Quantifying the Impact of Changes in Java Software

Ying-jie Zhang[1,2]

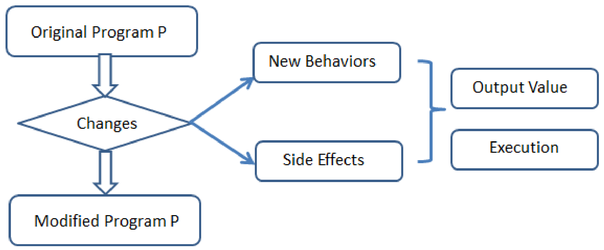
Advisor: Raul Santelices[1]

[1] Dept. of Computer Science and Engineering, University of Notre Dame, IN, 46556, USA
[2] Computer Science and technology ,Tsinghua University, Beijing, 10084, China

①Motivation

Sensibility Analysis is a technique used to determine how different values of an independent variable will impact a particular dependent variable under a given set of assumptions, which is widely used in many academic fields.

Program Slicing is a well-known analysis that can be used for various tasks. It can be used in debugging to locate source of errors more easily.



Drawbacks of the Traditional Program Slicing Technique:

-- it may identify affected statements that cannot be affected in any execution
-- it may report too many statements that are less necessary for software developers to review

②Technique

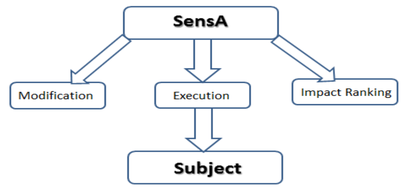
Dynamic Probabilistic Slicing (dynamic p-slicing) is a new technique we present in this research that makes the impact ranking closer to actual facts. Since this technique is based on real executions of representative test suites, our dynamic p-slicing will not only inherit the benefits of program slicing, but will also provide a relevance score for each statement to focus more on the actually impacted code so that the information provided is more effective. Sensibility analysis, which for many inputs modifies the values computed by A many times to determine the change/ frequency of effects on B, is the core idea of our dynamic probabilistic slicing approach.

SensA, based on Soot library, is a tool we are building to evaluate our new technique. It instruments and monitors the location of a prospective change in a Java program, and then reports the impacted statements and probability ranking. What's more, it is based on java byte-codes, rather than the source codes.

Use of the Technique

▲ Using SensA, software developers can focus on the "most impacted" parts of the program first.

▲ Also, SensA can be used for many other tasks, such as mutation testing and testability analysis.

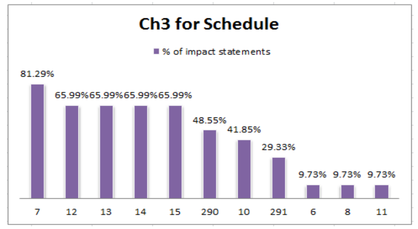


③Results

Schedule.

	Total Valid_Test*Run	Impacted Statements	Range (%)	Modified Value Type
CH1	1980	58	6.313 – 45.757	Short
CH2	596	12	7.382 -- 55.034	Float
CH3	791	11	9.735 – 81.290	Integer

	Lines of Code	Number of tests	Number of Changes
Schedule	290	500	3



- Number of impacted statements is small compared to the lines of code.
- Only a part of impacted statements possess a relative large percent (more than 50%).

④Future work

More Experiments

Since the main subjects we experiments on are Schedule and NanoXML, whose source codes are both relatively small, we will test our technique on other more complicated subjects to evaluate its effectiveness.

Extensions of SensA Tool

SensA will support more data types and provide more strategies (modification algorithms).

Moreover, it should be plug-ins by third parties in the future for further researches .

Study of Application on Change-impact Analysis

As software systems grow in size and complexity, their dependency webs of information also extend beyond most software engineers ability to comprehend them. And the major practical application of dynamic p-slicing is to improve the effectiveness of change-impact analysis whose task is to solve the problem mentioned above.

Combination with Other Program Analysis

Dynamic p-slicing has its own drawbacks, for example, it cannot guarantee that the observations generalize to all executions and thus we may combine it with static p-slicing to create a better impacted statements order for software development.

⑤Reference

[1]Raul Santelices, Mary Jean Harrold. Probabilistic Slicing for Predictive Impact Analysis
[2] Petar Tsankov, Wei Jin, Alessandro Orso. Execution Hijacking: Improving Dynamic Analysis by Flying off Course
[3] Yue Jia. An Analysis and Survey of the Development of Mutation Testing
[4] Jeffrey M. Voas. PIE: A Dynamic Failure-Based Technique