SEBASE

On Test Data Generation of Object-Oriented Software

Andrea Arcuri and Xin Yao
University of Birmingham, UK

Windsor, 13th of September, 2007

EPSRC

Outline

Introduction

Characteristics of Object-Oriented Software

State-of-the-Art

Final Discussion

- Different techniques exist for automating the generation of test data
  - Symbolic Execution
  - Search Based Techniques
  - etc.
- Most of the work has been concentrated on procedural software (e.g., C language)
- Object-Oriented (OO) software is more difficult to test
- White Box Testing: *branch coverage*
- Java as an example of OO language

What makes OO software more difficult to test?

- State Problem
- Information Hiding
- Polymorphism
- others

State Problem

- Software can have an internal hidden state (e.g, internal variables of an object)
- Before reaching the branch under test, the state needs to be put in the right configuration
- *Sequences* of function calls are hence required
- It affects procedural software as well (e.g., static variables in C)
- In general, internal states appear more often and in a more complex way in OO software
- More sophisticated techniques are hence required

Information Hiding

- Input data structures might have a hidden internal state
- In languages as C, even complex data structures have their state visible
- Object constructors might be not visible
  - singletons
  - internal classes
- Private methods cannot be called directly

Polymorphism

- The actual executed code is known only at runtime
- Source code analyses cannot always give the right answers
- Search space of the input parameters is enlarged (e.g, references to the class *Object*)

Other possible problems

- Exceptions
- Templates
- Subclassing classes which code is not available
- etc.

Conventional Techniques

- Exhaustive techniques with heuristics
  - symbolic execution
  - state matching
  - etc.
- Many problems:
  - state explosion (particularly for the sequence lengths)
  - non-linear predicates
  - non-primitive data types
  - loops
  - arrays
  - etc.

Search Based Techniques

- The task of generating test data is modelled as a *search problem*
- A *fitness function* f is used to judge the quality of a test case
- Several search algorithms:
  - Hill Climbing
  - Simulated Annealing
  - Genetic Algorithms
  - Memetic Algorithms
  - Estimation of Distribution Algorithms
  - etc.
- Successfully applied in many different contexts (e.g., scheduling, design of airplane wings and protein structure prediction)
- Do not particularly suffer from the previous limitations
- However, not enough evidence for claiming that they are “better”

Issues in the current State-of-the-Art

- tests done on small clusters of classes
- no common benchmark
- usually, no comparisons between different techniques
- lack of theoretical work
- little work with search algorithms (e.g., first paper in 2004 by Tonella)

Which search algorithm?

- Strong bias toward Genetic Algorithms
- Local search algorithms are often considered not suitable
- However, not all the test problems are so difficult
- *Memetic Algorithms* (MAs) combine together evolutionary algorithms and local search
- At least in our work, MAs outperformed several other search algorithms
- However, comparing search algorithms is not a trivial task
- Exploiting domain knowledge improves the performances

Scalability

- Scalability is an important factor
- Many research prototypes
- Might get high coverage, but at which computational/time cost?
- Can they scale up to industrial-size software?

Conclusion

- Object-Oriented languages are widely used in software development
- Testing OO software is more complex than testing procedural software
- Still many research questions
- Search Based techniques are giving promising results, in particular Memetic Algorithms