Tester-Centric Automated Testing: Bringing Humans Into the Loop

Matt Staats
KAIST
Outline

• Overview of software testing
• Highlight the problem
• Review of recent work toward solution
  – Empirical study demonstrating alternative approach is difficult to use
  – Application of technique demonstrating potential value
• Future work centered on concept of “tester-centric automated testing”
Software Development

Specification → Talented Developers? → Program

Implements
The Big Question

Does the program accurately represent the specification?

Specification

Program

Talented Developers?
Testing Process

- Specification
- Test Inputs
- Talented Developers?
- Program
  - Executed Over
  - Implements
- Oracle
  - Evaluates Correctness
- Correct/incorrect
  - Path through program
Testing Process

- Specification
- Talented Developers?
- Test Inputs
- Program
- Executed Over
- Oracle
- Evaluates Correctness
- Path through program
- Correct/incorrect
Testing Artifacts – In Practice

Specification

Talented Developers?

Program

Test Inputs

Implements

Executed Over

Oracle

Evaluates Correctness

Path through program

Correct/incorrect
Test Input Generation

Test Input Generation Tools

Symbolic Execution
- Symbolic Java Finder
- KLEE

Concolic Execution
- CUTE
- jCUTE
- DART
- SAGE
- SCORE

Random Testing
- Randoop
- Adaptive Random Testing
- Naive Random Testing

Genetic Algorithm-Based Testing
- Testful
- AUSTIN
- Iguana
Test Input Generation

- Symbolic Execution
  - Symbolic Java Finder
  - KLEE

- Concolic Execution
  - CUTE
  - jCUTE
  - DART
  - SAGE
  - SCORE

- Random Testing
  - Randoop
  - Adaptive Random Testing
  - Naïve Random Testing

- Genetic Algorithm-Based Testing
  - Testful
  - AUSTIN
  - Iguana

- Good work, progress in reachability, efficiency, etc.
Test Input Generation

- Symbolic Execution
  - Symbolic Java Finder
  - KLEE
- Random Testing
  - Randoop
  - Adaptive Random Testing
  - Naïve Random Testing
- Concolic Execution
  - CUTE
  - jCUTE
  - DART
  - SAGE
  - SCORE
- Genetic Algorithm-Based Testing
  - Testful
  - AUSTIN
  - Iguana

What now?

- Unclear how users can use tools
- We make a lot of work for people
Test Input Generation

Problems

Test Input Generation Tools

Symbolic Execution
- Symbolic Java Finder
- KLEE

Random Testing
- Randoop
- Adaptive Random Testing
- Naive Random Testing

Genetic Algorithm-Based Testing
- Testful
- AUSTIN
- Iguana

More Work

What now?

Problems

More Work

What now?
Test Input Generation

Problems

Test Input Generation Tools

Symbolic Execution
- Symbolic Java Finder
- KLEE

Random Testing
- Randoop
- Adaptive Random Testing
- Naive Random Testing

More Work

What now?

Problems
More Work
Problem: No Support for Test Oracles

Program

Test Input Generation Tools

Symbolic Execution
- Symbolic Java Finder
- KLEE

Concolic Execution
- CUTE
- jCUTE
- DART
- SAGE
- SCORE

Random Testing
- Randoop
- Adaptive Random Testing
- Naïve Random Testing

Genetic Algorithm-Based Testing
- Testful
- AUSTIN
- Iguana

Test Inputs

Oracle

Must evaluate test results
Existing Work: Automatic Oracle Generation

• Idea: automatically generate a test oracle from the system
• User then (necessarily) evaluates result
• Several approaches, varying result
  – Program invariant generation
    • Daikon, AutoInfer, Xie/Notkin work
  – Trace generation
    • EvoSuite
Existing Work: Automatic Oracle Generation

- Idea: automatically generate a test oracle from the system
- User then (necessarily) evaluates result
- Several approaches, varying result
  - Program invariant generation
    • *Daikon*, *AutoInfer*, Xie/Notkin work
  - Trace generation
    • EvoSuite
Automatic Invariant Generation

- Represents potential solution for supporting oracle creation
- Unclear how effective users are at classifying results
  - Problems if poor
  - Little evidence in favor of use
- Study: Daikon dynamic invariant generator
  - Approx. 30 students
  - 3 programs
- Thanks to 최윤자, 김문주 for loaning students

Program

Randoop (Test Input Generation) + Daikon

Program + Program Invariants

User

Correct Invariants

Incorrect Invariants

Randoop (Test Input Generation) + Daikon

Program + Program Invariants

User

Correct Invariants

Incorrect Invariants

Randoop (Test Input Generation) + Daikon

Program + Program Invariants

User

Correct Invariants

Incorrect Invariants

Randoop (Test Input Generation) + Daikon

Program + Program Invariants

User

Correct Invariants

Incorrect Invariants
Daikon: User Effectiveness

Results (Our Guess, Their Guess)

% of invariants in category

[(True, True)]
[(True, False)]
[(True, None)]
[(False, True)]
[(False, False)]
[(False, None)]
Daikon: User Effectiveness

% of invariants in category

Results (Our Guess, Their Guess)
Big takeaway: automatic invariant generation tools pretty hard to use
Daikon: User Effectiveness

Results to be submitted to ISSTA 2012 with:
Shin Hong, Moonzoo Kim, Gregg Rothermel

% of invariants in category

Results (Our Guess, Their Guess)

(0,0)
Test Oracle Generation Support

- As an alternative to complete construction, we thought we could support users in making oracles
- Select *oracle data*: part of system oracle defined over
- User still has to define oracle
Test Oracle Generation Support

- Mutation testing was used
  - Change program several small ways
  - Determine where and when we can detect changes
- Result is that for a set of test inputs, person has a list of useful variables
- Goal: do better than other methods of selecting oracle data

### Diagram

- **Program**
  - Generate Mutants
  - Run Against Test Inputs (Generated Externally)
- **Mutants**
  - Measure Variable Effectiveness
  - Variable Effectiveness Ranking
- **Oracle Data**
  - Tester Specifies expected value
  - Expected Value Test Oracle
Test Oracle Generation Support

Latctl

DWM_2

Person

DWM_1
Big takeaway: automatic oracle generation support shows some promise, avoids problems of user classification.
Test Oracle Generation Support

Results under submission to ICSE 2012 with: Greg Gay, Mats Heimdahl

DWM_2

DWM_1
Tester-Centric Automated Testing

• High-level problem: poor integration of users into automated testing techniques
  – Current techniques are very (maybe too) demanding on users
  – Our own approach provides direction, has promise

• Three takeaways
  – Users are necessary, but often ignored in automated testing
  – Existing methods of supporting users in test oracles have problems
  – Proposed method maybe can do better

• Can we do better?
Tester-Centric Automated Testing

- Leads to ideas for future work
- Several problems/issues left
  - Method of supporting oracle selection is coarse at best

Program — Mutants — Variable Effectiveness Ranking

Test Inputs (Generated Externally) — Tester — Oracle Data

Generate — Measure Variable Effectiveness — Run Against

Specifies expected value — Expected Value Test Oracle
Tester-Centric Automated Testing

- Leads to ideas for future work
- Several problems/issues left
  - Method of supporting oracle selection is coarse at best

![Diagram showing the process of testing and mutant generation]

**How many mutants?**

**One oracle data for all inputs?**

Crude, simple idea
Tester-Centric Automated Testing

• Leads to ideas for future work
• Several problems/issues left
  – Method of supporting oracle selection is coarse at best
  – Test input and oracle generation always separate
Tester-Centric Automated Testing

• Leads to ideas for future work
• Several problems/issues left
  – Method of supporting oracle selection is coarse at best
  – Test input and oracle generation always separate
Tester-Centric Automated Testing

• Leads to ideas for future work
• Several problems/issues left
  – Method of supporting oracle selection is coarse at best
  – Test input and oracle generation always separate
  – In generating inputs, no consideration of individual user preferences
    • Lots of inputs, unclear user understands / wants them
    • Some work on simplifying inputs, but...
    • Daikon study indicates people vary a lot
Tester-Centric Automated Testing

- Leads to ideas for future work
- Several problems/issues left
  - Method of supporting oracle selection is coarse at best
  - Test input and oracle generation always separate
  - In generating inputs, no consideration of individual user preferences
Tester-Centric Automated Testing

• Leads to ideas for future work
• Several problems/issues left
  – Method of supporting oracle selection is coarse at best
  – Test input and oracle generation always separate
  – In generating inputs, no consideration of individual user preferences

Can we use this?

Program → Test Input Generation → 1000s of Test Inputs → User

태스트 모두 딱ильно해!
Tester-Centric Automated Testing

- Add together potential solutions, view of automated testing changes considerably
- More about optimizing for user preferences and saving user time
• Need fast, effective method of determining oracle data
• Need fast, effective method of determining oracle data
• Prioritization / filtering is about maximizing result relative to cost of user time
Tester-Centric Automated Testing

- Need fast, effective method of determining oracle data
- Prioritization / filtering is about maximizing result relative to cost of user time
- Must incorporate user preferences into generation / filtering process
• Need fast, effective method of determining oracle data
• Prioritization / filtering is about maximizing result relative to cost of user time
• Must incorporate user preferences into generation / filtering process
• Need interface for users to work with
Now vs. Future

Test Input Generation Tools
- Symbolic Execution
  - Symbolic Java Finder
  - KLEE
- Concolic Execution
  - CUTE
  - jCUTE
  - DART
  - SAGE
  - SCORE
- Random Testing
  - Randoop
  - Adaptive Random Testing
  - Naive Random Testing
- Genetic Algorithm-Based Testing
  - Testful
  - AUSTIN
  - Iguana

What now?

Program

Test Inputs

Oracle Data

Prioritization / Filtering

User Preference Learning

Test Inputs

Test Input Generation

Expected Result Specifier / Executor

Result!
Questions