Improving Effectiveness of Automated Software Testing in the Absence of Specifications

Tao Xie
Department of Computer Science
North Carolina State University, Raleigh
http://www.csc.ncsu.edu/faculty/xie/

2005 July – Defended Ph.D. Dissertation
advised by David Notkin
at Dept. of Computer Science and Engineering
University of Washington, Seattle
Testing Setup

Test inputs + Program = Outputs

Expected Outputs
Problems

• Are some of those generated tests redundant?  [Xie et al. ASE’04]
Problems

- Are some of those generated tests redundant? [Xie et al. ASE’04]
- How can we generate non-redundant tests? [Xie et al. TACAS’05]
Problems

• Are some of those generated tests redundant? [Xie et al. ASE’04]

• How can we generate non-redundant tests? [Xie et al. TACAS’05]

• What if we don’t know expected outputs? [Xie&Notkin ASE’03, Xie&Notkin ICFEM’04]
Problems

- Are some of those generated tests redundant? [Xie et al. ASE’04]
- How can we generate non-redundant tests? [Xie et al. TACAS’05]
- What if we don’t know expected outputs? [Xie&Notkin ASE’03, Xie&Notkin ICFEM’04]
- How can we know changes don’t introduce error? [Xie&Notkin ICSM’04]
Specification-Based Testing

Specs often don’t exist in practice

• Are some of those generated tests redundant?
• How can we generate non-redundant tests?
• What if we don’t know expected outputs?
• How can we know changes don’t introduce error?

Previous Work

TestEra [Marinov et al. 01],
Korat [Boyapati et al. 02],
AsmIT [Grieskamp et al. 02],
JPF [Visser et al. 04],
ASTOOT [Doong et al. 94],
JML [Cheon et al. 02],
etc.
Contributions

- Effectively invest machine resources
  - detect redundant tests [Xie et al. ASE’04]
  - generate non-redundant tests [Xie et al. TACAS’05]
Contributions

- Effectively invest machine resources
  - detect redundant tests [Xie et al. ASE’04]
  - generate non-redundant tests [Xie et al. TACAS’05]

- Effectively invest human resources
  - select tests for inspection [Xie&Notkin ASE’03/ASEJ’06]
  - summarize tests for inspection [Xie&Notkin ICFEM’04]
  - expose regression faults [Xie&Notkin ICSM’04/TSE’05]
Contributions

Framework for Improving Effectiveness of Automated Testing in the Absence of Specifications

Test inputs + Program execution info

Test generation Behavior inference

Redundant-test detector Non-redundant-test generator

Test selector Test abstractor Program spectra comparator

Existing test generators

Feedback

Framework for Improving Effectiveness of Automated Testing in the Absence of Specifications
Relationships to Industry

- **ASE 03** paper on Jov: test generation/selection based on operational violations (integration of Daikon and Jtest)
  - 12/06/03: Highlighted in testdriver.com newsletter #1
  - 07/23/04: Visited Agitar upon their invitation
  - 07/17/06: Compared in an ISSTA 06 paper by Agitar people
- **ASE 04** paper on Rostra (redundant-test detection): Parasoft Jtest 4.5 generates 90% redundant tests
  - 09/16/04: Received an email from Parasoft VP
  - 11/05/04: Visited Parasoft upon their invitation
- **TACAS 05** paper on test generation using symbolic execution
  - Techniques reimplemented upon JPF by NASA Research people
  - Techniques reimplemented for C# by Microsoft Research people

09/2000
Reading, Thinking, Proposing, Exploring, …
Dynamic call graph extractors, code Instrumentation, etc.

Summer 2001
Avaya Labs Research supervised by Dave Weiss on SPL, processes, workflows
Regression test prioritization w/ Daikon invs
Fault localization with value spectra
Regression testing with spectra comparison

2001 Nov
SE Genealogy
later developed as [ICSM 04/TSE 05]

2001
SE Research Links

Summer 2002
IBM Research supervised by Gary Sevitsky on SoftViz, perf analysis, dyn analysis
Knew about Parasoft Jtest
Attended a talk by Susan Horwitz
Why not put Jtest and Daikon together?

2002 Sept.

Why not put Jtest and Daikon together?

Jov: test generation and selection guided by Daikon invariants  [ASE 03/ASEJ 06]

Early 2003

Why not infer algebraic specs (with Jtest generated tests) like Daikon?

After prototyping the idea, found out later Henkel&Diwan published an ECOOP 03 paper

Near end 2003

While playing around Jtest-generated tests, found state redundancy (initially w/ Daikon frontend)

Developed my own infrastructure to identify redundant tests and generate non-redundant tests

Rostra: redundant test detection and non-redundant test generation  [ASE 04]

Sabicu: statistical algebraic spec inference in special/common test identification

[3rd Place ACM SRC Grand Finals, ISSRE 05]

Near end 2003

Rostra: redundant test detection and non-redundant test generation [ASE 04]

Why not construct/visualize state transition diagrams after state exploration? Too complex!!

Mid 2004

Why not abstract them?

Abstract with observers [ICFEM 04]
Abstract with fields [SAVCBS 04]

Near end 2004

Why not try it for AOP?

Raspect: AspectJ-redundant-test detection, later [ISSRE 06]

Why not combine Rostra and sym exec?

Realized JPF people used sym exec in their TACAS 03 paper

Near end 2004

Symstra: sym-exec-based test generation [TACAS 2005]

Near end 2004

Stabilizer: helping users avoid bugs in GUI apps [ICSE 05]
Story Behind being Dr. XIE (Aug 2005- )

Test Abstraction [ICFEM 04, SAVCBS 04]
Raspect [ISSRE 06]
Symstra [TACAS 2005]
...
Stabilizer [ICSE 05]

Piled Higher & Deeper
Life (or the lack thereof) in Academia
http://www.phdcomics.com/

Regression Oracle Augmentation [ECOOP 06]
Substra: integration testing [AST 06]

AOP Testing
test gen, regres testing
pointcut testing/inference,
[AOSD 06, WTAOP 06, MUTATION 06, FSE 06 Poster, ICSE 07]

Symstra+Jov [ASE 06]

MAPO: API Usages
[MSR 06]
App interface analysis
[ICSE 06 ER]

Data Mining for SE
[KDD 06 Tutorial, ICSE 07 Tutorial]

Testing Access Control Policies
[POLICY 06, ICICS 06, WWW 07]
Attack Generation [ISSRE 06 SP]

Security
Verification

Recently: testing/verifying web services/SOA, regression test gen …

Property generation [ASE 06]
Interface inference [ISSRE 06]
Conclusion

● Framework for improving testing effectiveness
  ● effectively invest machine resources
    ● redundant-test detection [ASE 04]
    ● non-redundant test generation [TACAS 05]
  ● effectively invest human resources
    ● test selection [ASE 03/ASEJ 06]
    ● test abstraction [ICFEM 04]
    ● program-spectra comparison [ICSM 04/TSE 05]

● Lessons learned
  ● Working around industrial/existing tools helps
  ● Talking to researchers helps (collaboration also helps)
  ● One research idea leads to another (from hands-on experience)
  ● A mixture of holding hammers + nails
  ● Big picture (don’t get lost when too excited about low-level solution details)
Questions?
Lessons Learned

- Working around industrial tools helps
- Dynamic analysis tools can be integrated too
- Both measuring and managing redundancy are important
- Breaking into pieces helps
- Grouping pieces helps
- Looking inside helps
- Exploit the most out of artifacts that already exist
- Unavoidable for tools to ask help from developers (wisely)
- Automatically generating complex arguments is more difficult than expected
- Practical lightweight specifications may help
- Model-based testing may be a good way to go when doing integration or system testing