Identifying Stakeholders and Their Preferences about NFR by Comparing Use Case Diagrams of Several Existing Systems

Haruhiko Kaiya   Akira Osada   Kenji Kaijiri
Dept. of Computer Science, Shinshu University
4-17-1 Wakasato, Nagano 380-8553, Japan
kaiya@cs.shinshu-u.ac.jp   csada@cs.shinshu-u.ac.jp   kaijiri@cs.shinshu-u.ac.jp
`http://www.cs.shinshu-u.ac.jp/~kaiya/`

Abstract

We present a method to identify stakeholders and their preferences about non-functional requirements (NFR) by using use case diagrams of existing systems. We focus on the changes about NFR because such changes help stakeholders to identify their preferences. Comparing different use case diagrams of the same domain helps us to find the changes that can occur. We utilize the Goal-Question-Metrics (GQM) method to identify variables that characterize NFR. Thus, we can systematically represent changes about NFR using the variables. The use cases that represent system interactions help us to bridge the gap between goals and metrics (variables). Thus, we can easily construct measurable NFR. In order to illustrate and evaluate our method, we applied our method to an application domain of the Mail User Agent (MUA) system.

Keywords

Requirements Elicitation, Non-Functional Requirements (NFR), GQM, Stakeholders and their Preferences, Use Case Diagrams.

1. Introduction

We present a method to identify stakeholders and their preferences about non-functional requirements (NFR) by using use case diagrams of the existing systems. The result of our method can be used to prioritize NFR, for example, in order to maximize the preferences of all stakeholders. Since the policy on prioritizing NFR depends on each project, we do not handle such policies in this method.

In the field of requirements engineering, stakeholders are regarded as 'all those who have a stake in the change being considered, those who stand to gain from it, and those who stand to lose from it' [15]. Therefore, 'a stakeholder is much more than a product's eventual user' [3].

It is important to understand the stakeholders' preferences when eliciting the requirements. The preferences of stakeholders are largely related to NFR and/or quality requirements. On the other hand, the functions of a system are fundamental characteristics and they are relatively stable. For example, an airplane without the flying function is NOT an airplane. This is not a matter of preference. In case of airplanes, one airplane may fly with a high speed and another may fly safely with a low speed. Here, both perform the function of an airplane; however, speed is a matter of preference. One person may prefer high speed, but the same may not hold true for another person.

In general, it is not easy to decide whether or not one prefers one of the NFR; however, one can easily decide one's preference when one of the NFR is changed. For example, we can clearly state the preference for performance requirement when an airplane flies faster than before. Thus, we focus on the changes about NFR in our research.

Changes about NFR can be characterized by the changes of variables. In the case of an airplane's performance, its speed can be a variable. Such variables characterize more than one NFR in general. Therefore, changes in such variables can help stakeholders discover other unidentified NFR.

Goal-Question-Metrics (GQM) [4] is a method to identify the data that can be used to understand how a system achieves its goals. We utilize GQM to represent the changes in NFR because the metrics in GQM can be regarded as measurable and changeable variables for NFR.

Since NFR, or in other words conditions, should be measurable in well-formed requirements [1], such variables should contribute to the construction of well-formed requirements. Soft goals [6], which are not measurable, are also important in capturing NFR. However, stakeholders cannot decide their preferences only by referring to soft goals.

In GQM, deriving metrics from goals via questions is not easy because goals are relatively more abstract than metrics. This is one of the difficulties encountered in the GQM method. In this research, we use requirements specifications of existing systems as hints to bridge the gap between goals and metrics. Since such specifications include system interactions [9], it is easy to ascertain metrics that have to be focused on. Goals can be also assumed from such specifications, and we do not have to be concerned with whether the derived goals are the goals of actual users and developers, because we do not focus on reverse engineering but on requirements elicitation. Using such existing systems, we can easily validate and/or confirm system functionalities and/or non-functionalities.

We employ use case diagrams in UML to represent the existing systems. The first reason is that, currently, use case diagrams are the defacto standard to represent functional requirements. The second reason is that the diagrams use the concept of actors, which can be the first approximation of stakeholders. The third reason is that there are several techniques to construct use case diagrams and other related diagrams from the existing systems in the field of reverse engineering [18], [11].

In Section 2, we present concrete steps to carry out our method for eliciting NFR, stakeholders and their preferences. An example that uses our method is shown in Section 3. Finally, we summarize our current results and identify future work required.

2. Method

2.1 Terminologies and Notions

We introduce terminologies and notions for our method before introducing the procedure.

Use case diagrams and use cases are the same as those in UML.

System interactions and **user goals** are mentioned in [9]. These are two different styles in which the use case represents functionality. When use cases are written as system interactions, they represent the manner of interacting with the system. On the other hand, use cases represent the goals that are achieved when they are written as user goals.

Since our method starts with the existing systems, we use the system interaction style. It is easy to write in such a style because the systems already exist and we may execute them if needed.

Similarity of applications are basically decided subjectively. However, the number of similar actors and similar use cases will help in determining it. We may refer to pre-existing categories for applications in shops, catalogs, and/or web sites. When several applications are similar to each other, such a set of applications is called an application domain.

Similarity of actors and of use cases may be decided subjectively. However, words used in the specification of actors and use cases help decide it. In addition, use cases related to the same kinds of actors could be similar to each other.

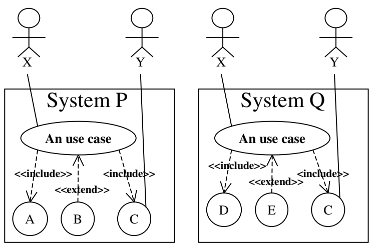

Figure 1. Example of Differences between Two Surroundings

The surrounding of a use case is a set of use cases directly connected to the use case by extend- or include-relationships and a set of actors directly connected to both the use case and connected use cases.

Differences among the surroundings of a use case consist of the use cases that are not included in the intersection of the surroundings.

In Figure 1, we have a small example of differences between two surroundings. We focus on a use case and identify the surroundings of the use case, one is in the system P and the other in system Q. In system P, it consists of actor X and Y and the use cases A, B, and C. In system Q, it consists of actors X and Y and the use cases D, E, and C. Therefore, the differences between the surroundings of the use case are use cases A, B, D, and E.

NFR taxonomy is a catalog of non-functional requirements in general. Currently, we use the software quality attributes provided in the ISO standard for software quality [12] and NFR types [6]. We will briefly introduce the contents of NFR taxonomy in the next section. NFR taxonomy helps us determine the goals of use cases in the differences among the surroundings.

Variables characterize a use case and differences among the surroundings of the use case. The variables should be both named and typed. For example, because 'the speed of the train' characterizes a use case of the train control system, we regard it as a variable. We name the variable as 'vSpeed' and type it as natural number (0, 1, 2, 3 $\cdots$).

Each type suggests the manner in which the variable can be changed. We can represent such a change of a variable in several ways.

- A change is represented by the assignment of values as in C or Java programs.
- A change is represented by denoting a pair of old and changed values as follows:
  $var1 : old\_value \rightarrow new\_value$
- A change is represented only by the changed

value when we do not need the old value as follows.

$var1 : * \rightarrow new\_value$

- We may simply explain the change with words or sentences as follows.
  $var1$: increase
  $var2$: x becomes a member of $var2$.

Currently, we use the following types and each type has the following notations for relative changes.

- nat: natural number. Two kinds of changes:
  $var++$ means 'var is increased'.
  $var--$ means 'var is decreased'.
- $boolean$: $true$ or $false$.
- set: enumeration of values. When $var2$ is typed with $\{x, y, z, ...\}$, changes can be represented as follows:
  $var2 = x$ means '$var2$ becomes x'.
  $var2 = y$ means '$var2$ becomes y'.
  $var2 = z$ means '$var2$ becomes z'.
- $pset$: subset of variables. Since values in this type are represented by a power set of the set, we denote this type as $pset$. When $var3 : pset\ of\ people$, these kinds of changes can be used:
  $var3 = \{\}$ means '$var3$ becomes empty'.
  $var3 \ni Jone$ means '$Jone$ becomes a member of $var3$'.

We may use general operators for sets or natural numbers, e.g., $\cup, \subset$ or $+, -, \times$.

Invariants among variables are mainly used to represent the correlation between variables. Currently, we use the following operators to show the invariants.

- $x \sim y$: when the variable x is increased the variable y is also increased, and vise versa.

2.2 NFR Taxonomy

As mentioned above, we use NFR taxonomy to determine the goals of the use cases written in user interaction style.

We mainly use the software quality attributes provided in the ISO standard for software quality [12] for this purpose. The attributes are as follows.

- Functionality: Suitability, Accuracy, Interoperability, Security, Compliance.
- Reliability: Maturity, Fault tolerance, Recoverability, Compliance.
- Usability: Understandability, Learnability, Operability, Attractiveness, Compliance.
- Efficiency: Time, Resource, Compliance.
- Maintainability: Analysability, Changeability, Stability, Testability, Compliance.
- Portability: Adaptability, Installability, Co-existence, Replaceability, Compliance.

Since the attributes are insufficient, especially for security, we also use NFR types [6]. The NFR types are as follows.

- Performance
  - Time: Response Time, Throughput, Process Management Time
  - Space: Main Memory, Secondary Storage
- Cost
- User-friendliness
- Security
  - Confidentiality
  - Integrity: Accuracy, Completeness
  - Availability

2.3 Procedure

We will now demonstrate the steps in our method by using the terminologies and the notations discussed above. The inputs for this procedure are manuals and/or help files of the existing applications and/or systems. You may employ use case diagrams for such applications if they already exist.

Step 1: Write or obtain use case diagrams of several systems. The systems should belong to the same and/or similar application domain. Use cases in the diagrams should be written as the system interaction, and not as the user goals.

Step 2: Find common and/or similar use cases in the use case diagrams of several systems. Such use cases do not have to be included in all diagrams.

Step 3: For each use case determined in Step 2, find the differences among the surroundings of the use case in the diagrams.

Step 4: Identify variables that characterize the use cases and their differences in Step 3. Decide the type of each variable so as to construct the changes of the variables.

Step 5: Identify stakeholders and their preferences about NFR based on Figure 2.

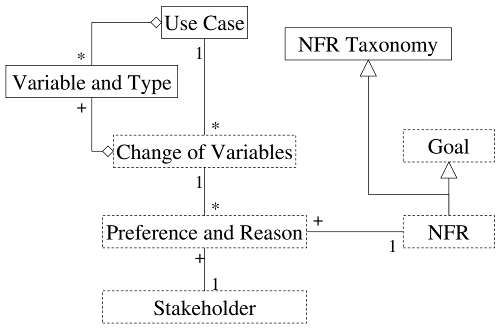

Figure 2. Simple Class Diagram for the Stakeholders, their Preference, and NFR

Classes represented in normal boxes in Figure 2 have been created or prepared before this step. Using these classes, we identify other classes represented in dashed boxes in Figure 2.

1. Identify instances of 'Change of variables'. Since the variables are typed in the previous step, we can identify them systematically.
2. Identify instances of 'Stakeholder' and 'Preference and Reason' related to an instance of 'Change of variables'. The intuitive meaning of the instances is that 'When several variables are changed in a certain way, a stakeholder prefers the change because of some reason'. We should identify them subjectively. However, we may interview real customers and related people by displaying the use case with such variables.
3. Generalize the instances of 'Preference and Reason' into NFR. We may refer to the taxonomy of NFR to make this generalization.

As shown in Figure 2, there can be several different instances of 'change of variables' for each use case. Therefore, Step 5 can be applied several times for each instance of 'change of variables'.

Step 6: Since object diagrams that are the instances of a class diagram in Figure 2 tend to become very complex to be read easily, we use the following form of matrices instead of the diagrams.

This matrix is written for each instance of 'Change of variables', it summarizes each stakeholders' preferences about each NFR.

a change	NFR$_1$	NFR$_2$	...
Stakeholder$_1$	pref$_{11}$	pref$_{12}$	...
Stakeholder$_2$	pref$_{21}$	pref$_{22}$	...
⋮	⋮	⋮	⋮

Each cell is occupied by an instance of 'Preference and reasons' that represents whether stakeholder$_i$ prefers NFR$_j$ or not. When stakeholder$_i$ is not interested in NFR$_j$, the corresponding cell may be empty. Concrete examples of this matrix are shown in the next section.

As a result, we identify stakeholders that are not actors, their preferences about NFR and the NFR that have not been previously mentioned.

3. Example

For illustrating and evaluating our proposed method, we will show an example. The application domain is Mail User Agent (MUA) systems, in other words, a Mailer. Although MUA is not a very large system, MUA systems are realistic and important applications today. There are many different MUA systems all over the world.

3.1 Existing Systems of MUA

In this example, we select four MUA systems: Outlook Express, The RAND MH System [21], AL-Mail [2], and

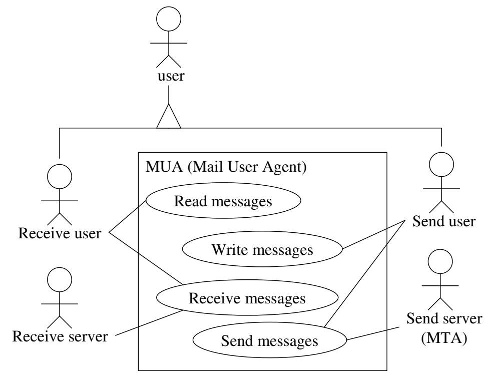

Figure 3. Use Case Diagram of MUA in General

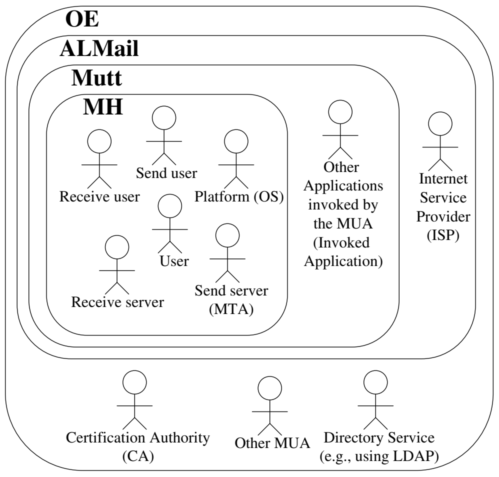

Figure 4. Venn Diagram for Actors in Each System

Mutt [20]. Overview of each MUA is shown in Table 1.

A generic use case diagram for MUA is shown in Figure 3, and all four MUA systems in this example also have such use cases (functions).

3.2 Use Case Diagrams for Each System: Step 1 of our method

We have described the use case diagrams of four different MUA systems. We referred to manuals and/or help files to describe them. Since the authors use MH, Mutt, and AL-Mail almost every day, the knowledge of these three MUA was perhaps complemented. We explicitly described use cases not as user goals but as system interactions according to our method. It was easy to describe use cases in such a manner because the systems already exist, and we may execute them as required. This is the first step of our method.

Since each MUA system is designed for different plat-

Table 1. Overview of the Existing Systems

System Name	Version	Release	Main Platform	Abbreviation used in this Paper	User Interface Type	Note
Outlook Express	6.00	Oct. 2002	Windows	OE	GUI	Japanese Extension
RAND MH System	6.8.3	1993	UNIX	MH	Console, Commands	Japanese Extension
AL-Mail	1.13	Jun. 2002	Windows	ALMail	GUI	Domestic System
Mutt	1.5.4	Mar. 2003	UNIX	Mutt	Console, Interactive	Japanese Extension

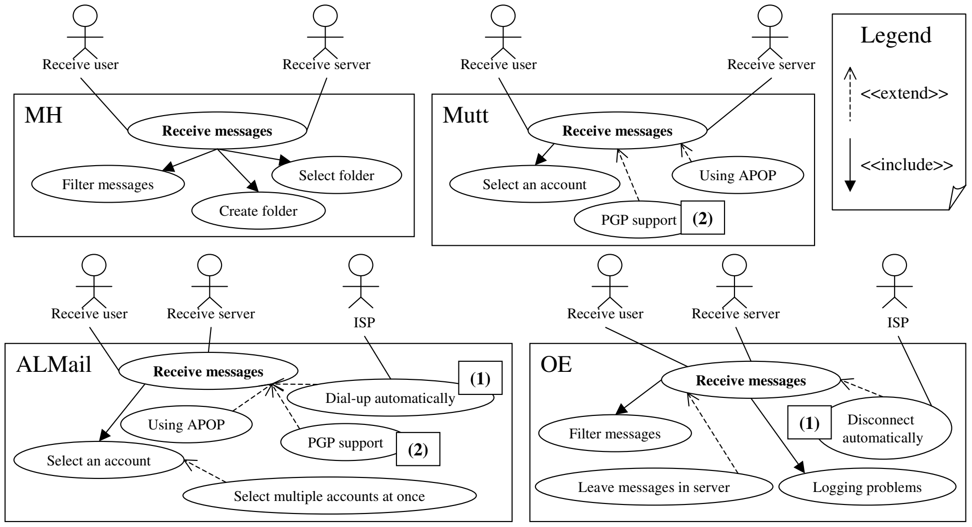

Figure 5. Use Cases and Actors around 'Receive messages'

forms and for different kinds of users, they have a variety of use cases. We will refer to such differences in Section 3.3. We have identified different actors as shown in Figure 4. This figure is written in a Venn diagram so as to know the relationships among the MUA systems. The number of use cases and the number of actors are shown in Table 2.

Table 2. Size for each MUA System

System	Number of use cases	Number of actors
OE	81	11
MH	38	6
ALMail	72	8
Mutt	53	7

3.3 Steps 2 to 6

In Step 2, we should find as many common use cases from the four diagrams as possible. However, we only focus on three common use cases: 'Receive messages', 'Read messages', and 'Import messages' in this example, because of the limitation of pages. The main steps of this method are Steps 3 to 6; these steps are applied to each set of differences in a use case. For convenience, we assign serial numbers to each set of differences in a use case, such as (1), (2), (3) $\cdots$.

3.3.1 UC: 'Receive messages'

Figure 5 shows four partial use case diagrams for each MUA. In each diagram, use cases directly connected to the use case 'Receive messages' and actors connected to them are described. For simplicity, we use simple arrows without stereotypes to show extend- and include- relationships between use cases. The manner of depicting arrows is shown in the legend of this figure.

Although we can determine many differences in Figure 5, we discuss the following two differences in these diagrams. The differences discussed here are marked by rectangles containing numbers in this figure.

(1) Connect/disconnect automatically

Step 3: Identify different use cases.
The use cases *'dial-up automatically'* in ALMail and *'disconnect automatically'* in OE are not included in other systems.

step 4: Identify variables characterizing the differences as shown in Table 3.

Table 3. Variables in (1)

Type	Name	Meaning and Invariant
boolean	$vIsAuto$	Automatically or not
nat	$vNumCon$	Number of time connected
nat	$vNumPas$	Number of time one's password is submitted over the network
		$vNumCon \sim vNumPas$

Steps 5 and 6: We focus on the three instances of 'Change of variables'. We write three matrices for each instance in Figure 6, 7, and 8. Due to the lack of in the cells in the matrix, instances are written outside the matrix. On writing the matrices, we determined three additional stakeholders and three NFR which are given in a thick-lined box.

In Figures 7 and 8, two variables are changed simultaneously. Similar to these examples, we may focus on the change of more than one variable.

If we depict these results as object diagrams, it is difficult to understand them at a glance. For example, the matrix in Figure 7 can be represented as the object diagram given in Figure 9.

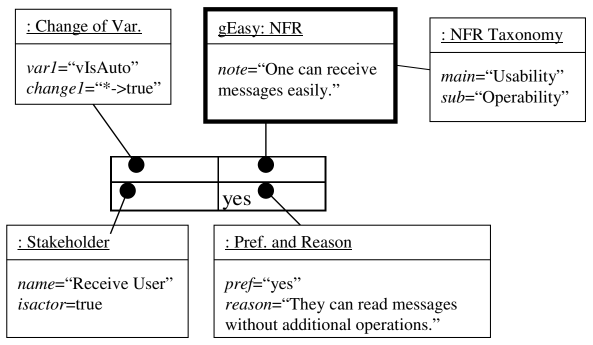

Figure 6. Matrix for vIsAuto in (1)

(2) PGP support

Step 3: Identify different use cases.

'PGP support' is implemented in Mutt and ALMail but not in other systems.

Step 4: Identify variables characterizing the differences as shown in Table 4.

Table 4. Variables in (2)

Type	Name	Meaning and Invariant
nat	$vCrypt$	Strength of the cryptography
pset	vCA	Set of agent authenticating identities

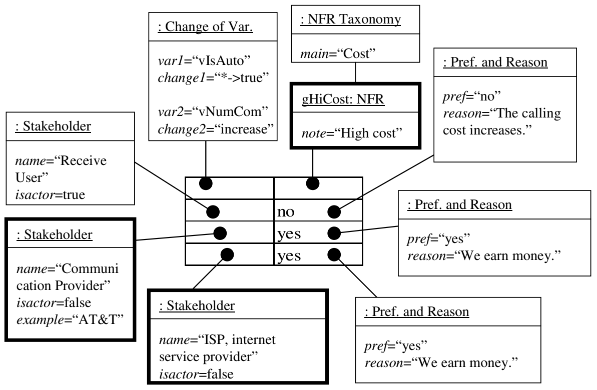

Figure 7. Matrix for vIsAuto and vNumCom in (1)

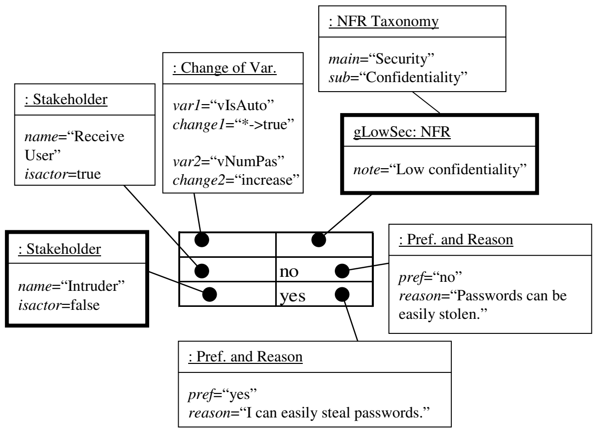

Figure 8. Matrix for vIsAuto and vNumPas in (1)

Steps 5 and 6: We focus on the two instances of 'Change of variables'. We write two matrices in Figure 10 and 11. On writing these matrices, we determined one additional stakeholder and four NFR.

3.3.2 UC: 'Read messages'

Figure 12 also shows four partial use case diagrams for each MUA around a use case 'Read messages'. We discuss the following two differences in these diagrams.

(3) Read messages directly from the server

Step 3: Identify different use cases.

Use cases *'from server'* and *'certification'* in Mutt, ALMail and OE are not in MH. These use cases imply protocols such as IMAP that can store messages on the server side.

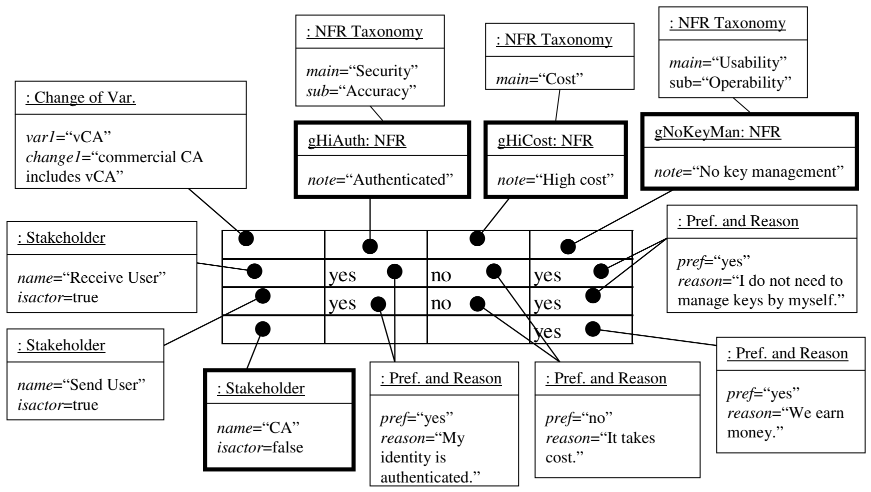

Figure 11. Matrix for vCA in (2)

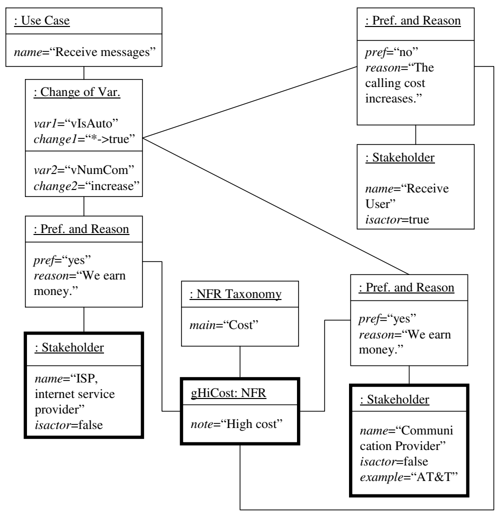

Figure 9. Object Diagram for vIsAuto and vNumCom (Another representation of a matrix in Figure 7) in (1)

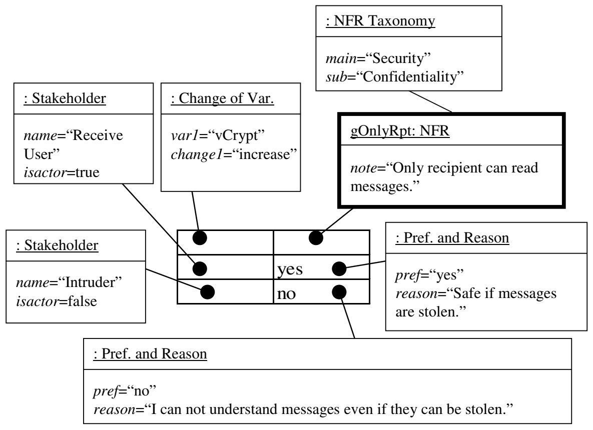

Figure 10. Matrix for vCrypt in (2)

Step 4: Identify variables characterizing the differences as shown in Table 5.

Steps 5 and 6: We focus on the two instances of 'Change of variables'. We write two matrices in Figure 13 and 14. Due to the limitation in the number of pages, we do not write all instances of 'Preference and reasons' in these figures. On writing these matrices, we determined two additional stakeholders and three NFR.

(4) Invoke external applications

Step 3: Identify different use cases.
The use case *'invoke external application'* in Mutt and ALMail and the use case *'execute active contents'* in OE do not appear in MH. OE also has the use case *'control execution of active contents'*. A use case *'allow hyperlink(HTML)'* in ALMail also involves similar functions.

Step 4: Identify variables characterizing the differences as shown in Table 6.

Steps 5 and 6: We focus on the two instances of 'Change of variables'. We write two matrices in Figures 15 and 16. On writing these matrices, we determined one additional stakeholder and four NFR.

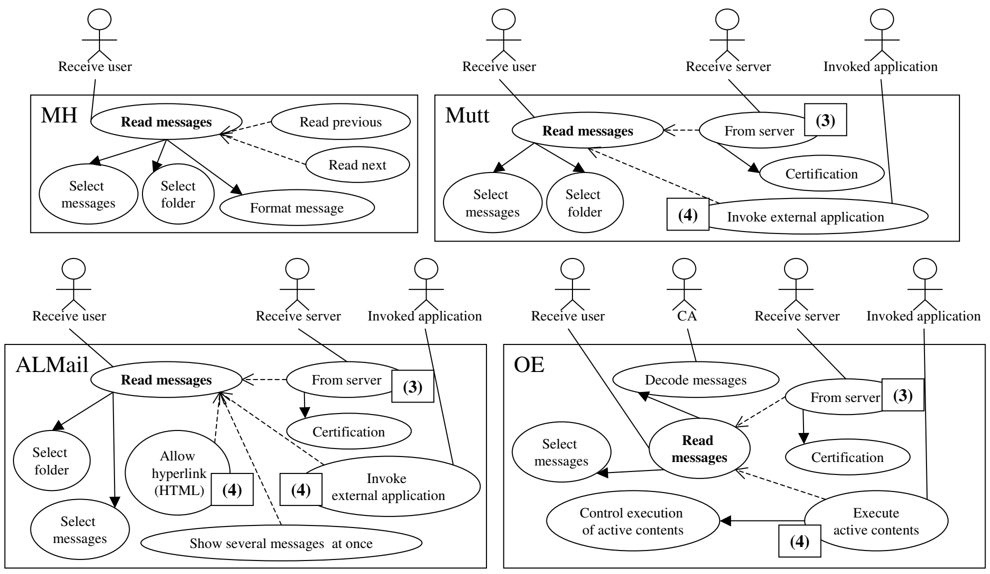

Figure 12. Use Cases and Actors around ‘Read messages’

Table 5. Variables in (3)

Type	Name	Meaning and Invariant
set	$vArea$	Area where MUA is used. {any, only in company, only in department, $\cdots$ }
set	$vResPer$	Person responsible for message archive. {self, server manager}

Table 6. Variables in (4)

Type	Name	Meaning and Invariant
pset	$vApp$	Set of applications
boolean	$vAutoEx$	Admit automatic execution or not

3.3.3 UC: ‘Import messages’

Figure 17 shows two partial use case diagrams for each MUA around a use case ‘Import messages’. We discuss the differences in these diagrams.

(5) Import messages from other MUA

Step 3: Identify different use cases.
A use case *‘Import messages’* in MH and OE is not in other systems. In the case of OE, it can import messages from other MUA, e.g., Eudora.

Step 4: Identify variables characterizing the differences as shown in Table 7.

Steps 5 and 6: We focus on the an instance of ‘Change of variables’. We write two matrices in Figure 18. On writing these matrices, we determined two additional

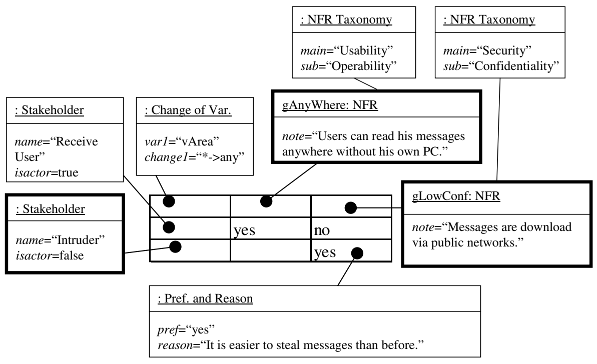

Figure 13. Matrix for vArea in (3)

Table 7. Variables in (5)

Type	Name	Meaning and Invariant
pset	$vMua$	Set of MUA

stakeholders and one NFR.

3.4 Evaluation

The use of our method in this example has resulted in several findings. In normal GQM, it is not easy to determine goals. In our method, this is not very difficult, because we can easily generalize goals from use cases written in system interaction style. In addition, such use cases can be easily written. NFR taxonomy also helps us to determine them.

It is easy to find malicious stakeholders, e.g., intruders,

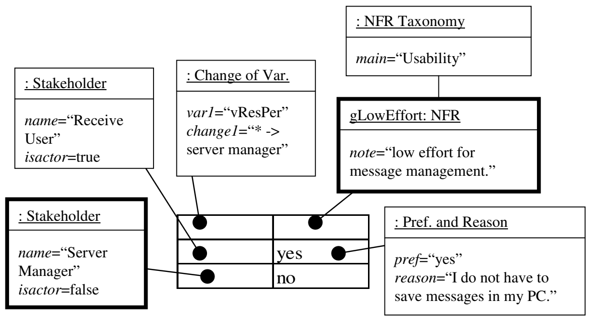

Figure 14. Matrix for vResPer in (3)

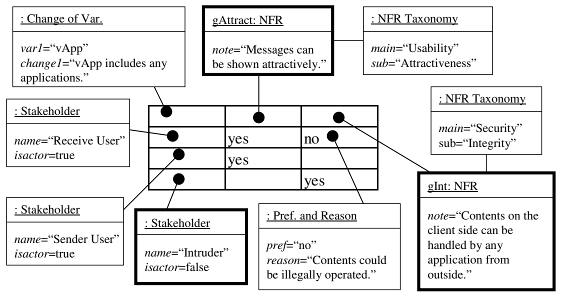

Figure 15. Matrix for vApp in (4)

or competitors by observing the change of variables. This is important because the world is insecure and competitive now.

NFR normally cut across several functions [17]. It is not so easy to identify such crosscuts only with normal use case diagrams; however, the variables in our method help us to identify them. For example, both $vIsAuto$ in (1) and $vAutoEx$ in (4) refer to a similar property of user operations, thus we can consistently design the operations. In general, one of the disadvantages of use case diagrams is

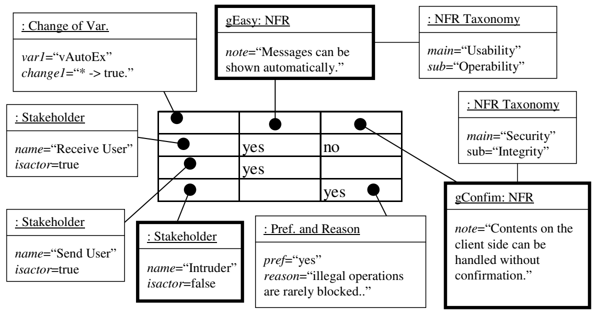

Figure 16. Matrix for vAutoEx in (4)

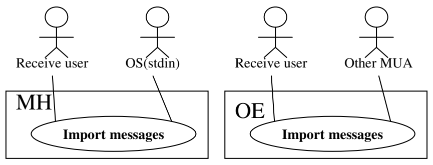

Figure 17. Use Cases and Actors around 'Import messages'

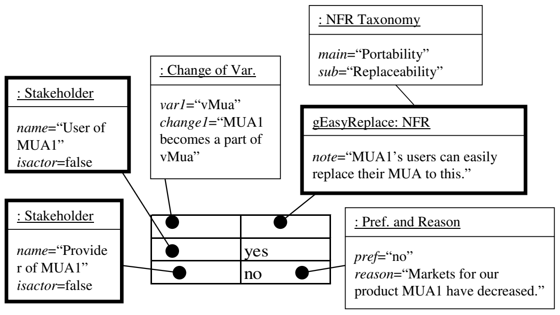

Figure 18. Matrix for vMua in (5)

that they cannot easily handle data or variables like data flow diagrams.

We can also determine the property related to 'compliance'. Figure 19 shows the differences between message deletion in UNIX based systems and Windows based systems. In general, Windows based systems store deleted files or data in the trash; however UNIX based systems do not. This difference is clearly represented in this Figure. However, our method cannot find this case, since use cases except 'delete messages' in Figure 19 are not included in the surrounding of 'delete messages' by definition.

4. Conclusions and Discussion

In this paper, we present a method to identify stakeholders, their preferences about NFR by comparing use case di-

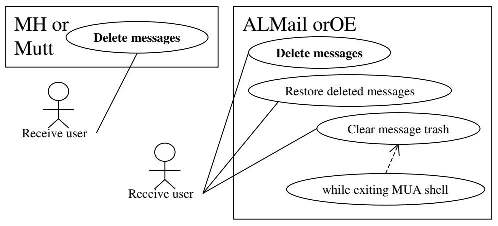

Figure 19. Use Cases and Actors related to 'delete messages'

agrams. The results of our method can be used to prioritize NFR and stakeholders. In WinWin approach [5], stakeholders should explore the trade-offs among their goals (win conditions). The method proposed here can contribute to this task, since we can understand the relationships among NFR goals through the variables. In DDP [8], exhaustive elicitation of requirements and risks is important. Our method will contribute to such an elicitation process, since unexpected situation can be generated by changing the values of variables related to NFR. Our method also can be used with AGORA [13] that clarifies the conflicts among stakeholders.

This method is based on the package oriented requirements elicitation method PAORE [14]. However, we did not handle NFR and preferences of stakeholders in PAORE. A simple functional decomposition represented by tables is used in PAORE. However, we could not naturally handle actors related to systems. As a result, the method in this paper overcomes several weaknesses of PAORE.

So as to use our method efficiently, we have to develop supporting tools. We at least need a database for use case diagrams and a tool for comparing the diagrams.

Although we do not handle internal structure of each use case, for example, scenario descriptions, we can obtain a sufficient number of stakeholders and preferences. Thus, we do not plan to handle scenarios in this method.

Apparently, our method cannot handle stakeholders that are related to the development process, e.g., software designers and programmers. So as to handle such stakeholders, we need specifications of the software development process.

Since stakeholders in this method can be regarded as viewpoints [16], one of the purposes of this method can be represented as 'finding new viewpoints'. Currently, we do not focus on inconsistency resolution among viewpoints. Stakeholders in this method are also similar to *Persona* [7], in other words, actual people.

There are few related works about requirements elicitation using GQM and quality standards. GQM, QFD and requirements specifications are used for quality control [19]. Quality standards are used for package selection [10].

References

[1] IEEE Guide for Developing System Requirements Specifications, Dec. 1998. IEEE Std 1233-1998, ISBN 0-7381-0337-3 SH94654 (Print).

[2] AL-Mail32. `http://www.almail.com/`. Japanese page only.

[3] I. Alexander and S. Robertson. Understanding Project Sociology by Modeling Stakeholders. *Software*, 21(1):23–27, Jan./Feb. 2004.

[4] V. R. Basili and D. M. Weiss. A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering*, SE-10(6):728–738, Nov. 1984.

[5] B. Boehm, P. Grunbacher, and R. O. Briggs. Developing Groupware for Requirements Negotiation: Lessons Learned. *IEEE Software*, 18(3):46–55, May/Jun. 2001.

[6] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[7] A. Cooper. *The Inmates Are Running the Asylum*. SAMS, 1999.

[8] S. L. Cornford, M. S. Feather, J. C. Kelly, T. W. Larson, B. Sigal, and J. D. Kiper. Design and Development Assessment. In *Proceedings of the Tenth International Workshop on Software Specification and Design (IWSSD'00)*, 2000.

[9] M. Fowler and K. Scott. *UML Distilled, Applying the Standard Object Modeling Language*. Addison-Wesley, 1st edition, 1997.

[10] X. Franch and J. P. Carvallo. Using Quality Models in Software Package Selection. *Software*, 20(1):34–33, Jan./Feb. 2003.

[11] M. Glinz. A Lightweight Approach to Consistency of Scenarios and Class Models. In *4th International Conference on Requirements Engineering*, pages 49–58, 2000.

[12] International Standard ISO/IEC 9126-1. Software engineering - Product quality - Part 1: Quality model, 2001.

[13] H. Kaiya, H. Horai, and M. Saeki. AGORA: Attributed Goal-Oriented Requirements Analysis Method. In *IEEE Joint International Requirements Engineering Conference, RE'02*, pages 13–22, Sep. 2002.

[14] J. Kato, M. Saeki, A. Ohnishi, M. Nagata, H. Kaiya, S. Komiya, S. Yamamoto, H. Horai, and K. Watahiki. PAORE: Package Oriented Requirements Elicitation. In *Proceedings of 10th Asia-Pacific Software Engineering Conference (APSEC 2003)*, pages 17–26, Chiang Mai, Thailand, Dec. 2003. IEEE Computer Society Press.

[15] L. A. Macaulay. *Requirements Engineering*. Applied Computing. Springer, 1996.

[16] B. Nuseibeh, J. Kramer, and A. Finkelstein. A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification. *IEEE Transations on Software Engineering*, 20(10):760–773, Oct 1994.

[17] M. Saeki and H. Kaiya. Transformation Based Approach for Weaving Use Case Models in Aspect-Oriented Requirements Analysis. In *The 4th AOSD Modeling With UML Workshop*, Oct. 2003. Joint workshop of UML 2003, `http://www.cs.iit.edu/~oaldawud/AOM/index.htm`.

[18] T. Systa. Understanding the Behavior of Java Programs. In *Seventh Working Conference on Reverse Engineering*, pages 214–223, 2000.

[19] S. Szejko. Requirements Driven Quality Control. In *COMPSAC'2002*, pages 125–130, 2002.

[20] The Mutt E-Mail Client. `http://www.mutt.org/`.

[21] The RAND MH Message Handling System UCI version 6.8.3. `http://www.ics.uci.edu/%7Emh/`.