Requirement and Interaction Analysis Using Aspect-Oriented Modeling

Sagar Mohite, Rashmi Phalnikar, and S.D. Joshi

Abstract

Aspect-oriented modeling (AOM) has been developed to modularize crosscutting concerns appropriately in UML models. In software engineering, aspects are concerns that cut across multiple modules. In 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 • Point-cuts • Crosscutting concerns

Introduction

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 take 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 [1]. 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 [2]. Aspect-oriented modeling covers many activities at early stages of the software development.

Related work

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. In AOM, each individual aspect model is small in size. To build larger systems, structure and behavior defined in one aspect

S. Mohite (✉) • R. Phalnikar • S.D. Joshi
Bharati Vidyapeeth Deemed University College of Engineering, Pune, India
e-mail: Sagarmohite44@gmail.com; rashmiphalnikar@yahoo.co.in; sdj@live.in

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 on Object Oriented Programming:-

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 point-cuts, 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.

Approach

Requirement Specification

In an object-oriented requirements specification the main usage scenarios are captured with use cases. Each such use case has to be described in more detail, either textually or by means of activity diagrams. Also, during requirements specification a structural model of the problem domain is captured with a class diagram. A use case diagram provides a system overview. Each use case is described by trigger, its actors, pre- and post-conditions and its key scenarios. Scenarios are specified using activity diagrams and use cases are the starting point for the aspect-oriented modeling. We model the so-called base of the system with use cases and an integrated behavior model. An aspect is modeled as a use case. The join point for an aspect is an activity of the base. The point cut of an aspect is specified in terms of the activities of the base. While up to now proposed for modeling techniques like UML, an integrated behavior model is also suitable and beneficial for aspect- oriented modeling:

It can naturally capture the functional and structural description of each aspect. An aspect may share the base domain model or add its own concepts. Each aspect can be analyzed for consistency, and the consistency of the entire system consisting of the base and aspects can be analyzed as well. Analysis is even more crucial for aspect-oriented models:

As an aspect is specified once but can be used in many different places of the system.

As an example, consider a simple Heath care air ambulance system. Patient can cancel booking of air ambulance. Patient can registered in the system and can be unregistered as well. Both are conducted by admin on behalf of patient. Also, air ambulance and flight segments are administered by staff members. When patient books air ambulance and if he wants to cancel the booking, then the booking are cancel. But after some time, he wanted to again book for the same air ambulance, then it is not possible because entire data has been deleted during cancellation process. Here the conflict will be created.

The domain classes are given in Fig. 1. A patient can book a book air ambulance.

When doctor and specialist generate the report of patient using machine or other test, in earlier Stage, they find the causes of some particular dieses and generate report for that, but later stage by using test report and medicine report, they find other dieses than earlier one. This could be one type of conflict

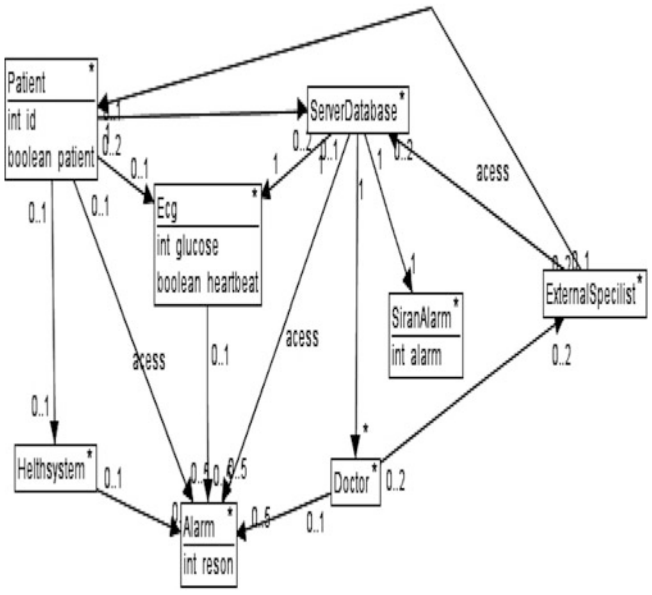

Fig. 1 Type graph for health care air ambulance system

Interaction Analysis Using Aspect Oriented Model

An overview of typical use cases is presented in Fig. 2.

It comprises use cases for administration of passenger data of patient, air ambulance and segment data. For the main usage scenarios it has use cases for booking and canceling air ambulance. In the following, we will focus on the use cases book air ambulance.

The steps, pre- and post- conditions of use case book air ambulance are described in Table 1.

The steps are also presented in the activity diagram in Fig. 3. Our approach uses integrated behavior models and extends them by aspect-oriented features. An integrated behavior model consists of a domain model and a set of activity models. The domain model provides the types of the domain objects.

Each activity is refined by pre- and post-conditions describing the effect of the activity in terms of domain objects. Typically, an initial configuration of the system is provided in terms of domain objects and their relations. The benefit of an integrated behavior model is an early and better integration of the structural domain model with the functional activity model. Pre- and post-conditions are formalized by the theory of graph transformation systems (Fig. 4).

It is conceivable, that the use case Finding Report is expressed as rules such as in Table 2, and Test crisis Management is express as rule in Table 3 (Figs. 5, 6, 7, 8, and 9).

Implementation

We are using AGG tool for implementing Heath care air ambulance system. With this tool we can perform conflict analysis and graph transformation. For graph transformation AGG tool is used which is a rule based visual language. A graph grammar contain in AGG program attributed by Java objects. Graph grammars contain a start graph and a set of rules which may have negative application conditions. A graph consists of two disjoint sets containing the nodes and the arcs of the graph. As a whole, the nodes and arcs are called the objects of the graph. Every arc represents a

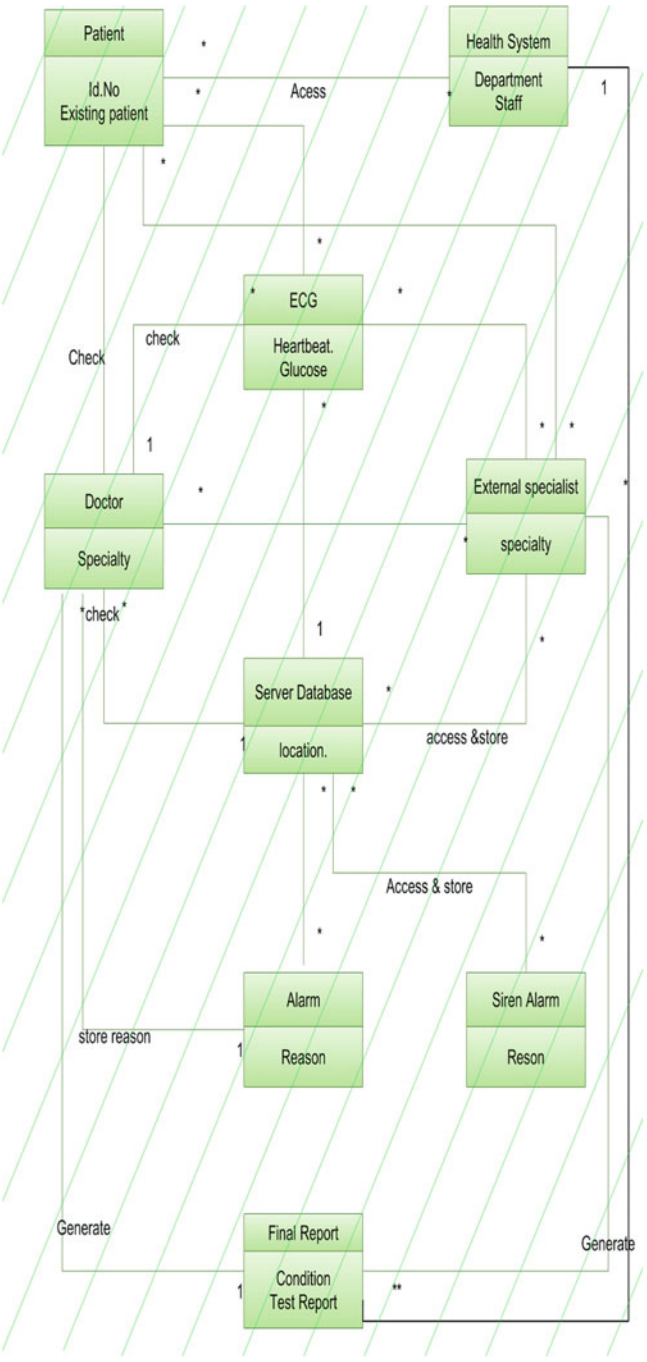

Fig. 2 Class diagram for health care air ambulance system

directed connection between two nodes, which are called the source and target nodes of the arc.

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 an after the action in question. Since states are modeled as graphs in AGG, it follows that basically an action can be described as a pair of two graphs modeling 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.

Table 1 Description of use case book air ambulance

Use case	Book air ambulance
Actor role	Admin
Trigger [condition]	Patient orders booking
Rule of precondition	Air ambulance exits
Rule of post condition	Each segment of the flight is booked
Use-case scenario	1. Select air ambulance 2. Select patient 3. Reserve segments 4. Book segments 5. Pay

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).

Our approach is towards detecting conflicts and dependencies. The tool was used for specifying the behavior of aspects and objects in terms of pre- and post-conditions and for analyzing conflicts and dependencies between them. The tool computed the necessary input in form of conflicts and dependencies which were then compared with the specified composition.

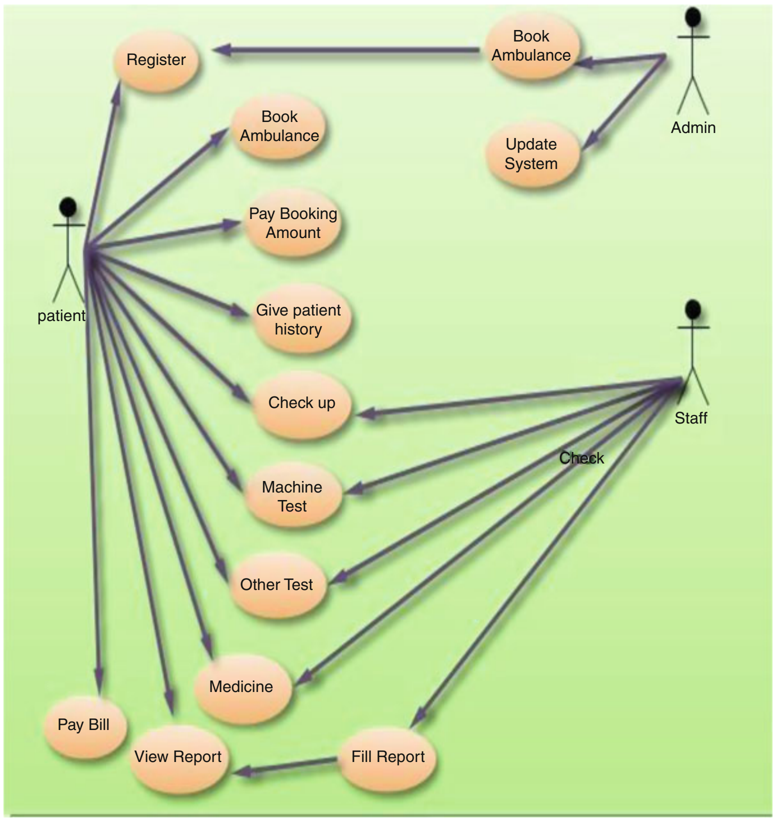

Fig. 3 Use case diagram for health care air ambulance system

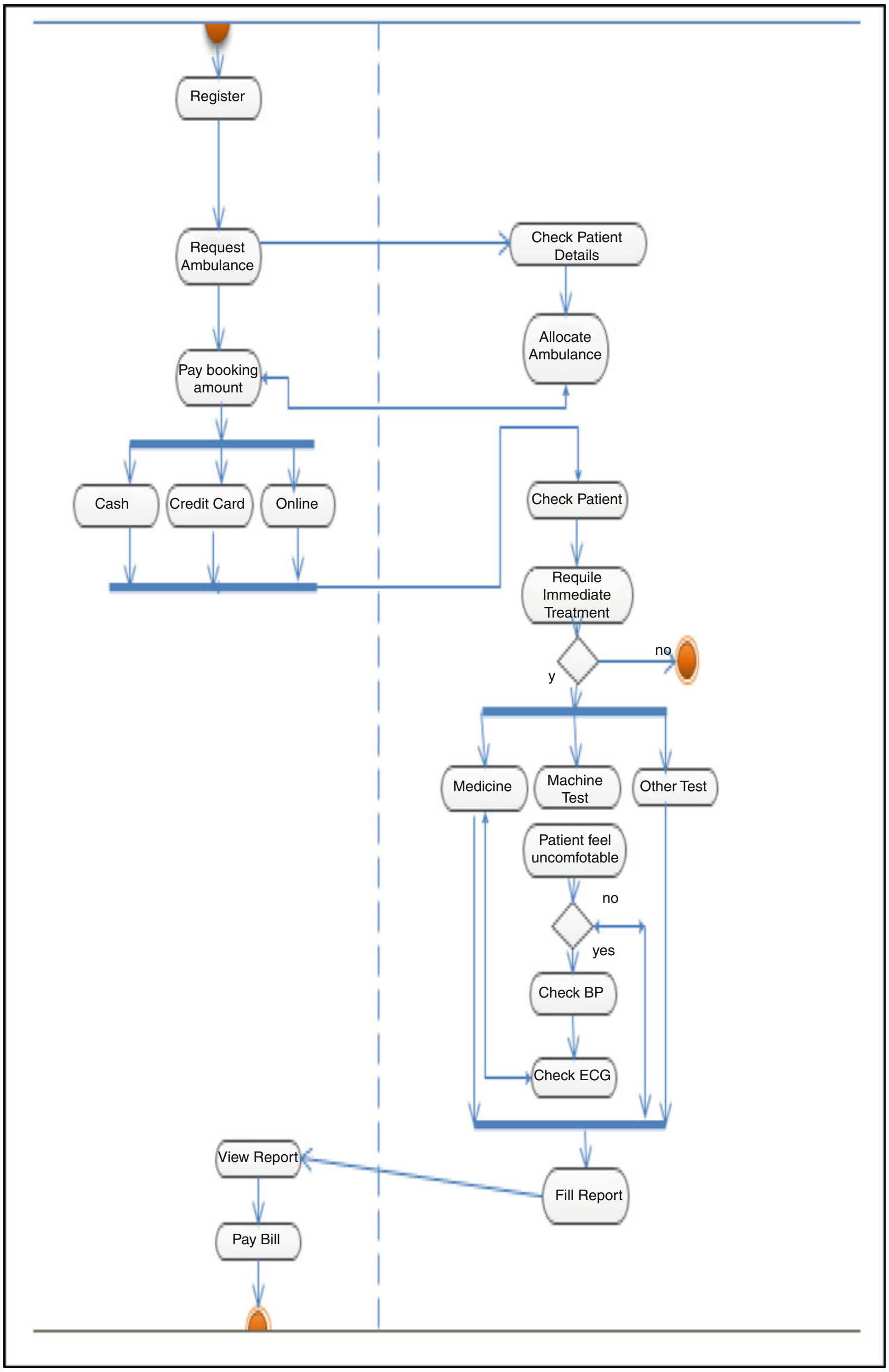

Fig. 4 Activity diagram for health care air ambulance system

Table 2 Finding report store given as rules

Finding report store scheme ( )
1) Only authorized staff i.e. doctor, nurse, staff etc can access the report of patient
2) Doctor checks the patient, if patient condition normal or abnormal
3) If patient condition is abnormal then store a patient report with time and report—name with a different server database on internet
4) If patient condition is normal then store a patient report to a local server on a system

Table 3 Test crisis management given as rules

Apply test crisis management ( )
Only doctor suggest medical test or other test to a patient
Doctor analyzes the heath condition of patient during test conduction
Patient feel normal or abnormal find in primary analysis
If patient feel abnormal, then conduct medical test with crises management system to a test
If patient feel normal, then conduct medical test without crises management system to a test

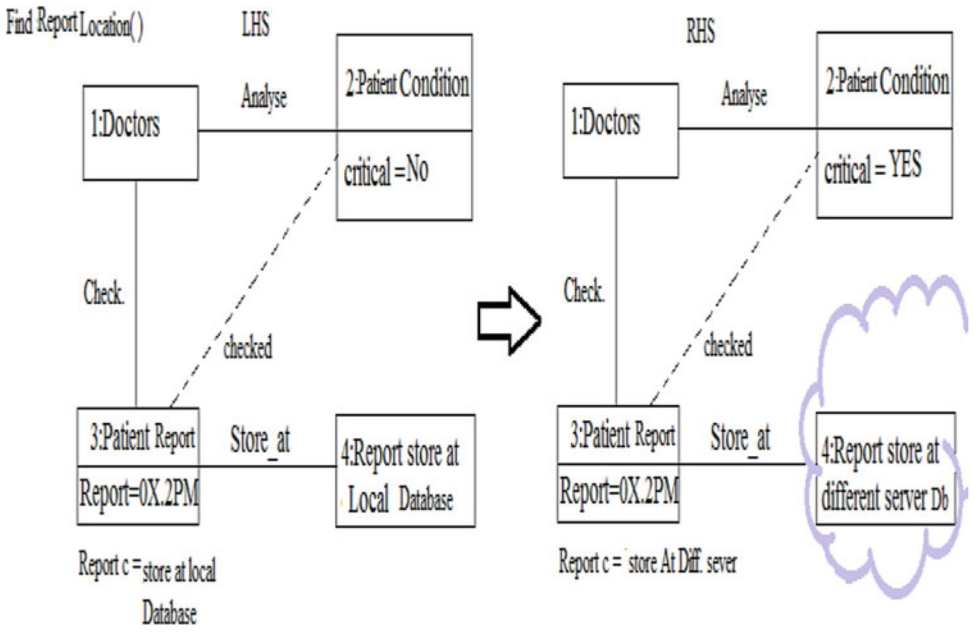

Fig. 5 Rule for report store

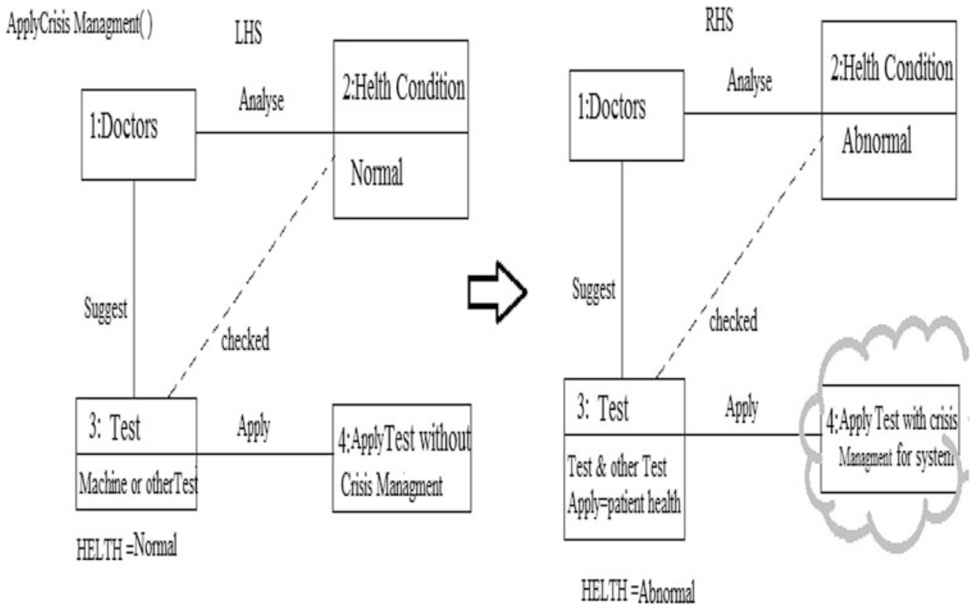

Fig. 6 Rule for crises management for test

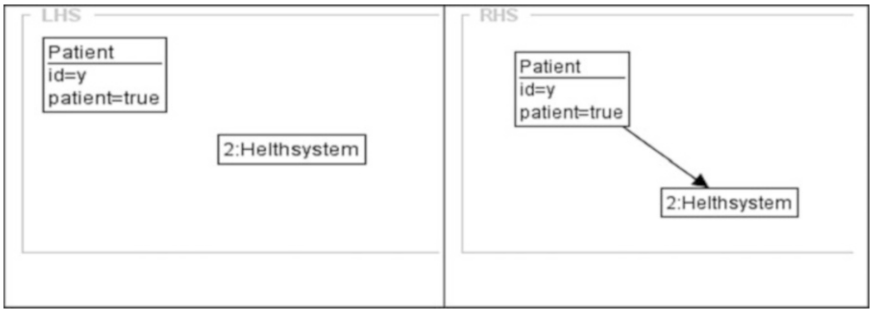

Fig. 7 Rule for patient login

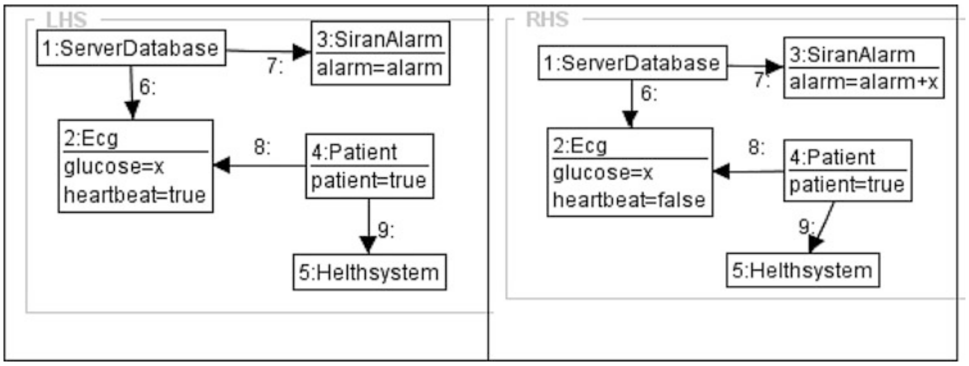

Fig. 8 Rule for siren alarm

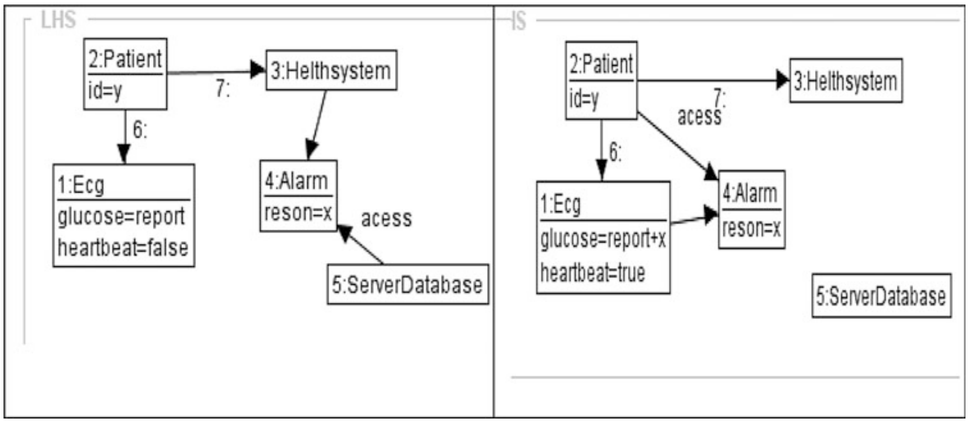

Fig. 9 Rule for alarm reason store in severs

Found conflicts and dependencies are potential conflicts (since they can be interactions) and not every possible conflict since it depends on the accurateness of the pre- and post-conditions. Graph transformation also allow to reason uniformly about object and aspect models [3]. Besides an editor for specifying the rules, the tool also provides all analysis functions as an API. Rules can be read from an XML file. Therefore, AGG is ideal to be used with existing UML CASE Tools [3].

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 to detect conflicts and dependencies. The approach uses formal aid to analyze systematically semi-formal specifications. We use graph transformation to detect conflicts.

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. D. Parnas. On the Criteria To Be Used in Decomposing Systems into Modules. Comm.
2. I. Jacobson and P.-W. Ng. Aspect-Oriented Software Development with Use Cases. Addison Wesley, 2005.
3. Katharina Mehner, Mattia Monga, and Gabriele Taentzer:" Interaction

Further Reading

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

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.

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

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.

E. Baniassad and S. Clarke. Theme: An Approach for Aspect-Oriented Analysis and Design.

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.unitrier.de/db/conf/models/models2007w.html WhittleJ07

S. Katz. Aspect Categories and Classes of Temporal Properties. *Transactions on Aspect-Oriented Software Development. LNCS 3880, Springer*, pp. 106-134, 2006.

Analysis in Aspect-Oriented Models"

G. Kiczales, E. Hisdale, J. Hugunin, M. Kersten, and J. Palm. An overview of AspectJ.

B. Nuseibeh, J. Kramer, and A. Finkelstein. AFramework for Expressing the Relationships betweenMultipleviews in Requirements Specifications.

J. Aráujo 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.

A. Rashid, P. Sawyer, A. Moreira, and J. Araujo. Early aspects: A model for aspect-oriented requirements engineering. In Proc.

AGG tool avialble at: http://user.cs.tu-berlin.de/~gragra/agg/index.html