Volume 4, Issue 5, May 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Functional and Non-Functional Requirement and Interaction Analysis using Aspect-Oriented Technology

Mr. Sagar Mohite.
Department of Computer Engineering
B.V.D.U College of Engineering.

Prof. Dr. S. D. Joshi.
Department of Computer Engineering
B.V.D.U College of Engineering,

Abstract— Aspect-oriented modeling (AOM) have been developed to modularize crosscutting concerns properly in UML models. In software engineering, aspects are concerns that cut across multiple modules. At requirements modeling, we analyze interactions and potential inconsistencies. We use UML to model requirements in a use case driven approach. During requirements specification a structural model of the problem domain is captured with a class diagram. Use cases refined by activities are the join points to compose crosscutting concerns. Graph transformation systems provide analysis support for detecting potential conflicts and dependencies between rule-based transformations.

Keywords— Aspect-Oriented Modeling, Rule-based graph transformations, Aspect, pointcuts, crosscutting concerns.

I. INTRODUCTION

Aspect-Oriented Modeling is applied beginning of the software development lifecycle and it now stretch to software modeling and design. Aspect-oriented represents aspects during requirements engineering. One of the advantages of aspect-oriented approaches is that they allow software developers to react easily to unanticipated changes in existing software systems, while promoting reusability of already tested and designed software components. Nevertheless, people are reluctant to apply AOP in serious and large projects, not because of a lack of good aspect-oriented programming languages and tools, but because they do not have aspect-oriented modeling and design techniques at their disposal. Aspect-orientation should be taken into account in all the stages of the software lifecycle, in particular, the design level.

One of the advantages of aspect-oriented approaches is that they allow software developers to react easily to unanticipated changes in existing software systems, while promoting reusability of already tested and designed software components. Nevertheless, people are reluctant to apply AOP in serious and large projects, not because of a lack of good aspect-oriented programming languages and tools, but because they do not have aspect-oriented modeling and design techniques at their disposal. Aspect-orientation should be taken into account in all the stages of the software lifecycle, in particular, the design level. Aspect-orientation clearly separate crosscutting concerns from non-crosscutting ones which provide modularization. Separation of concerns will reduce complexity of software design [5]. Aspect-orientation originally has applied at programming level; it now applied over other development phases. Aspect is quite important in software development. Aspects are identifying by analyzing a complex system from multiple viewpoints [12] .Aspect-oriented modeling covers many activities at early stages of the software development. Two kinds of model transformation in AOM, refinement as well as weaving, should be considered.

The model transformation, either refinement or weaving, is done manually, and some verification is needed between the model descriptions before and after the transformation.

II. EXISTING SYSTEM

When designing software, it is desirable to explore several different designs, i.e. consider several feasible solutions to implement a specific requirement or functionality and compare the advantages and disadvantages of each solution. Unfortunately, software modeling tools are often tedious to use when making significant changes within the design of a large software model. Aspect-oriented modeling (AOM) is a new modeling technique that allows developers to describe the design of their software using many aspect models. With AOM, each individual aspect model is small in size. To build larger systems, structure and behavior defined in one aspect model can be reused within other aspect models. This reuse is achieved by establishing a mapping between the model elements in the two aspect models. OOP already allows for modularizing concerns into distinct methods, classes and packages. However, some concerns are difficult to place as they cross the boundaries of classes and even packages. Security and logging, response time are examples for cross-cutting concern.

Disadvantage of existing system:-

UML, in its current state, allows us to capture the structure and interactions of our aspect-oriented program. However, the resulting model presents some major drawbacks:

- There is no difference between modularization by class and by aspect. The basic concepts of AspectJ, such as pointcuts, introduction and advice, are not explicitly modeled.

• The model does not show that the aspect is a "pluggable" entity. The diagrams give the impression that aspects are static entities, although, in reality, aspects are configured at weave-time, and triggering them can be based on various kinds of execution flows or conditions.

• The model does not provide way to find and remove conflict and dependency in system.

• The requirement, it can be functional or non functional are not properly analyze and manage.

III. IMPLIMENTATION

Implemented Functions

Main Functions implemented for project are listed as below in the project modules.

The total workflow is divided into following modules:

Module 1: Implementing rules from Type graph using AGG tool.

Attributed Typed Graph

In this module, we have taken input as a type graph. As in object-oriented modelling, types can be structured by an inheritance relation. Instances of a type graph are object graphs equipped with a structure-preserving mapping to the type graph. A class diagram can thus be represented by a type graph plus a set of constraints over this type graph expressing multiplicities and maybe further constraints.

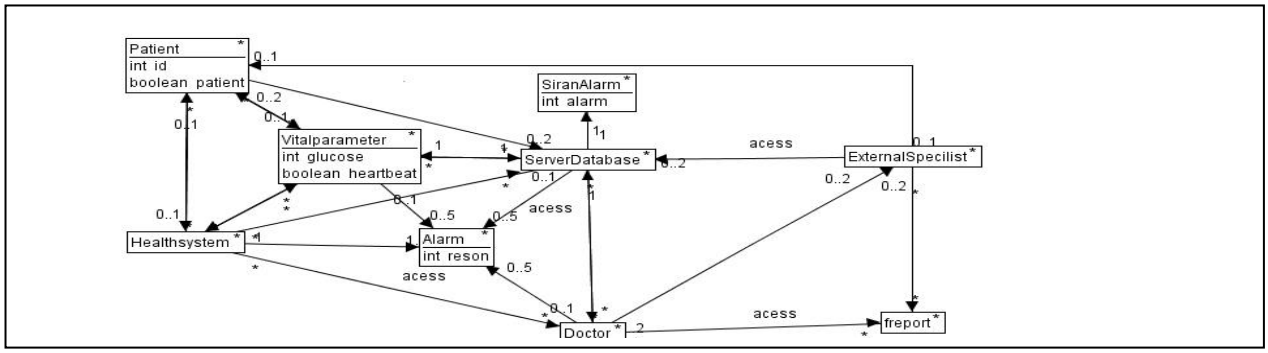


Figure.1 Type graph for Health Care air ambulance.

Then we created the functional and non-functional rules for system. After creating rules we have form the graph for transformation

Graph Rules for UML Attributed Typed Graph Transformations

In AGG attribute declaration is same as other programming language. An action can be viewed as a state transition, and obviously, a transition of states can be specified by giving descriptions of the states before and after the action in question. Since states are modelled as graphs in AGG, it follows that basically an action can be described as a pair of two graphs modelling the ``before" and ``after" states. In the ``before" state of an operation, we collect all the preconditions that have to be met for the operation to take place. The left-hand side of a graph rule states the necessary conditions for the specified operation to take place: A rule can only be applied if its conditions are fulfilled by the current concrete state graph. The effect of a rule application at a given match is a state graph transformation, also called derivation or graph transformation step.

1. Rule for Patient Login

If patient is authorized then and then only enter into the health system. This non-functional rule is for improving security.

Figure.2 Rules for Patient Login

2. Rule for Store Report

The above generated report is added to the server database.

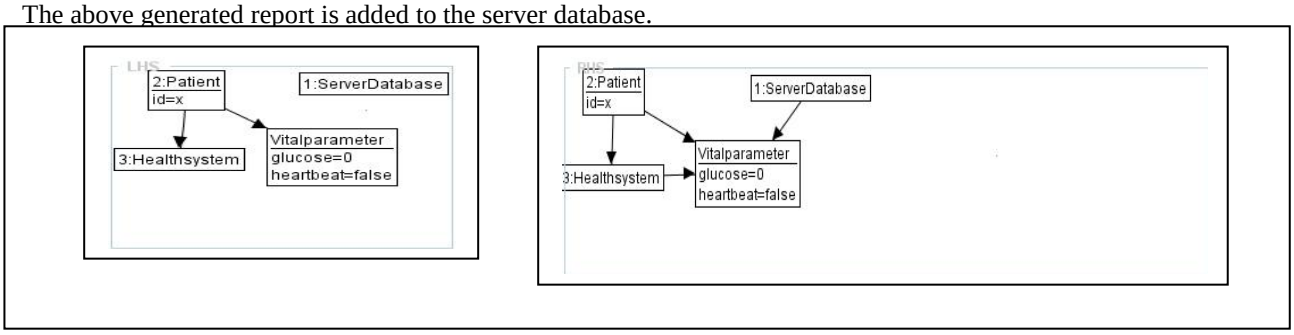
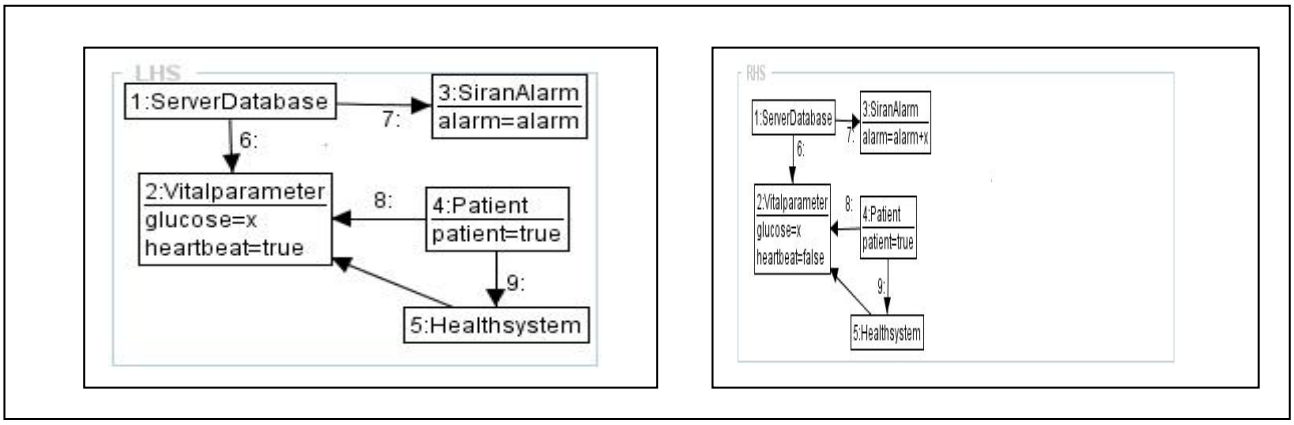


Figure.3 Rules for Store Report.

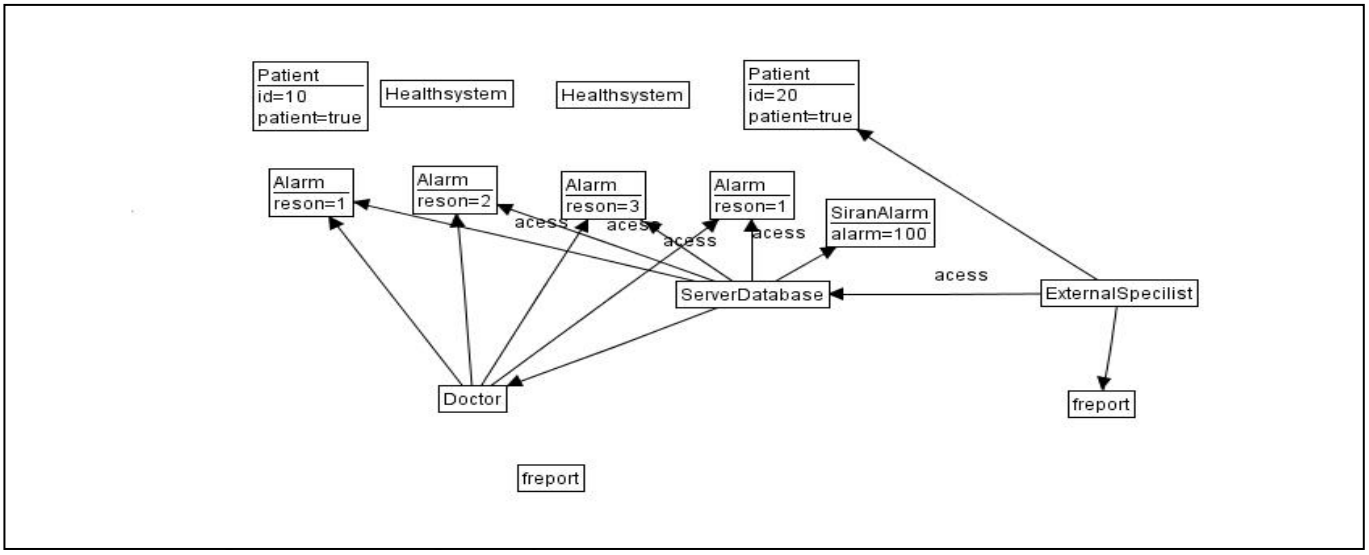
3 Rule for siren alarm condition

Depending on vital parameter, Server system will generate the alarm with specifying reason.



Graph of health management system:

Graph of health management system look like this before transformation. This graph is generated by using above rules.



Graph of health management system look like this after transformation. The transformation will depend upon RHS of the above rule.

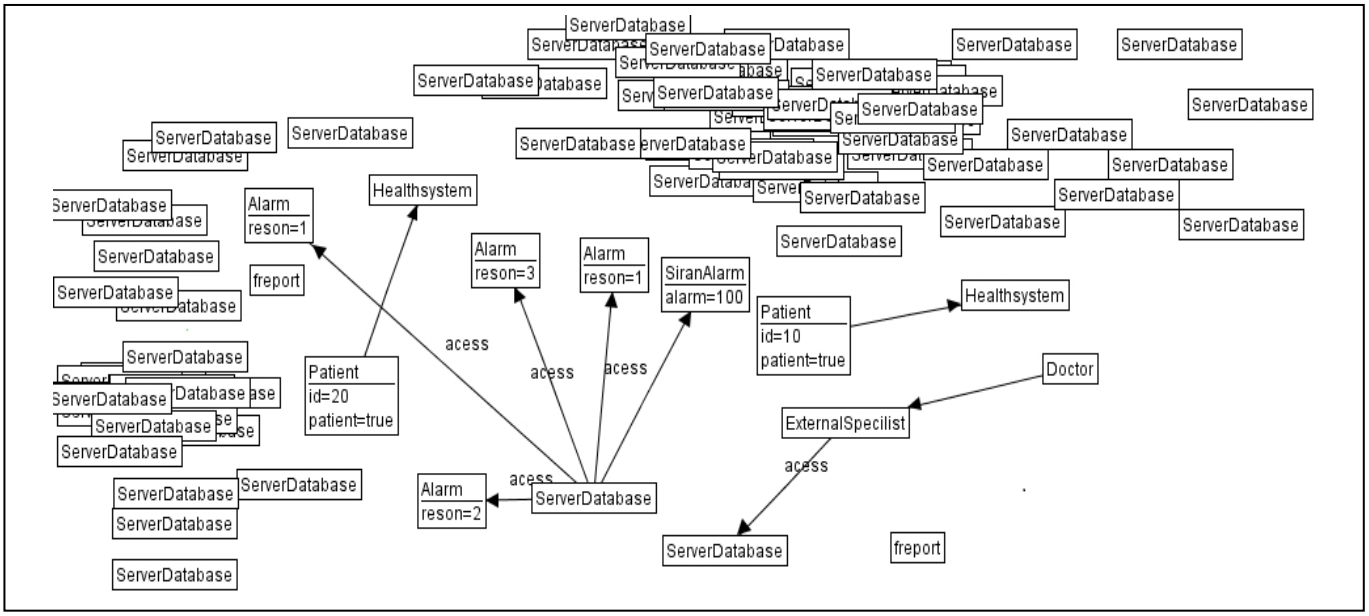
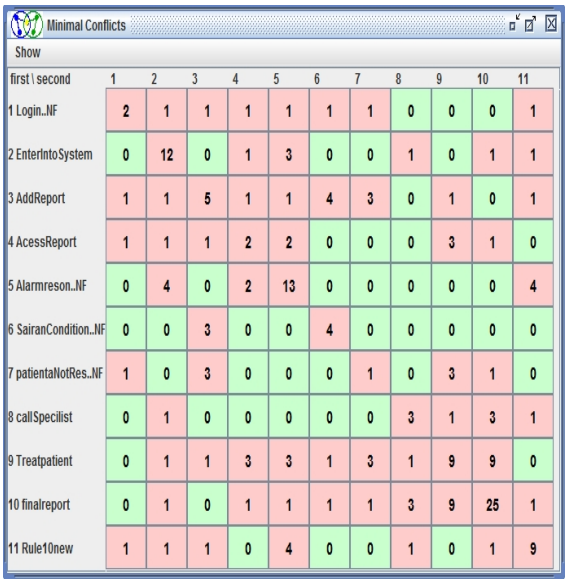


Figure.6 Transformed Graph for Health Care air ambulance.

Module 2: Implementing and Analysis of Conflict and Dependency

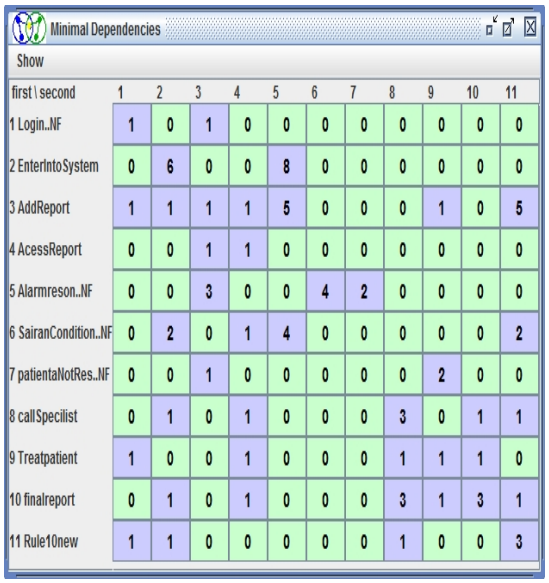
We have introduced graph transformation as the theoretical foundation for detecting conflicts and dependencies between activities specified with pre- and post-conditions.

We computed all potential conflicts and dependencies for travel agency example. The results are presented with a conflict and a dependency matrix.



first \ second	1	2	3	4	5	6	7	8	9	10	11
1 Login..NF	2	1	1	1	1	1	1	0	0	0	1
2 EnterIntoSystem	0	12	0	1	3	0	0	1	0	1	1
3 AddReport	1	1	5	1	1	4	3	0	1	0	1
4 AcessReport	1	1	1	2	2	0	0	0	3	1	0
5 Alarmreson..NF	0	4	0	2	13	0	0	0	0	0	4
6 SairanCondition..NF	0	0	3	0	0	4	0	0	0	0	0
7 patientaNotRes..NF	1	0	3	0	0	0	1	0	3	1	0
8 callSpecilist	0	1	0	0	0	0	0	3	1	3	1
9 Treatpatient	0	1	1	3	3	1	3	1	9	9	0
10 finalreport	0	1	0	1	1	1	1	3	9	25	1
11 Rule10new	1	1	1	0	4	0	0	1	0	1	9

Figure.7 Conflicts matrix



first \ second	1	2	3	4	5	6	7	8	9	10	11
1 Login..NF	1	0	1	0	0	0	0	0	0	0	0
2 EnterIntoSystem	0	6	0	0	8	0	0	0	0	0	0
3 AddReport	1	1	1	1	5	0	0	0	1	0	5
4 AcessReport	0	0	1	1	0	0	0	0	0	0	0
5 Alarmreson..NF	0	0	3	0	0	4	2	0	0	0	0
6 SairanCondition..NF	0	2	0	1	4	0	0	0	0	0	2
7 patientaNotRes..NF	0	0	1	0	0	0	0	0	2	0	0
8 callSpecilist	0	1	0	1	0	0	0	3	0	1	1
9 Treatpatient	1	0	0	1	0	0	0	1	1	1	0
10 finalreport	0	1	0	1	0	0	0	3	1	3	1
11 Rule10new	1	1	0	0	0	0	0	1	0	0	3

Figure.8 Dependency matrix.

The first column and first row contain the list of all activities. The number specifies how many different conflicts/dependencies were found.

• Conflict matrix: a positive entry indicates that column entry A *disables* row entry B; B is in conflict with A.

• Dependency matrix: a positive entry means that column entry A enables row entry B; B is dependent on A.

We will discuss the one dependency example from above rule.

Minimal dependency matrix shows the 3 dependency count of [10x8] field.

Three reasons of that discuss below with diagram.

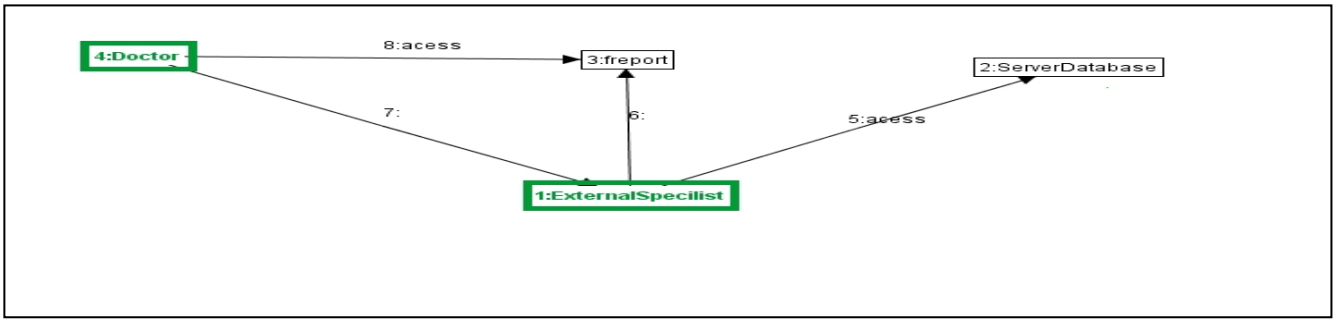


Diagram shows First reason, generation of report depend on doctor and external specialist.

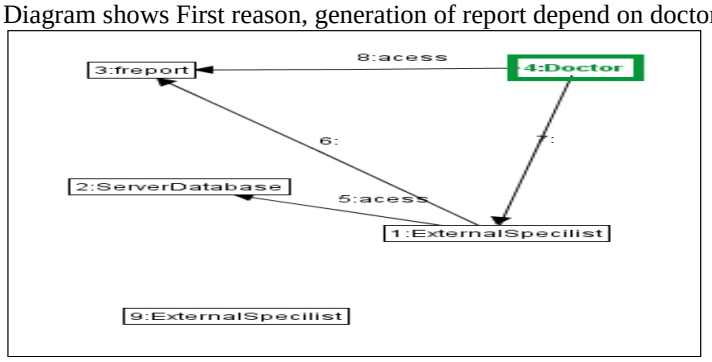


Figure.10 Dependency Rule2

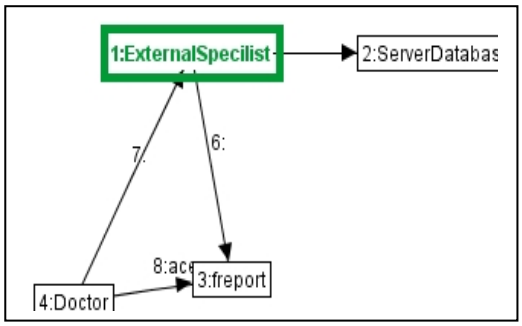


Figure.11 Dependency Rule3

Diagram shows second reason, generation of report depend on doctor.

Diagram shows third reason, generation of report depend on external specialist.

Module 3: Implementing web application for Health Care system .

We have created one web application for online Health Care system. We have Used java server page JSP and servlet for developing web application. Input to the web application is rules which we have developed by using the AGG tool. This

rules we have implemented in our web application by using JSP and servlet & aspect oriented programming applied for each rule for crosschecking by using aspect-tree & cross-reference section.

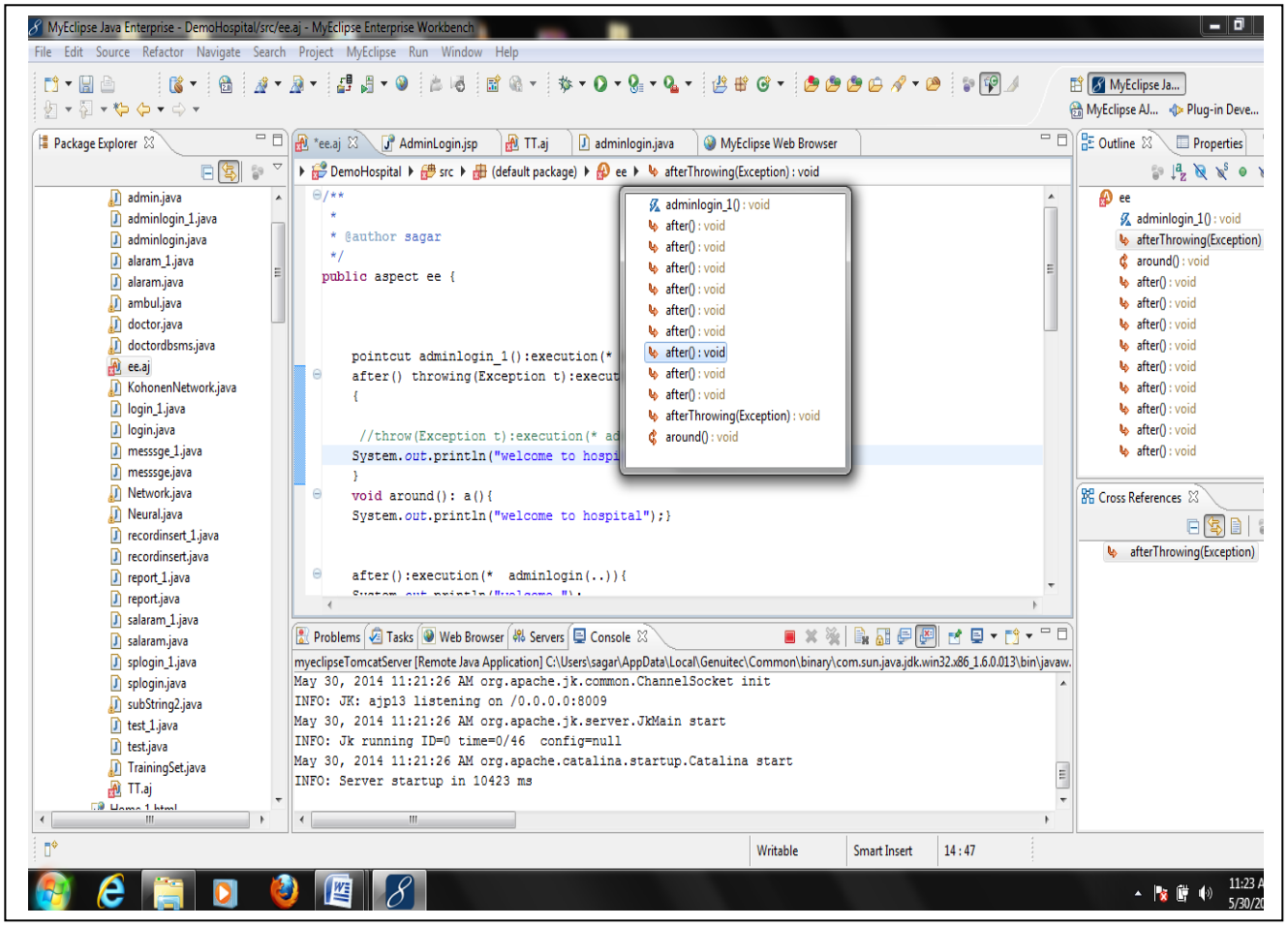


Figure 12 Aspect tree

IV. CONCLUSION

In aspect oriented modeling, the program with the main business logic and the cross cutting concerns are represented by models. There are many cross cutting concerns that are not part of the problem, such as security, logging, persistency, etc. Differently from traditional aspect oriented programming, in aspect oriented modeling there is no preferential entity. This means the weaving can be done in any model (e.g., weavings in the business models or in the cross cutting concerns).

We have proposed to integrate rule specifications with object oriented models in an aspect-oriented way. Graph transformation systems provide means to specify rule-based aspects directly as rules. We have tried for detecting conflicts and dependencies. The approach uses formal aid to analyze systematically semi-formal specifications. We use graph transformation to detect conflicts. we have create rule for functional & nonfunctional requirement in AGG & it's aspect apply to web design, web pages or web services to achieve security, response-time ,performance of a system.

The advantage of the aspect-oriented integration of graph transformation into the object-oriented control flow is that

In the future, we want to investigate the relationship between graph transformation and aspect-oriented languages further. We feel that pre- and post-conditions are an essential counterpart for an informal language like the UML, making modeling more rigorous. Pre- and post-conditions are analyze for activity diagrams. The approach is not restricted to functional aspects, as presented here. We consider both functional and nonfunctional aspect. Thereby, also interactions between functional and non-functional aspects are automatically covered.

REFERENCES

[1] T. Elrad, M. Aksits, G. Kiczales, K. Lieberherr, and H. Ossher: "Discussing Aspects of AOP". Communications of the ACM 44(10), pp. 33–38, October

[2] P. Tarr, H. Ossher, W. Harrison, and S. M. Sutton, Jr.: "N Degrees of Separation: Multi-Dimensional Separation of Concerns". In Proceedings of the 1999 International Conference on Software Engineering.

[3] H. Poor, an Introduction to Signal Detection and Estimation. New J. Rumbaugh, I. Jacobson, and G. Booch.

[4] Robert France, Indrakshi Ray, Geri Georg, and Sudipto Ghosh. Aspect-oriented Approach to Early Design Modelling. IEEE Proceedings Software, 151(4):173– 185, August 2004.

[5] D. Parnas. On the Criteria To Be Used in Decomposing Systems into Modules. Comm.

[6] E. Baniassad and S. Clarke. Theme: An Approach for Aspect-Oriented Analysis and Design.
[7] Jon Whittle & Praveen K. Jayaraman (2007): MATA: A Tool for Aspect-Oriented Modeling Based on Graph Transformation. In: Holger Giese, editor: MoDELS Workshops, Lecture Notes in Computer Science 5002. Springer, pp. 16–27. Available at http://dblp.uni trier.de/db/conf/models/models2007w.html WhittleJ07.
[8] K. Mehner and G. Taentzer. Supporting Aspect- Oriented Modeling with Graph Transformations. In P. Clements et al., editor, AOSD 05Workshop on Early Aspects, 2005.
[9] S. Herrmann, C. Hundt, and K. Mehner. Mapping Use Case Level Aspects to Object Teams/Java. In A. Moreira et. al, editor, OOPSLA Workshop on Early Aspects, 2004.
[10] G. Kiczales, E. Hisdale, J. Hugunin, M. Kersten, and J. Palm. An overview of AspectJ.
[11] I. Jacobson and P.-W. Ng. Aspect-Oriented Software Development with Use Cases. Addison Wesley, 2005.
[12] B.Nuseibeh, J. Kramer, and A. Finkelstein. AFramework for Expressing the Relationships betweenMultipleviews in Requirements Specifications.
[13] Object Teams home page. http://www.ObjectTeams.org.
[14] J. Ara´ujo and P. Coutinho. Identifying aspectual use cases using a viewpoint-oriented requirements method. In Early Aspects 2003: Aspect-Oriented Requirements Engineering and Architecture Design, Boston, MA, USA, March 2003
[15]. A. Rashid, P. Sawyer, A. Moreira, and J. Araujo. Early aspects: A model for aspect-oriented requirements engineering. In Proc.

