Quality Analysis of Information Technology Requirements to Support Knowledge Innovation

Noboru Hattori† *and Shuichiro Yamamoto*

Abstract

At the Research and Development Headquarters of NTT DATA, researchers are investigating whether it is possible to perform a more detailed analysis and evaluation of the quality of the requirements definition process by focusing on the structural components for requirements notation and classifying requirements defects detected during requirements specification reviews according to structural components.

1. Introduction

To achieve knowledge innovation, it is necessary to deal not just with technology but with the existence of needs as well and to understand those needs correctly. For that reason, in the area of information systems development, more is being expected of requirements engineering, which is a technological method for properly extracting and documenting customer requirements and appropriately managing requirements across all development phases.

There are still, however, a number of issues involved in the application of requirements engineering at the development site. Actual system development sites are currently characterized by increasing competition across the industrial world in recent years, as well as greater requirements of information systems. One cannot ignore the percentage of projects that fail because they are unable to meet at least one of the goals of quality, cost, or delivery. For example, the Japan Users Association of Information Systems (JUAS) conducted a survey of user satisfaction with system development and found that of large-scale projects of at least 500 man-months, 46% of the projects were not completed by the deadline and 38% experienced cost overruns [1]. JUAS has also reported on the results of a user survey asking the reasons for missed deadlines. The top two responses were "delays in deciding on required specifications" and "inadequate requirements analysis work". When the number of responses indicating "unsuitable request for proposal content" was included, 42% of the total responses indicated an awareness of problems related to the requirements definition process [2].

We believe that one of the issues related to this situation is the fact that although there is wide awareness that assuring quality is important during the requirements definition phase, the quality control and evaluation methods for the requirements specifications, which are the result of the process, are not very systematic, so it is hard to decide if the requirements definition process has been successful or not. During software development, quality evaluation normally entails a comprehensive evaluation based on both qualitative and quantitative factors. On the quantitative side, the result of the requirements defining process is a document, so in many cases only a rather rough method is used; namely, the number of items pointed out during the requirements specifications review and the number of defects actually discovered per unit page are calculated and compared with figures from earlier projects (below, the number of requirements defects per unit page is referred to as the defect detection rate). Although there have been cases where something close to a system of classifying defects occurring during design processes has been

† NTT DATA
Koto-ku, 135-8671 Japan
Contact: rdhkouhou@kits.nttdata.co.jp

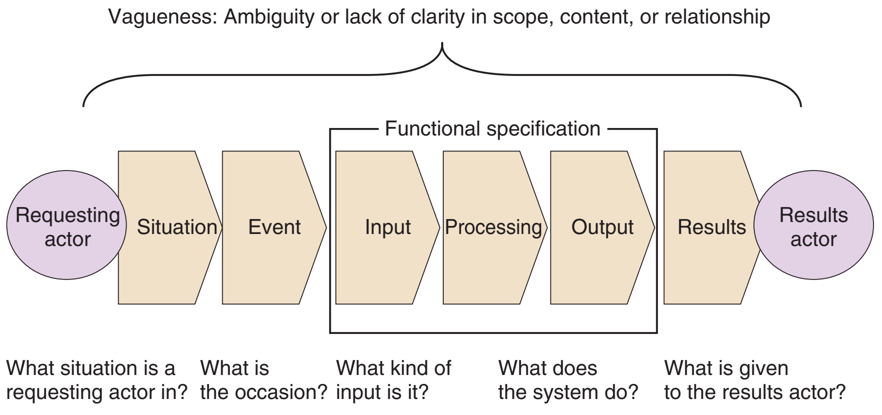
Vagueness: Ambiguity or lack of clarity in scope, content, or relationship
Functional specification
Requesting actor
Situation
Event
Input
Processing
Output
Results
Results actor
What situation is a requesting actor in?
What is the occasion?
What kind of input is it?
What does the system do?
What is given to the results actor?

Fig. 1. Requirements notation structural components.

used, there is currently not much of a system for classifying requirements defects for practical use in the development workplace.

For that reason, we have been pursuing research and development into a method of analyzing the quality of requirements specifications in more detail by focusing on the structural components of the requirements notation, reviewing requirements specifications, and classifying the requirements defects that occur at the requirements management stage according to the structural components in which the problem occurs. To date, actual software was produced first and then requirements defects were often discovered during the testing phase. We believe that by becoming aware of these structural components through the requirements definition phase, we should be able to detect many of these requirements defects earlier during the process of defining conditions.

2. Relationship between structural components of requirements notation and requirements defects

First, we discuss the structural components of requirements notation, which are fundamental to the requirements quality control method we are currently studying, as well as the relationship between these structural components and requirements defects. The discussion concerning these structural components is based on research published by S.Y., to which subsequent results of research have been added [3].

One may think of the functional requirements of software this way: Following some input from the actor under some circumstances on the occasion of some event, a specified process is executed and output, and the expected results are produced for the recipient of those results. This is shown in **Fig. 1**. Looking at the situation in the manner shown in Fig. 1, it is necessary to consider the relationship between requirements defects and the vagueness of the structural components of the requirements notation. The structural components of Fig. 1 are explained as follows.

(1) Requesting actor

The party who asks for a particular function. System users, subsystems, etc. correspond to this actor.

(2) Situation

The situation in which the requesting actor finds itself when it asks for a function to be executed.

(3) Event

The occasion when the function operates. It is necessary to look for the relationship between situations and events in order to determine whether the necessary event has been defined in response to each type of situation.

(4) Input

The input for a function. It is necessary to examine the correspondence to situations and events.

(5) Processing

The operation of the function. This is the portion that is actually achieved through the software, etc. It is necessary to define processing conditions based on input, limitations on processing, processing content, and so on.

(6) Output

The output of the function's operation result. This is achieved through an output screen, sent message, or other means.

(7) Results

The situations that are to be produced for the results actor as the result of processing output.

Table 1. Examples of requirements defects from actual system development projects.

No.	Classification by components	Examples
1	Requesting actor	Those who input value A were changed from customers to operators.
2		Those who operate terminals B were added to users of business C.
3	Situation	In a certain situation, information D may be input, so the corresponding processing was added.
4		Processing when users do not release the connection even if a certain time passed was added.
5	Event	Start time of business processing program E was changed.
6		It is necessary to generate instruction events to end process F internally.
7		Transmission interval of event G was changed.
8	Input	Checkboxes rather than drop down lists were used when users want to select following options.
9		Information H had to be input when logging in to achieve an adequate level of security.
10	Processing	Processing for checking the validity of input value J was added.
11		Default value for a table field K was not defined.
12		A defect in a formula was found and corrected.
13	Output	Defects associated with screen element definitions for output were found and corrected.
14		Information in popup window was changed.
15		Maximum display period of a certain output was defined.
16	Results	Personal data in output display screen had to be sorted into corresponding organizations.
17		New retrieval condition was added to improve screen response time and decrease the number of items displayed.
18	Results actor	User H was added as a recipient of delivered information.

(8) Results actor

The party who receives the results of the function's execution. The results actor may be the same as the requesting actor.

We conducted an analysis of requirements defect reports in an actual system development project in order to determine whether bad characteristics such as vagueness, incorrectness, ambiguity, or incompleteness or inconsistencies of requirements notation structural components can explain requirements defects that occur during actual software development. Analysis was performed on 336 requirements defects occurring after the completion of the requirements definition phase; in other words, those occurring during the design phase or later. Of these, 300 defects (about 90%) yielded analysis results indicating that they were the result of such characteristics in one of the structural components of Fig. 1. Some examples of requirements defects from an actual system development project for each of the structural components with vagueness, incorrectness, etc. are given in **Table 1**. In these examples, the expressions used have been generalized so that specific information could not be identified. The analysis results appear to indicate the possibility that requirements defects could be similarly explained by the incompleteness of these requirements notation structural components in other future development projects as well [4].

3. Analysis of requirements quality categorized by structural components

The results for defect detection rates for requirements categorized by each structural element in each subsystem in the actual system development project are given in **Table 2**. These values are relative to the value for the requesting actor in subsystem A. By comparing these values between subsystems, it may be possible to perform an analysis and make use of the comparative data as shown below, for example.

(1) In subsystem B, the defect detection rates for event and situation are higher than others. It is necessary to review the requirements specifications again and try to exhaustively examine defects concerning these components.

(2) In subsystem C, the defect detection rates for input, processing, and output are high, but the defects in these components may be reasonable if the components were examined exhaustively in the design phase but not in the requirements definition phase. It may be better to review other components, such as the requesting actor and requesting actor's situation, again because they may be related to these defects.

Table 2. Example of defect detection rates sorted by structural components of requirements notation in each subsystem.

	Requesting actor	Situation	Event	Input	Processing	Output	Results	Results actor	Total
Subsystem A	1.00	1.00	12.00	0.00	9.00	0.00	0.00	0.00	23.00
Subsystem B	0.00	3.70	3.23	1.39	3.23	7.85	1.39	0.00	20.79
Subsystem C	0.40	0.80	0.00	5.19	3.19	5.99	0.80	0.00	16.36
Subsystem D	0.00	2.06	1.65	0.41	1.24	6.18	1.65	0.41	13.59
Subsystem E	0.34	3.41	1.02	0.00	1.02	0.68	0.00	0.00	6.48

(3) The total defect detection rate for subsystem E is the lowest among the subsystems. However, the defect detection rate for the situation is the second highest. It may be necessary to try to detect defects concerning the situation because it is a more upstream component of the requirements description. If there is vagueness, incompleteness, etc. about the situation, it will be necessary to reexamine other components corresponding to the situation.

It is not possible to talk about these things by focusing only on the number of requirements defects without actually classifying the defects; this would appear to be a case where an analysis of the requirements quality based on the requirements notation structural components would be effective. By confirming the relationship between these structural components and requirements defects, it should be possible to examine the relative importance of requirements defects and the relative priority of dealing with them and to make revisions to other structural components corresponding to the structural components in which defects have been detected. Here, we have compared values between subsystems, but we believe that with more project data for this case, it should be possible to compare with data from earlier projects and even to detect signs of a project experiencing problems at an early stage.

4. Future issues

It will be necessary to determine whether it is possible to apply a method of analyzing requirements quality based on requirements notation structural components in actual development projects. It will also be necessary to test the hypothesis that if this method is applied to actual development projects from the beginning of the requirements definition phase, it will be possible to detect more requirements defects in the requirements definition phase and thus achieve a reduction in the number of requirements defects that occur after the requirements definition phase.

Even from the perspective of a review when performing requirements specifications reviews, these requirements notation structural components may still be effective. It will also be necessary to further refine the analysis method and make it easier to use by those connected with actual development projects. We hope that integrating such verifications into actual development projects as they are performed in the future will promote the establishment and progress of development-site-oriented requirements engineering.

References

[1] Japan Users Association of Information Systems (JUAS), "Report of Enterprise IT Trend Survey 2006," JUAS, Tokyo, 2006 (in Japanese).

[2] Japan Users Association of Information Systems (JUAS), "Survey of Software Metrics in IT User Enterprise 2006," JUAS, Tokyo, 2006 (in Japanese).

[3] S. Yamamoto, "Serial Requirements Engineering No. 38, Vagueness of Requirements," Business Communication, Vol. 44, No. 12, pp. 68–72, Business Communication, Inc., Tokyo, Dec 1, 2007 (in Japanese).

[4] N. Hattori and S. Yamamoto, "Requirements Quality Management based on the Requirements Structure," IEICE Technical Report, Vol. 107, No. 505, SS2007-73, pp. 97–102, 2008 (in Japanese).

Noboru Hattori

Deputy Senior Researcher, Center for Applied Software Engineering, Research and Development Headquarters, NTT DATA.

He received the B.Ec. degree from Keio University, Tokyo, in 1989. He joined NTT DATA in 1989. He has worked in R&D of communication protocols, empirical software engineering and requirements engineering, etc. He is a member of the Information Processing Society of Japan, the Association for Computing Machinery, and IEEE Computer Society.

Shuichiro Yamamoto

Fellow of NTT DATA Research and Development Headquarters.

He received the B.S. degree in information engineering from Nagoya Institute of Technology, Aichi, and the M.S. and D.Eng. degrees in information engineering from Nagoya University, Aichi, in 1977, 1979, and 2000, respectively. He joined Nippon Telegraph and Telephone Public Corporation (now NTT) in 1979 and engaged in the development of programming languages, CASE tools, network-based smart card environments, and distributed application development platforms. His research interests include distributed information systems, requirements engineering, ubiquitous computing, knowledge creation, and knowledge management. He moved to NTT DATA in 2002. He is currently responsible for research projects on systems science for knowledge innovation.