A Survey on Regression Test Selection Techniques on Aspect-Oriented Programming

Neerja Gupta
Department of Computer Science & Engineering
ASET, Amity University
Sector-125, Noida

Arun Prakash Agrawal
Department of Computer Science & Engineering
ASET, Amity University
Sector-125, Noida

Abhishek Singhal
Department of Computer Science & Engineering
ASET, Amity University
Sector-125, Noida

ABSTRACT
An aspect oriented programming is gaining a lot of popularity these days, there is a growing interest because of crosscutting concerns in object oriented systems. When the aspect oriented features are added into object oriented features the new program needs to be regression tested, and, to reduce the cost, Selection technique is used which eliminates the redundant test cases and thus makes them cost effective. Unfortunately the already existing approaches of object oriented programming does not work out for aspect oriented programming because of the following new features of aspects such as join points ,crosscutting concerns ,aspect weaving, etc. Therefore, this paper proposes the techniques used for object oriented programs and for aspect oriented programming.

Keywords
Aspect oriented programming, Object oriented programming, Regression testing, selection technique.

1. INTRODUCTION
As the software is modified during development and maintenance, the software needs to be regression tested to provide confidence that the changes do not introduce new errors and thus do not harm the system. Because the size of the test suite grows, regression-testing–selection-technique is used .It deals with the problem of selecting subset of test cases that tests the changed part of the software. A safe-regression – test selection selects all the test cases that contain faults in the modified software[1]. Various selection techniques have been described for procedural languages[2,3,4], object oriented languages[5,6,7] and aspect oriented languages[8,9,10]. Aspect oriented programming is a way of modularizing the crosscutting concerns and aspect-J is an implementation of aspect oriented programming as in the same way java is a way of modularizing common concerns for object oriented programming. Cross-cutting concerns are the parts of the program that affect many other parts of the system. They contain duplicate code as well as inter –dependent code. Logging and authentication system are the examples of cross-cutting concerns. In procedural and object oriented programming there is function or procedure calling whereas in aspect oriented programming ,the code to be implemented and the related crosscutting concerns i.e. dependent code can be accessed simultaneously . Aspect-j adds to java few constructs like point cuts, join points, advice, aspects. A join point is a well defined point in the program flow. A method call, exceptions are some examples of join points. Each method call at runtime is a different join point even if it comes from the same call expressions. Point cut picks out certain join points and values at those points .Point cuts don't do anything apart from that, to actually implement them we use advice. Advice is a code that is executed when a join point is reached. Advice brings together join point and body of code. Advice is of three types before, around and after. Aspects wrap up point cuts, advice and inter-type declaration (declarations that cut across classes). It is similar to class and can have methods in addition to crosscutting members[11].

2. Regression test selection for aspect oriented programs:
Software maintenance accounts for two-third of the overall software life cycle costs. Maintenance is necessary to fix defects, enhance functionalities and helps the software to work on different environments. Therefore, Resolution Test Cases are designed whenever the application program is modified and Regression Testing is done to ensure that no new errors are introduced. Regression Testing is carried out at all levels whether it is unit, integration or system level. As the software is released, changing reports or failure reports are compiled and the software is modified to provide necessary changes. Resolution Testing is responsible for testing the modified parts , whereas Regression Testing is responsible for unchanged parts of the code that may be affected by the code change. Regression Testing is an expensive process, therefore, minimization of test cases is done . Regression Testing involves three techniques , they are , Minimization -which seeks to reduce the test cases by eliminating the redundant test cases, Selection- that deals with the problem of selecting subset of test cases that tests the changed parts and the last one is Prioritization- it is concerned with the ordering of test cases. It uses Bee Colony algorithm[12]. *Rothermel and Harrold have defined the Regression Test Selection Problem as : Let P is application program and P' is the modified program . T is the test suite used to test P. RTS technique will select T' such that* $T' \subseteq T$ *which is to be executed on P' in such a way that every error detected when P' is executed with T is also detected when P' is executed with T'[13].* Regression Test Selection technique have been used for procedural languages, object oriented languages, component based , database oriented applications, web applications , aspect oriented paradigm , and many more.

Aspect oriented software development is a popular approach for modularizing the cross cutting concerns, which then simplifies software maintenance and evolution. When aspect oriented features (aspects, point cuts, advices)[14] are added to object oriented program or when the aspect oriented program is modified, it is need to be regression tested to ensure that the methods crosscuts and the aspects which has been introduced should behave as expected. An aspect

changes the behavior of the original code for example, without any change to the original program which is a java program, a single aspect is added, which changes the pre and the post conditions of the method. Regression testing is more important for aspect oriented than object oriented programs because of pervasive effects of small code changes. Existing object oriented selection technique represented by[15] are based on Graph Traversal algorithm. This paper focuses on regression test selection technique for object oriented and aspect oriented programs.

3. RTS Technique for Object Oriented Programs

3.1 Firewall Technique

This technique is proposed by Kung [5], Hsia [6], Abdullah and White [7] and Jang [16]. It was originally given by Leung and White [17]. Its aim is to identify the modified version of the software. It identifies only the affected area. It selects all the test cases which exercise at least one class from within the firewall. It can be class level or can be method level. In class level Object relation Diagram (ORD) which shows the relationships, associations and aggregations between the classes [5]. And, in method level, methods are considered as the unit of retesting and aim at identifying all affected methods [16].

3.2 Program Model Based

2.2.1 Class Dependency Based [18]

Rothermal and Harrold have divided the problem of RTS of object oriented programs into two parts: first one is for application program and the second one is for modified program. Application program uses inter-procedural class dependence graph for original and modified program. As they incur a lot of overhead, hence program dependence graph is used. Whereas modified program uses class dependence graph.

2.2.2 Extended Control Flow Graph [19]

Harrold et al. was the first one who proposed safe RTS for java programs based on control flow analysis. He uses Graph walk algorithm [20] .His method consists of three steps: constructing, intermediate representation of source program, analyzing the graph and detecting the dangerous arcs and test case selections. Harrold et al. uses the JIG representation for modeling java programs.

2.2.3 Partition Based [21]

Mansour and Statieh have proposed two phase RTS technique. Their technique is based on affected areas i.e. changed parts of the modified programs and constructs Affected Class Diagram (ACD). ACD represents modifications made at the level of class, interface, web, services. Their technique then uses test coverage criteria and selects the subset of test cases. It works on two phases- Partioning and selection. In partioning, the original and the modified programs are modeled as Inter Relation Graphs [22]. Partioning is further dived into statement and Declaration. Statement consist addition, deletion and modification of statements whereas declaration is addition, deletion, modification of methods. And, in Selection analysis of partitions are done.

3.3 Design Model Based Technique [23, 24]

Model Driven Development has made Model Based Testing very popular. In this design model is refined to obtain the code. The widespread usage of CASE tools has made close correspondence between design model and code. It describes Further various RTS techniques for Class and State machine [25, 26], Sequence diagrams [27], Use case diagrams [28].

3.4 Specification Based Technique [29]

In industry, practical difficulty in RTS is that testers may not have access to design code and source code. Hence code based testing and model based testing does not work out in this situation. So to avoid this problem specification based technique has been evolved. It uses UML activity diagram for modeling the potentially affected requirements and system behavior. They have also classified the test cases into two types safe and target test cases. Target test cases are those that exercise the affected requirements and safety test cases helps in achieving pre defined coverage target.

Author's Name	RTS Techniques for object oriented programs	Key Features	Advantages	Disadvantages
Leung And White[17]	Firewall Based Technique	It analysis the data and control dependencies and also shows the dependencies among modified and interacting modules. It uses call graph. It identifies the affected area.	Computationally efficient, captures the dynamic behavior of class	It is imprecise and also do not handle exceptions in java
Rothermal And Harrold[19] , Mansour and stetieh[21]	Program-Model Based Technique	It Shows dependencies among methods, classes, and there control flow graphs and two phased technique	Applicable to both derived and modified programs, safe, more precise than firewall, more efficient than firewall, it combines the techniques that work at higher level of abstraction.	It does not handle few constructs of object oriented language such as exception handling, and, it is expensive, less efficient because of high overhead of inter-procedural graph. It is also imprecise for handling polymorphic calls
Ali [25]and Gorthi[28]	Design Model Based Technique	It is based on model driven development which consist of different UML diagrams such as sequence diagram, use case diagram	More efficient , suited for large programs , independent of implementation, higher level of abstraction as compared to code based technique	It is not safe , It requires close correspondence between requirement artifacts, design models, code, test cases which sometimes impossible to collect, Limited to model driven environment
Chen[29]	Specification Based Technique	Based on specifications such as activity diagram	More efficient, platform independent, can be extended to a wide class of programs, do not depend on static analysis of source code , largely dependent on quality and accuracy	It is not safe and it is less precise as compared to others

Table 1: RTS techniques for object oriented Paradigm.

4. RTS Technique for Aspect Oreinted Program:

4.1 Rothermal and Harrold:[30]

Rothermal and harrold proposes an approach which is based on graph traversal algorithm. In their approach aspect oriented features (such as point cuts, join points, aspects, advice [14]) are added into object oreinted programs, and the program is regression tested to make sure that the newly introduced features do not affect the code. He assumes two versions of program P and P' such that P is the original program which is basically a java program and P' is the modified program (aspect is added to java program). He presents a graph traversal algorithm which runs the test suite for the original program and obtains the coverage information and constructs the java interface graph (JIG)[20] for both original and modified program and then compare the CFG's .Comparison helps in detecting the dangerous arcs. An arc is dangerous if target of the CFG of both original and modified program differ[31]. Dangerous arcs are then rerun and the test cases are selected safely.

4.2 Guoquing Xu: [32]

The approach given by Rothermal and Harrold is based only on static analysis but most of the time dynamic analysis is required because of calling of external methods. Guoquing Xu gave an approach which is based on RETSA framework. This technique uses dynamic analysis to record coverage information for P and P' and static analysis for safety and precision. RETSA framework consist of five components : First one is Dynamic Coverage Recorder which takes the byte code of the program as input , runs the old test suite for P and P' and maps the dynamic execution path and outputs the coverage matrix. Second one is CGF Comparator which is used to compare the CFG's of the two programs. Third is the Safe Edge Identifier, which takes the two coverage matrices as its input and does the pointer analysis to detect the safe edges. Fourth is Candidate Selector which accepts the dangerous arcs and coverage matrix of P, looks into the coverage matrix the tests which covers the dangerous arcs. And, the last one is Final Selector which removes the tests output by computing the safe edge for each test.

4.3 Guoquing Xu:[33]

He gave another approach on aspect oriented program. This approach is an extension of JIG used by Rothermal and Harrold i.e. AJIG (Aspect-J Inter Module Graph). It is a new control flow representation for aspect-J softwares which captures the semantic intricacies of aspect-related interactions. An AJIG includes(1) CFGs that model the control flow within Java classes, within aspects, and across boundaries between aspects and classes through non-advice method calls, and(2) interaction graphs that model the interactions between

methods and advices at certain join points .It does not introduce extra nodes or edges to represent the lower level details of compiler specific code. They depend only on the input program and not on the implementation code of weaving program. He uses two phase graph algorithm in which first phase is defined as Inter-procedural Traversal and the second phase is defined as the Intra-procedural comparison. In first phase it compares the invocation order of two IG's and outputs the dangerous edges and the set of FP further processing advices whose invocation order remains same. In the second Phase, for each advice in FP it traverses its CFG in P and P'.

4.4 Romaine Delaware:[34]

His goal was to analyze which test cases are impacted by the introduction of an aspect in the base program. He divided the methods and test cases into two categories: Impacted (I) and Non –Impacted. The base program used is Java program and its test case is implemented into Joint. His main objective is to determine which test cases have modified , which test cases should be kept unchanged , which test cases should be removed in aspect oriented evolution. The analysis starts by detecting the methods in base code that are impacted by an aspect .An impacted method is a method where an aspect is woven. The test case is detected as impacted if it can reach the impacted method. His analysis consist of two parts: First, He builds a model that captures the relationship of aspects and base program. Then in second , He analyses the static call graph of each test case in order to determine whether iimpacted or not. In Impacted method model, join points are identified where the aspects have been weaved when java and aspects are run through aspect-j compiler. To evaluate the impact , he represents the relationship between the aspects and base code. The aspect-J compiler executes the java code and weaves the aspects , but he has extended the aspect-J compiler by offering an interface to add build listener, it then provides a list of relations between aspects and java code. And in second, static call graph checks whether test cases can reach impacted method or not.

4.5 Mark Harman & Tao Xie: [35]

He develops an approach for Automated Test Data Generation (ATDG) for aspect oriented programs. It is used to generate test data which covers aspectual behavior (achieving aspectual branch coverage [36]). He uses ASPECTRA framework, which takes input as aspects or java programs in which aspects are woven. To measure aspectual behavior, it uses the metrics of aspectual branch coverage, which measures the branch coverage within aspect code. The technique is based on dependence analysis based on slicing .It consist of four major components, that are, Aspectual- branch –identifier which identifies branches inside aspects. Relevant- parameter –identifier which identifies only relevant parameters because not all parameters of the methods of base class are relevant. Evolutionary Tester, it conducts evolutionary testing on relevant parameters. Aspectual branch coverage measurer measures the aspectual behavior. It is based on ASPECTRA framework, a novel framework for automatically generating test inputs for AspectJ programs. To test aspects in an AspectJ program, developers can construct base classes, which can be woven with aspects to produce woven classes. Aspectra synthesizes a wrapper class for each woven class. The wrapper mechanism allows test-generation tools to indirectly exercise advice related to call join points and public non-advice methods in aspects during test generation. At the same time, the mechanism prevents the methods in generated test classes from being advised by unwanted advice. Given wrapper classes, Aspectra leverages existing test-generation tools for generating test inputs.

Author's Name	Year	Technique used	Advantages	Disadvantages
Rothermal and Harrold[30]	2001	They run the test suite for original program and modified program and obtain the coverage information, and , then detects the dangerous arcs and the test cases containing dangerous arcs are rerun. Based on MSIL(Microsoft Intermediate Language).	• Safety, higher precision • It enables the developer to quickly recognize interaction pattern that supports modular reasoning and focuses on the causes of potentially non modular interaction	• code-weaving of AOP messes up the execution path hence results in redundant tests cases. • For external methods , JIG does not represent the intra – method control flow • It does not considers multiple advices apply at shadow
Guoquing Xu [32]	2006	He uses RETESA framework which involves dynamic coverage of information for both P and P' and static analysis to achieve safety and precision.	• Safety and higher precision • It involves dynamic coverage of information. • Better than above approach	

Guoquing Xu[33]	2007	He also gave AJIG approach which is an extension of JIG approach. It captures the semantic intricacies of aspect –related interactions. It uses two phase graph algorithm. The first phase is Inter-procedural traversal in which it compares the invocation order of advices and the second phase is Intra-procedural traversal in which, for each advice it traverses its cfg.	• It is independent of the low-level implementation code introduced by a weaving compiler. • It not only serves as a basis of regression test selection but also does the static analysis of aspect-J. • It reduces the number of test cases effectively • *AspectJ Inter-module Graph* (AJIG) extends the Java Interclass Graph from [8] with representations	• This approach does not appear to be effective for woven byte code.
Romain Delamare[34]	2008	He categorizes the test cases as well as the methods into two parts : Impacted and Non Impacted. Impacted methods are those where an aspect is woven , or the methods that are affected. He builds a model that represents the relation between aspect and java code called as Impacted method model and uses static call graph to check whether test cases can reach impacted method or not.	• It provides confidance. • It is used to validate the preservation of certain behaviours after the addition of a new aspect or the evolution of an aspect already woven. • It selects the test cases that must be executed to validate this behaviour preservation. • It takes the advantage of the point cut expression to evaluate the impact of aspects on the test cases.	• This approach helps in identifying the test cases that do not need to be executed but they do not consider the evolution of the impacted test cases.
Mark Harman and Tao Xie[35]	2009	He develops an approach for automated test data generation .Its objective is to cover the aspectual branches. It uses ASPECTRA framework. , a novel framework for automatically generating test inputs for AspectJ programs. To test aspects in an AspectJ program, developers can construct base classes, which can be woven with aspects to produce woven classes. Aspectra synthesizes a wrapper class for each woven class.	• Manual test data generation for achieving coverage is tedious, error-prone and expensive. Therefore, for some time, automation has been considered to be the key to effective and efficient test data generation. • Because of the widespread practitioner usage of branch coverage, this testing goal has received a lot of attention from the software testing research community.	• It generates a lot of redundant test cases.

Table 2: RTS Techniques for Aspect Oriented Programs

5. Tools for Aspect- Oriented Programs

5.1 ORTS tool: [37]

ORTS tool is used to generate the optimized regression test suite for java applications. It helps in capturing the runtime traces of test execution, and, identifies the change points during build update. In this no artifacts are missed during execution. It captures all artifacts of java applications such as Java script, JSP, etc. It has three features: first is, it is scalable for runtime profiling of commercial java applications. Second is, it has the ability to identify complete change points, and last is , it prioritizes the test suites in terms of risk and guides the rerun schedule.

5.2 Automatic Debugging tool (AutoFlow):[38]

It integrates the delta algorithm for debugging as well as the change impact analyzer to reduce search for faulty changes. It first uses the change impact analyzer is responsible for identifying the subset of changes for faulty or failed test cases (i.e., it indicates the likelihood that is contributed to failure)

Author's Name	Name of the tool	Year	Features	Advantages
Sheng Huang[37]	ORTS	2009	A tool used for generating the optimized regression test suite for java applications. It helps in capturing the runtime traces of test execution and identifies the change points during build update.	• It improves efficiency, reduces cost and is lightweight, thus making regression test selection process more automated and effective. • Scalable with resource and time constraint
Sai Zhang[38]	Autoflow Debugging Tool	2008	It identifies the subset which is responsible of the changes for a failed test through impact analysis, and indicates the likelihood that is contributed to failure.	• It automatically reduces a large portion of irrelevant changes in early phase only.
Sai Zhang [39]	Celadon	2007	It analyses the source code of two aspect-J software versions and divide them into two parts of atomic changes along with their dependence relationship and outputs the impacted program and affected results. It provides the extended concept of atomic change.	• It helps in fault localization by isolating failure inducing changes for one specific affected test case.

Table 3 : Tools for Aspect –Oriented Programs

and then it uses delta debugging algorithm to determine the minimal set of faulty changes. When a regression test fails unexpectedly after a session of source changes, AutoFlow works as follows, first it decomposes the code modifications into a set of atomic changes (at method-level); then it employs change impact analysis to isolate a subset of responsible changes for that failed test; in the third step, AutoFlow ranks these changes according to our proposed heuristic, and finally employs an improved delta debugging algorithm to determine a minimal set of faulty changes.[38]

5.3 Celadon tool:[39]

It automates the change impact analysis for aspect-J. It is implemented in the context of eclipse environment and is designed as a plug-in. It analyses the source code of two aspect-j software versions and divide them into two set of independent changes along with their dependence relationship. It outputs the impacted program and the affected results.

Conclusion and Future Work:

In this paper survey has been done on various regression test selection techniques of object-oriented programs as well as on aspect–oriented programs. Advantages and Limitations of them is also discussed along with the techniques they have proposed. It has also discussed the tools used for aspect oriented programs. We will use any of these tools in our future research work for regression selection techniques. These tools will help to simplify the use of regression selection techniques in aspect-oriented software systems. This will also reduce the testing effort, cost and time.

6. References

[1] S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: a survey. Software Testing, Verification and Reliability, 1(1):121–141, March 2010.

[2] Thomas Ball, On the limit of control flow analysis for regression test selection, Proceedings of the 1998 ACM SIGSOFT ISSTA, pp.134-142, March 02-04, 1998, Clearwater Beach, Florida, United States.

[3] Yih-Farn Chen , David S. Rosenblum , Kiem-Phong Vo, Test Tube: a system for selective regression testing, Proceedings of the 16th ICSE, pp.211-220, May 16-21, 1994,Sorrento, Italy.

[4] H. K. N. Leung and L. J. White. A cost model to compare regression test strategies. In *Proceedings of ICSM '91*, pages 201- 208, Oct. 1991.

[5] **D**. Kung, J. Gao, P. Hsia, F. Wen, Y.Toyoshima, and C. Chen. On regression testing of object oriented programs. Journal of Systems and Software, 32(1):21–40, January 1996

[6] P. Hsia, X. Li, D. Kung, C. Hsu, L. Li, Y. Toyoshima, and C. Chen. A technique for the selective revalidation of object-oriented software. Journal of Software Maintenance: Research and Practice, 9(4):217–233, 1997.

[7] K. Abdullah and L.White. A firewall approach for the regression testing of object-oriented software. In Proceedings of 10th Annual Software Quality Week, page 27, May 1997.

[8] L. Bergmans and M. Aksits. Composing crosscutting concerns using composition filters. *Commun. ACM. 44(10):51-57*, 2001.

[9] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J. M. Loingtier, and J. Irwin. Aspect-Oriented Programming. In and J. Hugunin. Advice weaving in *Proceeding of 11th ECOOP, pp. 220-242*, 1997.

[10] K. Lieberherr, D. Orleans, and J. Ovlinger. Aspect-oriented programming with adaptive methods. *Commun. ACM, 44(10): 39-41*, 2001.

[11] The AspectJ Team. The AspectJ Programming Guide. August 2004.

[12] L. P. Wong, M. Y. H. Low, and C. S. Chong, "A bee colony optimization algorithm for travelling salesman problem," in Proceedings of Second Asia International Conference on Modelling & Simulation (AMS 2008), 2008. pp. 818-823.

[13] G. Rothermel and M. Harrold. Selecting tests and identifying test coverage requirements for modified software. In Proceedings of the International Symposium on Software Testing and Analysis, pages 169–184, August 1994.

[14] http://en.wikipedia.org/wiki/AspectJ

[15] G. Rothermel and M. J. Harrold. Empirical studies of a safe regression test selection technique. IEEE Transactions on Software Engineering, 24(6):401{419, Jun.1998.

[16] Y. Jang, M. Munro, and Y. Kwon. An improved method of selecting regression tests for C++ programs. Journal of Software Maintenance: Research and Practice, 13(5):331–350, September 2001.

[17] H. Leung and L. White. A firewall concept for both control-flow and data-flow in regression integration testing. In Proceedings of the Conference on Software Maintenance, pages 262–270, 1992.

[18] G. Rothermel and M. Harrold. Selecting regression tests for object-oriented software. In International Conference on Software Maintenance, pages 14–25, March 1994.

[19] G. Rothermel, M. Harrold, and J. Dedhia. Regression test selection for C++ software. Software Testing, Verification and Reliability, 10(2):77–109, June 2000.

[20] M. Harrold, J. Jones, T. Li, D. Liang, A. Orso, M. Pennings, S. Sinha, S. A. Spoon, and A. Gujarathi. Regression test selection for Java software. In Proceedings of the 16th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications, pages 312–326, January 2001.

[21] N. Mansour and W. Statieh. Regression test selection for C# programs. Advances in Software Engineering, 2009:1:1–1:16, January 2009.

[22] A. Orso, N. Shi, and M. Harrold. Scaling regression testing to large software systems. In Proceedings of the 12th ACM SIGSOFT Twelfth International Symposium on Foundations of Software Engineering, pages 241–251, November 2004.

[23] M. Fahad and A. Nadeem. A survey of UML based regression testing. In Zhongzhi Shi, E. Mercier-Laurent, and D. Leake, editors, Intelligent Information Processing IV, volume 288 of IFIP Advances in Information and Communication Technology, pages 200–210. Springer Boston, 2008.

[24] L. Briand, Y. Labiche, and G. Soccar. Automating impact analysis and regression test selection based on UML designs. In Proceedings of the International Conference on Software Maintenance (ICSM'02), pages 252–261, 2002.

[25] Q. Farooq, M. Iqbal, Z. Malik, and A. Nadeem. An approach for selective state machine based regression testing. In Proceedings of the 3rd international workshop on Advances in model-based testing, AMOST '07, pages 44–52. ACM, 2007.

[26] Q. Farooq, M. Iqbal, Z. Malik, and M. Riebisch. A model-based regression testing approach for evolving software systems with flexible tool support. In 17th IEEE International Conference on Engineering of Computer-Based Systems (ECBS), pages 41–49. IEEE Computer Society, March 2010.

[27] L. Naslavsky and D. Richardson. Using traceability to support model-based regression testing. In Proceedings of the twenty-second IEEE/ACM international conference on automated software engineering, ASE '07, pages 567–570. ACM, November 2007.

[28] R. Gorthi, A. Pasala, K. Chanduka, and B. Leong. Specification-based approach to select regression test suite to validate changed software. In Proceedings of the 2008 15th Asia-Pacific Software Engineering Conference, pages 153–160, 2008.

[29] Y. Chen, R. Probert, and D. Sims. Specification based regression test selection with risk analysis. In CASCON '02: Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research, page 1, 2002.

[30] M. Harrold, J. A. Jones, T. Li, D. Liang, A. Orso, M. Pennings, S. Sinha, S. A. Spoon, and A. Gujarathi. Regression Test Selection for Java Software. Proc. ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications, pp.312-326, October 2001.

[31] G. Rothermel and M. J. Harrold. A Safe, Efficient Regression Test Selection Technique. ACM Transactions on Software Engineering and Methodology, Vol. 6, No. 2, pp.173-210, April 1997.

[32] G. Xu. A regression tests selection technique for aspect oriented programs. In *Workshop on Testing Aspect-Oriented Programs*, pages 15–20, 2006.

[33] G. Xu and A. Rountev. Regression test selection for AspectJ software. In ICSE '07: Proceedings of the 29th international conference on Software Engineering, pages 65–74, 2007.

[34] Romain Delamare, Benoit Baudry, Yves Le Traon. Regression Test Selection when Evolving Software with Aspects

[35] M. Harman, F. Islam, T. Xie, and S. Wappler, "Automated test data generation for aspect-oriented programs," in AOSD, 2009, pp. 185–196.

[36] T. Xie and J. Zhao. A framework and tool supports for generating test inputs of AspectJ programs. In Proc. AOSD, pages 190–201, 2006.

[37] Sheng Huang, Jun Zhu, Yuan Ni . ORTS: a tool for optimized regression testing selection. In Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, Pages 803-804, OOPSLA '09.

[38] Sai Zhang, Zhongxian Gu, Yu Lin and Jianjun Zhao . AutoFlow: An Automatic Debugging Tool for AspectJ Software In Proc. 24th IEEE International Conference on Software Maintenance (ICSM 2008 demo).

[39] S. Zhang and J. Zhao. Change impact analysis for AspectJ programs. Technical Report SJTU-CSE-TR-07-01, Center for Software Engineering, SJTU, Jan 2007.