International Journal of Computer Science and Engineering ( IJCSE )
Vol.1, Issue 1 Aug 2012 55-64


International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

# A HIERARCHICAL TEST CASE PRIORITIZATION TECHNIQUES FOR ASPECT-ORIENTED SOFTWARE

**[1]SUSHIL KUMAR & [2]NARESH CHAUHAN**

[1]Assistant Professor (Deptt of Computer Engg), YMCA University of Science & Technology, Faridabad Haryana- 121006, India.

[2]Professor (Deptt of Computer Engg.), YMCA University of Science & Technology, Faridabad Haryana - 121006, India.

## ABSTRACT

Aspect Oriented Programming is a new paradigm for developing software. It is the way to modularize the cross-cutting concerns. As it is in its evolving phase it poses some challenges, one of which is testing AO programs. As there are some basics differences between AOP and OOP, there is need of new testing approach for AO programs. One of the approaches used to test AO programs is state-based incremental testing. Testing of aspect(s)-class block is done incrementally. As the aspects are incremented there is need of regression testing with the objective to ensure that the integration is done without affecting the original behaviour of the class.

One of the major problems encounter during state-based incremental testing is: as the number of aspects to be added increases, the number of test cases on which regression testing has to be performed also increases exponentially. This scenario leads to exhaustive testing which is both impractical and inefficient.

In the work presented a new framework for state-based incremental testing of Aspect Oriented Program has been proposed. As the focus of the work is improving the efficiency of regression testing performed therefore a new algorithm, Hierarchical Test Case Prioritization(HTCP) in State-based Incremental Testing for Aspect Oriented Programs has been proposed. HTCP takes hierarchical prioritization into consideration with the goal of maximizing the rate of fault detection at first level and at the second level goal is to increase the rate of detection of high-risk faults, locating those faults earlier in the testing process. Evaluation and Analysis of the framework has been performed using Average Percentage of Faults Detection (APFD) metric. The analysis is done by comparing the Prioritization Test Suite, which is the result of proposed HTCP algorithms and Non-prioritized test suite.

**KEYWORDS:** AOP, APFD, AO, HTCP, OO, OOP

## INTRODUCTION

Aspect Oriented Programming (AOP) is a new paradigm having foundations on OOP. AOP is based on the idea that concerns crosscutting several modules of an application can be developed as single unit of modularity and weaved into application, through a process of composition, using join points (a

construct of AOP). AOP offers a way of dealing with system behavior which does not fit cleanly into the programming models currently being used in the industry. The system behavior that cannot be encapsulated in classes because of its impact across the whole system is called crosscutting behavior AOP encapsulates this kind of behavior in aspects.

AOP offers a way of dealing with system behavior which does not fit cleanly into the programming models currently being used in the industry. The system behavior that cannot be encapsulated in classes because of its impact across the whole system is called crosscutting behavior.

Aspect Oriented Programming (AOP) is a new paradigm having foundations on OOP. AOP is based on the idea that concerns crosscutting several modules of an application can be developed as single unit of modularity and weaved into application, through a process of composition, using join points

There are two scenarios in which aspects can be defined.

1. In one scenario the aspects are the result of refactoring and aggregating the common code from primary abstraction in one place i.e. aspect
2. In second scenario which is inverse of first, and is the area of concern , aspect is defined independently with respect to crosscutting concern that is not present in primary abstraction(e.g. synchronization or security policy)

Testing is an essential part of software development process that ensures software correctness

There exist several testing techniques of different types such as unit testing, integration testing, system testing and others. These and other techniques have been developed, researched and applied on different programming paradigms

AOP is relatively new programming paradigm and aspect oriented (AO) programs provides different characteristics which differ from OO programs. There are new challenges regarding testing due to some characteristics of aspects, like dependency on the context of classes, tight coupling to class etc.

AOP cannot be addressed using traditional unit or integration test techniques, these techniques are applicable to class that implement core concerns but not applicable to aspects because aspect depend on woven context.

## AREA OF CONCERN

In the second scenario defined above aspect is defined independently with respect to crosscutting concern that is not present in primary abstraction e.g. synchronization. Class and methods of primary concerns are developed and tested as before using object oriented programming paradigm however, code regarding cross-cutting concerns is not embedded into bodies of methods instead it is contained in separately defined aspects.

Later aspects are woven with classes and it results in woven program composite of behavior of

both core concerns and cross-cutting concerns. here aspects are introduced as a result of change in requirement or addition of a new feature to already developed software. State-based incremental testing technique has been chosen for testing the aspect(s) - class blocks. In this approach to testing aspect-oriented programs, first UML statechart diagram of the class under test is developed and then converted into transition tree and testing of the class is done separately based on test sequences generated from transition tree. Then the statechart diagram is extended to incorporate the aspect that has been incremented to the class and extended transition tree is generated for both classes of the core concern and aspects of the crosscutting concern, which implies a test suite for adequately testing object behavior and interaction between classes and aspects in terms of message sequences.

## PROBLEM IDENTIFICATION

As the aspects are woven with class incrementally there is the need of regression testing. The Regression testing is done with the objective to ensure that the interaction of aspect and class have not affected the original behavior of class. Regression testing is also done in case of a change instantiated on a class or on one of its related aspects.

As the integration of aspects proceeds, a major problem that is encountered is, ***exponential growth of test cases*** this problem becomes acute in case of regression testing. This problem leads to exhaustive testing. Therefore there is a need to provide the solution for the problem so that efficiency of testing and overall quality of the software could be improved.

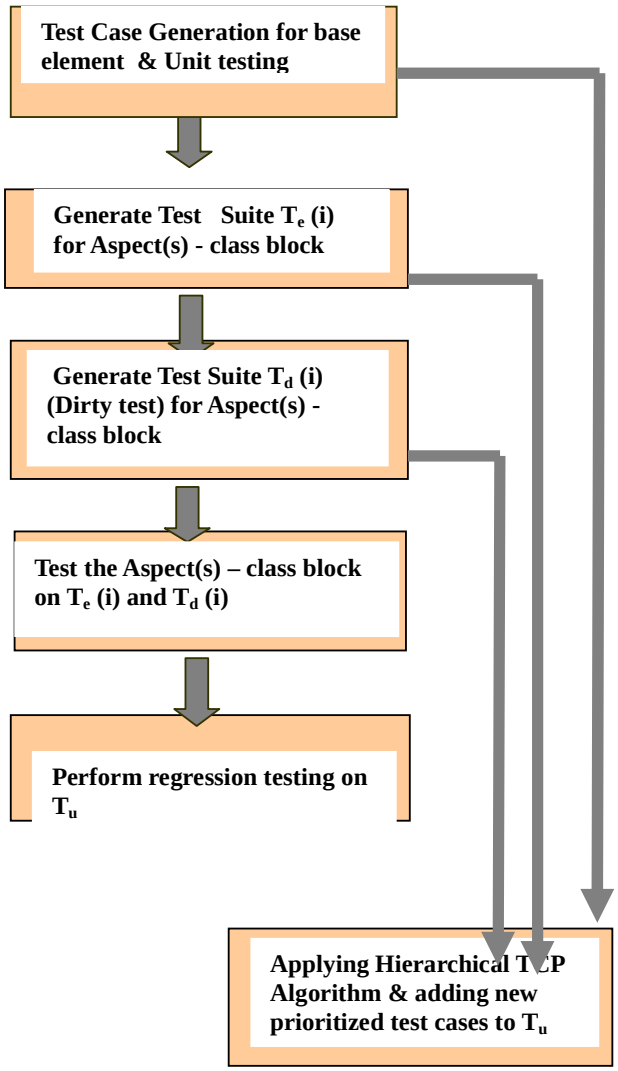
## ISSUES

Some of the major issues that need to be taken care of, while providing solution to the problem, are:

1. Aspects do not have independent identity or existence. They depend upon the context of some other class for their identity and execution context.
2. Aspect implementation can be tightly coupled to their woven context. Aspects depend on the internal representation and implementation of classes into which they are woven. Changes to these classes will likely propagate to the aspects[2].
3. When a failure occurs, the first challenge is to diagnose the failure and detect the fault. For non-aspect oriented programs, one examines the code and possibly instruments it with probes to isolate and localize a fault. Dealing with failures in aspect-oriented programs requires a similar approach. However, to detect a fault in an AOP, the code of the woven aspects must also be examined.[2]

## PROPOSED FRAMEWORK FOR STATE BASED INCREMENTAL TESTING IN AOP

In this paper a new framework has been proposed for the state-based incremental testing in AOP. Framework consists of algorithm for test case prioritization. Hierarchical prioritization has been taken into consideration with the goal of maximizing the rate of fault detection at first level and at second level increase the rate of detection of high-risk faults, locating those faults earlier in the testing process.




**Hierarchical Test Case Prioritization (HTCP) Algorithm**

The framework presented in this work implements a new

**Input:** Test suite $T_b$, $T_e(i)$, $T_d(i)$, number of faults due to part of code affected by *join points* fj, detected by each test case and total number of faults detected ft.
**Output:** Prioritized Test suite $T_p$.
**begin**
2. set $T_p$ empty
sort $T_d(i)$ in descending order based on the value fj of each test case
**if** more than one test case in $T_d(i)$ have same values of fj
**then** decide the priority on the values of ft
sort $T_e(i)$ in descending order based on the value fj of each test case
**if** more than one test case in $T_e(i)$ have same values of fj
**then** decide the priority on the values of ft
$T_p = \{ T_d(i) , T_e(i) \}$
Adding $T_p$ to $T_u$
Applying first level prioritization to $T_u$
**ends**

## EVALUATION AND ANALYSIS

As the aspects are woven with class incrementally there is the need of regression testing. As the integration of aspects proceeds, a major problem that is encountered is, exponential growth of test cases this problem becomes acute in case of regression testing. This problem leads to exhaustive testing. Therefore there is a need to provide the solution for the problem so that efficiency of testing and overall quality of the software could be improved.

A new framework has been proposed for the state-based incremental testing. Framework consists of prioritization algorithm. Hierarchical prioritization has been taken into consideration with the goal of maximizing the no of fault detection at first level and at second level increase the rate of detection of high-risk faults, locating those faults earlier in the testing process.

Here evaluation of the proposed framework has been by applying it on example of *Stack* class and

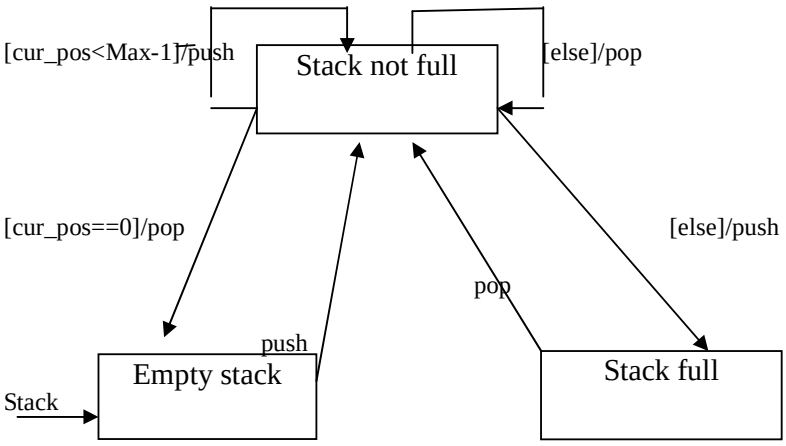
*StackAspect* aspect. Analysis of framework has been done by using the APFD metrics.

## EVALUATION

For evaluating the proposed frame work, considered a class *Stack* that implements stack of integer type. The class contains the following methods and variables:

1. Stack [ Constructor to create an Stack class instance which is initially empty]
2. Push [ this method is used to insert an item on the top of stack]
3. Pop. [ this method returns the item from the top of stack]
4. cur_pos is the variable which is used to index the top of stack
5. Max is variable whose value shows the size of stack
6. item is variable whose value is input to stack
7. Stack can be one of the following states
8. Empty Stack if cur_pos = = -1 [ it shows initial position ]
9. Stack Not Full if 0 <= cur_pos < Max-1
10. Stack Full if cur_pos = = Max - 1

State Chart Diagram of the Stack Class




**Statechart Of Simple Stack Class**

## ANALYSIS

The performance of the framework proposed, depends on the effectiveness of the prioritization algorithm, HTCP, proposed in the framework so it is necessary to analyze the effectiveness. The APFD metric is used to analyze the effectiveness.

**Graphical analysis**

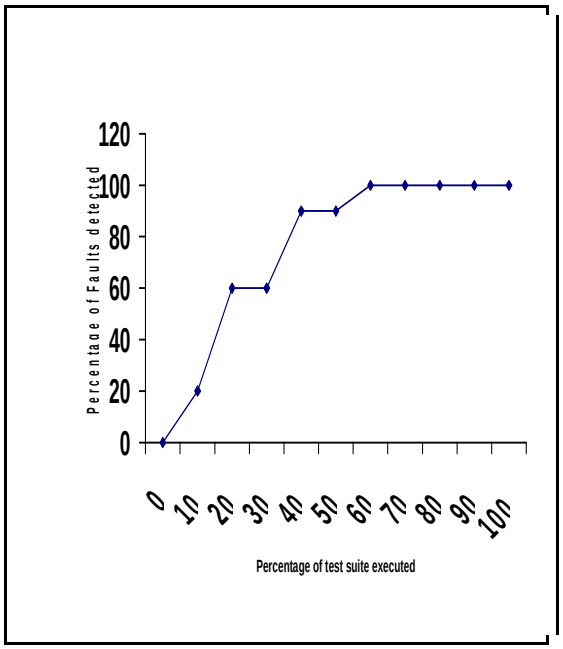



**fig 1 APFD Graph for Non Prioritized Test Suite**

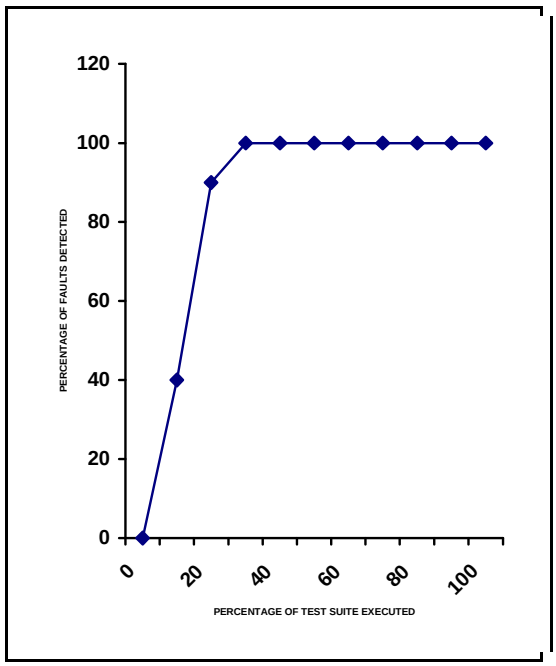



**fig 2 APFD Graph for Prioritized Test Suite**

The comparison drawn between prioritized and non-prioritized case, shows that the value obtained for the prioritized test suite ($T_p$) which is resultant of the new approach is more than non prioritization approach, hence proposed algorithms effective.

To analyze the results more clearly graphs has been plotted for percentage of faults detected vs. percentage of test suite executed. Graphs have been plotted for both Prioritization approach and for non prioritization approach. So that results could be compared.

As it can be seen from the graphs(refer Fig. 1) itself that for Prioritized Test Suite 100% of faults were detected when only 30 % of test suite got executed whereas in case of Non Prioritized test suite(refer Fig 2) 100% of faults were detected when 60% of Non- Prioritized Test Suite got executed. Therefore it can be concluded that proposed algorithm Hierarchical Test Case Prioritization for State based testing in Aspect-Oriented Software, is effective as compared to non prioritized test suite.

## CONCLUSIONS & FUTURE WORK

In the work presented, a new framework has been proposed for incremental state based testing of Aspect Oriented Programs (AOP). The framework focuses on the integration of one or several aspects to a class. The objective is to ensure that the integration is done without affecting the original behavior of the class. In present work Java and AspectJ has been used as implementation language. The approach is based on state-based incremental testing of aspect(s)-class block. This leads to regression testing approach. After the each aspect added to the class test cases are generated to test the behavior of the class and added aspect it should behave as intended. The test cases are generated on the basis of Transition Tree.Regression testing is also performed on previously generated test cases to ensure that the integration is done without affecting the original behavior of the class. But as the number of aspects to be added increases the number of test cases on which testing has to be performed also increases exponentially. This problem becomes acute in case of regression testing.

To overcome the problem of exponential growth of test cases, an algorithm for test case prioritization, Hierarchical Test Case Prioritization (HTCP) has been proposed. This algorithm is designed keeping in mind the environment and issues related to aspect oriented programs. Hierarchical prioritization is done at three levels. Objectives of prioritization are to maximize the number of fault detected and to increase the rate of detection of high-risk faults, locating those faults earlier in the testing process.

The proposed framework has been evaluated using the example of a class and weaving it with one aspect. To analyze the effectiveness of the proposed prioritization algorithm, Hierarchical Test Case Prioritization, in an observable manner a metric called Average Percentage of Faults Detected(APFD) developed by Elbaum [3,4,5], is used. The APFD measures the average rate of fault detection per percentage of test suite execution. It is calculated by taking the weighted average of the number of faults detected during the run of the test suite.

The proposed algorithm results in highly prioritized test suite, which has been found effective as compared to non prioritized test suite. Further the results have been analyzed by plotting the graphs for percentage of faults detected vs. percentage of test suite executed. The comparison has been done between the proposed HTCP and Non prioritized test suite.

The work can be expanded as a future work in the following ways:

- Proposed framework has not taken into consideration AOP constructs such as introduction, aspect inheritance, and aspect composition. The proposed framework could be modified to incorporate these constructs of the Aspect-Oriented Programming.
- Since the proposed framework has been evaluated using a Toy example, the proposed framework can be evaluated using a real time software in the industry. The empirical results collected thereof may expose new problems and challenges in aspect oriented software.

## REFERENCES

1. Dianxiang Xu, Weifeng Xu and Kendall Nygard: "A State-Based Approach to Testing Aspect-Oriented Programs", In Proc. of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'05), July 14-16, Taiwan.
2. Roger T. Alexander, James M. Bieman and Anneliese A. Andrews: "Towards the Systematic Testing of Aspect-Oriented Programs", Department of Computer Science, Colorado State University, Fort Collins, Colorado, USA. Technical Report CS-4-105, March 2004.
3. Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel, Sebastian Elbaum, "Cost-cognizant Test Case Prioritization", 2006.
4. S. Elbaum, A. Malishevsky, and G. Rothermel "Test Case Prioritization: A family of empirical studies". IEEE Transactions on Software Engineering, Feb 2002.
5. S. Elbaum, Gregg Rothermel, Satya Kanduri, Alexey G. Malishevsky: Selecting a Cost-Effective Test Case Prioritization Technique", April 2004.
6. Roger S. Pressman, Software Engineering a practitioner's approach 6/e, 2005
7. H. Ossher and P. Tarr," Multi-dimensional separation of concerns and the In Software architectures and Component Technology: The Hyperspace approach", 2000. State of the Art in Research and Practice.
8. A. Restivo and A. Aguiar, "Towards Detecting and Solving Aspect Conflicts and SPLAT'07, Vancouver, British Columbia, Canada, Interferences Using Unit Tests", ACM, 2007.
9. R. T. Alexander and James M. Bieman, "Challenges of Aspect-oriented Workshop on Software Quality (WoSQ'2002), Orlando, Florida, ACM, Technology", 2002.
10. E. Baniassad and S. Clarke, "Theme: An Approach for Aspect-Oriented Analysis In Proceedings of International Conference on Software Engineering and Design", 2004, IEEE Computer Society. (ICSE '04)
11. G. Kiczales, et al. "An Overview of AspectJ", In 15th European conference on Object-Oriented Programming, Budapest, Hungary, 2001.

12. Aida Atef Zakaria, Dr. Hoda Hosny and Dr. Amir Zeid: “A UML Extension for Modeling Aspect-Oriented Svstems”, Second International Workshop on Aspect-Oriented Modeling with UML .Dresden, Germany, 2002.

13. M. L. Bernardi and G. A. D. Lucca, “Testing Aspect Oriented Programs: an “Approach Based on the Coverage of the Interactions among Advices and Methods”, In Proceedings of Sixth International Conference on the Quality of Information and Communications Technology (QUATIC’07), Lisbon, IEEE, 2007.

14. S. A. A. Naqvi, S. Ali and M. U. Khan “An Evaluation of Aspect Oriented Testing International Conference on Emerging Technologies, Techniques”, In Proceedings of Mohammad Ali Jinnah University, Islamabad, Pakistan, IEEE, September, 2005.