PAPER Special Section on Knowledge-Based Software Engineering

Trade-Off Analysis between Concerns Based on Aspect-Oriented Requirements Engineering

Abelyn Methanie R. LAURITO[†*], *Nonmember and* **Shingo TAKADA**[†a)], *Member*

SUMMARY The identification of functional and non-functional concerns is an important activity during requirements analysis. However, there may be conflicts between the identified concerns, and they must be discovered and resolved through trade-off analysis. Aspect-Oriented Requirements Engineering (AORE) has trade-off analysis as one of its goals, but most AORE approaches do not actually offer support for trade-off analysis; they focus on describing concerns and generating their composition. This paper proposes an approach for trade-off analysis based on AORE using use cases and the Requirements Conflict Matrix (RCM) to represent compositions. RCM shows the positive or negative effect of non-functional concerns over use cases and other non-functional concerns. Our approach is implemented within a tool called E-UCEd (Extended Use Case Editor). We also show the results of evaluating our tool.

key words: *use case, AORE, trade-off analysis, concern*

1. Introduction

During the requirements specification phase of software development, stakeholders identify different concerns of a system. A *concern* can be defined as anything of interest or significance to a stakeholder. It can be a *functional* concern, which is a statement of a service the system should provide, or a *non-functional* concern, which is a constraint on the services offered by the system.

When analyzing concerns, the developer will also discover its effects, relationships, and conflicts with other concerns. Issues regarding conflicts need to be clarified by performing trade-off analysis. This is important because when these issues are resolved early in the software development process, a feasible design that satisfies stakeholder needs can be created [10].

Aspect-Oriented Requirements Engineering (AORE) is an approach which has trade-off analysis as one of its goals. AORE is defined as a set of techniques focusing on the separation of crosscutting concerns before the detailed design is derived [13]. A concern is considered to be crosscutting when it affects other functional or non-functional concerns. The basic steps in the AORE process are as follows:

1. Identification and description of concerns
   The initial requirements document or information elicited from stakeholders is used to identify functional and non-functional concerns.
2. Identification of crosscutting relationships
   Crosscutting relationships between concerns are identified.
3. Composition and conflict analysis
   The identified crosscutting relationships are summarized and used as a reference for the next stages of software development and for trade-off analysis.

However, only a few approaches actually explicitly offer trade-off analysis. In some techniques such as [14] and [2], trade-offs are discovered between viewpoints and concerns. Others, such as [17], describe functional and non-functional concerns in the same manner and consider trade-offs by using a formal method. Some AORE approaches that utilize use cases to describe functional concerns, such as [5] and [9], also support description of non-functional concerns but lack a trade-off analysis process. The approach in [20] does not support non-functional concerns but includes the Use Case Editor (UCEd) as tool support for the use case analysis.

In this paper, we propose an approach for supporting trade-off analysis based on AORE. We use use cases as the basic building blocks for representing concerns. Furthermore, we employ Requirements Conflict Matrix (RCM) to show the effects between concerns. RCM is used within the Conflicting Forces method [1] for resolving trade-offs between conflicting requirements.

This paper aims to support the trade-off analysis of the different concerns of the system. The contributions are as follows:

1. Show how the AORE process is used to create a representation for trade-off analysis in the form of the RCM.
2. Provide details on how trade-off analysis is performed using the RCM.
3. Provide tool support for our approach.

The rest of this paper first begins with a review of related work. Section 3 describes our proposed approach. Section 4 describes our tool E-UCEd (Extended Use Case Editor). Section 5 provides the evaluation and observation results. Finally, Sect. 6 makes concluding remarks.

2. Related Work

This section discusses related work and summarizes their issues.

Manuscript received June 29, 2011.
Manuscript revised October 26, 2011.

[†]The authors are with the Graduate School of Science and Technology, Keio University, Yokohama-shi, 223–8522 Japan.
[*]Presently, with Netsuite Philippines, Inc.
a) E-mail: michigan@ics.keio.ac.jp

DOI: 10.1587/transinf.E95.D.1003

2.1 Non-AORE Approaches

The following studies focused on trade-off analysis without using AORE.

The basic is to analyze a requirement in some form and a quality attribute. The approach by Kazman et al. [6] performed trade-off identification between scenarios and quality attributes with high priority. Liu [8] used a matrix called House of Quality, which contains data such as customer requirements and measureable engineering characteristics of the product.

Some approaches considered requirements "levels". Sadana et al. [16] proposed an approach for analyzing conflicts between requirements by going down the requirements hierarchy until low-level description(s) of conflict(s) between function/non-functional requirements are found. Pasternak [12] based his approach on the observation that some non-functional requirements are implemented by functional requirements at a lower level.

Although the actual identification of trade-offs are often done manually, Lee et al. [7] used fuzzy logic to determine conflicting and cooperative relations of requirements.

Finally, Castiglioni et al. [1] proposed the Conflicting Forces method to analyze component models of the system. This method views the different requirements in the system as opposing forces, and uses the *Requirements Conflict Matrix (RCM)* to summarize the effects of non-functional concerns to other concerns.

2.2 AORE Approaches

The following studies are concerned with AORE.

Rashid et al. [14] used viewpoints to represent functional concerns and "concerns" to label non-functional concerns. They are analyzed to discover how the concerns are related to the viewpoints, and summarized in a matrix with viewpoint columns and concern rows. Concerns that affect more than one viewpoint are then identified as candidate aspects. Candidate aspects are given weights to show how much a viewpoint is affected, and to analyze trade-offs.

Chitchyan et al. [2] proposed the Multidimensional Requirements Analysis Tool and the Analytical Hierarchical Process to support resolution of conflicts that occur when composing concerns, specifically semantic conflicts between the concerns themselves and the temporal ordering of compositions.

Work based on use case diagrams include [20] and [9]. Somé et al. [20] focused on functional concerns, and described how a Petri net can be used to simulate the flow of use cases to detect conflicts. Moreira et al. [9] showed non-functional concerns as use cases and included it in use case diagrams.

2.3 Issues

Existing studies on requirements trade-off analysis provide different methods for trade-off analysis using various forms of requirements. Some also classify concerns as functional or non-functional in order to view their relationships. Unfortunately, most studies lack details on how to provide independent descriptions of the requirements that they analyze, especially those concerned with non-functional concerns.

Although one of the goals of AORE is trade-off analysis, in reality most work do not explicitly support it. Those that do support it perform them using a representation such as viewpoints. However, representations such as viewpoints cannot be easily used in the next stages of software development. Widely used representations such as use cases can be used as a better representation for functional concerns.

AORE approaches that utilize use cases either do not support non-functional concerns [20] or supports non-functional concerns by depicting it as a use case [9]. The latter approach is an issue because a non-functional concern itself cannot become a use case according to the UML definition ("means for specifying required usages of a system" [11]). Thus, the omission of non-functional concerns in use case based AORE leads us to the idea to propose a process that describes use cases, non-functional concerns, and their relationships. We consider this process to be a proper representation for trade-off analysis.

3. An Approach for Trade-Off Analysis Using AORE

This section proposes an approach to perform trade-off analysis based on the AORE process. Through our approach we create a representation for trade-off analysis in the form of the Requirements Conflict Matrix (RCM). We explain how to describe the concerns and create the RCM. We also include the details of how to perform trade-off analysis using the RCM.

3.1 Methodology

AORE is incorporated in our approach by considering the crosscutting relationship between functional concerns and non-functional concerns. The developer has the task of identifying both functional and non-functional concerns, as well as where or which functional concerns that a non-functional concern crosscuts. This is analogous to a programmer identifying a base component and an aspect as well as the join point in aspect-oriented programming. We virtually weave them together by showing the crosscutting relationship within the RCM. In our proposed tool, we also virtually weave them together and see how they affect each other with a simulator (described in Sect. 4.4).

The basic steps in our methodology is as follows:

1. Identify and describe concerns.
2. Create RCM.
3. Trade-off analysis with RCM.

Figure 1 shows how our methodology maps to the AORE process.

3.1.1 Identifying and Describing Concerns

The first step in our methodology is the identification of functional and non-functional concerns. We employ use cases to describe functional concerns as use cases are widely used. Details of a use case is based on Cockburn's format [3], which includes primary actor, system to be developed, goal, and steps, which are the actions performed by the actor and system.

Non-functional concerns are abstract concepts that can be further described by measurable non-functional requirements. Examples of non-functional concerns can be found in the IBM FURPS+ model [4]. Non-functional requirements under the non-functional concerns are described by system constraints and quality metrics. For example, a non-functional concern is *Implementation* and a non-functional requirement that describes it is *Network Speed*.

3.1.2 Creating the Requirements Conflict Matrix

After describing the concerns, the Requirements Conflict Matrix (RCM) is created to summarize their relationships. This step corresponds to two steps in the AORE process – identification of crosscutting relationships and composition (Fig. 1).

In our approach each row in the RCM represents a use case or a non-functional requirement, while each column represents a non-functional requirement. Each cell in the matrix indicates the effect (positive, negative, or neutral) of the non-functional requirements in the columns to the use cases and non-functional requirements in the rows. The value of each cell in the matrix is decided based on the first two steps in the Conflicting Forces method [1]:

1. Identify conflicts between use cases and non-functional requirements.
2. Identify conflicts among non-functional requirements.

First, we consider the relationship between use cases and non-functional requirements. A non-functional requirement crosscuts a use case if it has an effect on its steps or general goal. We use the following rules to decide the cell values (and color) that represent the relation between a non-functional requirement and a use case:

- If the use case is likely to be negatively affected by the non-functional requirement, place a *"−1"* in the cell and change the color to red.
- If the use case is likely to be positively affected by the non-functional requirement, place a *"1"* in the cell and change the color to green.
- If the use case is likely to be not affected by the non-functional requirement, place a *"0"* in the cell and do not change its color.

Next, we consider the relationships among non-functional requirements using the following rules:

- If a non-functional requirement in the column is the same as the non-functional requirement in the row, leave it blank and change the color to gray.
- If a non-functional requirement degrades or conflicts with another non-functional requirement, the effect is negative. Place a *"−1"* in the cell and change the color to red.
- If a non-functional requirement supports or enhances another non-functional requirement, the effect is positive. Place a *"1"* in the cell and change the color to green.
- If a non-functional requirement does not affect another non-functional requirement, there is no relationship between them. Place a *"0"* in the cell and do not change its color.

Note that we only use the integers −1, 0, and 1 to represent the relationship. In future work, we are considering using a wider range to show that for example some relations are more important than others.

Figure 2 shows an example of an RCM. This example is based on the AmGro-from-Home case study described in [1]. The *"−1"* cell value for the *Network infrastructure* non-

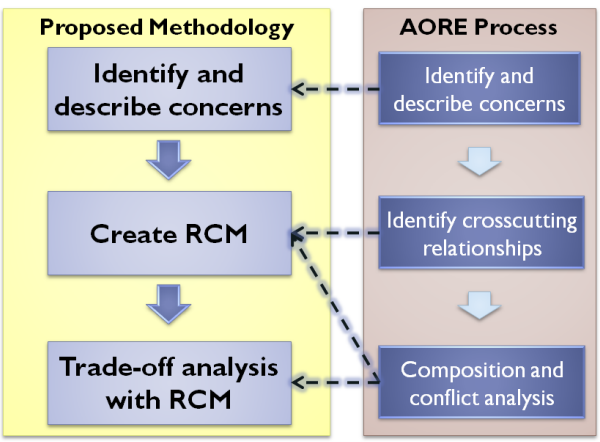

Fig. 1 Proposed methodology and AORE process.

	Performance	Implementation				
	Response times	Network infrastructure	Data volatility (Orders, customer)	Data volatility (Products)	Unscheduled outages	Developed in Java
Use Cases:						
Create order	-1	-1	-1	1	-1	0
Submit order	-1	-1	-1	0	-1	0
Non-functional Requirements:						
Performance						
Response times		-1	0	1	0	0
Implementation						
Network infrastructure	0		0	0	0	0
Data volatility (Orders, custon	0	-1		0	0	0
Data volatility (Products)	0	0	0		0	0
Unscheduled outages	0	0	-1	0		0
Developed in Java	0	0	0	0	0	

Fig. 2 Requirements Conflict Matrix (RCM).

functional requirement column and the *Create order* use case row is interpreted as: *"Network infrastructure has a negative effect on Create order"*. This is considered to be negative because the network bandwidth allocated for the system is limited and it may cause problems for the *Create order* use case.

3.1.3 Trade-Off Analysis Process

The approach for trade-off analysis is based on the third step of the Conflicting Forces method [1]: *"Analyze the consequences of the stress case on the components involved in the use cases execution."* Basically, this entails the developer to conduct the following:

1. Identify stress cases.
2. Identify points for resolution based on the effect of each conflicting requirement.

The completed RCM is first used to identify stress cases. A *stress case* is a row in the matrix that has at least one conflicting pair of non-functional requirements, which in our case is indicated by *"−1"*. Each stress case has the following:

- a source requirement: requirement in the row being affected by the non-functional requirements
- conflicting requirements: set of non-functional requirements that negatively affect the source requirement

Several stress cases exist in our RCM example (Fig. 2). For example, the source requirement *Create Order* has the following conflicting requirements: *Network infrastructure*, *Data volatility (Orders, customer)*, *Response time*, and *Unscheduled outages*.

After identifying the stress cases, the effect of each conflicting requirement to the source is analyzed to identify points for resolution. A point for resolution is a question to be clarified with the stakeholders, and there are three types:

- *Questions on trade-offs between non-functional requirement and use case*: These are questions about the negative effect of non-functional requirements to use cases. For example, for *Unscheduled outages*, we may consider *"How should the system handle unscheduled outages?"*
- *Questions on trade-offs among non-functional requirements*: These are questions about issues on the prioritization of conflicting requirements. Conflicting requirements may have a trade-off when the value of their relation in the RCM is negative. For example, among the conflicting requirements in Fig. 2, *Network infrastructure* negatively affects *Data volatility (Orders, customer)*. The reason behind this is that delays or failures in the network may affect data modification. This trade-off may impact the use case *Create order*. A point for resolution that can be added for this trade-off is: *"What should be done to assure data consistency in case of failure in the modification because of the network?"*
- *General questions*: Aside from questions about the trade-offs, questions about discrepancies in the concern descriptions can also be used as points for resolution.

4. Implementation

We implemented a tool called Extended Use Case Editor (E-UCEd) that supports our approach. E-UCEd is extended from the Use Case Editor (UCEd) [20] to support the description of non-functional concerns and their relationships to other concerns. The descriptions are used to generate the RCM for the trade-off analysis between non-functional concerns and use cases.

UCEd was originally proposed in [18] as a tool to support use case based requirements engineering and only handles functional concerns. It requires use cases to be written in a restricted natural language with formal semantics. A *normal* use case includes description of a complete interaction sequence, while an *extension* use case lists additional behavior for a normal use case. UCEd can parse and validate use case descriptions, as well as generate a Petri net to represent the flow of use cases [19] and conduct use case simulation.

4.1 Architecture

Our E-UCEd supports non-functional concerns by providing a user interface to describe them and their relationships with other concerns. Composition is performed by generating the following based on the descriptions:

- *Requirements Conflict Matrix (RCM)* with positive or negative values that indicate relationships among non-functional requirements and blank cells that indicate relationships between non-functional requirements and use cases.
- *Petri net with non-functional concern information*.
- *Use case simulation with non-functional requirements* based on the Petri net.

We added the *NFR Editing Tool* and *RCM Generator* to support non-functional concerns. Aside from these modules, we also modified the UCEd's *Parser*, *Petri Net Generator*, and *Simulator*. The architecture of our E-UCEd tool is shown in Fig. 3. The following subsections will discuss the main modules.

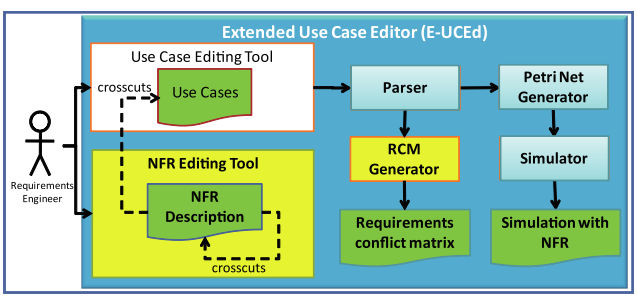

Fig. 3 Extended use case editor architecture.

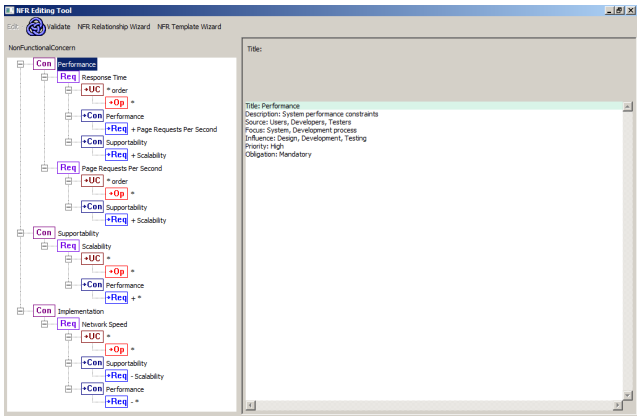

Fig. 4 NFR editing tool.

4.2 NFR Editing Tool

As we noted in Sect. 2.3, most studies lack details on how to provide independent descriptions of the requirements that they analyze, especially those concerned with non-functional concerns. Our E-UCEd provides the *NFR Editing Tool* (Fig. 4) to identify and describe non-functional concerns and their relationships with other concerns. The left side shows the non-functional concerns, requirements and crosscutting relationships, while the right side shows details of the chosen element.

Elements on the left side include the following:

- *Non-Functional Concern (Con)*: This is further described with fields based on work by Moreira, et al. [9], such as *Title* and *Description*.
- *Non-Functional Requirement (Req)*
- *Related Use Case (→UC)*: This is the name of the use case that is affected by the non-functional requirement. The reason behind the relationship is explained in the *Relation Reason* field.
- *Related Operation (→Op)*: These are the specific steps affected under *→UC*.
- *Related Non-Functional Concern (→Con)*: This is a non-functional concern that is affected by the non-functional requirement.
- *Related Non-Functional Requirement (→Req)*: This is a non-functional requirement that is actually affected under for a specified *→Con*. A "+" or "−" indicates the positive or negative effect of the affecting non-functional requirement.

Crosscutting relationships between concerns is specified under *Req* by indicating the name of the affected concern and its specific requirements. The wildcard character "*" can also be used for pattern matching to indicate multiple concerns. In other words, the wildcard character is used to show that a non-functional concern crosscuts multiple concerns.

For example, Fig. 5, which shows details of the lower half of the left side of Fig. 4, describes the *Implementation* concern, which explains the implementation constraints of

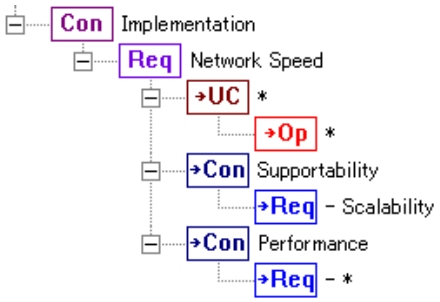

Fig. 5 Non-functional concern description.

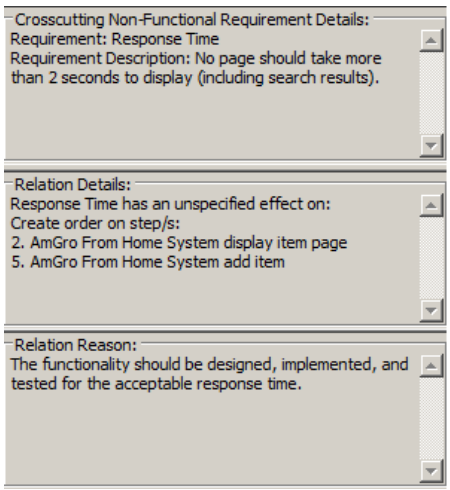

Fig. 6 Information for selected RCM cell.

the system. The *Network Speed* requirement is a measurable constraint, and affects all use cases (→UC*) and operations (→Op*). It also negatively affects the *Scalability* requirement under the *Supportability* concern and all requirements under the *Performance* concern.

The *NFR Editing Tool* provides templates for non-functional concerns that are based on the FURPS+ model [4]. It also has a wizard for customizing reusable non-functional concern descriptions (*NFR Template Wizard*) and a wizard for specifying relationships under a *Req* element (*NFR Relationship Wizard*).

4.3 RCM Generator

The *RCM Generator* automatically generates an RCM based on the requirements and use cases that were specified with the *NFR Editing Tool*. When a cell in the matrix is selected, relation information is displayed in the *Crosscutting Non-Functional Requirement Details*, *Relation Details*, and *Relation Reason* panels (Fig. 6).

It also automatically determines the cell values for the relation between non-functional requirements based on the value specified in the *→Req* description. The value is *"1"* if the description is *"+"*, *"−1"* if the description is *"−"*, *"0"* if there is no relation specified, and blank if the non-functional requirement in the column is the same as in the

	Performance		Supportability	Implementation
	Response Time	Pages Per Second	Scalability	Network Speed
Use cases:				
Create order	-1	-1	-1	-1
Submit order	-1	-1	-1	-1
Non-functional concerns:				
Performance				
Response Time		0	1	-1
Pages Per Second	1		1	-1
Supportability				
Scalability	1	1		-1
Implementation				
Network Speed	0	0	0	

Fig. 7 Completed RCM in E-UCEd.

row. The color of the cell is set according to the value.

The cell values that represent the relationship between a use case and non-functional requirements are manually assigned by referring to the *Crosscutting Non-Functional Requirement Details*, *Relation Details*, and *Relation Reason* panels (Fig. 6). Figure 7 shows a completed RCM.

4.4 Simulator

Our E-UCEd also simulates a use case with non-functional requirements. This can help when deciding the values for the relationship of non-functional requirements to use cases in the RCM. The original simulation feature in UCEd was modified to include information about non-functional requirements.

Simulation is performed in the *Simulator* (Fig. 8). When a user selects an actor event (left side of Fig. 8), the non-functional concerns and requirements that crosscut the resulting system reactions are displayed in the *Related non-functional concern(s)* panel (lower right panel in Fig. 8; also shown in Fig. 9). By using this information, the value for the blank input fields in the RCM can be decided based on how the non-functional requirement crosscuts the use case based on actor events. This can also be used for the trade-off analysis, which will be described at the end of the next section.

4.5 Trade-Off Analysis with E-UCEd

As we described in Sect. 3.1.3, the two basic steps in the trade-off analysis are as follows:

1. Identify stress cases.
2. Identify points for resolution based on the effect of each conflicting requirement.

Figure 10 summarizes the trade-off analysis process.

First, the identification of stress cases is done automatically. A stress case summary is generated in HTML format from the RCM window (Fig. 11). It describes each stress case in the RCM by providing the name of the source requirement and a table of its conflicting requirements. The conflicting requirements table contains the name of each requirement that has a negative effect on the source requirement and the description of its effect. This description is based on the value specified for the Relation Reason field when the relationship was added.

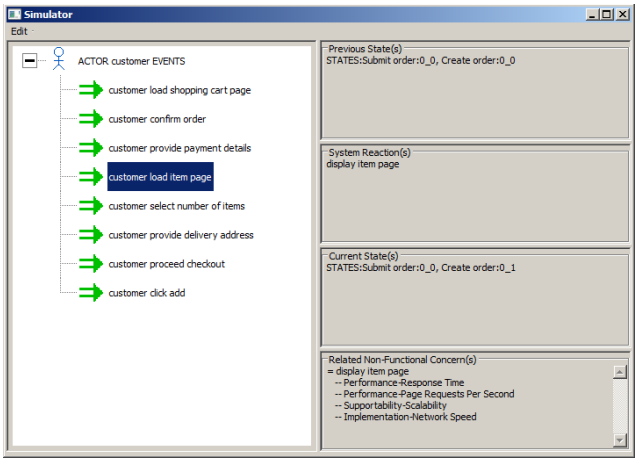

Fig. 8 Simulator.

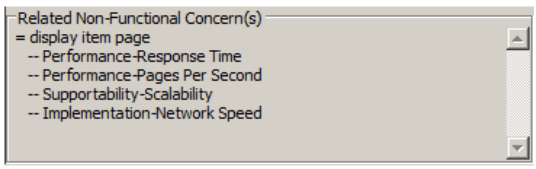

Fig. 9 *Related Non-Functional Concern(s)* panel in simulator.

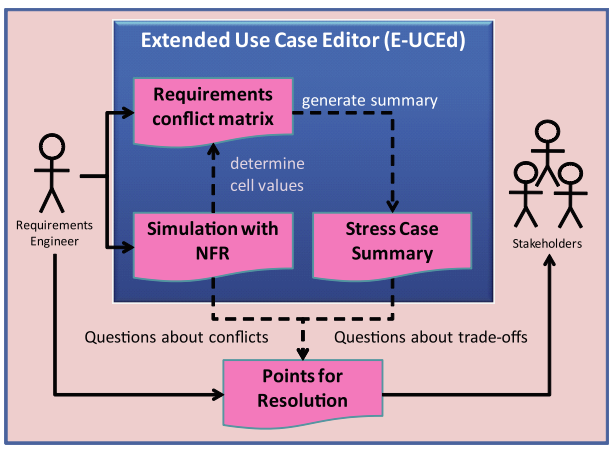

Fig. 10 Trade-off analysis process with E-UCEd.

Source Requirement: Create order

Conflicting Requirements:

Name	Effect
Scalability	Scalability needs to be considered in design and implementation.
Network Speed	Existing infrastructure unlikely to allow orders to be submitted 24x7.
Page Requests Per Second	The functionality should be designed, implemented, and tested for the acceptable pages per second.
Response Time	The functionality should be designed, implemented, and tested for the acceptable response time.

Fig. 11 Stress case summary.

Next, the developer identifies the points for resolution using the stress case summary. Each stress case is considered one by one to see if it can be considered as a point for resolution based on the three types that were given in Sect. 3.1.3. For example, the first line in Fig. 11 concerning

scalability can be considered as a question on trade-off between a non-functional requirement and a use case. Specifically, the point for resolution could be considered as "How should we consider scalability in design and implementation for the Create Order requirement?"

Finally, the developer can also use conflicts found with the use case simulation as a point for resolution. Since our approach adds the Related Non-Functional Concerns for each actor event, points for resolution between non-functional requirements and other concerns can be interpreted. For example, Fig. 8 shows the simulation result when a user selects the actor event "customer load item page". This event occurs when the customer (actor) clicks on a link for an item offered by the company. From the Related Non-Functional Concerns, we can list the question: "How much information should be displayed on the page to support the metric for Response Time and Page Requests per Second?" A user can also detect conflicts between concerns that occur when an actor event is performed. Figure 8 shows that the event results in the System Reaction "display item page", which displays the information for the item selected by the customer such as its features, quantity left, and price. A point for resolution that can be listed for this sequence is "If another customer orders the same item in the page, will the quantity of available items be changed during page load?"

In sum, our approach and tool supports developers by providing information in the form of stress case summary and simulation which can be used during trade-off analysis. We believe this to be an improvement over existing approaches where trade-off analysis is conducted based on information in the form of a matrix such as RCM.

5. Evaluation

This section evaluates and discusses our tool.

5.1 Evaluation Method

Evaluation was performed with five English-speaking test subjects: two were software developers and three were graduate students with professional software development experience. Their development experience ranged from two to eight years, but most of the subjects did not have much professional experience with non-functional requirements and trade-off analysis.

The subjects were first given a tutorial of our E-UCEd tool and were given time to get used to it. They were also asked to manually conduct trade-off analysis with a *Hotel Management System* [5] to help understand the process. They were then provided with a problem statement and a description of functional concerns (eight use cases) of a *Cafeteria Ordering System* [21]. Functional concerns were provided because we wanted to focus on non-functional concerns. They identified and described non-functional concerns using the E-UCEd to generate the RCM and filled the blank cells. Then they used the completed RCM to identify points for resolution. The specific steps and identified artifacts are as follows:

1. Read and understand the problem statement.
2. Identify and describe non-functional concerns.
3. Identify crosscutting relationships and complete the RCM.
4. Identify points for resolution.

We recorded questions, comments, problems encountered and identified artifacts.

Table 1 Identified number of requirements.

Subject ID	# Reqs
Developer 1	5
Developer 2	9
Student 1	4
Student 2	11
Student 3	2

5.2 Analysis of Results

(1) Identify and describe non-functional concerns

Table 1 shows the number of requirements that each subject identified. Some of the identified requirements were as follows:

- Performance – Response Time: "Responses to queries shall take no longer than 7 seconds to load onto the screen after the user submits the query."
- Reliability – Availability: "The Cafeteria Ordering System shall be available to users on the corporate Intranet and to dial-in users 99.9% of the time between 5:00 A.M. and midnight local time and 95% of the time between midnight and 5:00 A.M. local time."

Three out of the five subjects identified requirements just from the problem statement, while the other two subjects identified requirements based on their own ideas for the system.

Subjects commented that the use of the available non-functional concern description templates helped in their analysis.

(2) Identify crosscutting relationships and RCM creation

Figure 12 shows the percentage of RCM cells with values filled in. Although only an average of 16% of the cells were given a value, this differed between subjects. The reason for this was because there were few commonly identified non-functional requirements.

The description of each relation was helped by the *NFR Relationship Wizard*. They also found it easy to generate the RCM and to fill in the values by referring to the information for each cell. However, some subjects had problems on deciding the correct value of the relationship between non-functional requirements and use cases due to lack of experience.

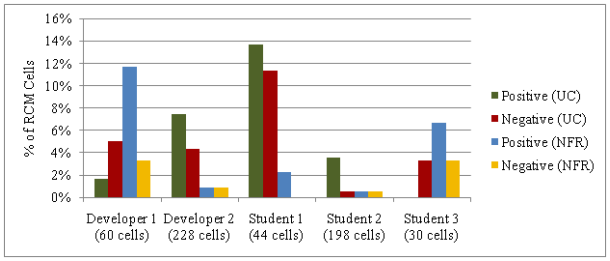

Fig. 12 RCM cells with values.

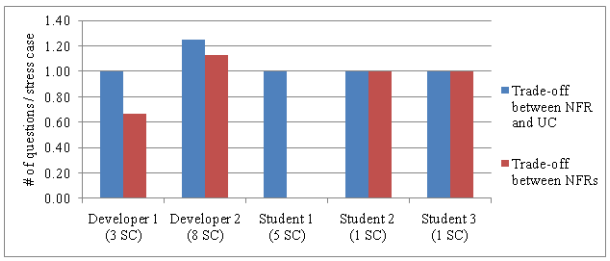

Fig. 13 Points for resolution per stress case.

(3) Identify points for resolution

Figure 13 summarizes the average number of points for resolution identified by each subject for each stress case. On average, 3.6 use cases were considered as a stress case from the RCM out of the 8 use cases.

Most subjects had similar results finding about one question for each type of trade-off. However, *Student 1* was not able to identify any points for resolution about trade-offs between non-functional requirements because unlike the others subjects, *Student 1* used requirements not found in the problem statement.

Stress cases were not considered a problem in this step since they were already summarized in an HTML file. Subjects commented that the summary helped, but they still found it difficult to think of points for resolution.

5.2.1 Level of Difficulty

The subjects also gave a rating for each step in the process by answering a questionnaire. Each question was to be answered with a rating from 1 (very easy) to 5 (very difficult). Figure 14 shows the average rating of the subjects for the difficulty of each step. The *Manual Analysis* results is the rating from the manual analysis for the *Hotel Management System* while the *Tool Analysis* is the rating based on using our tool.

The difficulty of the identification and description of non-functional concerns for both manual and tool analysis was on a *Moderate* level. The identification of crosscutting relationships, RCM creation, and identification of points for resolution were given a *Difficult* rating during manual analysis but only considered as *Moderate* during tool analysis. This indicates that our tool helps, but at the same time experience and skill in trade-off analysis is needed.

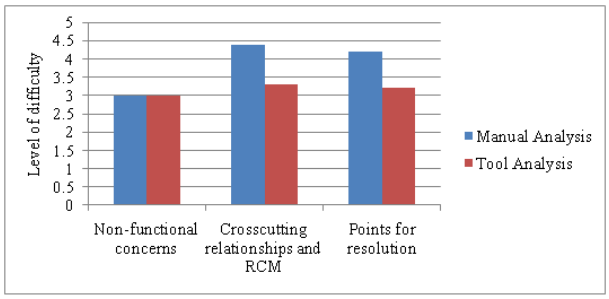

Fig. 14 Difficulty of each step in manual and tool analysis.

5.2.2 Discussion

Our proposed approach uses aspect-oriented techniques that are more commonly available for analysis between functional requirements and applied it to analyze conflicts between use cases and non-functional requirements. Through the support tool, these relationships are summarized and representations that guide in listing the points for resolution for trade-off analysis can be generated. These representations support the process of analyzing trade-offs by:

1. Helping the user understand what type of information is necessary for trade-off analysis;
2. Displaying information in a way that helps developers consider the trade-offs.

The subjects commented that the NFR Relationship Wizard made the description of the relationships easier and the stress case summary helped in the identification of points for resolution.

On the other hand, our tool supports trade-off analysis and does not automatically conduct the analysis itself. It guides the developer when inputting information for trade-off analysis, and also displays information in a way that helps the developer conduct trade-off analysis. This means that, as with any CASE tool, experience is important. In our case, experience and skill in trade-off analysis is very important to produce better points for resolution.

For example, if developers have different ideas on how to identify non-functional requirements and indicating their effect on other concerns, this would affect the resulting analysis, because these are the input to our tool (through the NFR Editing Tool). In our evaluation, most of the requirements identified were performance, usability, reliability, and security concerns for use cases. Design requirements, implementation requirements, and supportability, which are usually addressed during architecture design, were not given enough attention. This was because the subjects were used to relating the constraints to use cases since their experience was more on software design and implementation.

Furthermore, even though subjects were able to produce points for resolutions, they still had a difficult time thinking of questions about trade-offs and this led to less time to list questions about use case conflicts.

6. Conclusion

This study proposed an approach for trade-off analysis between different concerns in the system. Our approach is based on AORE and used RCM to represent how functional and non-functional concerns relate to each other. The developer uses the RCM to identify stress cases and gather points for resolutions to be used in the trade-off analysis with the stakeholders. We implemented our approach in the E-UCEd tool to support the description of non-functional concerns, generation of RCM, simulation with non-functional concerns, and identification of stress cases.

Compared with previous techniques, our approach and tool have the following advantages:

- We took RCM and considered it from an AORE viewpoint, i.e., how the functional and non-functional concerns relate (or crosscut) each other.
- We provide a means (NFR Editing Tool) for describing non-functional requirements. Information provided with the NFR Editing Tool is used to automatically fill in the RCM cells as well as provide the basis for the simulator.
- Trade-off analysis is supported by automatically providing a summary of stress cases, as well as a simulator to see how concerns affect each other.

Our evaluation showed that our tool helps to a certain extent. However, experience and skill in trade-off analysis are necessary to produce better points for resolution.

For future work, we are considering how values other than -1, 0, and 1 can be used in the RCM cells. Another topic is how the tool can be extended to support group analysis.

References

[1] F. Castiglioni and P. Cripps, "Analyze requirements for complex software systems in a new, holistic way," IBM developerWorks Library, 2009.
[2] R. Chitchyan, A. Rashid, R. Waters, I. Brito, A. Moreira, and J. Araujo, "Requirements-level aspectual trade-off analysis approach," AOSD-Europe deliverable, ULANC-28, 2007.
[3] A. Cockburn, Writing Effective Use Cases, Addison-Wesley Longman Publishing, 2000.
[4] P. Eeles, "Capturing architectural requirements," IBM developerWorks Library, 2005.
[5] I. Jacobson and P. Ng, Aspect-Oriented Software Development with Use Cases, Addison-Wesley Professional, 2004.
[6] R. Kazman, M. Klein, and P. Clements, "ATAM: Method for architecture evaluation," Carnegie Mellon University Software Engineering Institute, Technical Report CMU/SEI-2000-TR-004, 2000.
[7] J. Lee and J. Kuo, "New approach to requirements trade-off analysis for complex systems," IEEE Trans. Knowl. Data Eng., vol.10, no.4, pp.551–562, 1998.
[8] X.F. Liu, "Software quality function deployment," IEEE Potentials, vol.19, no.5, pp.14–16, 2000.
[9] A. Moreira, J. Araujo, and I. Brito, "Crosscutting quality attributes for requirements engineering," Proc. 14th International Conf. on Software Engineering and Knowledge Engineering (SEKE '02), pp.167–174, 2002.
[10] D. Oliver, T.P. Kelliher, J.G. Keegan Jr, Engineering Complex Systems with Models and Objects, McGraw-Hill, 1997.
[11] Object Management Group, OMG Unified Modeling LanguageTM (OMG UML) – Superstructure, version 2.3, 2010.
[12] T. Pasternak, "Using trade-off analysis to uncover links between functional and non-functional requirements in use-case analysis," IEEE International Conf. on Software Science, Technology and Engineering (SwSTE03), pp.3–9, 2003.
[13] A. Rashid, P. Sawyer, A. Moreira, and J. Araujo, "Early aspects: a model for aspect-oriented requirements engineering," Proc. 10th Anniversary IEEE Joint International Conf. on Requirements Engineering (RE '02), pp.199–202, 2002.
[14] A. Rashid, A. Moreira, and J. Araujo, "Modularisation and composition of aspectual requirements," Proc. 2nd International Conf. on Aspect-Oriented Software Development (AOSD '03), pp.11–20, 2003.
[15] A. Rashid and R. Chitchyan, "Aspect-oriented requirements engineering: A roadmap," Proc. 13th International Workshop on Early Aspects (EA '08), pp.35–41, 2008.
[16] V. Sadana and X.F. Liu, "Analysis of conflicts among non-functional requirements using integrated analysis of functional and non-functional requirements," Proc. 31st Annual International Computer Software and Applications Conf. (COMPSAC '07), pp.215–218, 2007.
[17] E. Soeiro, I.S. Brito, and A. Moreira, "An XML-based language for specification and composition of aspectual concerns," Proc. 8th International Conf. on Enterprise Information Systems: Databases and Information Systems Integration (ICEIS '06), pp.410–419, 2006.
[18] S. Somé, "Supporting use case based requirements engineering," Information and Software Technology, vol.48, no.1, pp.43–58, 2006.
[19] S. Somé, "Petri nets based formalization of textual use cases," SITE - University of Ottawa, Technical Report TR-2007-11, 2007.
[20] S. Somé and P. Anthonysamy, "An approach for aspect-oriented use case modeling," Proc. 13th International Workshop on Early Aspects (EA '08), pp.27–34, 2008.
[21] K. Wiegers, Software Requirements: Practical Techniques for Gathering and Managing Requirements Throughout the Product Development Cycle, Microsoft Press, 2003.

Abelyn Methanie R. Laurito is currently a software architect at Netsuite Phillippines, Inc. She received her B.S. in Computer Science from University of Santo Tomas in 2003. She received her M.E. in Computer Science from Keio University in 2010. From 2003 to 2008, she was a software engineer at NEC Telecom Software Philippines, Inc.

Shingo Takada is currently an associate professor at the Department of Information and Computer Science, Keio University. He received his B.E. in Electrical Engineering, and M.E. and Ph.D. in Computer Science from Keio University, in 1990, 1992, and 1995, respectively. From 1995 to 1999, he was a Research Associate at Nara Institute of Science and Technology. His interests include software engineering and service orientation. He is a member of IPSJ, ACM, and IEEE-CS.