PAPER

Refining Behavioral Specification for Satisfying Non-functional Requirements of Stakeholders

Haruhiko KAIYA†, *Nonmember and* **Kenji KAIJIRI**†, *Member*

SUMMARY System specifications should be refined to meet stakeholders' requirements as much as possible, because the first specification does not satisfy all stakeholders in general. This paper presents a procedure to refine behavioral specification to satisfy stakeholders. Non-functional requirements are used for checking stakeholders' satisfaction. With this procedure, stakeholder-dissatisfaction can be reduced and new possibilities to satisfy or dissatisfy other stakeholders can be found, since a modification to cancel dissatisfaction can sometimes influence the satisfaction of the others.

key words: *Requirements Elicitation, Requirements Change, Non-Functional Requirements, Sequential Diagram.*

1. Introduction

One of the reasons that it is difficult to elicit system requirements is that it is not easy for users and/or customers to state what they want clearly [1]. Merely representing a current task or an intended system does not help them identify their requirements but showing task changes does. A system's non-functionalities are particularly useful in this respect. For example, response time becomes worse, usability goes up, confidentiality is lost and so on.

Many people participate in or are related to system development, and each may have different requirements of the system. Such people are called stakeholders [2] in this field. Some stakeholder requirements conflict with others. Therefore, we should coordinate their requirements as much as possible, or we should find a trade-off among their requirements. To coordinate the requirements among stakeholders or to find a trade-off, criteria shared by the stakeholders are required. Non-functional requirements can play this role, because non-functional requirements influence the concerns of most of stakeholders.

As mentioned above, many kinds of stakeholders participate in one system development. Therefore, requirements eingineers want to elicit requirements in parallel if possible. To enable parallel elicitation processes, they need a tool to merge several specifications. They also need a tool to compare one specification with another because a pair of such specifications is not always consistent and therefore, a specification may have to be foregone depending on the circumstances.

This paper proposes a procedure for requirements engineers to elicit requirements specification with its evaluation of stakeholders. In this procedure, a specification is stepwise refined to meet the non-functional requirements of stakeholders. Because each specifcation is scored based on the evaluations of stakeholders, they can be easily compared to select the most suitable one. To enable requirements engineers to elicit the requirements simultaniously, this procedure also provides a procedure to merge specifications that are refined in different ways.

This procedure focuses on a behavioral aspect of requirements, especially the ordering of messages among the objects in an intended system and system users. Therefore, this procedure is suitable for interactive systems, such as Web based information system. It is assumed that several kinds of stakeholders relate to system development, use and maintenance, and that requirements engineers can contact them. Therefore, this procedure is suitable for the custom-made software rather than the off-the-shelf software.

The rest of this paper is organized as follows. Section 2 defines the requirements types for categorizing stakeholders's desire. Section 3 introduces the procedure for refining specifications stepwise, by evaluating the value of each refinement and merging several results of refinements. Section 4 presents an example for applying the procedure to the program of the program chair's task of the technical conference. Section 5 compares the results with other results, and finally, the results are summarized and future work is discussed.

2. Requirements Types based on Non-functional and Ease Requirements

As mentioned above, non-functional requirements (NFR) are used for measuring stakeholders' satisfaction to a behavioral specfication. In [3], NFR are categorized as follows;

- Performance
  - Time: Response Time, Throughput, Process Management Time
  - Space: Main Memory, Secondary Storage
- Cost
- User-Friendliness
- Security
  - Confidentiality: Guarding against unauthorized disclosure.
  - Integrity: Accuracy and Completeness.
  - Availability: Guarding against interruption of service.

These non-functional requirements show how a stakeholder

†Department of Information Engineering, Faculty of Engineering, Shinshu University. 4-17-1 Wakasato, Nagano City, JAPAN.

is satisfied or dissatisfied by an intended system represented in a specification.

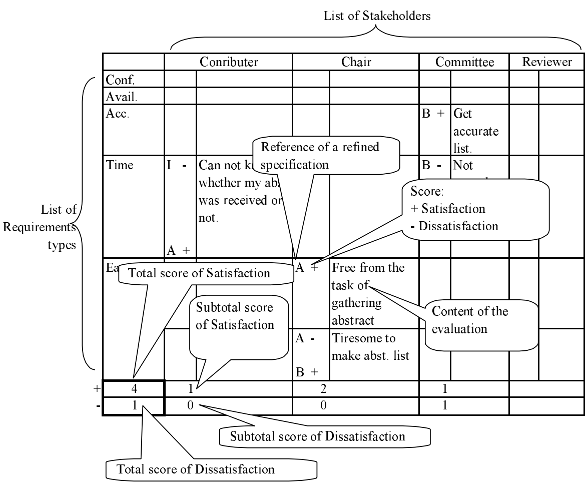
The following requirements are also used for measurement.

- Ease: This is *not* the same as user-friendliness in [3], but means the delegation of a task from users to systems. A computer system is essentially expected to perform tasks instead of human workers and users, to reduce their load by automating these tasks. Functions that can be performed by a computer system instead of by system users are represented in ease requirements. Ease requirements reflect the change in the boundary between users and systems and essentially follow the trade-off among the stakeholders. For example, as the amount of ease requirements increases, users become comfortable but it becomes hard for developers. In addition, because a computer system does not always perform the functions in ease requirements exactly in the same way as a human, some stakeholders lose the advantages of functions that were performed by human workers. For example, a human worker can handle unexpected exceptions manually, and sometimes accept ill-formed inputs. However a system sometimes can not handle such exceptions, and only accpet well-formed inputs.

These requirements are called *requrements types* in this paper.

3. The Procedure: Dissatisfaction Driven Approach

Table 1 Legend of an Evaluation Table



This section introduces steps of the procedure to refine specification. The outputs of this procedure are the refined specifications written in a sequence diagram and *evaluation tables* as shown in Table 1.

An evaluation table is used to store the evaluations of each stakeholder. The evaluations are also categorized by requirements types as shown in Section 2. The rows in an evaluation table corresopond to requirements types. The columns of the table correspond to the kinds of stakeholders. Each cell in the table shows a stakeholder's evaluation by the type of requirement. Each cell is labeled by three attributes; a reference of a refined specification, a score and the content of the evaluation. Currently, the score is either '$+$' or '$-$'; the former shows satisfaction and the latter shows dissatisfaction.

Stakeholder's scores are calculated as follows.

- Determine the satisfaction and dissatisfaction scores respectively.
- Add the number of '$+$' as the satisfaction score.
  In this manner, we add the number of '$+$' which is paired with '$-$'. This $+$ score is used to represent that a refinement complements an existing dissatisfaction. The complemental refinement is regarded as the discovery of satisfaction in this procedure.
- Add the number of '$-$' which are not paired with '$+$'.
  The total score of dissatisfaction acts as a flag to show whether a current specification should be still refined.

If a cell is labelled by '$-$', the current specfcation should be refined to cancel the evaluation. This is the basic concept of this procedure, namely, *a dissatisfaction driven appoarch*. Figure 1 shows an overview of this procedure. This procedure is explained in detail in Section 3.1, an iterative step. In an iterative step, one requirements engineer with several stakeholders refines specification and evaluation table stepwise.

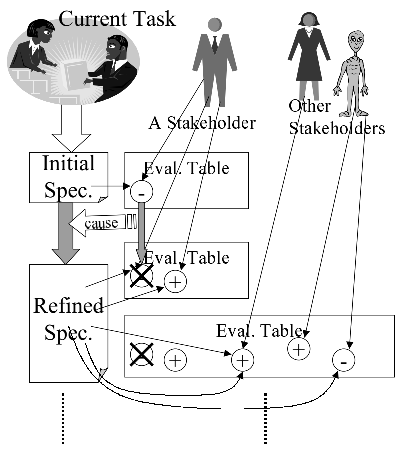


Fig. 1 Overview of an Iterative Step

This procedure also has another step, namely, a merge step as shown in Figure 2. In the merge step, engineers merge two different refined specifications safely, therefore more than one requirements engineers can work simultaneously and the procedure scales along the size of a specification.

3.1 Iterative Step

Iterative steps are explained in detail as shown in Figure 1. This procedure is accomplished by a requirements engineer.

1. Generate a sequence diagram of a current task. This diagram is an initial specification.
2. Let a stakeholder find his dissatisfaction of a current task, and categorize it into requirements types in Section 2. Write '−', the reference of the evaluated specification and the content of his dissatisfaction in a suitable cell on the evaluation table.
   For example, if an operator is dissatisfied by a certain part of specification X because he has to wait a long time, fill (X, '−', 'Wait long time') in the cell of 'an operator' by 'Time'.

Table 2 Example of Evaluation Table

	Operator		Cust	
Conf.				
Avail.				
Acc.				
Time	X - Y +	Wait long time		
Ease				

3. Refine the specification to cancel the dissatisfaction. Validate the cancellation by showing refined specification to the dissatisfied stakeholder. Add the reference of refined specification, say 'Y' in the example in Table 2, and '+' in the cell with the canceled evaluation as shown in Table 2 if the cancellation is validated.
4. Show the refined specification to the other stakeholders, and let them find their satisfaction and dissatisfaction. If satisfaction is found, fill it in on the evaluation table. This can be regarded as a gain of the previous refinement. If disatisfaction is found, fill it in on the evaluation table as in step 2 and go to step 3 to cancel it. If disatisfaction is not found, go to step 2 or step 5.
   Because a specification, which represents the procedure of their task, is slightly modified by the previous step, the other stakeholders can easily find their satisfaction and/or disatisfaction.
5. Sum up the score of '+' evaluations. This score shows the worth of refined specification.
   You may give priority to certain requirements or a certain stakeholder to synchronize the results with the real world.

As a result of this step, an engineer and his stakeholders obtain the sequence of refined specifications, each of which has its evaluation table.

From the procedure of the iterative step, it can be seen that a sequence of refined specifications is not always totally ordered. For example, if two refinements are applied to a specification and there is no intersection between the refinements, these two refinements may be applied in any ordering.

3.2 Merge Step

Two refined specifications may be merged, if a pair of the specifications passes *a horizontal check* and *a vertical check* on sequence diagrams as shown below. The merge step consists of these two checks. Because both checks depend on a representation of a specification, these checks should be redeveloped when we use another representation. A concrete example is shown in the next section.

The inputs of these checks are two sequences of refined specifications which share an initial specification, and evaluation tables of each specification. The outputs are a merged specification and a merged evaluation table. Figure 2 shows the outline of a merge step.

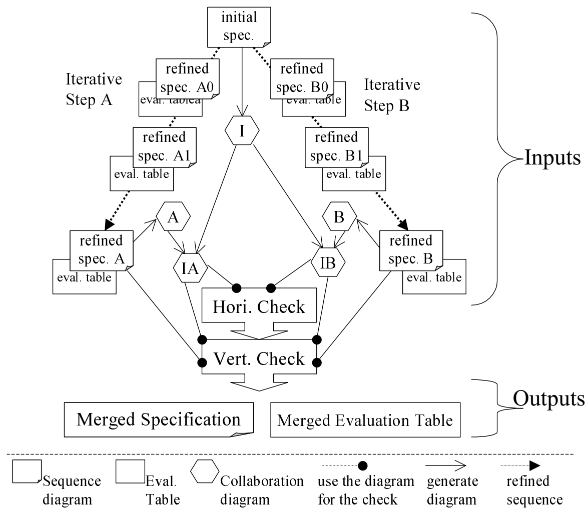


Fig. 2 Overview of Merge Step

If two such sequences in Figure 2 are merged successfully in a merge step, the merged specification is a union of refined specification A and B, and the merged evaluation table is a union of evaluation table A and B. Therefore, the total satisfaction score of the merged evaluation table is the sum of total satisfaction score of the evaluation table A and B, if there is no intersection between the evaluation table A and B. If the sequences cannot be merged, select one side by refering to the score of each specification.

3.2.1 Horizontal Check

On a sequence diagram, a message shows the relationship among objects. At least, if a source or a destination of a message in an initial specification represented by a sequence diagram is modified in different ways in each refined specification, two refined specifications cannot be safely merged. Check this point in the horizontal check. Collaboration diagrams generated from refined sequence diagrams are used because the topology among the objects in a sequence diagram may be the only focus.

1. Write three collaboration diagrams, namely A, B, I, corresponding to an initial sequence diagram and two refined sequence diagrams as shown in Figure 2. Collaboration diagrams can easily generated from a sequence diagram.
2. Create a copy of diagram I and mark the difference between I and A on the copy. This copy is called IA. The differences to be marked are as follows.

    - Cut path: an arc on IA that does not exist on A.
    - Mod path: an arc on IA that is modified on A, e.g. the ordering of the messages on the arcs are changed.
    - Add path: an arc on IA that exists on A but does not exist on I. Therefore, the add path should be added to IA before marking.

    These marks correspond to the modified parts of an initial specification. Develop IB in the same way.
3. If there are no overlaps between a mark set on IA and a mark set on IB, the pair of specifications passes the horizontal check. Intuitively, no path is modified or cut in both specifications.
4. If there are such overlaps, shorten the sequence of refinements of A or B, and retry the check. Select the sequence which is shortened as follows.

    - Pay attention to the last element of the shortened sequence. The element consists of the pair of refined specification and its evaluation table. Total dissatisfaction score of the evaluation table should be zero. If the score is not zero, shorten the sequence again.
    - Shorten the sequences to eliminate the overlap between the mark sets. As mentioned in the last part of Section 3.1, a sequence of refinements is not always totally ordered. Therefore, there are several choices of which refinement step is abandoned.
    - If the overlap between the mark sets is eliminated by shortening either sequences, the shortened sequence may be freely selected. However, this procedure recommends selecting a shortening where the loss of total satisfaction score is smaller.

    Of course, the score of refinements is decreased if the sequence is shortened.

Examples of the horizontal check are shown in Section 4.3.1 and 4.3.2. Because the shortest sequences of both refinements, each of which consists of only an initial specification, is identical, this procedure is always terminated.

Figure 3 shows the distributions of cut, mod and add paths with respect to specification A. The intersection area in this Figure can be divided into two parts; one for mod paths and another for paths that are completely shared with the initial specification and specification A.

Figure 4 shows the posibilities that paths in specification B exist in each area in Figure 3. If a pair of specifications A and B has passed the horizontal check, mod, add and cut paths in specification B do not exist in $\boxed{B_1}$, $\boxed{B_2}$ and $\boxed{B_5}$ in Figure 4. Therefore, the paths excluding those in $\boxed{B_1}$, $\boxed{B_2}$ and $\boxed{B_5}$ may be checked. Paths in the area of $\boxed{B_3}$ are shared by all three specifications; initial specification, specification A and B. Paths in the area of $\boxed{B_4}$ are shared by initial specification and specification A, but are modified or cut in specification B. Paths in the area of $\boxed{B_6}$ are add paths of specification B.

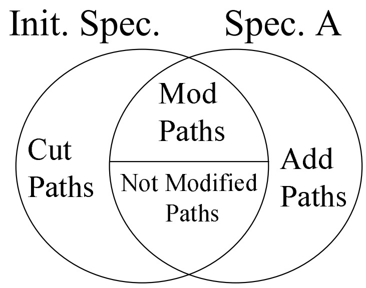


Fig. 3 Differences between init. spec. and spec. A

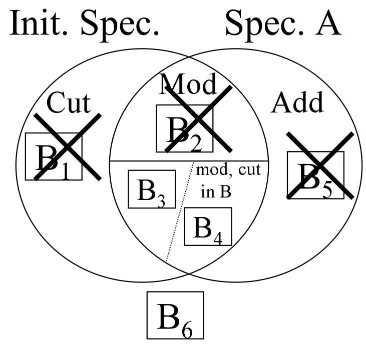


Fig. 4 Possible Area where the Parts of Specification B can Exist

3.2.2 Vertical Check

The horizontal check guarantees that there are no direct inconsistent modifications between specification A and B in Figure 2. However, differences between the initial specification and specification A can effect parts in B that do not appeare in the initial specification. These parts in B exist in the area of $\boxed{B_4}$ or $\boxed{B_6}$ as shown in Figure 4.

For example in Figure 5, assume that an input message to a person is slightly postponed in specification A, and that an output message that depends on the input is also advanced in specification B. If specification A and B are merged, dependency between the input and the output can be inconsistent. In other words, the output can be sent before the input!

We will examine the area of $\boxed{B_3}$ in Figure 4, that is the rest area where no different paths among the three specifications; the initial specification and specification A and B. Because any part in $\boxed{B_3}$ is also a part of specifcation A, any

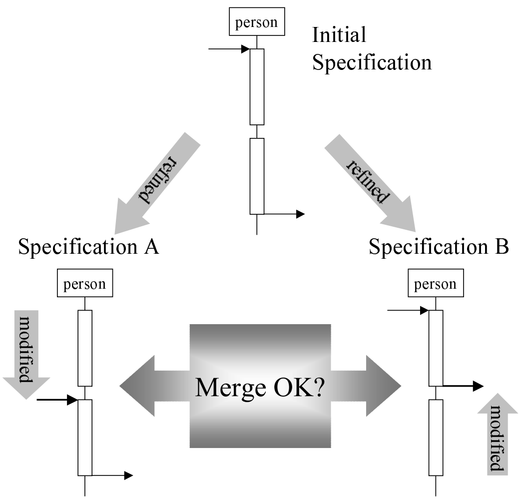


Fig. 5 Example of Causal Dependency

part in $\boxed{B_3}$ is consistent with specification A. Therefore, inconsistency is not a concern in these cases. As a result, the cases in $\boxed{B_4}$ or $\boxed{B_6}$ may only be checked.

Here, we show the procedure to check whether differences between initial specification and specification A can give some effects to parts in B.

1. Pick up the paths in IA marked with cut, mod, or add. We call the set of the paths as *IAM*.
2. Pick up the paths in IB marked by mod or cut (corresponding to $\boxed{B_4}$) or add (corresponding to $\boxed{B_6}$). We call the set of the paths as *IBM*.
3. Find the objects that meet the following conditions.
   - The object appears in both initial specification and refined specification B.
   - The object is a source of a message in *IBM*.

   We call the set of such objects as *ObjB*.
4. Find the objects that meet the following conditions.
   - The object appears in both initial specification and refined specification A.
   - The object is a destination of a message in *IAM*.

   We call the set of such objects as *ObjA*.
5. Exit this procedure as a success, if no intersection exists between *ObjA* and *ObjB*. We call the intersection set between *ObjA* and *ObjB* as *ObjAB*.
6. Check each object, say *obj*, in *ObjAB* as follows.
   a. From *IBM*, find messages that source is *obj*. Check each message, say *m*, as follows.
      i. From *IAM*, find messages which destination is *obj*. We call the set of messages md.
      ii. Exit this procedure as a success, if no message in md is occurred after the message *m*.
      iii. Check the causal relationships between the messages in md and the message *m*. Because no formal specification is specified in the sequence diagram, we should check such causal relationships manually.
   b. Exit this procedure as a success, if we can not find the messages that source is *obj* from *IBM*.
7. If this procedure does not finished as a success, shorten the one of sequences in the same way in the horizontal check, and do the horizontal and vertical check again.

We should check whether differences between initial specification and specification B can give some effects to parts in A in the same way. There is an example of a vertical check in Section 4.3.3.

3.2.3 Merging more than Three Sequences

Because this procedure allows more than three engineers elicit requirements simultaniously, more than three refined sequences can be merged with this procedure. The procedure for merging more than three sequences is as follows.

1. Select two sequences out of the set of refined sequences.
2. Merge selected sequences according to the Section 3.2.1 and 3.2.2.
3. Stop this procedure if only one sequence is in the set, otherwise, go to the next step.
4. Select two sequences out of the set and merged sequence in step 2.
5. Go to step 2.

This procedure does not tell which pair of refined sequences should be merged first, because the priority of each refined sequence cannot be decided simply. Priority of a refined sequence depends on many factors, such as total satisfaction score of the last elements in the sequence, the stakeholders participating in the elicitation process of the sequence, the length of the sequence, the skill of an engineer who elicits the sequence and so forth.

4. Example

This section presents an example for applying this procedure to the task of a program chair in a technical conference such as JCKBSE. The example here is based on a problem of Requirements Engineering Working Group, SIGSE of IPSJ [4]. This problem only includes both general goals, developing a system for supporting a program chair of the technical conference, and a description of the ordinary and traditional work of the chair. An *initial specification* is developed from this problem. Figure 6 shows a part of the initial specification written in a sequence diagram.

Using the initial specification as a start point, refine the diagram stepwise. Two different sequences of refinements are shown in Figure 7, one is the sequence A, B, C, D in Section 4.1 and another is X, Y, Y in Section 4.2.

In this example, a merge step between specification D and Z is failed, and a sequence should be shortened for

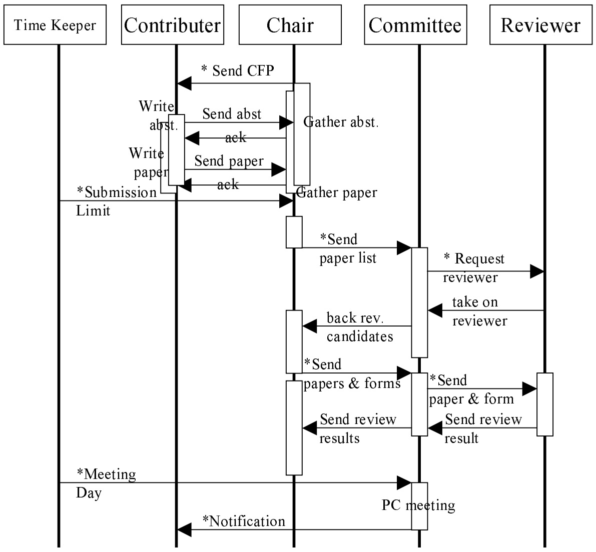


Fig. 6 Sequence diagram of initial specification

merging. The sequence of X, Y, Z is shortened, so that a merge step between specification D and Y is successed.

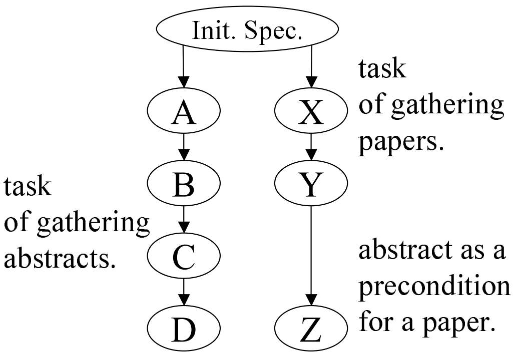


Fig. 7 Two sequences of refinements

4.1 Task of gathering abstracts

4.1.1 Refinement A: Evaluation by a contributor

Table 3 Evaluating init. specification by a contributor

		Conributer		Chair		Committee		Reviewer	
	Conf.								
	Avail.								
	Acc.								
	Time	I -	Can not know whether my abst. was received or not.						
	Ease								
+	0								
-	1	1							

In the first refinement, a contributor is allowed to evaluate the initial specification in Figure 6. The contributor complains that he can not quickly know whether his abstract is recieved successfully or not, because he is impatient and a worrier. His dissatisfaction can be represented by an evaluation table shown in Table 3, which is the result of step 2 in Section 3.1.

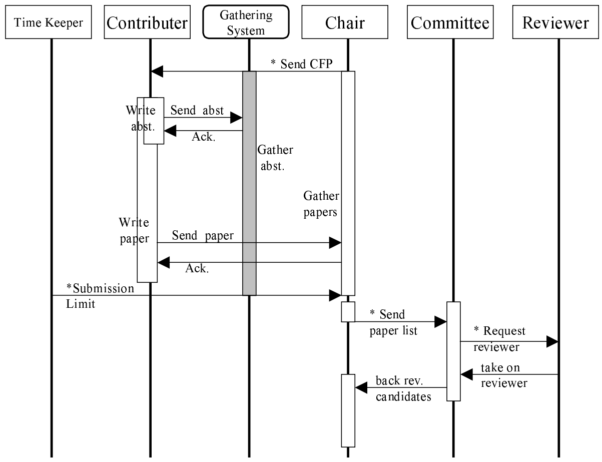


Fig. 8 Refined Specification A

The initial specification is refined to a new one as shown in Figure 8 so that a computer system instead of a chair gathers abstracts. Then the dissatisfaction of the contributor in Table 3 can be canceled as shown in Table 4, because the system can work around the clock and contributers can receive the receipt for their abstracts immediately. This is the result of step 3 in Section 3.1.

Table 4 Exploring other evaluations in specification A

		Conributer		Chair		Committee		Reviewer	
	Conf.								
	Avail.								
	Acc.								
	Time	I - A +	Can not know whether my abst. was received or not.						
	Ease			A +	Free from the task of gathering abstract				
+	2	1		1					
-	0	0							

Because this refinement can give positive or negative influences to the other stakeholders, these influences are explored with the stakeholders. New satisfaction of a chair can be found so that he can become free from the task of gathering abstracts manually. This new satisfaction is also filled in Table 4. Even though this new satisfaction of a chair can be found without the refinement shown in Figure 8, oversight can be prevented with this refinement. This is the result of step 4 in Section 3.1, and as no dissatisfaction can be found, go back to step 2 in Section 3.1.

4.1.2 Refinement B: Evaluation by a chair

A chair evaluates specification A in Figure 8. The chair finds that the task for making the list of abstracts is also tiresome. His dissatisfaction can be represented by the evaluation shown in Table 5. Though he can find this fact in the initial specification in Figure 6, first refinement A in Figure 8, the delegation of a task of gathering abstracts, would contribute to his dissatisfaction in Table 5 as an analogy. This is the result of step 2 in Section 3.1.

Table 5 Evaluating specification A by a chair

	Conributer		Chair		Committee		Reviewer	
Conf.								
Avail.								
Acc.								
Time	I - A +	Can not know whether my abst. was received or not.						
Ease			A +	Free from the task of gathering abstract				
			A -	Tiresome to make abst. list				
+ 2	1		1					
- 1	0		1					

According to his dissatisfaction above, specification A is refined in Figure 8 to delegate the task to a system mentioned by the chair. New refined specification B is shown in Figure 9, and his dissatisfaction can be canceled as shown in Table 6. This is the result of step 3 in Section 3.1.

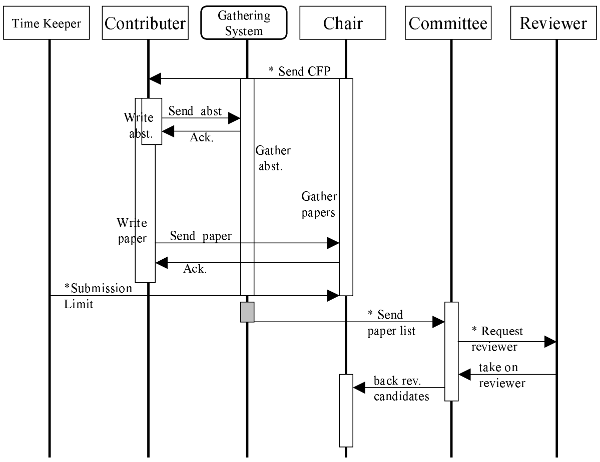


Fig. 9 Refined Specification B

Again, this new specification B is shown in Figure 9 to other stakeholders. Then a member of the committee can find that members can get a more accurate list of abstracts because it is generated not manually but automatically. The specification B in Figure 9 increases its value by finding this fact, therefore this proposing system also meets the stakeholder needs. This is the result of step 4 in Section 3.1.

Table 6 Exploring other evaluations in specification B

	Conributer		Chair		Committee		Reviewer	
Conf.								
Avail.								
Acc.					B +	Get accurate list.		
Time	I - A +	Can not know whether my abst. was received or not.						
Ease			A +	Free from the task of gathering abstract				
			A - B +	Tiresome to make abst. list				
+ 4	1		2		1			
- 0	0		0					

4.1.3 Refinement C: Evaluation by a committee member

A committee member evaluates specification B in Figure 9. The committee member pays attention to his task in the figure, which has not been changed since the initial specification. Because the member plans to ask his fellows to review several papers, he wants to know what kind and how many papers should he process as soon as possible. If he can know this, he can ask his fellows to schedule their review task beforehand. His hope, in other words his dissatisfaction to the specification in Figure 9, can be represented as shown in Table 7, 'B-: Not enough time to find reviewers'. This is the result of step 2 in Section 3.1.

Table 7 Evaluating specification B by a committee member

	Conributer		Chair		Committee		Reviewer	
Conf.								
Avail.								
Acc.					B +	Get accurate list		
Time	I - A +	Can not know whether my abst. was received or not.			B -	Not enough time to find reviewers		
Ease			A +	Free from the task of gathering abstract				
			A - B +	Tiresome to make abst. list				
+ 4	1		2		1			
- 1	0		0		1			

According to this dissatisfaction, specification B is refined to specfcation C in Figure 10, where committee memebers directly receive abstracts from a system. This is the result of step 3 in Section 3.1.

When a contributor looks at this new specification C in Figure 10, he complains that his idea in his abstract can be leaked to others before the deadline of a submission.

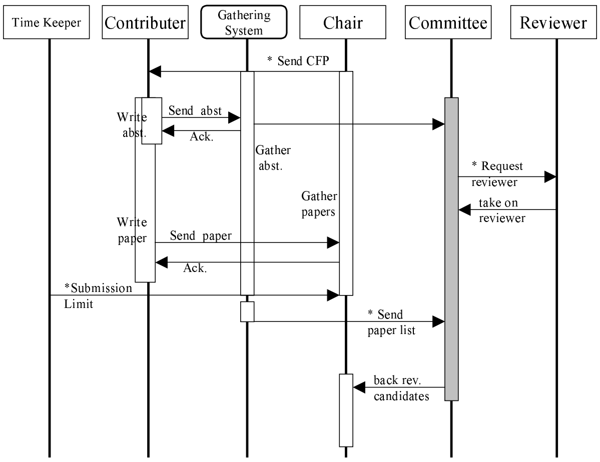


Fig. 10 Refined Specification C

Though the committee members and their fellows are trustworthy enough to glance over the abstracts, this worry of contributors should be removed as much as possible. Because there are several solutions to remove it and because the decision for selecting a solution depends on other situations, what we do now is to record the worry itself. It is recorded as a dissatisfaction of contributors about confidentiality as shown in Table 8. This is the result of step 4 in Section 3.1, go to step 3 because a dissatisfaction is found.

Table 8 Exploring other evaluations in specification C

	Conributer		Chair		Committee		Reviewer	
Conf.	C -	Leak his original idea to the others.						
Avail.								
Acc.					B +	Get accurate list.		
Time	I - A +	Can not know whether my abst. was received or not.			B - C +	Not enough time to find reviewers	C +	Schedule his review task in advance.
Ease			A +	Free from the task of gathering abstract				
			A - B +	Tiresome to make abst. list				
+ 6	1		2		2		1	
- 1	1		0		0			

4.1.4 Refinement D: Evaluation by a contributor again

As mentioned above, a contributor is already aware of his disadvantage as shown in Table 8. A refined specfcation is proposed as shown in Figure 11, where a filter system is introduced, to cancel his dissatisfaction. The role of the filter system is both to prevent abstracts from a leak of ideas and to give a hint to select a suitable reviewer for the paper. This is the result of step 3 in Section 3.1.

As a result, no dissatisfaction remains and no new dissatisfaction is discovered as shown in Table 9. However, it

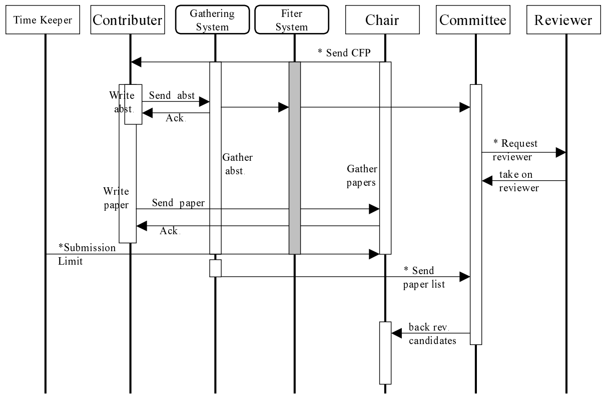


Fig. 11 Refined Specification D

would be difficult to realize the filter system above, therefore this refinement is not desirable for software developers, who are also stakeholders of this system, with respect to the cost requirements. This is not memtioned to keep the example simple.

Table 9 Result of Evaluation of D

	Conributer		Chair		Committee		Reviewer	
Conf.	C - D +	Leak his original idea to the						
Avail.								
Acc.					B +	Get accurate list.		
Time	I - A +	Can not know whether my abst. was received or not.			B - C +	Not enough time to find reviewers	C +	Schedule his review task in advance.
Ease			A +	Free from the task of gathering abstract				
			A - B +	Tiresome to make abst. list				
+ 7	2		2		2		1	
- 0	0		0		0			

4.2 Task of gathering papers and else

Back to the initial specfication in Figure 6, other specification refinement possibilities are considered.

4.2.1 Refinement X: Evaluaton by a chair again

Focusing on the submission of papers, a chair evaluates the initial specification in Figure 6 again. Once the chair has learnt the ease of a system to gather abstracts, he asks the system to gather papers as well. His request is represented in Table 10.

The initial specification is refined to specfication X in Figure 12 to meet the evaluation in Table 10.

A contributor remembers that an introduction of a system contributes to a decrease in the time to get a response

Table 10 Evaluating init. specification by a chair

	Conributer		Chair		Committee		Reviewer	
Conf.								
Avail.								
Acc.								
Time								
Ease			I -	Tiresome to gather papers manually.				
+ 0								
- 1			1					

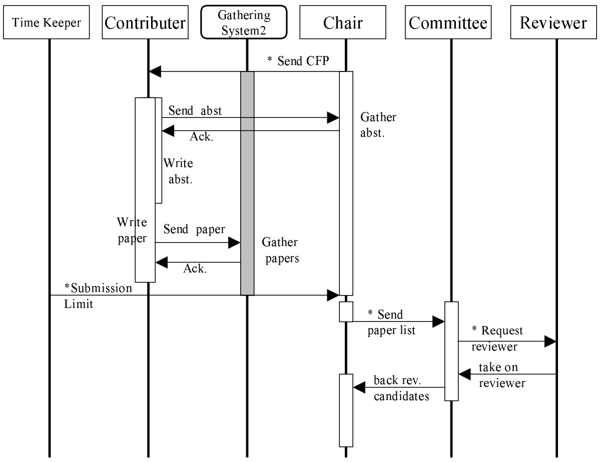


Fig. 12 Refined Specification X

from a receiver as explored in Section 4.1.1. Therefore, the contributor can point out that refined specification X also contributes to his own desire. These evaluations are represented in Table 11.

Table 11 Exploring other evaluations in specification X

	Conributer		Chair		Committee		Reviewer	
Conf.								
Avail.								
Acc.								
Time	X +	Know quickly whether my paper was received or not						
Ease			I - X +	Tiresome to gather papers manually.				
+ 2	1		1					
- 0			0					

4.2.2 Refinement Y: Evaluation by a contributor

A contributor is allowed to evaluate a specification X moreover, and the contributor remembers that his paper was returned from a chair of another conference because of a format error in his electric submission. For example, PDF documents with Japanese fonts are not receieved by a chair of an international conference because most countries except Japan cannot view or print such documents. Therefore, the contributor fills up his dissatisfaction as shown in Table 12 because gathering systems do not always check the format of submitted papers.

Table 12 Evaluating specification X by a contributor

	Conributer		Chair		Committee		Reviewer	
Conf.								
Avail.	X -	Can not know the validity of my paper format immediately.						
Acc.								
Time	X +	Know quickly whether my paper was received or not.						
Ease			I - X +	Tiresome to gather papers manually.				
+ 2	1		1					
- 1	1		0					

To cancel the dissatisfaction above, introduce a format checker in the system as shown in Figure 13.

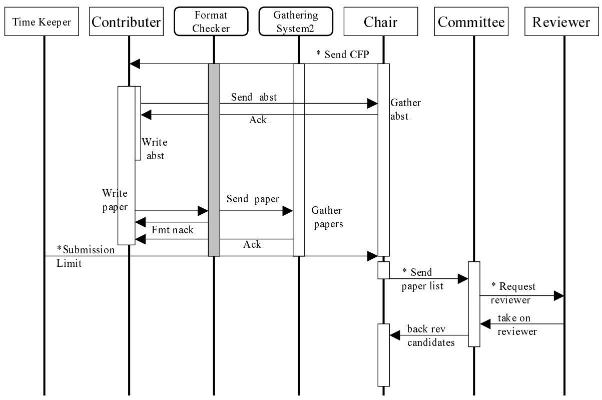


Fig. 13 Refined Specification Y

The refinement in Figure 13 shows another advantage of the gathering system with a format checker, that a chair or others should check the format of electric submission before. Therefore, refinements in specification X and Y give another benefit for the chair as shown in Table 13.

4.2.3 Refinement Z: Evaluation by a chair

In this example, contributors do not have to submit their abstracts before submitting their papers. However, each paper should be paired with an abstract because reviewers are selected based on the contents of abstracts. For preserving this condition, the chair does not send the acknowledgement for the abstracts directly to the contributors, but he registers the data of abstracts to the gathering system2 in Figure 13. Instead of the chair, the system sends the acknowledgement automatically. In addition, the system do not accept the paper that abstract is not been submitted yet. The refined specification along with above is shown in Figure 14.

Table 13 Resulting Evaluation of Y

	Conributer		Chair		Committee		Reviewer	
Conf.								
Avail.	X -	Can not know the validity of my paper format immediately.						
	Y +							
Acc.								
Time	X +	Know quickly whether my paper was received or not.						
Ease			I -	Tiresome to gather papers manually.				
			X +					
			Y +	Free from file format check.				
+ 4	2		2					
- 0	0		0					

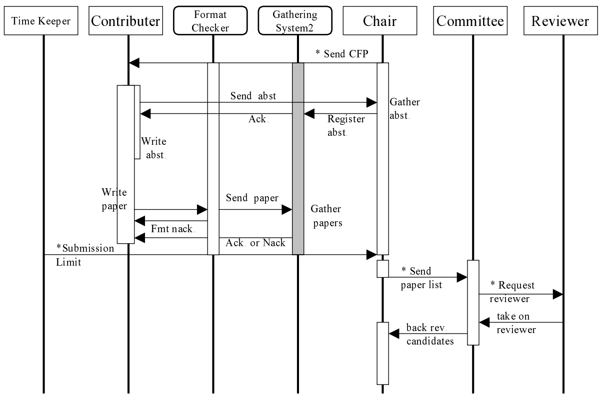


Fig. 14 Refined Specification Z

As a result, the chair can preserve the consistency about the papers and the abstracts, and contributors can know whether they have already submit both. Though the ordering for submitting a paper and its abstract is fixed, it would give no effect to stakeholders. The resulting evaluation table is shown in Table 14.

4.3 Merge two resulting Specifications

Up to here, two different sequences of refinements have been shown, one is sequence A, B, C, D and another is X, Y, Z as shown in Figure 7. As a result of refinements A, B, C, D, the refined specification D is shown in Figure 11 with its evaluation in Table 9. As a result of refinements X, Y, Z, the refined specification Z is shown in Figure 14 with its evaluation in Table 14. Now check whether these specifications can be merged or not.

4.3.1 First Horizontal Check: Failed

Collaboration diagrams are shown in Figure 15, based on sequence diagrams in Figure 6, 11 and 14. As shown in Figure 15, collaboration diagram of specification D has two cut path and one add path. A mark set of specification D consists of these three paths. On the other hand, a collaboration diagram of Z has three cut path and two add path. A mark set of specification Y consists of these five paths.

Table 14 Resulting Evaluation of Z

	Conributer		Chair		Committee		Reviewer	
Conf.								
Integrity			Z +	Guarantee all paper with abstract.				
Avail.	X -	Can not know the validity of my paper format immediately.						
	Y +							
	Z +	Check both abstract and paper are submitted.						
Acc.								
Time	X +	Know quickly whether my paper was received or not.						
Ease			I -	Tiresome to gather papers manually.				
			X +					
			Y +	Free from file format check.				
+ 6	3		3					
- 0	0		0					

Because one overlap path is found between two mark sets, a pair of specification D and Z does not pass the horizontal check. In fact, a message for sending acknowledgement of an abstract is generated by a system in specification D. On the other hand, the message is generated another system. Therefore, we can not merge these two specifications in this stage.

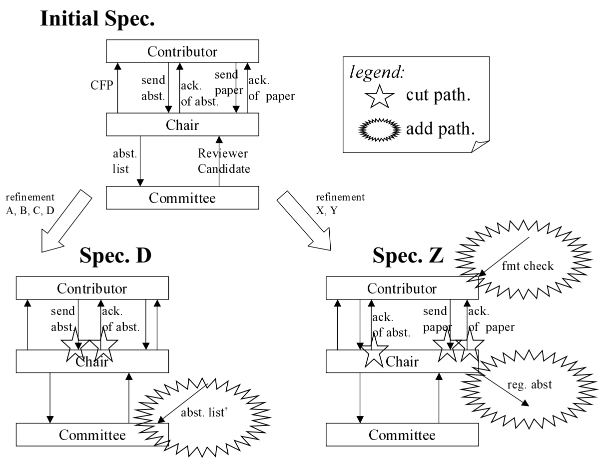


Fig. 15 Horizontal Check between D and Z

If we abandon the overlapped message in the sequence A, B, C and D, we should throw away the most of all refinement steps in the sequence. On the other hand, we only need the last refined step in the sequence X, Y and Z to abandon the overlapped message. Therefore, We shorten the sequence X, Y and Z by a step.

4.3.2 Horizontal Check Again

As mentioned above, the sequence X, Y and Z are shortened, therefore the evaulated pair of specification becomes the pair of D and Y. Collaboration diagrams are shown in Figure 16, based on sequence diagrams in Figure 6, 11 and 13 again. As shown in Figure 16, Y's collaboration diagram has two cut path and one add path. A mark set of specification Y consists of these three paths. Because there is no overlap between two mark sets above, the pair of specification D and Y passes the horizontal check.

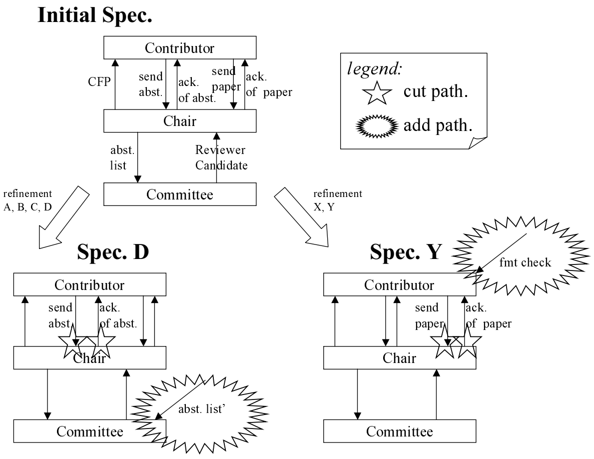


Fig. 16 Horizontal Check between D and Y

4.3.3 Vertical Check

We check whether differences between initial specification and specification Y can give some effects to parts in D according to the procedure in Section 3.2.2. We have following variables in this case.

- $ObjB$ = $ObjA$ = $ObjAB$ = {Contributor}
- m = 'send abst'
- md = { 'ack of paper', 'fmt nack'}

Because messages in md can be occurred after the message m, we should check the causal relationship between md and m as shown in step 6.a.iii in Section 3.2.2. In this case, we successfully finish the half of vertical check because m does not depend on any message in md.

We also check whether differences between initial specification and specification D can give some effects to parts in Y in the same way.

- $ObjB$ = $ObjA$ = $ObjAB$ = {Chair}
- m = 'ack of abst'
- md = {'send paper'}

Because message 'send paper' can be occurred after the message m, we should check the causal relationship between md and m. In this case, we also successfully finish the rest of vertical check because m does not depend on any message in md.

As a result, we find no inconsistency between specification Y and D. Threrefore, a merge of specification D and Y passes the vertical check, and we conclude that we may merge two specification D and Y safety. This implies we can get an evaluation score 11 because of evaluations in both Table 9 and 13.

5. Related Work

This procedure is compared with other similar research results with respect to the following criteria. Table 15 shows the results.

1. Notation for design decisions:
   Many others use graphical notation to represent precise and complex descriptions of decisions. However, it is too complex to use requirements elicitation. Moreover, they can easily spread and it is hard to manage or preview the results. This procedure uses a tabular form to easily manage and preview the results for both engineers and stakeholders.
2. Have a process support or only a notation:
   Some also support the notation of design decision because the notation is simple. Though the notation is simple, an instance of the notation is not always simple. Therefore, an explicit process support is necessary for requirements elicitation.
3. Represent the history of design decisions or not:
   Most support the history of design decision. It seems to be reasonable to record design spaces[5] rather than process history. However, it is useful to log the history of the process when backtracking and redoing the elicitation. In the proposed procedure, backtracking plays a large role in resolving inconsistency as shown in Section 3.2.1.
4. Represent multiple views of stakeholders explicitly:
   Some others also support the multiple view. But explicit agents are needed who have several criteria to the intended system to give priority to the set of criteria. Therefore, it is insufficient to provide only a notion of the evaluation criteria like QOC[5] or gIBIS[6]. Stakeholders in the proposed procedure play these agent roles.
5. Represent the relationship of the views, especially represent the gains and losses of each stakeholder:
   The proposed procedure uses a refined part of a specification as a mediator among the related stakeholders This permits direct understanding of the trade-off for the part with respect to the stakeholders.
6. Represent the relationship between the specification and the decisions:
   As mentioned above, the specification is a good mediator for relating stakeholders in the proposed procedure. Therefore, the relationship between the decisions and the specifications is very natural.

7. Support divergence and convergence of decision process or not:
   The proposed procedure includes a merge step in Section 3.2 and allows the decision processes to be diverse or merged. Moreover, one can easily compare the candidates of decisions because each decision has its own score.
8. Support for resolving inconsistencies:
   In the proposed procedure, it is passive to resolve inconsistencies because the sequence of refinement is backtracked to be consistent as shown in Section 3.2.1. Therefore, most effort for refinement can be omitted.
9. Represent an explicit score of a decision:
   The proposed procedure and QFD[7] provide a numerical score of evaluation. Currently it is too simple because there is no prioritization among the stakeholders.

Quality function deployment(QFD) [7] is a matrix notation to relate the customers requirements with final product control characteristics. QOC[5], gIBIS[6] and Lee's notation[8] are famous and traditional notations for recording design rationale [11]. Inquiry based requirements analysis [9] is a procedure to elicit the requirements by interview. KAOS[10] is a system based on the goal-driven requirements engineering with lightweight formal method.

As shown in Table 15, the proposed procedure is qualitatively better than the other results listed above except KAOS. Because formal methods are introduced in the KAOS method, it is suitable for a later phase of the requirements process, where the requirements are almost clarified. In contrast, the proposed procedure is suitable for an early phase where the requirements are vague. For example, the behavior is rigorously checked by temporal logic in KAOS, whereas it is intuitively checked in a sequence diagram in the proposed procedure.

Because the KAOS system has powerful descriptive capacities, tool support seems to be indispensable. A tool for KAOS is already proposed[12]. On the other hand, the proposed procedure is more portable than the KAOS system, because the procedure can be used without specific tool support. Such portability contributes to start the correctness[13] check, because requirements are not always elicited under the fully equipped environment.

6. Conclusion

This paper introduces a procedure for requirements engineers to elicit the requirements with its reasons from stakeholders stepwise. Several requirements types are used based on the non-functional requirements. These requirements types are useful not only for stakeholders who participate in the interactions during the task, but also for stakeholders who do not, such as developers and managers. For example, there is a trade-off between programmers and users, because an increase of users' desires causes an increase of programmers works. The proposed procedure clearly represents such a trade-off in an evaluation table. The reasons are as follows.

- In an evaluation table, we may add a stakeholder, who do not appear the sequence diagram but can evaluate the interactions in the diagram.
- When a refined specification satisfies a stakeholder, it sometimes dissatisfies the others too much. We may easily find such imbalance numerically, because each evaluation in an evaluation table has the same reference of a refined specification, and is categorized into stakeholders. Conflicts among stakeholders are shown in Table 8.

In an early phase of elicitation, several stakeholders sometimes do not state their satisfaction or dissatisfaction. These stakeholders can remember what they want, when another stakeholder states what he wants and the specification is changed slightly. Requirements engineers may play the part of such a stakeholder in the first stage. The proposed procedure also allows elicitation processes simultaneously, because the results can be merged.

Because the proposed procedure uses a sequence diagram as a specification to attack a behavioral aspect of a system, it is suitable for the interactive system, especially human-computer interactive system. However, it is not suitable for the machine-computer interactive system like embedded real-time system, because we only focus on the ordering of messages.

The notation of the proposed procedure is simple but useful for stakeholders to validate the correctness[13]of specifications on the spot. However, this procedure is not suitable for the cases where requirements engineers can not interactively elicit requirements from stakeholders.

Currently, the scores in an evaluation table are calculated and the specification is evaluated with the table. However, priority or bias should be given to some kind of requirements types, or to some kind of stakeholders, when a specific analysis is needed.

Acknowledgments

This work has been supported by Grant-in-Aid for Encouragement of Young Scientists #12780210, the Ministry of Education, Science, Sports and Culture, Japan. The authors would like to thank the members of the Requirements Engineering Working Group, SIGSE of IPSJ, especially Prof. Motoshi Saeki at Tokyo Inst. of Technology.

References

[1] N. V. Patel. The Spiral of Change Model for Coping with Changing and Ongoing Requirements. *Requirements Engineering Journal*, 4(2):77–84, 1999.
[2] Linda Macaulay. Requirements Capture as a Cooperative Activity. In *Proc. of the IEEE International Symposium on Requirements Engineering*, pages 174–186, San Diego, Jan. 1993.
[3] Lawrence Chung, Brian A. Nixon, Eric Yu, and John Mylopoulos. *Non-functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

Table 15 Comparison with others

criteria	1	2	3	4	5	6	7	8	9
Proposed proc.	Table+Graph	Notation+Process	Yes	Yes	Yes	Yes	Yes	Yes	Yes
QFD[7]	Table	Notation	No	No	No	Yes	No	No	Yes
QOC[5]	Graph	Notation	No	No	No	No	No	No	No
gIBIS[6]	Graph	Notation	Yes	No	No	No	No	No	No
Lee[8]	Graph	Notation	Yes	Yes	No	Yes	No	No	No
Inq[9]	Graph+List	Notation+Process	Yes	No	No	No	No	No	No
KAOS[10]	Logic+Graph	Notation+Process	Yes	Yes	Yes	Yes	Yes	Yes	Yes

[4] Common Problem of Requirements Engineering Working Group, SIGSE of IPSJ (In Japanese). `http://www.selab.cs.ritsumei.ac.jp/~ohnishi/RE/problem.html`, Feb. 2001.

[5] MacLean, A., Young, R. M., Bellotti, V. M. E., and Moran, T. P. Questions, options and criteria: Elements of design space analysis. *Human-Computer Interaction*, 6(3 & 4):201–250, 1991.

[6] Jeff Conklin and Michael L. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. In *CSCW'86 Proceedings*, Dec. 1986.

[7] D. Daetz. QFD: a method for guaranteeing communication of the customer voice through the whole product development cycle. In *IEEE International Conference on Communication*, pages 1329–1333, 1989.

[8] J. Lee. Extending the Potts and Bruns Model for Recording Design Rationale. In *13th International Conference on Software Engineering*, pages 114–125, 1991.

[9] Colin Potts, Kenji Takahashi, and Annie I. Anton. Inquiry-based requirements analysis. *IEEE Software*, 11(2):21–32, Mar. 1994.

[10] Axel van Lamsweerde, Robert Darimont, and Emmanuel Letier. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, 24(11):908–926, Nov. 1998.

[11] Thomas P. Moran and John M. Carroll. *Design Rationale Concepts, Techniques, and Use*. Lawrence Erlbaum Associates, Publishers, 1996.

[12] P. Bertrand, R. Darimont, E. Delor, and A. van Lamsweerde. GRAIL/KAOS: An Environment for Goal-Driven Requirements Engineering. In *Proc. of ICSE'98 Vol. II*, pages 58–62, Kyoto, Japan, Apr. 1998. Poster & Research Demonstrations.

[13] IEEE Recommended Practice for Software Requirements Specification, Oct. 1998. IEEE Std 830-1998, ISBN 0-7381-0332-2 SH94654(Print).

Haruhiko Kaiya is an associate professor of Software Engineering at Shinshu University, Japan

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