Weaving Multiple Viewpoint Specifications in Goal Oriented Requirements Analysis

Haruhiko Kaiya
Shinshu University
kaiya@cs.shinshu-u.ac.jp

Motoshi Saeki
Tokyo Institute of Technology
saeki@se.cs.titech.ac.jp

Abstract

Goal oriented requirements analysis is one of the useful method to bridge the gaps between stakeholders needs and a requirements specification. Goals are structured as a directed graph, and its upper parts show the needs and its lower parts show the requirements. Although goals come from several different viewpoints, such viewpoints are not separated explicitly in such a goal graph. As a result, following kinds of problems can be occurred. First, we cannot easily remove or modify one viewpoint which affects several different goals. Second, it is difficult to analyze several different viewpoints separately and/or incrementally. For example, we cannot analyze a family of products simultaneously. Third, such a graph is not intrinsically scalable. In this paper, we propose a method to weave several goal graphs each of which represents a viewpoint. Several candidates of a weaved graph are systematically generated based on the structural characteristics of graphs for each viewpoint. By using this method, We can overcome the problems above, and we can easily propose alternative requirements specification if a specification is rejected by stakeholders.

keywords requirements elicitation, goal oriented requirements analysis, multiple viewpoints, aspect-oriented requirements analysis

1 Introduction

Since requirements analysis step is the first one in software development processes, the analysts' activities have great effects on the quality of the produced software and on its development cost. In particular, requirements elicitation from stakeholders is one of the most crucial steps, and the methodological techniques for it are indispensable. 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. The resulting artifact is an AND-OR graph whose nodes are elicited goals. 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]. The combination of these methods are being put into practice[1, 9]. However the existing goal-oriented methods have the following two problems;

1. They have not explicitly mentioned the activities of identifying sub-goals from the goal to be developed as a method, or not provided powerful guidelines for goal decomposition. As a result, the resulting goal graphs may include the sub-goals into which a parent goal has been decomposed in multiple perspectives, and
2. They do not consider the collaborative activities by stakeholders to elicit goals and to construct a goal graph. In other words, they do not provide the suggestions how multiple stakeholders can participate in requirements elicitation as a method.

Some studies provided guidelines and templates how to decompose a goal into sub-goal, e.g. KAOS[3] and AGORA[4]. However they did not consider that goal elicitation should be the collaborative tasks done by a team of stakeholders, who have knowledge of different domains. Stakeholders as the knowledge source play an important role on eliciting requirements of high quality and all of them should participate in requirements elicitation activities. Some extended versions of goal-oriented methods to support collaborative tasks such as embedding idea-generation methods have been proposed. However they did not consider the support on how to decompose goals following a single perspective.

Viewpoints approach is a method where an analyst gathers the fragments of the requirements of a stakeholder (or of a specific group of stakeholders) and integrates them[10]. It does not support not only to structure and to manage the requirements from each perspective specific to a stakeholder, but also includes the supports of the collaborative tasks. However, this approach has a specific method and notation, and is not for goal-oriented methods or for use case modeling.

In this paper, we propose the integration of viewpoints approach into a goal-oriented method + a use case modeling one. As a result, we can make up for the above two deficiencies; 1) support for collaborative tasks and 2) the mixture of goal decompositions from multiple perspectives in a goal graph. More concretely, each stakeholder develop a goal graph with use cases whose goal decomposition follows his perspective, and then the developed graphs are conceptually weaved into a final goal graph. Simultaneously, the use cases attached to each goal graph are also weaved into a set of the use cases that can achieve the final goals. Since each stakeholder can decompose their goals following his unique perspective for himself and have a goal graph of his own, stakeholders' knowledge and intents can participate in the process of developing a final goal graph and use cases. As a result, our approach encourages stakeholder participatory requirements elicitation in collaborative way. The key point of our weaving technology is a cross-cutting table which represents inter-relationships among the goals. A goal in a graph may be related into several goals included in the other graphs which were constructed based on different perspectives. The table is used for identifying which goals should be weaved, and as a result identifying which use cases should be weaved.

The rest of the paper is organized as follows. In the next section, we summarize the issue of the current version of goal-oriented methods that were mentioned above again. In particular, we focus on the mixture of multiple decomposition criteria in a goal graph. Section 3 presents our technique of weaving. We have two types of weaving; one is for goal graphs and another is for use cases. In the case of weaving goal graphs, since they are essentially AND-OR graphs, the idea of logical combination made of AND and OR operators is sufficient. On the other hand, since a use case describes the behavior of the corresponding function, we should consider how to weave the behavior of the use cases. We pick up the example consisting two concerns; functional requirements (FR) concern and non-functional concern (NFR), and apply a transformational approach to weaving use cases. Sections 4 and 5 are the discussion through a case study and the concluding remarks respectively.

2 Goal Decomposition

2.1 Functional Requirements v.s. Non-Functional Requirements

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[11]. In this section and the next one, these two concerns (perspectives) are used as a typical example of multiple separated concerns. Because FRs 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 clear example to explain how our approach works on these two concerns.

2.2 Goal-Oriented Analysis + Use Case Modeling

In the first step of our requirements analysis processes, 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. Figures 1 and 2 illustrates the examples of a FR goal graph and an NFR one respectively. 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[12]. 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[13]. 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.

The example of the NFR graph is based on ISO 9126 related to Software Quality. Many researchers used it for clarifying a part of NFRs and we use it as an example. The complete separation of the FR graph and the NFR one like this, i.e. separation of concerns resulted from the idea of viewpoints approach[10] or aspect-orientation[14, 15, 16, 17]. Their separation enables us to have components of functional requirements and non-functional ones. Although a goal-oriented method can support elicit NFRs in the same way as FRs' elicitation[11], 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. In fact, as mentioned in the next subsection, the mixture of FRs and NFRs frequently appears in the same goal graph.

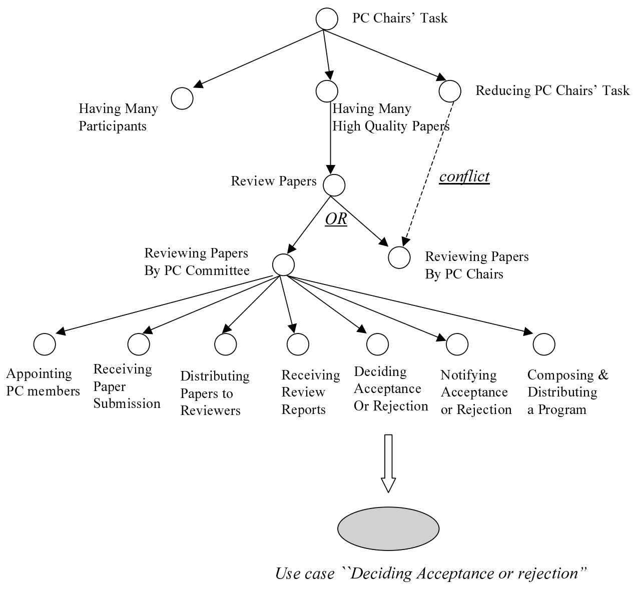

PC Chairs' Task
Having Many Participants
Having Many High Quality Papers
Reducing PC Chairs' Task
conflict
Review Papers
OR
Reviewing Papers By PC Committee
Reviewing Papers By PC Chairs
Appointing PC members
Receiving Paper Submission
Distributing Papers to Reviewers
Receiving Review Reports
Deciding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

Figure 1. Goal-Oriented Analysis and Use Cases

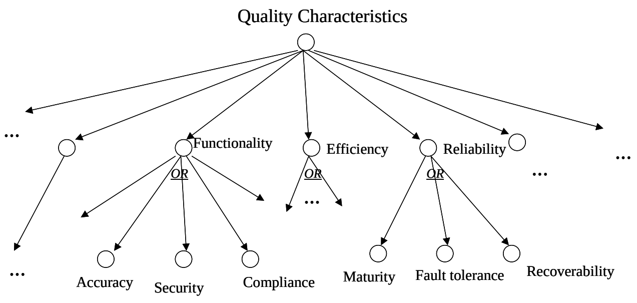


Figure 2. A Goal Graph for Non-Functional Requirements

2.3 Goal Decomposition by Multiple Concerns

When the analysts apply a goal-oriented method, they can put the sub-goals from the other concerns, because there are no powerful guidelines for goal decomposition. In Figure 1, the goal "Reviewing papers By PC Committee" can be decomposed into seven sub-goals, e.g. "Appointing PC members" from a functional concern (more precisely, the concern of business tasks). Additionally, it can have the sub-goals "High Reliability" and "High Security" from non-functional concern. As shown in Figure 3, the NFR sub-goals are embedded in the FR goal graph in a usual goal-oriented method. Because the goal "Reviewing papers By PC Committee" has them as its sub-goals. That is to say, to achieve the goal, it is necessary to achieve not only the seven FR sub-goals but also "High Reliability" and "High Security" goals. Consequently, there appears multiple concerns included in a goal graph and they lead to the difficulty to maintain the goal graph. Furthermore, as shown in the figure, the goal decomposition of "High Reliability" from NFR concern progresses and the graph gets the sub-goals "Maturity", "Fault-tolerance" and "Recoverability". This decomposition is done in the same goal graph, and we will have got the graph much more difficult to maintain. To avoid this kind of issue, as mentioned in the previous subsection, goal decomposition in a goal graph should be done following a single concern. If the analysts manage more than one concern, they should construct the different graphs which are separated concern by concern. After constructing goal graphs and their use cases from separated concerns, they weave both the graphs and the use cases.

2.4 Related Work : Aspect-Oriented Approach

Aspect-oriented programming (AOP) [14, 15] is a programming 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. 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 [17]. 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, and what transformation can be applied is identified from the relationships between the

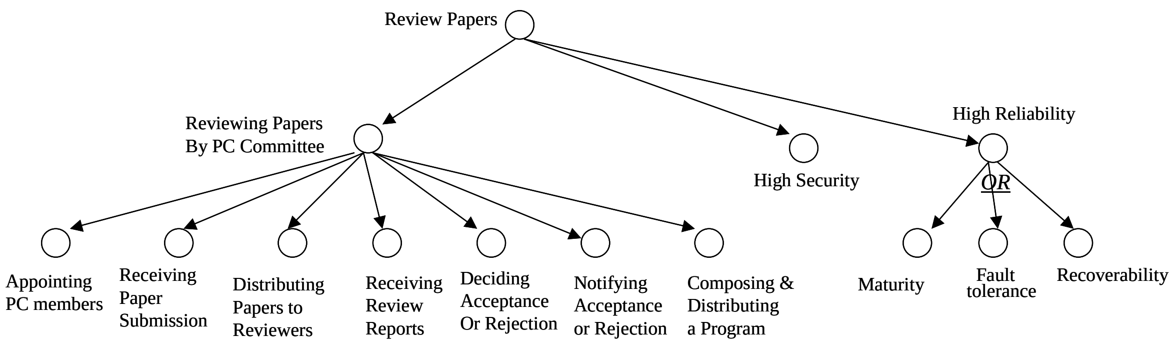

Review Papers
Reviewing Papers By PC Committee
High Security
High Reliability
OR
Appointing PC members
Receiving Paper Submission
Distributing Papers to Reviewers
Receiving Review Reports
Deciding Acceptance Or Rejection
Notifying Acceptance or Rejection
Composing & Distributing a Program
Maturity
Fault tolerance
Recoverability

Figure 3. Mixture of FRs and NFRs in a Goal Graph

FR and NFR sub-goals.

3 Weaving Goals and Use Cases

3.1 How to Weave: An Overview

Let's consider how to weave goal graphs and use cases. We should consider two points; one is how to specify which parts of the graphs are weaved and another is how to weave them. As for the former point, we use a matrix so called Cross-cutting Table, which represents which two goals belonging to the different concerns are related. In the latter point, we should consider two types of weaving; one is weaving on goal graphs and another is on use cases. Weaving goal graphs can be based on the logical meaning of AND-OR decomposition, while weaving use cases depends on concerns. As general technique for weaving use cases, we adopt a transformational approach. In this paper, we focus on FR concern and NFR one and illustrate weaving use cases on the FR concern and the NFR one. Figure 4 depicts two types of weavings and their details will be explained in the later subsections, and illustrates the weaving of the goal B to the goal A2 in the graph A. The bottom graph of the figure is the resulting one. The point is that weaving goals causes weaving the use cases derived from its leaf goals. In the figure, we weave the goals A2 and B. This weaving triggers the weaving of the use cases U_A21, U_A22, U_B1 and U_B2. We can get four combinations of them and they result from the characteristics of OR decomposition of the graph B.

3.2 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 5 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

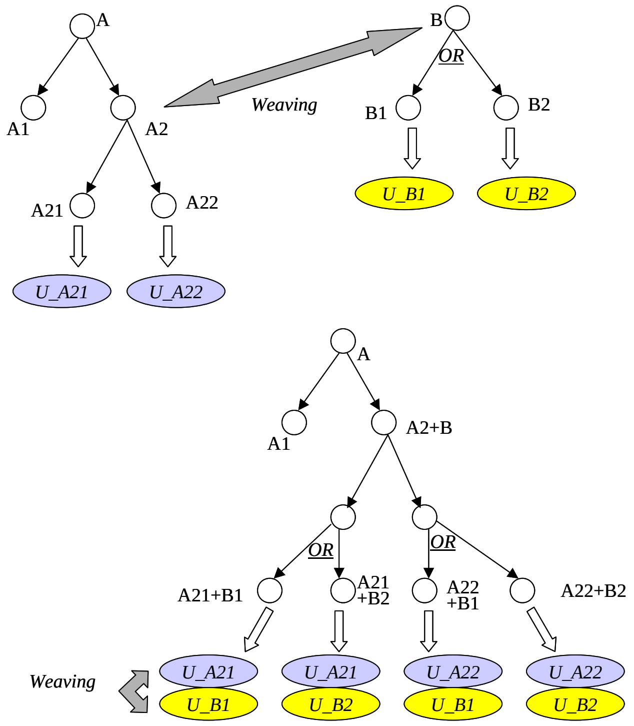
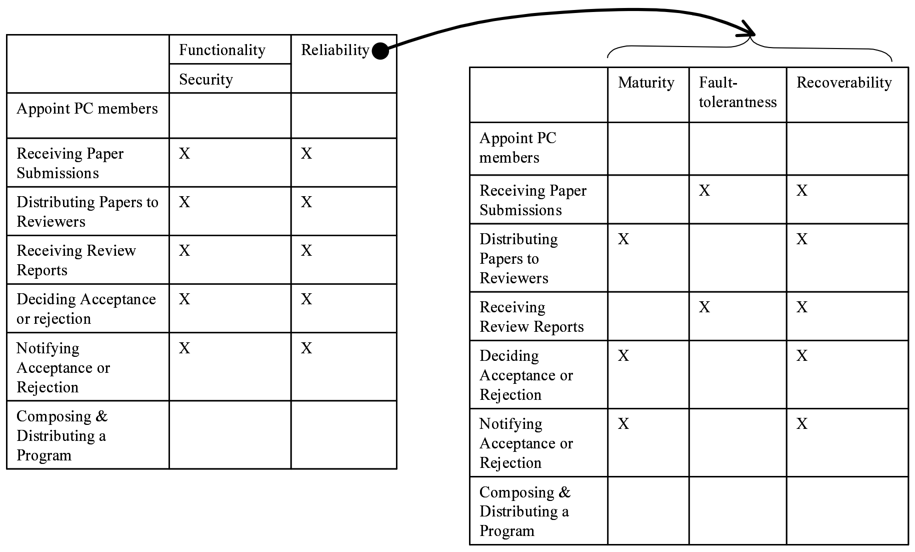


Figure 4. Weaving Goal Graphs

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-tolerance" 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.



	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 5. Cross-Cutting Table

3.3 Weaving goal graphs

The logical meaning of a goal graph is simple an AND-OR graph. If we achieve all of the sub-goals, we can achieve their parent goal. We can formulate the relation between a goal and its sub-goals with logical expressions. As illustrated in Figure 4, suppose that the goal A2 is decomposed into the sub-goals A21 and A22 in AND composition. To achieve the goal A2, both achievements of A21 and A22 are necessary. So we can write this decomposition as a logical formula $(A21 \wedge A22) \rightarrow A2$. On the other hand, in the figure, we can write the OR-decomposition of the goal B into B1 and B2 as $(B1 \vee B2) \rightarrow B$. The logical meaning of weaving the goals A2 and B is that both A2 and B are achieved, i.e. $A2 \wedge B$ holds. From the above two logical formulas, we can have $(A21 \wedge A22) \wedge (B1 \vee B2) \rightarrow (A2 \wedge B)$ as a logical consequence. The formula is identical to $(((A21 \wedge B1) \vee (A21 \wedge B2)) \wedge ((A22 \wedge B1) \vee (A22 \wedge B2))) \rightarrow (A2 \wedge B)$, and it expresses the result of weaving shown in the bottom of Figure 4.

As for weaving goals in goal graphs, we simply weave the goals that are specified with the cross-cutting table. More concretely, the row and the column marked in the table are weaved. The sub-graphs of the goals being weaved are also hierarchically weaved following the meaning of AND-OR decomposition, as shown in the Figure 4.

Suppose that the FR graph shown in Figure 1 and the NFR one in Figure 2 are weaved according to the cross-cutting table of Figure 5. The two goals "fault-tolerance" and "recoverability" in the NFR graphs are weaved into "Receiving paper submission". Although the two goals are OR-decomposed in Figure 2, the cross-cutting table in Figure 5 requires that both goals shall be related to a function "receiving paper submission". Thus, the two goals are AND-weaved into the function. As a result, we have the two additional sub-goals "Receiving paper submission with fault tolerance" and "Receiving paper submission with recoverability" in AND composition.

3.4 Weaving Use Case Description

The next issue is how to weave the use cases derived from leaves of goal graphs. A use case model consists of a set of use cases and their dependencies such as control dependency (execution order of the use cases) and data one. Each use case includes interaction relationships to actors and the behavioral scenarios called use case descriptions. In the case of the NFR concerns, what use cases we can have? To achieve some of the NFRs, we have specific behavior scenarios. For example, to achieve high security, we embed the scenarios of encoding data by using cryptography and of decoding them, into the use cases where high security is required. It means that we can consider the use cases of high security as patterns of embedding additional actions such as encoding and decoding. Assume that some of NFRs and their hierarchical decomposition are compliant to ISO9126 (shown in Figure 2). We can prepare how to weave the use cases, as rules of embedding new actions to a use case or use cases, and these rules can be considered as transformation of FR use cases into ones where embedding the new actions.

We capture use case 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 6 summarizes the process of weaving the use case models by using transformation. 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 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 6. In the transformation, a new use case "check", a new actor "C" in Figure 6 and their interactions are also added according to a pattern written in the bottom half in Figure 6. 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 5, 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.

Let's show and 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 7 and 8. Figure 7 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 7 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 8 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 Example and Discussion

In this section, we illustrate how to write and weave multiple viewpoint specifications through a case study.

4.1 Overview of an Intended Business

In this case study, the way to support business persons is discussed. The business persons here mainly make contract with their customers on the customers site. After a contract is made, the business person visits the next customer. Because they cannot easily estimate the spending time to make a contract and there are several business persons in the company, they can not divide their works beforehand. Thus, each business person asks their business office to tell his/her next customer. At the same time, he/she submits the report of a finished contract to the office. Business persons usually have to collect public information, about their business, e.g., stock markets information, so as to succeed their contract. Under such circumstance, the company explores a system of data exchanges among business persons, the business office and public information resources.

4.2 Viewpoints and Cross-Cuttings

Figure 9 shows four viewpoints each of which is written in a goal graph. A viewpoint V1 shows main functionalities related to this system. Only this viewpoint is deeply related to the business supported by an intended system. Viewpoints V2, V3 and V4 are related to the methods for data communication. Each of them shows a non-functional requirement for data communication and alternatives that satisfy the requirement. When we weave these viewpoints,

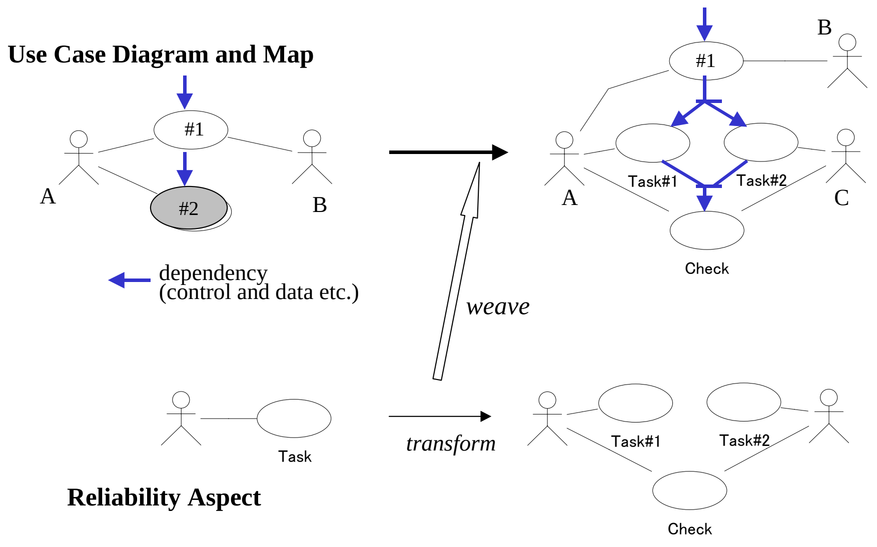
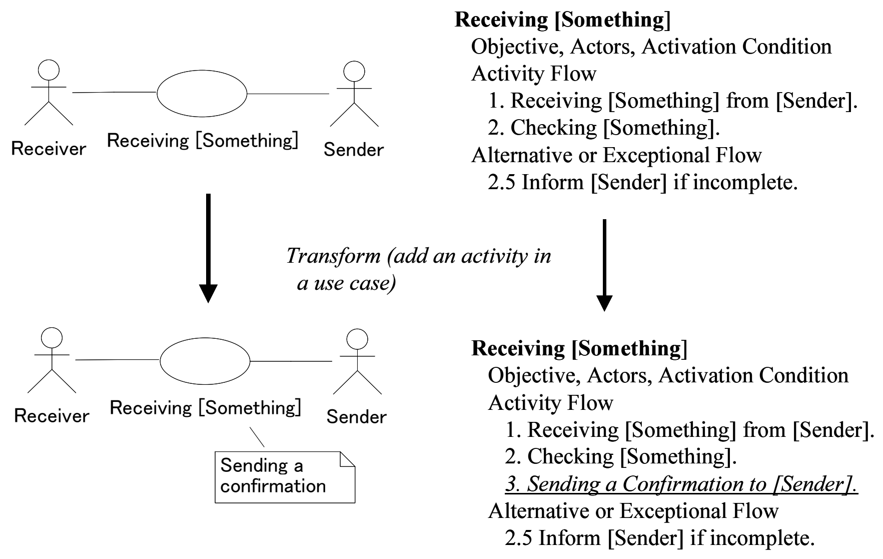


Figure 6. Transformation Based Weaving



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 7. Weaving Use Case Descriptions

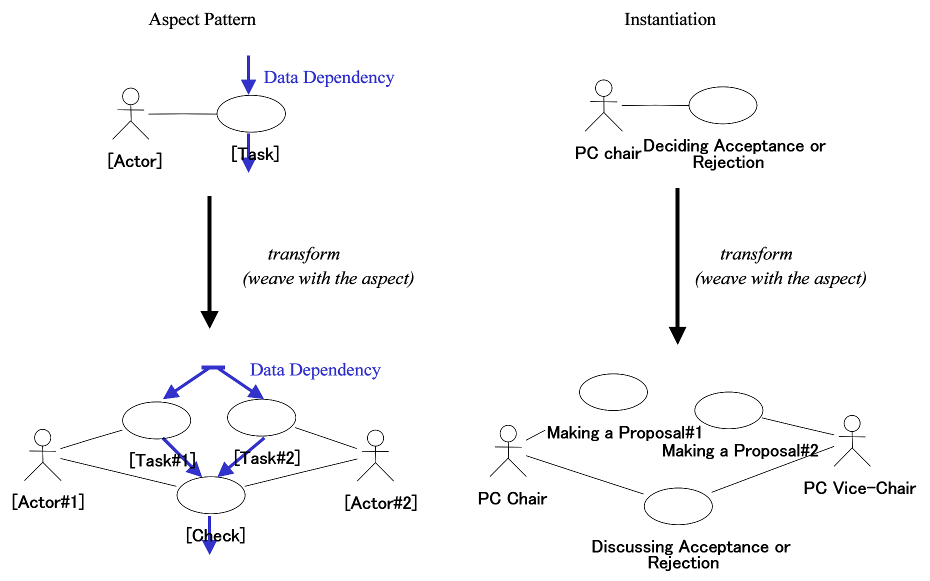


Figure 8. Weaving Use Case Diagram and Maps

there seem to be too many possibilities. At least, all leaf nodes on V1 are related to all other viewpoints because such nodes require communication features. However, we do not need to take care of all combinations because of the features of alternatives in each viewpoint. We can clearly show this fact by using cross-cutting tables in Figure 10.

We explain each cross-cutting table a little bit. So as to collect public information, we need to communicate with any peers. This is the reason why one cell is not checked in a table of V1 and V2. Two cells are not also checked in a table of V1 and V3 because using keys for encryption is not necessary for anonymous communication. In addition, we cannot exchange private keys beforehand with anonymous peers. The functions to report the business results and to get next customers are definitely achieved with specific peers, e.g., their business office. On the other hand, public information could be acquired from anonymous peers. We also define the cross-cutting tables among non-functional viewpoints. A table of V2 and V3 shows encryption is meaningless on a private channel because such a channel is inherently protected from malicious users. A table of V2 and V4 shows that we cannot communicate with anonymous peers on the private channel. The reason of a table of V3 and V4 is the same as the reason of the table of V1 and V3.

4.3 Weaving Viewpoints and Discussion

Figures 11 and 12 show two instances of weaved goal graphs. Thick ovals represent weaved goals in these figures. In both graphs, a goal "exchange contract data" has a subgoal "specific peer" and a goal "collect public information"

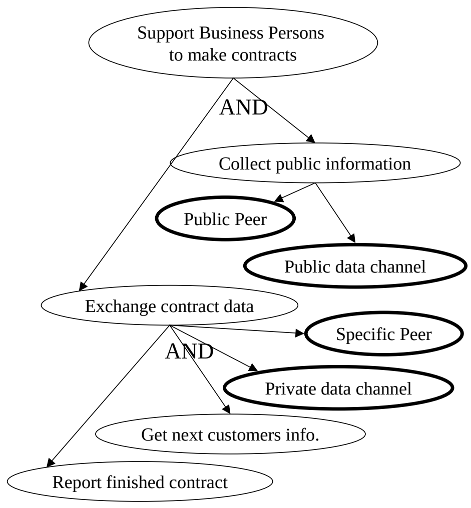


Figure 11. Weaved Goal Graph A

has a sub-goal "public peer" according to the cross-cutting table of V1 and V4. This weaving is fit for our intuition because of the characteristics of data in each case. Although non-functional viewpoints like V2, 3 and 4 should be basically weaved into the leaf goals, we do not do so in Figures 11, i.e. the goal "specific peer" is weaved not into both a leaf goal "report finished contract" and another leaf goal "get next customers info." but into the goal "exchange contract data". We should weave the goal "specific peer" into these two leaf goals formally, but we simplify the goal graphs like in Figures 11 because of the limitation of the space. There are similar simplifications in Figures 11 and 12.

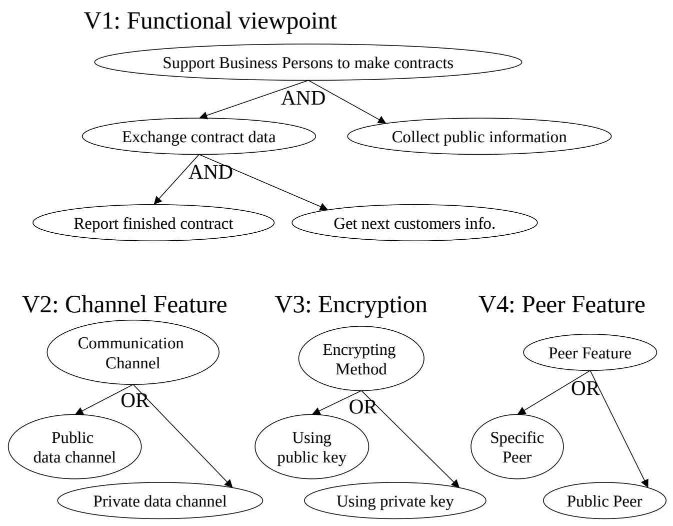


Figure 9. Four viewpoints of a system supporting business persons

V2 \ V1	Report	Get next	Collect
Pub chan.	X	X	X
Pri. chan.	X	X	

V2 \ V3	Pub. key	Pri. key
Pub chan.	X	X
Pri. chan.		

V3 \ V1	Report	Get next	Collect
Pub key	X	X	
Pri. key	X	X	

V2 \ V4	Specific peer	Public peer
Pub chan.	X	X
Pri. chan.	X	

V4 \ V1	Report	Get next	Collect
Specific	X	X	
Public			X

V3 \ V4	Specific peer	Public peer
Pub. key	X	X
Pri. key	X	

Figure 10. Cross-cutting Tables among viewpoints in Figure 9

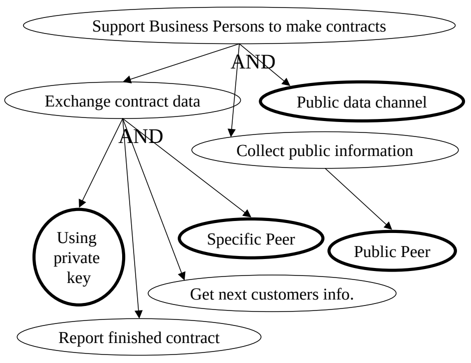


Figure 12. Weaved Goal Graph B

In the case of Figure 11, a goal "private data channel" is weaved to a goal "exchange contract data", and a goal "public data channel" is weaved to a goal "collect public information". According to the cross-cutting table of V2 and V3 and the table of V3 and V4, we do not mind encryption in this case. This shows that the private channel is enough secure for communication. On the other hand, suppose the public data channel should be used all over the business in the case of Figure12. According the cross-cutting table of V2 and V3, we should use encryption mechanism. In this case, we may use both private and public keys, and we use private keys. If peers are not specific, we cannot use private keys according to the cross-cutting table of V3 and V4.

As shown in Figures 11 and 12, we can elicit two different requirements for the same business in the different situation and/or constraints. Because we analyze viewpoints respectively, we can concentrate on each viewpoint. Because we can weave such viewpoints in any time and under any situation, we can elicit requirements for product families simultaneously. Because one viewpoint is weaved into another viewpoint incrementally in this method, we can iteratively elicit requirements for a system. In addition, because we may analyze several viewpoints simultaneously, more than one requirements analyst can cooperatively work together.

5 Conclusion

This paper discusses a technique to weave multiple artifacts in goal-oriented method + use case modeling method for requirements elicitation processes. Weaving can be formalized with logical AND-OR composition for goal graphs and 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 decomposition 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 few examples 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. We should also apply our method to the problems in the real word so as to validate our method. Exploring how to construct class diagrams from use case descriptions is also one of the future work.

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] Annie I. Anton and Colin Potts. The use of goals to surface requirements for evolving systems. In *Proceedings of the 20th International Conference on Software Engineering (ICSE 98)*, pages 157–166, April 1998.

[10] B. Nuseibeh, J. Kramer, and A. Finkelstein. Viewpoints: Meaningful Relationships are Difficult! In *Proc. of 25th International Conference on Software Engineering (ICSE'2003)*, pages 676–681, 2003.

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

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

[13] 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.

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

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

[16] 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.

[17] 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.