PAPER Special Section on Knowledge-Based Software Engineering

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

Haruhiko KAIYA[†a], **Akira OSADA**[†], *Nonmembers, and* **Kenji KAIJIRI**[†], *Member*

SUMMARY 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 changes to be occurred. We utilize Goal-Question-Metrics (GQM) method for identifying variables that characterize NFR, and we can systematically represent changes about NFR using the variables. Use cases that represent system interactions help us to bridge the gap between goals and metrics (variables), and we can easily construct measurable NFR. For validating and evaluating our method, we applied our method to an application domain of Mail User Agent (MUA) system.

key words: *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 existing systems. By using our method, we can prioritize NFR, for example, to maximize preferences of all stakeholders. Because the policy for prioritizing NFR depends on each project, we do not handle such policies in this method.

In requirements engineering field, 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' [1]. Therefore, 'stakeholder is much more than a product's eventual user' [2]. To avoid fruitless development and useless software systems, suitable stakeholders should be identified as soon as possible. However, there are few methods to identify stakeholders [3]. Our method contributes to finding stakeholders at the early phase of software development, i.e., requirements elicitation phase.

It is important to know stakeholders' preferences when we elicit requirements. Preferences of stakeholders are largely related to NFR and/or quality requirements, because non-functionalities are relatively unstable. On the other hand, functions of a system are fundamental characteristics and they are relatively stable. For example, an airplane without flying function is NOT an airplane. It is not a problem about preference. However both an airplane flying very fast and another flying slowly and safely are airplanes, and one prefers the former but another person does not.

In general, it is not easy to decide whether someone prefers one of NFR or not, but he/she can easily decide one's preference when one of NFR is changed. For example, we can clearly state the preference about performance requirement when an airplane flies faster than before. So 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 airplane's performance, its speed can be a variable. Such variables characterize more than one NFR in general. So changes of such variables can help stakeholders to find unidentified other NFR.

Goal-Question-Metrics (GQM) [4] is a method to select the data which can be used to know the way how a system achieves its goals. We utilize GQM for representing the changes about NFR because metrics in GQM can be regarded as measurable and changeable variables for NFR.

Such variables contribute to constructing well-formed requirements [5] because NFR, in other words conditions, should be measurable in well-formed requirements. Soft goals [6], that are not measurable, are also important to catch NFR. However, stakeholders cannot decide their preferences only by referring soft goals.

In GQM, it is not so easy to derive metrics from goals via questions because goals are relatively more abstract than metrics. This is one of the difficulties of GQM. In this research, we use requirements specifications for existing similar systems as the hint to bridge the gap between goals and metrics. Even if two systems are similar with each other, there are several differences between the two systems. Such differences of systems are caused by the differences of goals, especially non-functional goals, and are specified by some metrics. Therefore, we can identify relationships between goals and metrics by focusing on differences among similar systems.

We use use case diagrams in UML to represent existing similar systems. First reason is that use case diagrams are defacto standard to represent functional requirements now. Second reason is that the diagrams have the concept of actors, that can be first approximation of stakeholders. Third reason is that there are several techniques to construct use case diagrams and other related diagrams from existing systems in the reverse engineering field [7], [8].

In Sect. 2, we show the concrete steps to perform our method to elicit NFR, stakeholders, their preferences. An example using our method is shown in Sect. 3. Finally, we summarize our current results and show the future direction.

Manuscript received July 2, 2007.
Manuscript revised October 12, 2007.

[†]The authors are with the Graduate School of Science and Technology, Shinshu University, Nagano-shi, 380–8553 Japan.
a) E-mail: kaiya@cs.shinshu-u.ac.jp

DOI: 10.1093/ietisy/e91–d.4.897

2. Method

2.1 Terminologies and Notions

Before introducing the procedure of our method, we introduce terminologies and notions for our method.

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

System interactions and **user goals** are mentioned in [9]. They are two different styles in which use case represents functionality. When use cases are written as system interactions, they represent how to interact with the system. On the other hand, use cases represent what are achieved when they are written as user goals.
Because our method is started from the existing systems, we use system interaction style. It is easy to write in such a style because we have had the systems and we may execute them if we want.
Whether a use case is written as a system interaction or a user goal, the use case can be any levels of granularity. In our method, use cases are decomposed up to user goal levels (blue level) and subfunctions level (indigo level) defined in [10]. Elementary business processes are written in use cases in blue level, and use cases in indigo level correspond to goals that are required to carry our goals in blue level. Because goals in blue level rarely have mutual relationships [10] and our method needs such relationships among use cases, use cases in indigo level are encouraged to be written.
Because use case diagrams used in our method are not the results of requirements elicitation but the results of analysis of existing systems, use cases in indigo level can be written without decision by analysts using our method. Use cases in indigo level usually contain architecture and design issues, and such issues largely give influences to satisfaction of NFR. This is another reason why use cases in indigo level are required in our method.

Similarity of applications are decided by the amount of similar actors and of similar use cases. When several applications are similar with each other, such set of applications are called an application domain.

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

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

Differences among surroundings of a use case consist of the use cases that are not included in the intersection among the surroundings.
In Fig. 1, we show small example of differences between two surroundings. We focus on a use case and identify two surroundings of the use case; one is in the system P and another in system Q. First surrounding consists of actor X, Y and use cases A, B and C. Second one consists of actors X, Y and use cases D, E and C. Therefore differences between two surroundings of the use case are use cases A, B, D and E.

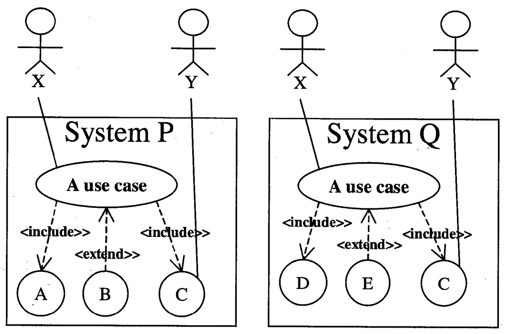


Fig. 1 Example of differences between two surroundings.

NFR taxonomy is a catalog of non-functional requirements in general. Currently, we use software quality attributes in ISO standard for software quality [11], and NFR types [6]. We will briefly introduce the contents of NFR taxonomy in the next section. NFR taxonomy helps us to find the goals of use cases in differences among surroundings.

Questions are almost the same in GQM. In this method, we derive questions not directly from goals but from both use cases and related NFR. So it is not so difficult to make questions.

Variables are included in questions for checking the achievement of goals. They should be both named and typed. For example 'speed of the train = $var1$: *nat*', the name of variable is $var1$, the type is *nat* and 'speed of the train' shows the intuitive meaning of the variable.
Each type suggests how the variable can be changed. A relationship between pre and post values represents the change of the variable. We use notation that is used in a typical formal methods such as Z notation [12] to represent such changes. For example, when a variable $var1$ is increasing by a change, the change can be specified by the following logic formula.

$$var1' > var1$$

Note that $var1$ corresponds to a value of $var1$ before the change, and $var1'$ corresponds to a value after the change.
Currently, we use the following types and changes in each type are specified by the following formulas.

- *nat*: natural number. Two kinds of changes:
  $var' > var$ means 'var is increased'.
  $var' < var$ means 'var is decreased'.
- *boolean*: *true* or *false*. In our method, we define $true > false$. Therefore, the following two formulas say the same fact.

$var = false \wedge var' = true$
$var < var'$

- $\{x, y, z, \ldots\}$: enumeration of constants. Among the elements of an enumeration, there can be orderings. When these is no explicit axiom about orderings, there is a transitive ordering between any two different elements. In the case of $\{x, y, z, \ldots\}$, there is a ordering such as $x > y > z > \cdots$.
  When $var2 : \{x, y, z, \ldots\}$, we can show a change by using the formula below:
  $var2 \neq x \wedge var2' = x$
  This formula means '$var2$ was not x, but it becomes x'. This formula can be represented in the following formula because there is a transitive ordering between any two elements in $\{x, y, z, \cdots\}$.
  $var2' = x \wedge var2' > var2$
- *set of constants*. Suppose $var3$ recorded a set of people such as $\{Tom, Bill, John\}$, and another guy Kim is added to the set $var3$. The following formulas are true, and specify this change.
  $|var3| < |var3'|$
  $var3 \subset var3'$
  $var3 \cup \{Kim\} = var3'$

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 the variables. Currently we use the following operators to show the invariants.

- $x \sim y$: When the variable x is changes, the variable y is also changed, and vise versa.

2.2 NFR Taxonomy

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

We mainly use software quality attributes in ISO standard for software quality [11] for this purpose. The list of the attributes is 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.

Because the attributes are not sufficient, especially in security, we also use NFR types [6]. The list of NFR types is 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

Now we will show the steps of our method by using terminologies and notations above. The inputs of this procedure are manuals and/or help files of existing applications and/or systems. You may use use case diagrams for such applications if they are available.

Step 1: Write or get use case diagrams of several systems. The systems should belong to the same and/or similar application domain, and for each system at least one diagram should be written or gotten. Use cases in the diagrams should be written as the system interaction, not as the user goals. In addition, use cases are decomposed up to blue and indigo level [10].

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

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

Step 4: Analyze the reasons of differences, and identify goals that are achieved by use cases in the differences. In this step, requirements analysts have to interact with domain experts including technical experts to explore the goals that are satisfied by use cases identified in step 3. Especially, analysts have to find the reasons why these use cases exist, especially what are advantages of these use cases. To categorize such reasons and advantages, NFR taxonomy is used. Several keywords contribute to narrowing the candidates of NFR. For example, the word "automatically" is frequently related to usability. Understanding technical words also contributes to identifying NFR. For example, "PGP" is a technique for security, thus the analyst can focus on security related NFR.

Step 5: Create questions for checking the achievement of the goals in the same way as GQM. We may refer system interactions in use cases that achieve the goals, when we create questions.

Step 6: Identify the variables that characterize the questions. Variables should be typed for identifying the directions and ranges of their changes.
We may add variables that are not directly related to the questions but related to a variable already identified by using technical knowledge and/or experiences. Write

Table 1 Overview of existing systems.

System Name	Version	Release	Main Platform	Nickname in this paper	Uer 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

invariants among variables.

Step 7: Assume the changes of variables. For each change, identify stakeholders that are affected by the change and how their preferences about NFR are changed. For example, suppose a variable 'the length of password for a system' is shortened. A stakeholder prefers such a change with respect to NFR 'usability' because he/she does not have to mange long phrase. At the same time, the stakeholder does not prefer such change with respect to NFR 'security' because someone spoofs the system easily. We use the following kinds of table to show these preferences.

change of variables	NFR_1	NFR_2	...
$Stakeholder_1$	+ or −	+ or −	...
$Stakeholder_2$	+ or −	+ or −	...
⋮	⋮	⋮	⋮

As shown in a simple example above, each cell represents whether $stakeholder_i$ prefers the change with respect to NFR_j. + means that he prefers the change with respect to NFR_j, and − means that he does not. When $stakeholder_i$ is not interested in the change with respect to NFR_j, the cell is empty. We also attach the reason why the stakeholder prefers or not in this table.

To find stakeholders that are not actors and to find NFR that are not focused in step 4, we may freely imagine the impacts to NFR when the variables are change in a certain way.

As a result, we identify stakeholders that are not actors, their preferences about NFR and NFR that are not mentioned before. Newly found stakeholders and NFR are underlined in examples in Sect. 3.3.

Steps 4, 5 and 6 are mainly imported from GQM [4].

3. Example

For validating and evaluating our proposed method, we will show an example for applying this method. The application domain is Mail User Agent Systems (MUA), in other words, Mailer. Although MUA is not so large system, MUA is realistic and important application today. There are many different MUA all over the world.

3.1 Existing Systems of MUA

In this example, we select four MUA systems: Outlook Express, The RAND MH System [13], AL-Mail [14] and Mutt [15]. Overview of each MUA is shown in Table 1.

Generic specification for MUA is shown in Fig. 2, and

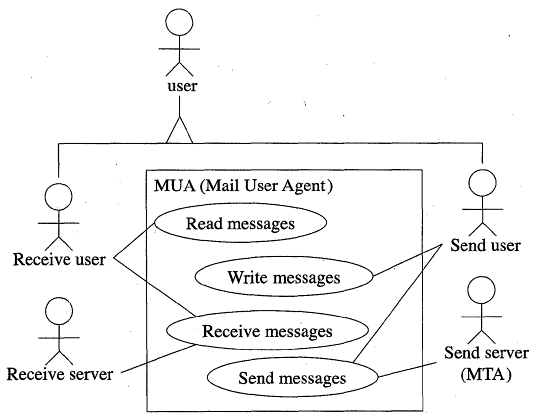


Fig. 2 Use case diagram of MUA in general.

all four MUA systems in this example also have such functions (use cases).

3.2 Use Case Diagrams for Each System: Step 1 of Our Method

We have described use case diagrams of four different MUA systems. We referred manuals and/or help files for describing them. Because authors use MH, Mutt and ALMail almost every day, we would complement knowledge for these three MUA. 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 way because we have already had the systems, and we may execute them, as we want. This is the first step of our method.

Because each MUA system is designed for different platforms and for different kinds of users, they have many different use cases. We will refer such differences in Sect. 3.3. We have identified different actors as shown in Fig. 3. This Figure is written in a Venn diagram to know the relationships among the MUA systems. The number of use cases and the number of actors are shown in Table 2.

3.3 Results

In step 2, we should find common use cases from four diagrams. In this example, we only focus on three common use cases: 'Receive messages', 'Read messages' and 'Import messages'.

3.3.1 UC: 'Receive messages'

Figure 4 shows four partial use case diagrams for each MUA. In each diagram, use cases directly connected to a use

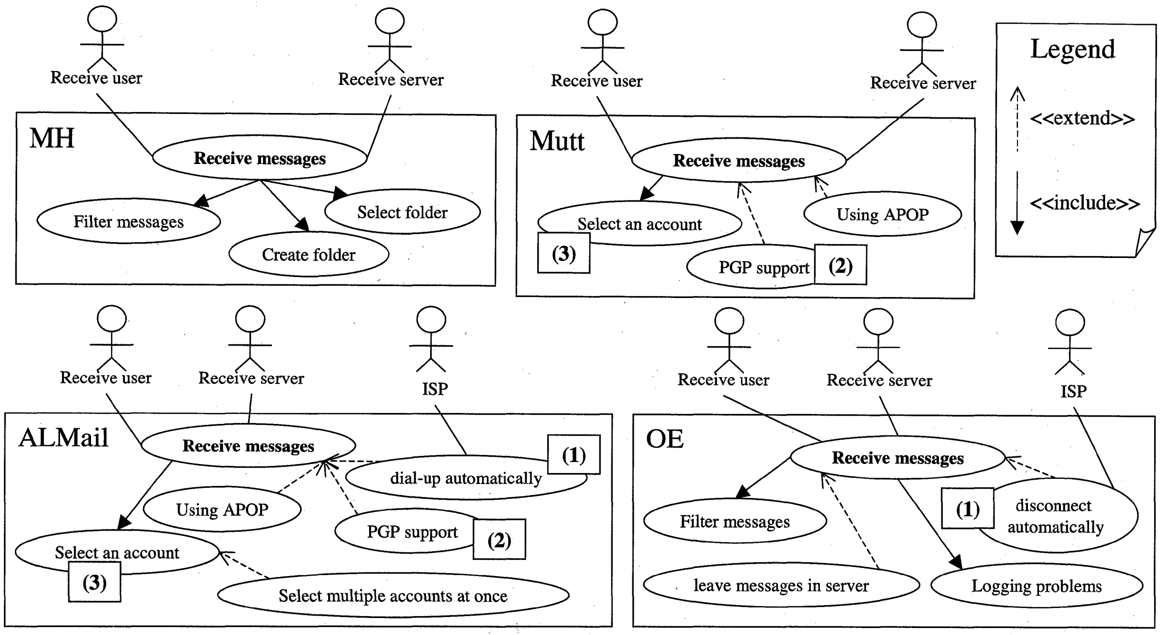


Fig. 4 Use cases and actors around 'Receive messages'.

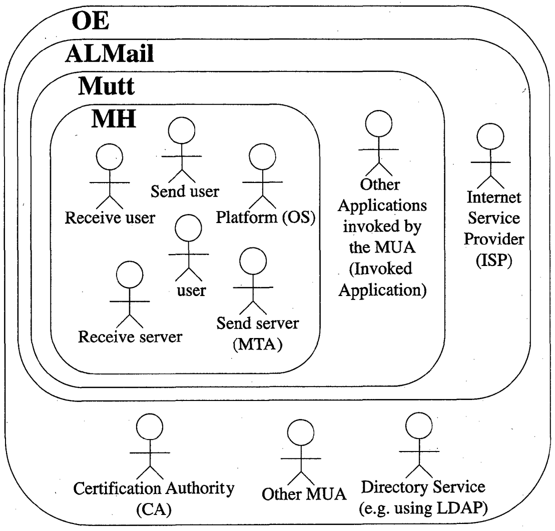

OE
ALMail
Mutt
MH
Receive user
Send user
Platform (OS)
user
Receive server
Send server (MTA)
Other Applications invoked by the MUA (Invoked Application)
Internet Service Provider (ISP)
Certification Authority (CA)
Other MUA
Directory Service (e.g. using LDAP)

Fig. 3 Venn diagram for actors in each system.

Table 2 Size for each MUA system.

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

case 'Receive messages' and actors connected to them are described. For simplicity, we use simple arrows to show extend and include relationships between use cases as shown in the legend in this Figure.

Although we can find many differences in Fig. 4, we discuss the following three differences about these diagrams. The differences discussed here are marked by rectangles with number in this Figure.

(1) Connect/disconnect automatically

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

step 4: Identify goals by using NFR taxonomy.
A requirements analyst regarded the term "automatically" was related to usability of the system. Therefore, the analyst asked a domain expert if the use cases above contributed to using the system easily. In addition, the analyst asked the expert there are another reasons why the use cases existed. The expert answered yes, and mentioned the cost for network connection. Based on the interaction above, the analyst identifies the goals below.

g1: One can receive messages without additional operation like connecting networks. (Operability.Usability).

g2: One can save cost for communication. (Cost).

step 5: Create questions for checking goal achievement.

q1: Whether the connection is accomplished automatically or not?

q2: How many times or hours one can connect the network?

step 6: Identify variables in questions.

- Automatically or not = *var*1 : *boolean* related to g1.
- Number of connecting times = *var*2 : *nat* related to g2.

The following variables are related to the variables above.

- Number of submitting one's password over the network = *var*3 : *nat* such that *var*2 ~ *var*3

step 7: Identify stakeholders and their preferences.

*var*1 > *var*1′	g1
Receive user	−

*var*2 < *var*2′	g2	Security
Receive user	−	−
ISP	+	
Communication provider (e.g. AT&T)	+	
Intruder		+ r1

r1: *var*2 ~ *var*3.

As a result, we found two additional stakeholders and one of NFR that are underlined.

(2) PGP support

step 3: Identify different use cases.
'*PGP support*' in Mutt and ALMail is not in other systems.

step 4: Identify goals by using NFR taxonomy.
A requirements analyst asked a domain expert what was PGP. The export answered that PGP was a kind of software of cryptography for email. The analyst tried to identify the goal(s) of PGP by asking questions with security related NFR in the NFR taxonomy. For example, "Can PGP contribute to keeping our secret?" Based on such interactions, the analyst identified the goals below.

g3: Only recipient can read messages. (Security.Confidentiality).
g4: One's identity is authenticated. (Security.Accuracy)

step 5: Create questions for checking goal achievement.

q3 for g3: How strong the used cryptography is?
q4 for g4: Who authenticates users identities?

step 6: Identify variables in questions.

- Strength of the cryptography = *var*4 : *nat* related to g3.
- Agent authenticating identities = *var*5 : {*commercial CA, private CA, no CA*}
  Note that CA is certificate Authority, and we assume companies of CA are more reliable than others. The element '*no CA*' means no one authenticates users identities.

step 7: Identify stakeholders and their preferences.

*var*4 < *var*4′	g3
Intruder	−
Receive user	+

*var*5′ > *var*5	g4	Cost	Operability
Receive user	+	−	+ r2
Sender	+	−	+ r2
CA			− r3

The marks r2 and r3 in the table are references to the reason why stakeholders have such preferences. The reasons are as follows.

r2: They do not have to manage keys by themselves.
r3: They can get charges.

As a result, we found additional stakeholders and NFR that are underlined.

(3) Multiple accounts

step 3: Identify different use cases.
'*Select account*' in Mutt and ALMail and '*Select multiple accounts at once*' in ALMail are not in other systems.

step 4: Identify goals by using NFR taxonomy.
A requirements analyst asked a domain expert the reason(s) for each use case. Especially, the analyst asked if the latter use case 'select multiple accounts at once' was related to usability because the use case seemed to contribute to decreasing user's efforts. The expert answered the former use case was used to identify and to authenticate a user who wanted to read email. Thus, the analyst checked security related NFR taxonomy and identified the goal below. The expert also answered latter use case was useful because users with several accounts could read his/her email at once. Based on the interaction above, the analyst identified the goals below.

g5: One's identity is authenticated. (Security.Accuracy)
g6: One can handle multiple accounts at once without additional operation. (Operability.Usability)

Because g5 is the same as g4 in (2), we only discuss g6 later.

step 5: Create questions for checking goal achievement.

q5: How many accounts can one handle?
q6: How to notify ID and passwords to the system?

step 6: Identify variables in questions.

- Number of accounts = *var*6 : *nat*
- Managing Schema for accounts = *var*7 : {*save disk, input every time*}

step 7: Identify stakeholders and their preferences.

*var*6 < *var*6′	g6
Receive user	+

*var*7 < *var*7′	g6	Confident.
Receive user	+	− r4

r4: Storing accounts in client machines is insecure in general.

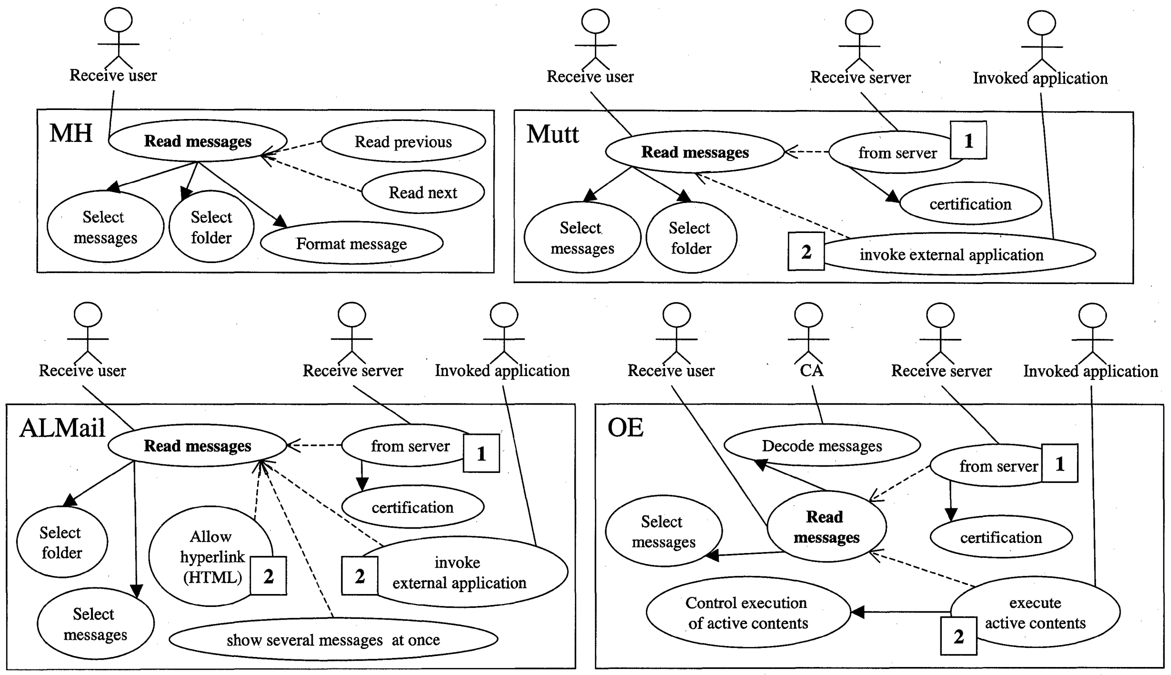


Fig. 5 Use cases and actors around 'Read messages'.

As a result, we found additional NFR that is underlined.

3.3.2 UC: 'Read messages'

Figure 5 also shows four partial use case diagrams for each MUA around a use case 'Read messages'. We discuss the following two differences about these diagrams.

(1) Read messages directly from server

step 3: Identify different use cases.
Use cases '*from server*' and *certification* are in Mutt, ALMail and OE, but such use cases are not in MH.

step 4: Identify goals by using NFR taxonomy.
A requirements analyst asked a domain expert what was the role of these use cases. The expert explained IMAP, which could leave messages in the server side. The analyst asked the expert what were the advantages of IMAP to identify goals that are satisfied by these use cases. The expert explained the advantages of IMAP. Based on these interactions, the analyst identified goals below.

g7: One can read his messages anywhere even when he does not bring his own PC. (Operability.Usability).

g8: One does not have to prepare the space for messages. (Resource.Efficiency)

step 5: Create questions for checking goal achievement.

q7 for g7: From where can one connect to the server?

q8 for g8: How much space does one prepare for his message?

q9 for g8: Who has the responsibility for archiving messages?

step 6: Identify variables in questions.

- Area where using MUA = *var*8 : {*any*, *only in company*, *only in department*} related to g7.
- Space for messages in recipient's machine = *var*9 : *nat* related to g8.
- Responsible person for message archives = *var*10 : {*server manager*, *self*}

step 7: Identify stakeholders and their preferences.

*var*8 < *var*8′	g7	Confident.
Receive user	+	−
Intruder		+

*var*9 < *var*9′	g8
Receive User	−
Server manager	+ r5

r5: Manager does not have to prepare spaces for clients' messages if each client store their messages more.

*var*10 < *var*10′	g9
Receive User	+
Server manager	−

As a result, we found underlined stakeholders.

(2) Invoke external applications

step 3: Identify different use cases.

'*invoke external application*' in Mutt and ALMail and '*execute active contents*' in OE are not in MH. OE also has a use case '*control execution of active contents*'. A use case '*allow hyperlink (HTML)*' in ALMail involves also similar functions.

step 4: Identify goals by using NFR taxonomy.
A requirements analyst asked a domain expert what were the advantages of these use cases. The expert answered that users could send/receive attractive email e.g., with colors or beautiful type sets, and they could also accept and execute dynamic contents even they did not know how to execute them. Based on these interactions, the analyst identified goals below.

g9: The sender of the message can show his message attractively. (Attractiveness.Usability).
g10: When the recipient do not have knowledge to execute specific application, he/she can easily execute the message. (Learnability.Usability).

step 5: Create questions for checking goal achievement.

q10: Which applications can be executed in the recipient side?
q11: How applications can be executed in the recipient side?
q12: Can recipient easily execute the application attached to a message?

step 6: Identify variables in questions.

- Set of application which can be executed = $var11$: *set of applications*.
- Access control for applications = $var12$: *nat*. Note that the more $var12$ is, the fewer resources applications can handle.
- automatic execution or not = $var13$: *boolean*

step 7: Identify stakeholders and their preferences.

$var11 \subset var11'$	g9	Security
Sender	+	
Receive user	+	−
Intruder		+

$var12 < var12'$	g10	Security
Sender	−	
Receive user	−	+
Intruder		−

$var13 > var13'$	g10	Security
Sender	−	
Receive user	−	+
Intruder		−

As a result, we found underlined new stakeholders and NFR. In addition, $var12$ and $var13$ seems to have a correlation because preferences about security are the same.

3.3.3 UC: 'Import messages'

Figure 6 shows two partial use case diagrams for each MUA around a use case 'Import messages'. We discuss a difference about these diagrams.

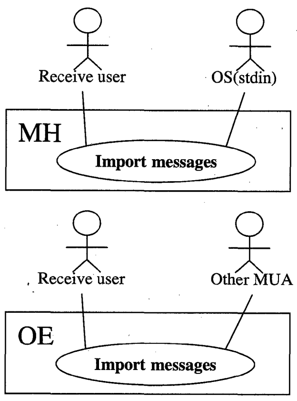


Fig. 6 Use cases and actors around 'Import messages'.

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 goals by using NFR taxonomy.
A requirements analyst asked a domain expert why users required the use case. The expert answered that few users used several different MUA at once, but some users tried to change his/her own MUA. This use case was used in such a change. Based on these interactions, the analyst identified a goal below.

g11: Users can easily change his MUA to e.g. OE. (Replaceability.Portability).

step 5: Create questions for checking goal achievement.

q13: Which kind of message can be converted to the MUA's format?

step 6: Identify variables in questions.

- set of MUA that format can be imported = $var14$: *set of MUA*

step 7: Identify stakeholders and their preferences.
Suppose an instance of MUA, say *mua*.

$\neg(var14 \ni mua) \wedge var14' \ni mua$	g11
Provider of *mua*	−
Current user of *mua*	+

As a result, we found underlined new stakeholders.

3.4 Evaluation

Through the experience to use our method in this example, we have found several findings. In normal GQM, it is not easy to find goals and to identify questions. In our method, it is not so difficult to find them because we can easily imagine both goals and questions from use cases written in system interaction style. In addition, such use cases can be easily written. NFR taxonomy also helps us to find them.

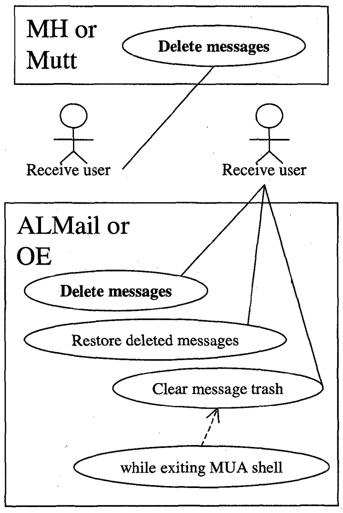


Fig. 7 Use cases and actors related to 'delete messages'.

It is easy to find bad stakeholders, e.g. intruders, or competitors by observing the change of variables. This is important because the world is insecure and competitive now.

NFR are normally cross-cut across several functions [16]. In normal use cases, it is not so easy to identify such cross cuttings only with normal use case diagrams, but variables in our method help us to identify them. For example, both *var*1 in Sect. 3.3.1 (1) and *var*3 in Sect. 3.3.2 (2) refer similar property of user operations, and we can consistently design the operations. In general, one of disadvantages of use case diagrams is that they cannot easily handle data or variables like data flow diagrams.

Software systems largely depend on the external environment, and such environment is partially defined by configuration of each system. In the case of MUA, accepted protocols are defined in both configuration files for building the application and configuration files in runtime. Our method does not take variability caused by the differences of configurations into account. In our method, requirements analysts focus on each system under a specific configuration. Therefore, same systems under different configurations are regarded as different systems. Because the external environment itself is represented as external actors in a use case diagram, differences caused by an external environment can be handled in our method.

In some cases, use cases for shared data improve usability of a system. As shown in Fig. 7, ALMail and OE provides a function for restoring deleted messages. This function improves usability, especially operability. In the case of MH and Mutt, users will be able to restore deleted messages to manipulate a file system directly, but usual computer users today cannot/do not do that. Our method cannot identify this difference, because use cases except 'delete messages' in Fig. 7 are not included in a surrounding of 'delete messages' by its definition.

4. Conclusions and Discussion

In this paper, we present a method to identify stakeholders, their preferences about NFR by comparing use case diagrams. Our method can be used to prioritize NFR and to identify stakeholders. In WinWin approach [17], stakeholders should explore trade-off among their goals (win conditions). The method proposed here can contribute to this task, because we can know relationships among NFR goals by variables. In DDP [18], exhaustive elicitation of requirements and risks is important. Our method will contribute to such elicitation process, because changing the values of variables related to NFR can generate unexpected situation. Our method also can be used in AGORA requirements analysis method [19] that clarify the conflicts among stakeholders.

Our method is based on the package oriented requirements elicitation method PAORE [20]. However, we did not handle NFR and preferences of stakeholders in PAORE. Simple functional decomposition represented by tables is used in PAORE. However we could not naturally handle actors that are related to systems. As a result, the method in this paper overcomes several weak points of PAORE.

To use our method efficiently, we have to develop supporting tools. At least, we need database for use case diagrams and comparing tool for the diagrams.

Currently, we do not handle internal structure of each use case, for example, scenario descriptions. There already exists a research focusing on differences in scenarios [21]. There is a possibility to find more stakeholders and NFR by using the technique, but it takes additional efforts. In the future, we would like to examine scenario based techniques contribute to finding more stakeholders and NFR cost-effectively.

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

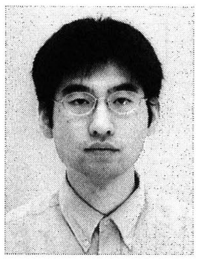
There are few related works about requirements elicitation using GQM and quality standards. GQM, QFD (Quality-Function Deployment) and requirements specification are used for quality control [22]. Quality standards are used for package selection [23].

References

[1] L.A. Macaulay, Requirements Engineering, Applied Computing, Springer, 1996.
[2] I. Alexander and S. Robertson, "Understanding project sociology by modeling stakeholders," Software, vol.21, no.1, pp.23–27, Jan./Feb. 2004.
[3] N. Rozanski and E. Woods, Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, Addison Wesley, 2005.
[4] V.R. Basili and D.M. Weiss, "A methodology for collecting valid software engineering data," IEEE Trans. Softw. Eng., vol.SE-10, no.6, pp.728–738, Nov. 1984.
[5] "IEEE guide for developing system requirements specifications,"

Dec. 1998. IEEE Std 1233-1998, ISBN 0-7381-0337-3 SH94654 (Print).
[6] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos, Non-functional Requirements in Software Engineering, Kluwer Academic Publishers, 2000.
[7] T. Systa, "Understanding the behavior of Java programs," Seventh Working Conference on Reverse Engineering, pp.214–223, 2000.
[8] M. Glinz, "A lightweight approach to consistency of scenarios and class models," 4th International Conference on Requirements Engineering, pp.49–58, 2000.
[9] M. Fowler and K. Scott, UML Distilled, Applying the Standard Object Modeling Language, 1st ed., Addison-Wesley, 1997.
[10] A. Cockburn, Writing Effective Use Cases, Addison-Wesley, 2000.
[11] International Standard ISO/IEC 9126-1, "Software engineering - Product quality - Part 1: Quality model," 2001.
[12] B. Potter, J. Sinclair, and D. Till, An Introduction to Formal Specification and Z, Prentice Hall International, 1991.
[13] The RAND MH Message Handling System UCI version 6.8.3, "http://www.ics.uci.edu/%7Emh/."
[14] AL-Mail32, "http://www.almail.com/." Japanese page only.
[15] The Mutt E-Mail Client, "http://www.mutt.org/."
[16] M. Saeki and H. Kaiya, "Transformation based approach for weaving use case models in aspect-oriented requirements analysis," 4th AOSD Modeling With UML Workshop, Oct. 2003. Joint Workshop of UML2003, http://www.cs.iit.edu/~oaldawud/AOM/index.htm
[17] B. Boehm, P. Grunbacher, and R.O. Briggs, "Developing groupware for requirements negotiation: Lessons learned," IEEE Softw., vol.18, no.3, pp.46–55, May/June 2001.
[18] S.L. Cornford, M.S. Feather, J.C. Kelly, T.W. Larson, B. Sigal, and J.D. Kiper, "Design and development assessment," Proc. Tenth International Workshop on Software Specification and Design (IWSSD'00), pp.105–114, 2000.
[19] H. Kaiya, H. Horai, and M. Saeki, "AGORA: Attributed goal-oriented requirements analysis method," IEEE Joint International Requirements Engineering Conference, RE'02, pp.13–22, Sept. 2002.
[20] J. Kato, M. Saeki, A. Ohnishi, M. Nagata, H. Kaiya, S. Komiya, S. Yamamoto, H. Horai, and K. Watahiki, "PAORE: Package oriented requirements elicitation," Proc. 10th Asia-Pacific Software Engineering Conference (APSEC 2003), pp.17–26, Chiang Mai, Thailand, IEEE Computer Society Press, Dec. 2003.
[21] M. Makino and A. Ohnishi, "A scenario generation method using a differential scenario," ICSOFT, pp.279–282, 2006.
[22] S. Szejko, "Requirements driven quality control," COMPSAC'2002, pp.125–130, 2002.
[23] X. Franch and J.P. Carvallo, "Using quality models in software package selection," Software, vol.20, no.1, pp.34–33, Jan./Feb. 2003.

Haruhiko Kaiya is an associate professor of Software Engineering in Shinshu University, Japan. http://www.cs.shinshu-u.ac.jp/~kaiya/



Akira Osada is a graduate school student in Shinshu University, Japan.

Kenji Kaijiri is a professor of Software Engineering in Shinshu University, Japan.