Transformation Based Approach for Weaving Use Case Models in Aspect-Oriented Requirements Analysis

Motoshi Saeki
Dept. of Computer Science, Tokyo Institute of Technology
Ookayama 2-12-1, Meguro-Ku, Tokyo 152-8552, Japan
saeki@se.cs.titech.ac.jp

Haruhiko Kaiya
Dept. of Computer Science, Shinshu University
Wakasato 4-17-1, Nagano 380-8553, Japan
kaiya@cs.shinshu-u.ac.jp

ABSTRACT

This paper discusses techniques for combining non-functional requirements (NFRs) with functional requirements (FRs) in requirements analysis phases, based on aspect-oriented approach. In our approach, we elicit both types of requirements by using goal-oriented analysis method, and then we specify the relationships between the FRs and NFRs with a cross-cutting table because an elicited NFR can be related to multiple FRs. These relationships help us to evolve the goal-graphs of FRs and NFRs in goal-oriented analysis processes. We can identify use cases from the elicited FRs. To weave the NFRs, from the cross-cutting table and the use cases of the FRs, we design transformation rules to automatically produce use case diagrams, use case descriptions and use case maps that achieve the NFRs. The paper illustrates a simple example to clarify our method.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications – Methodologies

General Terms

Design

Keywords

Use Case, Aspect, Pattern, Reuse, Requirements Analysis, Template

1. INTRODUCTION

In recent software development, many industrial practitioners come to adopt object-oriented approach, especially in the upper stream of development, e.g. requirements analysis and design, and they frequently use UML diagrams, use case diagrams and class ones. However, generally speaking, it is difficult to identify use cases and objects, and to construct diagrams in the early stage of requirements analysis, e.g. requirements elicitation stage. To solve these difficulties, several researchers and practitioners integrate requirements elicitation methods with use case modeling.

A family of goal-oriented requirements analysis(GORA) methods [1, 2, 3, 4] is one of the promising approaches to support requirements elicitation, and is a top-down approach for refining and decomposing the needs of customers into more concrete goals that should be achieved for satisfying the customers' needs. We can also find several researches and case studies where use case modeling techniques are combined with a goal-oriented method, in order to elaborate both the identification of use cases and the decomposition of goals[5, 6, 7, 8]. Figure 1 depicts a general process for completing a requirements specification document. In this process, we can use a goal-oriented method for eliciting requirements, and use case modeling technique to describe requirements, and this style of combination is putting into practice. Requirements are classified into two categories; one is functional ones (FRs) and the other is non-functional ones (NFRs) such as reliability, performance, security and so on. In addition to elicitation of functional requirements, a goal-oriented approach could be applied to the elicitation of non-functional ones[9]. However, functional requirements can be cleanly decomposed into components, e.g. sub-goals and use cases that are units of the system functions, while non functional requirements cross-cut the functional units and are tangled on them. It leads to the difficulty in decomposing cleanly functional requirements and non- functional ones. As a result, it is hard to get reusable and maintainable components as unit of requirements. Aspect-oriented programming (AOP) [10, 11] is a programming

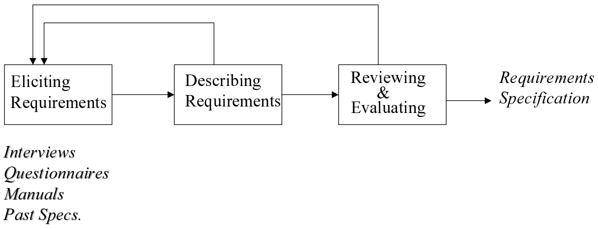


Figure 1: Requirements Analysis

technique to have separated coding of functional parts from non-functional ones, called aspects, and to weave them into a final implementation. We apply this idea to requirements specification in order to make the modularity of components higher. That is to say, in our technique, we describe functional requirements separating from non-functional requirements and after specifying both of them, we weave them together into a final requirements specification written with use cases. Their separation enables us to have components of functional requirements and non-functional ones. In [12], the aspects of componentware was considered as interfaces of components in order to have the suitable and correct combination of the components, but its aim and approach are different from this paper and AOP.

Although a goal-oriented method can support elicit NFRs in the same way as FRs' elicitation[9], it is a problem how we can embed the elicited NFRs into the FRs and get an integrated description based on use case modeling, because the NFRs are scattered to the FRS, so called cross-cutting concerns to the FRs. This paper provides one of the solutions to deal with weaving FRs and NFRs to get the integrated descriptions of the elicited use cases. The essential idea is as follows; During the independent elicitation of FRs and NFRs by using a goal-oriented method, we specify the relationship between sub-goals of FRs and those of NFRs. The relationship is represented in a table form in the same way as [13]. After completing a goal graph of FRs, we extract use cases as a use case model for FRs. By transformation, the FR use case model is evolved to a use case model where the NFRs are satisfied, and what transformation can be applied is identified from the relationships between the FR and NFR sub-goals. The rest of the paper is organized as follows. The next section presents the overview of our approach. In section 3, we show a transformational approach on use case models to weave FRs and NFRs, in example-based style.

2. REQUIREMENTS ANALYSIS PROCESS

2.1 Goal-Oriented Analysis and Use Case Modeling

Figure 1 depicts the flow of a requirements analysis process based on our aspect-oriented requirements analysis with goal-oriented method and use case modeling one. In the first step, an analyst elicits requirements by using a goal-oriented analysis method. In most cases, he or she has interviews and/or questionnaires with stakeholders including customers and users. In the case that a similar system had been developed before, he investigates its documents such as manuals and specifications. After gathering information, he starts goal-decomposition activities, in some cases together with users and customers, to construct a goal-graph for functional requirements (FR goal graph) and a graph for non-functional ones (NFR goal graph) such as reliability, performance, memory space and so on. In the FR goal-graph, its leaves include operational descriptions, so we can make them correspond use cases in use case modeling. A use case description of the corresponding use case results from the contents of the leaf goal. Figure 3 illustrates this process. The example that we used in the figure is a tool for supporting the task that program committee chairs (PC chairs) of academic international conferences have to perform. The PC chairs should organize the committee to have many high quality papers and to reduce the PC chairs' tasks. See the first decomposition in the goal graph of the figure. The chairs receive the submitted papers and distribute them to the reviewers, normally PC members. After getting the review reports from the reviewers, they summarize them and have PC meetings to decide which papers will be accepted or rejected. The authors of the papers are notified of their acceptance or rejection by the PC chairs. Since each leaf of the decomposed graph includes operational descriptions, we can identify it as a use case.

We adopt a use case diagram of UML and a use case description for specifying the behavior of each use case[14]. In addition them, to specify control dependencies, e.g. execution order on use cases, and data dependencies on use cases by using use map technique[15]. We adopt two additional dependencies among use cases; data dependency and control dependency, and they are represented by using a technique of use case map. For example, the use case "Deciding Acceptance or Rejection" of the submitted papers can be performed only after "Receiving Review Reports" of the papers from the reviewers. These two use cases have control dependency.

2.2 Cross-Cutting Table

During or after the goal-decomposition processes, the analyst can relate the sub-goals of the FR goal graph and those of the NFR graph. Figure 4 shows the example of the relationship between the FRs and NFRs, and the description in a table form, called cross-cutting table. This table is useful to identify and to understand which FRs the NFRs are cross-cutting over. The vertical axis of the table stands for a list of the FR sub-goals, and on the other hand the horizontal one is NFR sub-goals. For example, "Receiving Paper Submission" should satisfy high security and reliability requirements. According to ISO9126, Reliability can be decomposed into three sub-goals "Maturity", "Fault-tolerantness" and "Recoverability". So we can have another table, more detailed, that expresses the cross-cutting information of the decomposed NFRs. For example, Maturity are crossed over "Distributing Papers to Reviewers", "Deciding Acceptance or Rejection" and "Notifying Acceptance or Rejection". It means that we should consider how Maturity should be satisfied when we decompose these FR sub-goals further or specify use cases of these three goals.

2.3 Aspect Weaving

We capture aspect weaving as transformation of the use case structures into one that can hold the non-functional requirements. The analyst considers the transformations that are suitable for the NFRs, and apply them to the use case model of the functional requirements. New use cases may be added and/or the structure of the use case description may be changed so that the NFRs are satisfied. Figure 5 summarizes the process of weaving the aspects (transforming). Suppose that the analyst finds "high reliability" requirements is imposed on a use case #2 of the figure. The use case #1 has a dependency relationship to the use case #2, which is represented with a gray arrow in the figure. If the relationship is a control dependency, the use case #1 is executed and then the use case #2 is activated. The arrows stand for an instance of execution sequence of the whole of the system, i.e. a scenario.

To get high reliability, we often take a duplicate style of performing tasks. In this example, we allocate the same

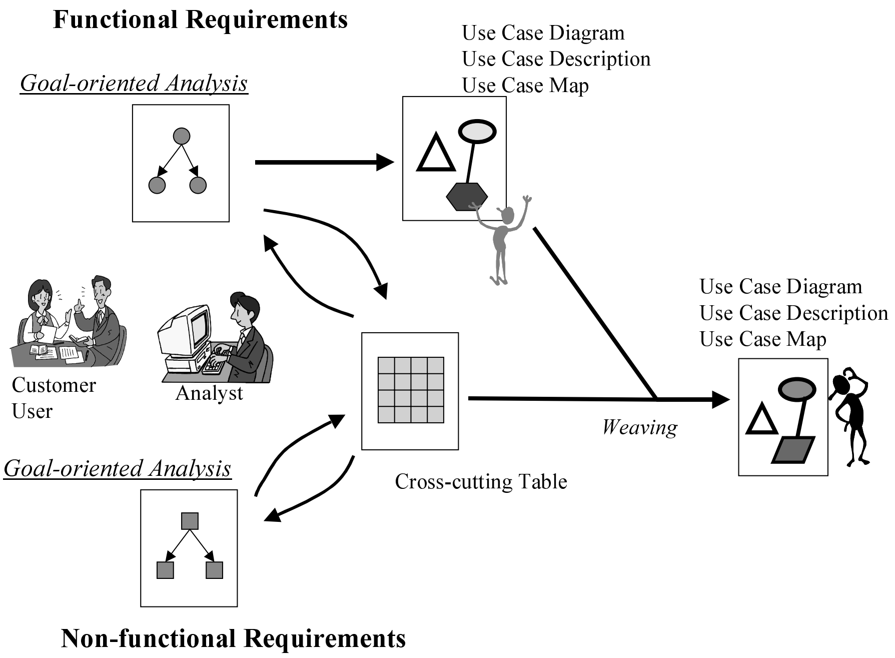


Figure 2: Aspect-Oriented Requirements Analysis

or similar "task" to another actor and after the two actors' completing the task, their results are checked against each other. This is a typical strategy of working to keep its high reliability. We can have this strategy as a weaving knowledge and it can be represented with a transformation of a use case "task" into the duplicated ones "task#1" and "task#2" as shown in Figure 5. In the transformation, a new use case "check", a new actor and their interactions are also added. The analyst selects a suitable aspect pattern for the non-functional requirements and applies it to the use case description. Finally he or she gets the final use case descriptions that are satisfied with the functional requirements and the non-functional ones. This weaving technique can be catalogued as patterns that are formally defined with the rules on graph grammar since a use case diagram is considered as a graph. By separating NFRs with FRs, maintainability and traceability of requirements are improved.

Let's turn back our example. According to the cross-cutting table shown in Figure 4, Reliability can be related to five use cases. Thus we should apply this transformation to these five use cases if we adopt the "duplicate" strategy.

3. WEAVING EXAMPLES

In this section, we discuss more examples of transformation based weaving to get the use case model from FR use case model, by using the example problem of the PC chair's task. And we will discuss briefly the possibility of transformations as rules to get reusable weaving knowledge. In this example, we can consider the reliability and fairness of making a program as examples of the NFRs. The strategies for achieving these non-functional requirements are 1) sending a confirmation whenever authors contact to the PC chair, 2) having two PC chairs, and so on. These can be added to the task of the PC chairs. The transformation rules are shown in Figures 6 and 7. Figure 6 is for adding an activity "sending a confirmation for the receipt" to the abstract use case whose type is "receiving something". The examples of the use cases to which this rule can be applied, i.e. to which the activity can be added are "Appointing PC members", "Receiving Paper Submissions", "Receiving Review Reports" and so on. Note that the pattern of Figure 6 is parameterized and the parameters are parenthesized with "[" and "]". They can be considered as hot spots within a use case description. In the sense that the parameterized use case "Receiving [Something]" includes hot spots, it is an abstract use case and a pattern of a use case.

Figure 7 is for introducing another PC chair in order to make high reliability and fairness of making a program. The pattern adds a new use case to check the results of the tasks performed by them. If we apply the pattern to the use case "Deciding Acceptance or Rejection", we can get a system where two PC chairs would decide the acceptance or rejection of papers independently and after that they discuss their results to get a conclusion. The system seems to improve the reliability. Note that the use case "Deciding Acceptance or Rejection" has data dependency to "Receiving Review Report", i.e. the received review reports are input data to "Deciding Acceptance or Rejection". This dependency should inherit to the duplicate-generated use cases.

4. DISCUSSION AND FUTURE WORK

This paper discusses transformation based weaving aspects in requirements analysis. Weaving can be formalized with transformation rules or transformation patterns to derive the use cases and their structures that are satisfied with non-functional requirements.

Cross-cutting tables are really useful to design how transformations should be applied, and also to proceed the de-

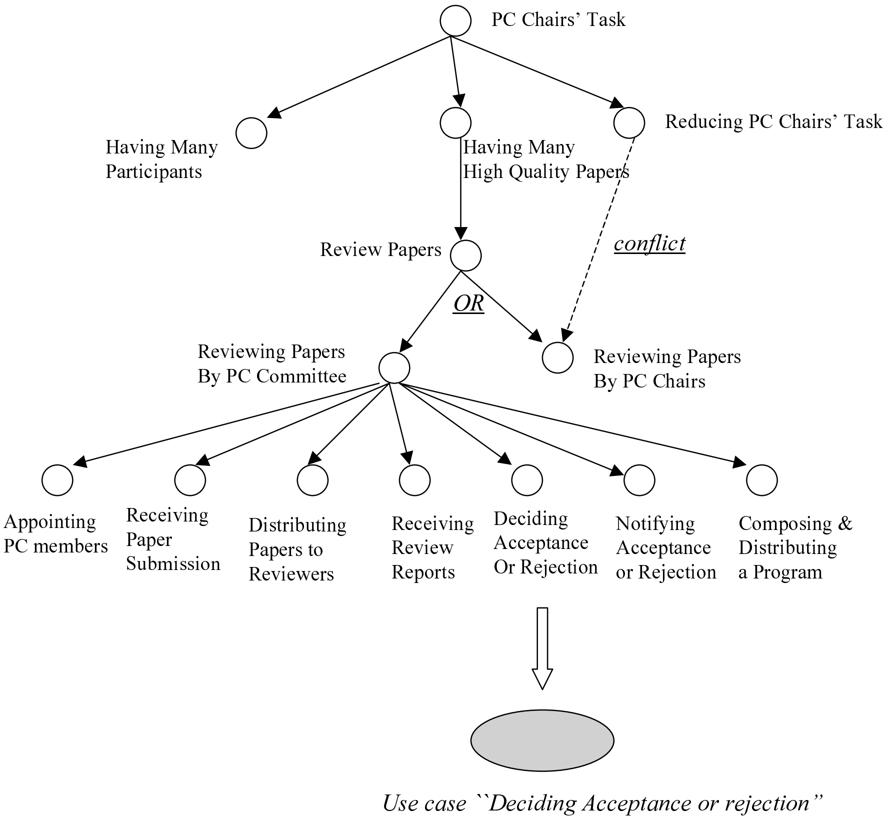

PC Chairs' Task
Having Many Participants
Having Many High Quality Papers
Reducing PC Chairs' Task
conflict
Review Papers
OR
Reviewing Papers By PC Committee
Reviewing Papers By PC Chairs
Appointing PC members
Receiving Paper Submission
Distributing Papers to Reviewers
Receiving Review Reports
Deciding Acceptance Or Rejection
Notifying Acceptance or Rejection
Composing & Distributing a Program
Use case ``Deciding Acceptance or rejection"

Figure 3: Goal-Oriented Analysis and Use Cases

composition of sub-goals. In particular, the tables suggest which set of use cases should be transformed so as to satisfy the NFRs. The examples in the paper were very simple and the transformations were applied once, not multiple applications of different transformations. In practical setting, the analyst applies multiple transformations to several use cases. In this situation, we should consider the order of transform application and how to specify the order.

This paper picked up only one example in a specific domain. Therefore, to construct a practical pattern base system that stores the use case patterns and aspect patterns, we should explore much more case studies and extract patterns from various kind of problem domain. Exploring how to construct class diagrams from use case descriptions is also one of the future work.

5. REFERENCES

[1] Annie I. Anton. Goal-based requirements analysis. In *Proceedings of the Second IEEE International Conference on Requirements Engineering(ICRE'96)*, 1996.

[2] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.

[3] Axel van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering(RE'01)*, pages 249–263, 2001.

[4] Haruhiko Kaiya, Hisayuki Horai, and Motoshi Saeki. AGORA : Atributed Goal-Oriented Requirements Analysis Method. In *Proceedings of the 10th Anniversary IEEE Joint International Requirements Engineering Conference(RE'02)*, pages 13–22, Sep 2002.

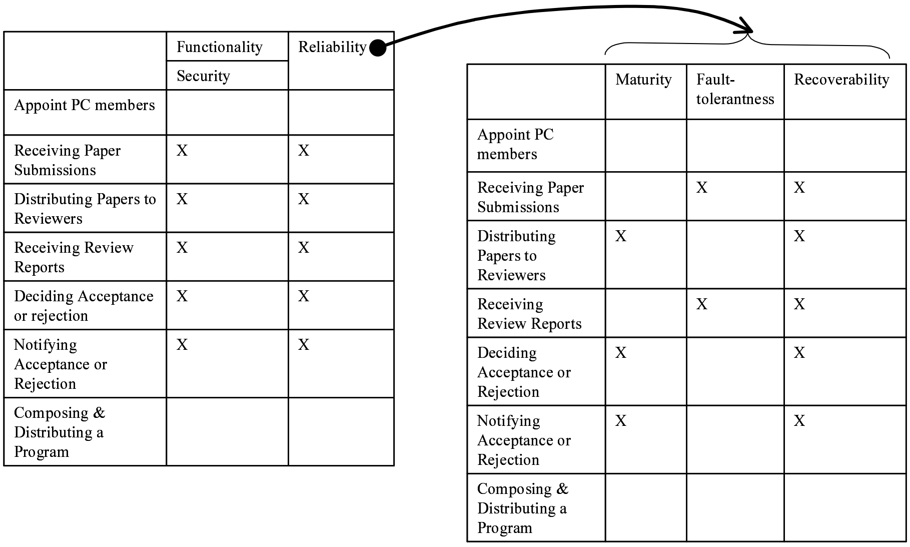
[5] Annie I. Anton, W. Michael McCracken, and Colin Potts. Goal decomposition and scenario analysis in business process reengineering. In *Proceedings of the 6th International Conference of Advanced Information Systems Engineering (CAiSE 94)*, pages 94–104, June 1994.

[6] C. Rolland, C. Souveyet, and C. Ben Achour. Guiding goal modelling using scenarios. *IEEE Transaction on Software Engineering*, 24(12):1055–1071, December 1998.

[7] Annie I. Anton, Ryan A. Carter, Aldo Dagnino, John H. Dempster, and Devon F. Siege. Deriving goals from a use-case based requirements specification. *Requirements Engineering Journal*, 6:63–73, 2001.

[8] Victor F.A. Santander and Jaelson F.B. Castro. Deriving use cases from organizational modeling. In *Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering(RE'02)*, pages 32–39, 2002.

[9] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic, 1999.



	Functionality	Reliability
	Security	
Appoint PC members		
Receiving Paper Submissions	X	X
Distributing Papers to Reviewers	X	X
Receiving Review Reports	X	X
Deciding Acceptance or rejection	X	X
Notifying Acceptance or Rejection	X	X
Composing & Distributing a Program		

	Maturity	Fault-tolerantness	Recoverability
Appoint PC members			
Receiving Paper Submissions		X	X
Distributing Papers to Reviewers	X		X
Receiving Review Reports		X	X
Deciding Acceptance or Rejection	X		X
Notifying Acceptance or Rejection	X		X
Composing & Distributing a Program			

Figure 4: Cross-Cutting Table

[10] G. et. al. Kiczales. Aspect-Oriented Programming. In *Lecture Notes in Computer Science (Proc. of ECOOP'97)*, volume 1241, pages 220–242, 1997.

[11] L. Bergmans and M. Aksit. Composing Crosscutting Concerns Using Composition Filters. *CACM*, 44(10):51–57, 2001.

[12] J. Grundy. Aspect-Oriented Requirements Engineering for Component-Based Software Systems. In *Proc. of 4th IEEE International Symposium on Requirements Engineering (RE'99)*, pages 84–91, 1999.

[13] A. Rashid, A. Moreira, and J. Araujo. Modularisation and Composition of Aspectual Requirements. In *Proc. of 2nd International Conference on Aspect-Oriented Software Development (AOSD2003)*, pages 11–20, 2003.

[14] Alistair Cockburn. *Writing Effective Use Case.* Addison-Wesley, 2000.

[15] D. Amyot, L. Logrippo, R. Bruhr, and T. Gray. Use Case Maps for the Capture and Validation of Distributed Systems Requirements. In *Proc. of 4th IEEE International Symposium on Requirements Engineering (RE'99)*, pages 44–53, 1999.

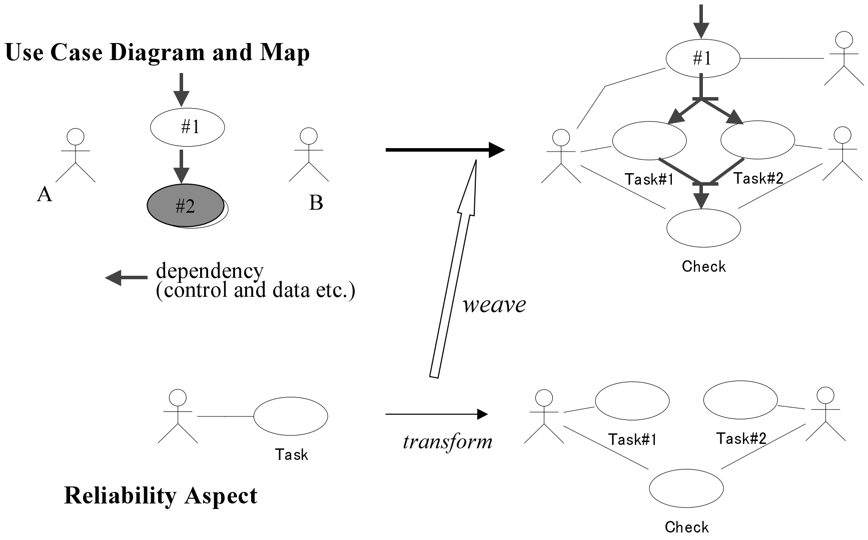
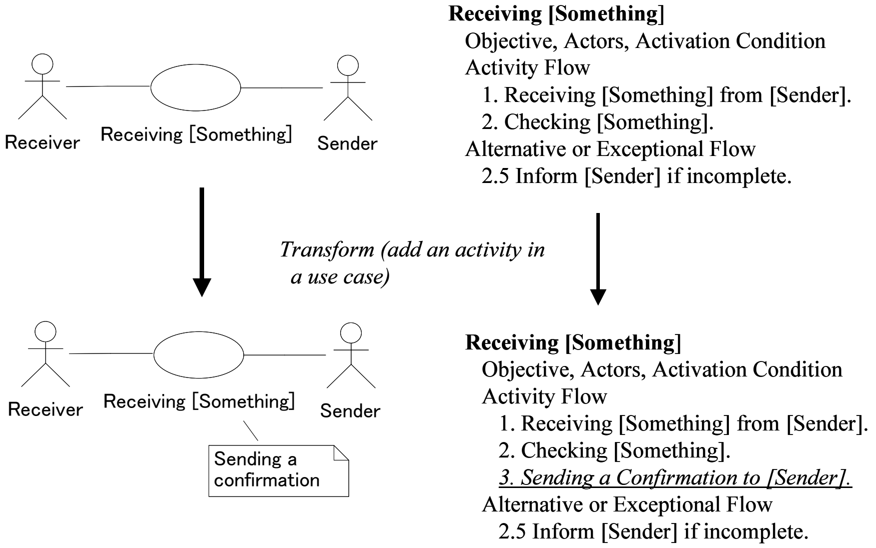


Figure 5: Transformation Based Weaving



Receiving [Something]
Objective, Actors, Activation Condition
Activity Flow
1. Receiving [Something] from [Sender].
2. Checking [Something].
Alternative or Exceptional Flow
2.5 Inform [Sender] if incomplete.

Receiving [Something]
Objective, Actors, Activation Condition
Activity Flow
1. Receiving [Something] from [Sender].
2. Checking [Something].
3. Sending a Confirmation to [Sender].
Alternative or Exceptional Flow
2.5 Inform [Sender] if incomplete.

Figure 6: Weaving Use Case Descriptions

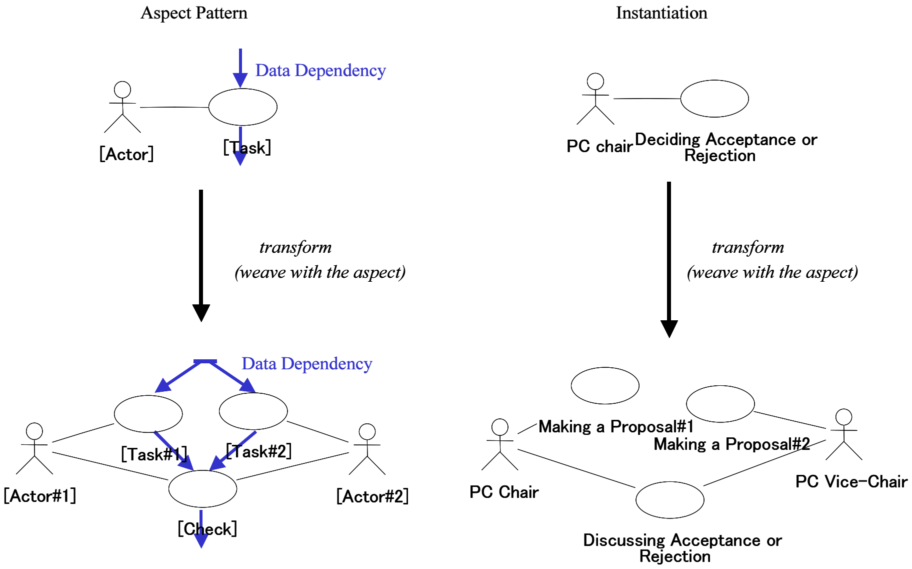


Figure 7: Weaving Use Case Diagram and Maps