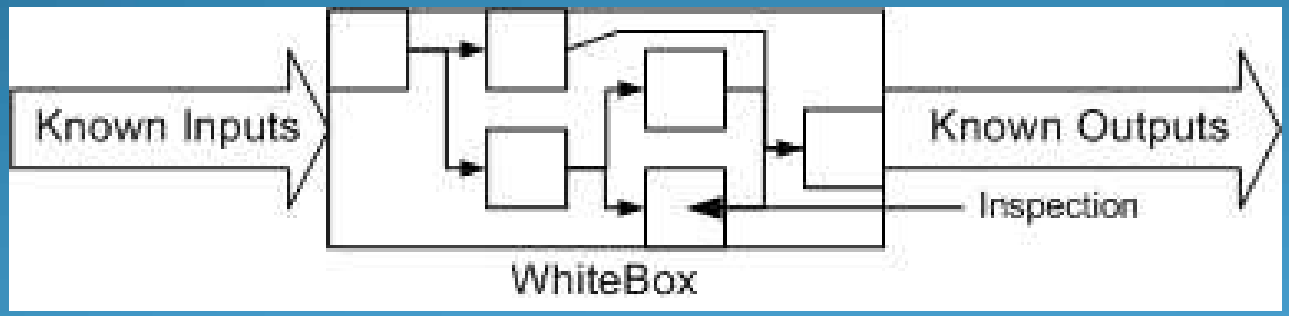
Automated White-box Test Generation

Does it help Software Testers?



Sonam Gupta
COMP 597 (Software Testing)
Spring 2014

What is and testing

White-box Testing

- Testing is done by accessing the internal code
- Has the ability to change the code
- Can be tested with all possibilities

Black-box Testing

- Testing is done only with the external perspective of the code
- Does not have the ability to change the code
- Checks the output if its according to the input

What is this research about?

- To investigate whether automated test generation really helps software testers or not?
- Comparison of the testers writing test cases and generation of test cases by the automated test tool
- White-box test generation adopted to answer the unanswered question…
- EVOSUITE- test suite tool used to support the research

Now, how does this EVOSUITE work? Let's see.....

How does EVOSUITE work?

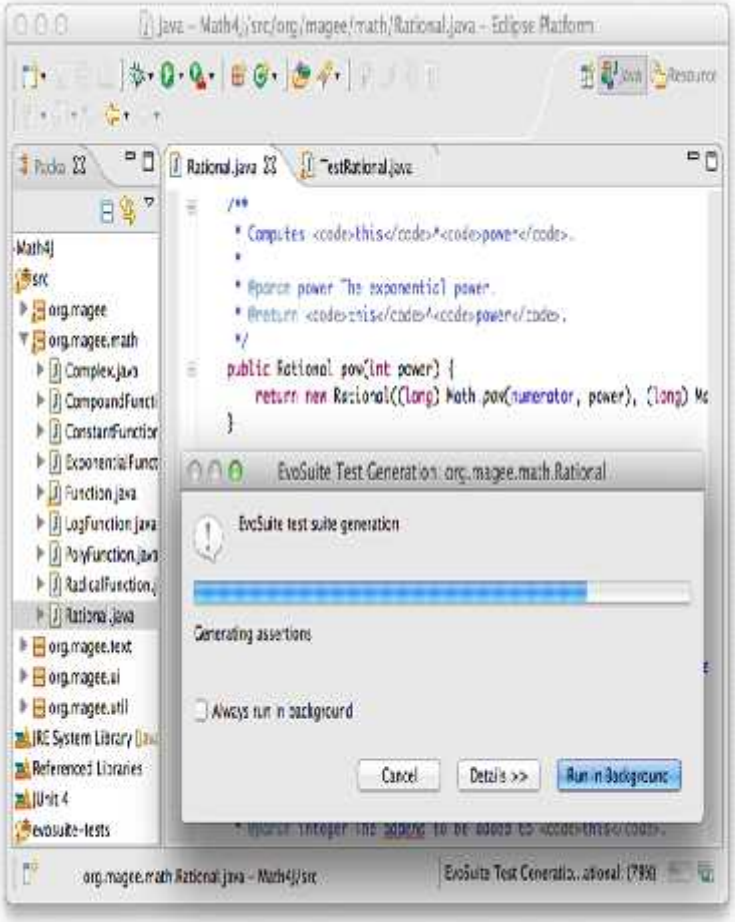


Fig.1: EVOSUITE-eclipse plugin, generating test cases for a class [1]

- Fig.1 shows that Eclipse has a plugin for EVOSUITE
- Used for Java codes
- Test suite tool that generates test cases with assertions
- Search based approach
- Genetic algorithms used to evolve test suites [2]
- EVOSUITE guides towards reaching the goal of testing a program
- Handles dependency among predicates (true-false statements)

Their study and experiments…

- Controlled experiment with involvement of 49 human subjects and an EVOSUITE
- **Experiment**: three Java classes given to every subject with already added seeded faults (known bugs)
- To produce JUnit with the help of EVOSUITE
- Oh! Yes, EVOSUITE automatically creates JUnit test suites
- Performance based on coverage, seeded faults detection, mutation score, erroneous tests production

Detailed study (three results)

- Testing tool effectively work for what they are designed with high code coverage
- Detailed study revealed: testing tools does **not** confirm high code coverage in finding faults
  - Number of faults found by EVOSUITE = number of faults found by manual testing
  - No high mutation score (measurement of sufficiency of test set)
- Developers spend most of their time on analyzing how the tool works… need to give more thoughts towards test suite tool

The basic idea…Methodology

- The whole idea for the research: already developed Java class, need to create a test suite to find faults
- The future thought: regression testing (no change in any part of the software because of newly created class)
- They based their study in order to answer the four questions

Those four questions…? ? ? ?

- How will the use of automated tools for testing affect:
  - Structural code coverage ?
  - Ability of manual testing in detecting faults in the Java class ?
  - Mismatch of the behavior of tests ?
  - Detection of regression faults by test suite ?

Solve the assignment…

- **The Java class :**
  - already created (Well, that's why it is needed to be tested ☺)
  - Pre-tested using 25 candidate classes (appropriate difficulty) by the authors with test suites and had a pilot study of those Java classes. Regular programmers took a long time. (Replacement)
  - The table below shows the classes used in the experiment
  - Five deliberate faults in the classes using MAJOR (mutation analysis tool) and one original class (no seeded faults)

Project	Class	LOC	Methods	Branches	Mutants
XOM	DocType	296	26	242	186
Commons CLI	Option	155	42	96	140
Math4J	Rational	61	19	36	112

Solving the assignment (contd.)

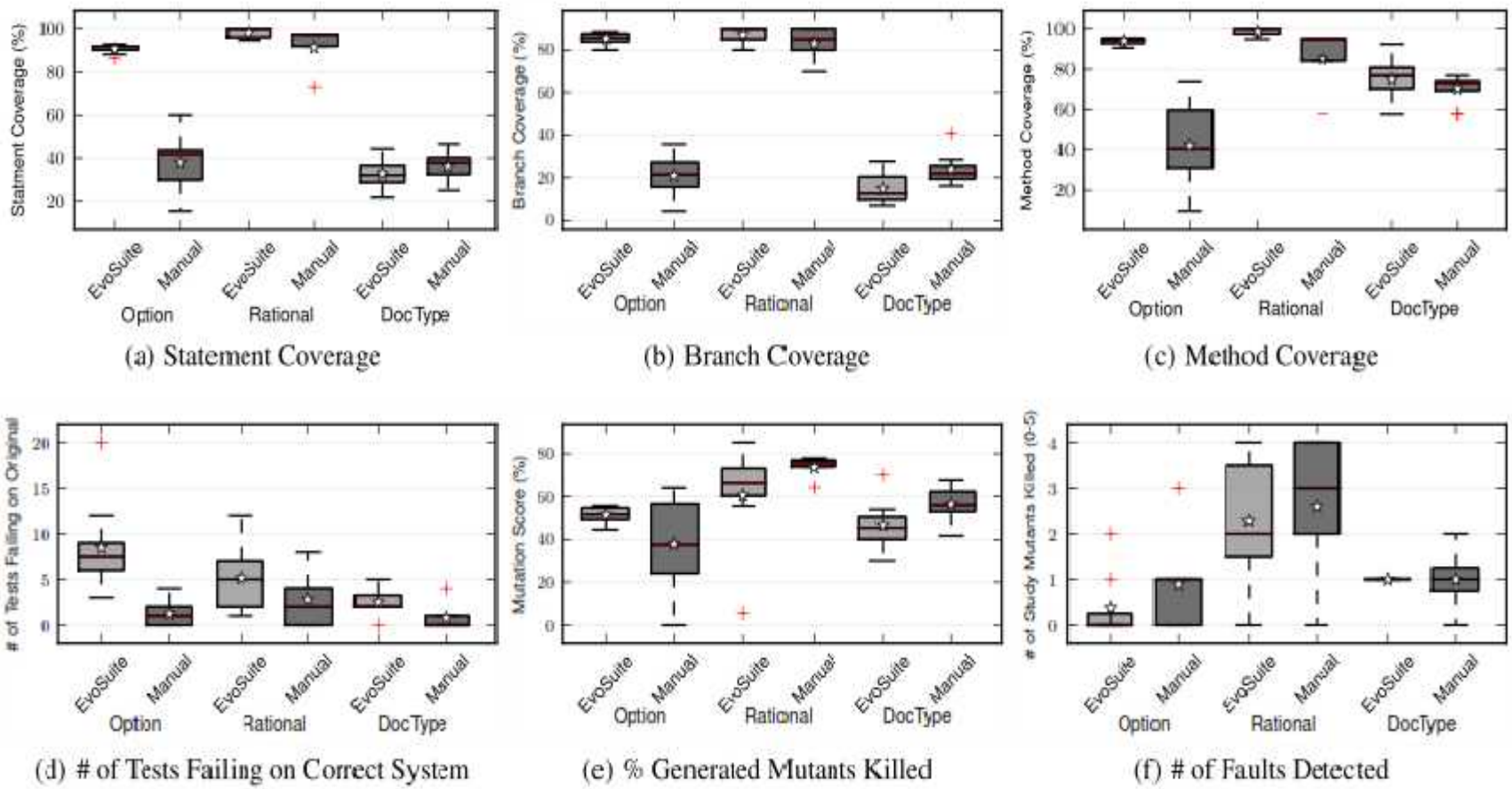
- Human subjects were limited to 49 which restricted creation of Java classes to three (for better statistics)
- Java classes then were based on few criteria
- Experiment process:
  - Different subject IDs
  - Experiment pack included-subject ID, statement of consent, questionnaire on background, instructions for performing the experiment, exit survey
  - Goal: to test the class carefully, detect the fault but not fix the code
- Modification in Eclipse for storing a copy of test suite when subjects run their suite.

Analysis of Results

Variable	Option		Rational		DocType	
	$\hat{A}_{12}$	p-value	$\hat{A}_{12}$	p-value	$\hat{A}_{12}$	p-value
# failing tests on original	0.97	0.001	0.70	0.289	0.83	0.026
# of detected faults	0.31	0.165	0.43	0.738	0.50	-
Mutation score	0.65	0.321	0.37	0.530	0.22	0.065
% Statement coverage	1.00	0.001	0.79	0.109	0.37	0.399
% Branch coverage	1.00	0.001	0.64	0.431	0.19	0.040
% Method coverage	1.00	0.001	0.91	0.014	0.70	0.178

- Basis of the evaluation of results: shown in the figure
- Those 5 individual study with faults detected had different versions of classes with one correct version.
- Combination of results by using EVOSUITE and manual testing
- Small number of subjects for statistical effect between the testing approaches

Statistics using Mann-Whitney U test



(a) Statement Coverage (b) Branch Coverage (c) Method Coverage

(d) # of Tests Failing on Correct System (e) % Generated Mutants Killed (f) # of Faults Detected

Figure 2: Test suite properties, comparing EVOSUITE against manual testing (boxes spans from 1st to 3rd quartile, middle lines mark the median, whiskers extend up to 1.5× the inter-quartile range, while plus symbols represent outliers and stars signify the mean).

Results with respect to those four Questions

- 1. Structural code coverage:
  - Referring figures a-c from above slide shows the structural coverage
  - Testing *Option* and *Rational* classes, code coverage has a high improvement for branch, statement, method coverage
  - For DocType, did not improve much due to configuration error in it
  - Assumption that EVOSUITE will anyway work better fo r DocType
  - **Automated test tools achieves higher code coverage than manual testing**

Answer to Question # 3

- Mismatch between the tests and specifications:
  - For *Option*, the number of tests failed (EVOSUITE) on original class was larger than the ones by the subjects
  - Initial test suite using EVOSUITE were expected to give such results but this happened for all the remaining test suites
  - Constant amount of failed tests for both EVOSUITE and manual testing
  - Actually EVOSUITE has a slight increase in mismatching with the desired class behavior

Answer to Question # 2

- Surprisingly sad results
- Number of faults detected (EVOSUITE and manual testing)
  - EVOSUITE did not perform well
  - Fault detection decreased for *Option (3.0 faults)* and *Rational (2.86 faults)* when compared with manual testing (without EVOSUITE) results. For DocType, fault detection by EVOSUITE = fault detection by subjects
  - To determine the reason of these results- incorrect assertions for EVOSUITE
  - Thus EVOSUITE proved to be not of much help to the testers

Oh well, now how does the assertion help increase the efficient in fault detection… We'll see in sometime….

Answer to Question # 4

- Regression Fault detection: (examines the mutation score)
  - Mixed opinion to this.
  - EVOSUITE performs really well for the class *Option* but worse for the other two
  - EVOSUITE covers high code coverage in *Option* so, gave a higher mutation score even though not well in fault detection
  - Manual testing for the other two classes killed more mutants
  - From the statistics, EVOSUITE users created better test suites
  - Studies say that EVOSUITE may improve in regression fault detection
  - Testers struggle to generate large number of test cases manually
  - **Conclusion**: automated test generation does not guarantee better mutation score

Researchers discussed…

- Answers to the four questions : researchers think that automated generation techniques need more improvement
- Better understanding required
- Use of automated testing can have greater impacts on software testing

Assertions are Influential

- Assertions are helpful for the discovery of bugs in the program
- The subjects found confirming the correctness of assertions is easier than generating them
- Assertions created by EVOSUITE users (1.41 to 4.9) > assertions created by subjects (1.44 to 1.68)
- The configuration error in the class DocType could have been corrected depending on the assertion
- Need of development in the methods to select the assertions

Future work

- Testers' trust on the automated tools being wasted
- Developing the testing tools might earn the trust back
- Testers could have gained more benefit from the tool (for example, removing the configuration error)
- Getting high code coverage is not the only thing
  - User studies needed
  - Careful consideration of how assertions are generated
  - Selecting the test oracle (determines if test passed or failed) needs improvement (helps in detecting faults, as per the previous empirical studies)
  - User suggestions/feedback

Finally their Conclusion…

- Constant work on white-box test generation only focused on technical aspects
- Researchers' study concludes that automated tools are very good for high coverage test cases but are not really useful in detecting faults
- The words of the researchers…. " It is time for software testing research to consider the follow-up problem to white-box generation: once we have generated our test data, how should the developer use it? " [1]

What do I think?

- I agree with the researchers that automated test generation needs more improvement
- EVOSUITE seems to be very helpful (if developed more) for Java programmers and software testers
- Testers need to study more about the automated testing techniques

References

[1] Gordon Fraser, Matt Staats, Phil McMinn, Andrea Arcuri, Frank Padberg. 'Does Automated White-Box Test Generation *Really* Help Software Testers?', ISSTA 2013, ACM, NY, USA.

[2] Gordon Fraser, Andrea Arcuri. 'Evolutionary Generation of Whole Test Suites', QSIC, Los Alamitos, California, 2011.

[3] Lionel C.Briand. 'A Critical Analysis of Empirical Research in Software Testing', IEEE Transactions on Software Engineering, 2002.

Thank you !