, Young)
- Expectation-Driven Event Monitoring (EDEM) (Hilbert, Redmiles, Taylor)
- Skoll (Porter, Schmidt, et al.)
- Bug Isolation (Liblit, Aiken, et al.)

Static and Dynamic Analysis

- Trace-based impact analysis (Law and Rothermel)
- Too numerous to list!
Summary and Conclusions

- **Gamma**
  - Dynamic impact analysis
  - Replay of users’ executions
- **Empirical studies**
  - User-based dynamic impact analysis is practical
  - Importance of using field data
Ongoing and Future Work

Gamma
- Hybrid technique for impact analysis
- Replay of executions
- Stability of users’ behavior
- New analyses (e.g., memory-leak detection)
- Use of statistical analysis
- Larger live experiment

Testing of OO software
- Regression testing of large industrial software
- Augmentation based on change analysis
- Automated support for testing and maintenance

Static and dynamic analysis for security
- Analysis to prevent DOS attacks from mobile code
Collaborators on this work

- Taweesup Apiwattanapong
- Anil Chawla
- Mary Jean Harrold
- Bryan Kennedy
- Jim Law
- Gregg Rothermel

For more information:

- [http://gamma.cc.gatech.edu](http://gamma.cc.gatech.edu)