The Analyzing Method of Root Causes for Software Problems

Tomomi KATAOKA*, Ken FURUTO and Tatsuji MATSUMOTO

In this technical paper, the authors propose an analyzing method of the root causes for software problems. To prevent the recurrence of the same problems, it is necessary to logically identify the root causes and take appropriate measures. Therefore, the authors applied "the 5 whys analysis," a technique that has been used mainly in the manufacturing process, to the trouble shooting of software. First, the authors visualized the procedures of software design by arranging documents illustrating work pieces made in each process to find out the procedures that involve errors. Next, they prepared many "5 whys" samples related to software development, so that even inexperienced users can be guided by referring to them. Consequently, the method effectively worked on the software development and successfully shortened the lead time required for identifying errors and analyzing their causes.

Keywords: software, trouble, analyze method, 5 whys

1. Introduction

The quality of software needs to be secured through a proper development process, and that development process must be improved day to day based on the feedback of problems that occurred in actual use. If a bug is found in software, in particular, it is necessary to investigate the root cause of the bug in order to work out a proper measure to prevent it from recurring[1],[2].

On the one hand, "5 whys analysis" is adopted as an analysis approach to identifying any and all root causes of problems[3] systematically. Forming part of total quality management (TQM), this approach is well-known as an improvement method mainly used in the manufacturing industry[4].

Our software development sections have also adopted the 5 whys analysis approach to analyze the causes of bugs. However, there was no specific procedure applicable to software development, and analysis operations depended on analyzers' skills and capabilities. In addition, rework frequently occurred because of failure to obtain satisfactory quality of analysis. As a result, it took time to analyze bugs, and the lead time from the occurrence of a bug to process improvement tended to become long.

We developed a procedure for identifying issues relating to the application of the 5 whys analysis approach to software development and conducting quality analysis in an efficient manner. This is a report on the method.

2. Issues in Applying the 5 Whys Analysis Approach

To develop the procedure for analyzing root causes, we studied the trend of the past application of the 5 whys analysis approach in our company. As a result, we identified the problems in which analyzers were trapped in applying the 5 whys analysis approach to software development. We consequently discovered three major problems peculiar to software development.

The first problem is a case in which 5 whys analysis results are used only to track down the mechanism of causing program bugs (**Fig. 1 (a)**). In example (a), an "error in the initialization of the communication driver" is derived as the factor responsible for the bug event "A specific frame is not transmitted." However, this is merely the cause that manifests the bug event. It leads us to a method for correcting the bug concerned but cannot reveal the factor of the process that caused the bug to exist in the software, making it difficult to work out a proper measure to prevent the recurrence of the bug. In this context, the case shown in **Fig. 1 (a)** is not thoroughly analyzed.

The second problem is a case in which the analyzer performs analysis relying on his/her preconception or impression. In the 5 whys analysis approach, it is important to sort out objects or events that should be examined and grasp only facts in advance[3]. In example in **Fig. 1 (b)**, however, this fact finding is subjective and vague in that the design engineer's feelings and the considerations in the review are mentioned although there is no positive proof that can be considered to be a fact. If a problem occurs to manufacturing equipment in a plant, subjective facts regarded as the causes of the problems can be collected from the manufacturing equipment, materials, or the equipment's operation records. On the other hand, almost all processes of software development are carried out by personnel. Errors in processes are human mistakes in many cases, and fact finding is likely to depend greatly on the subjectivity of the personnel involved in the processes. Facts should, therefore, be verified based on objective positive proof, such as design documents.

The third and last problem is a case in which the investigation is not performed in depth although there are sufficient opportunities to ask why because the downstream processes focus attention on the error in information input by the upstream process (**Fig. 1 (c)**). Certainly, in this case, the process causing the error is identified by scrutinizing the "operation processes" of personnel in charge and tracing back "design documents," or objective positive proof,

from detailed design documents to basic design documents. However, as a result of focusing the "why" analysis step on the identification of the process in which the error penetrated into the product, the indispensable root cause analysis for investigating "why" the wrong operation was carried out is not performed. The steps up to the identification of the process causing the operation error must be completed before a full-scale "5 whys analysis" operation for a deeper root cause analysis.

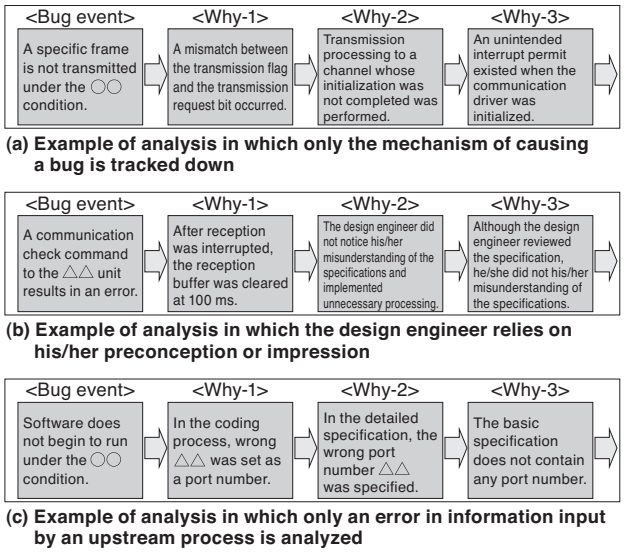

Fig. 1. Examples of past application of 5 whys analysis

3. Development of Problem-Solving Policies

To develop an intended procedure for analyzing root causes when software bugs are found, we studied the problem-solving policies presented in Chapter 2. Our development process is a V-shaped model, and we made endeavors to contour the problem-solving policies to this development process. The V-shaped model is a software development process approach. As shown in **Fig. 2**, this development approach is made up of a process for detailing and implementing requirements, which flows from upstream to downstream on the left side of the V, and a process for verifying that the implemented results properly function as required, which proceeds from downstream to upstream on the right side of the V.

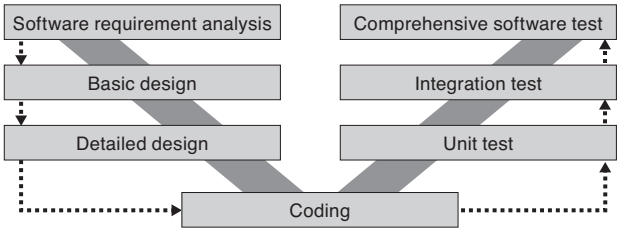

Fig. 2. V-shaped model and development process

[Solution 1] To deal with the first problem "insufficient process-related factor analysis," we developed a fact finding step that does not rely on "program-related mechanism" investigation by separately performing it and "operation process-related error" investigation (**Fig. 3 [Step 1]**)

[Solution 2] To resolve the second problem "insufficient collection of objective information," we collected positive proof that does not depend on subjectivity by listing from registered documents for the project all documents worked out from the initial stage to the program creation stage of the software development process and investigating which process design document contains an error.

[Solution 3] To solve the third problem "insufficient investigation into the root cause in the process," we identified the process in the V-shaped model software development process in which the error was included in software by arranging the design documents collected as described in **[Solution 2]** in order of processes and investigating to which process the design documents were correct and from which process the design documents contained the error (**Fig. 3 [Step 2]**).

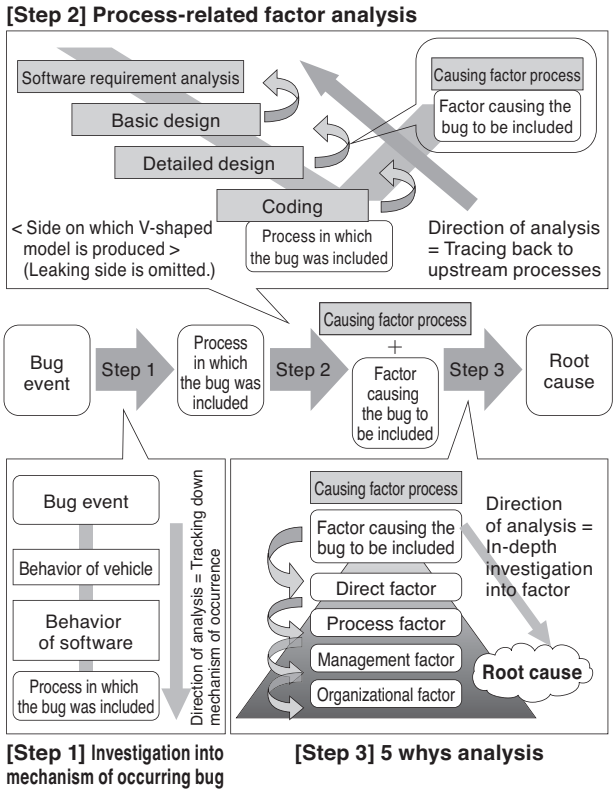

Fig. 3. Image of ideal analysis flow

We developed an approach that enables us to spend more time on analyzing deeper factors behind the occurrence of errors, including process rule-related factors, management-related factors, and the culture of the organization as shown in **Fig. 3 [Step 3]**. That effect is accomplished by performing the analysis procedure steps mechanically.

4. Development of a Procedure for Analyzing the Root Cause When a Software Bug is Found

4-1 Flow of root cause analysis when a bug is found

According to the policies stated in Chapter 3, we developed a procedure for analyzing the root cause when a bug is found. Root cause analysis takes place through Steps 1 to 3 below along the "ideal analysis flow" shown in **Fig. 3**.

(1) **[Step 1]** Analyzing the mechanism of occurrence
Track down the mechanism causing the bug event concerned and identify the location on the program where it is included.

(2) **[Step 2]** Analyzing process-related factors
Trace the design documents along the development process of the V-shaped model to make clear the possible processes in which the bug is included (causing factor processes) and details of the bug (factor causing the bug to be included).

(3) **[Step 3]** 5 whys analysis
Investigate factors in depth, and derive the root cause of the inclusion of the bug.

To reflect the solutions presented in Chapter 3 in the steps shown above, we prepared Tools (1) to (3) shown in **Fig. 4**. Each of the Tools is detailed below.

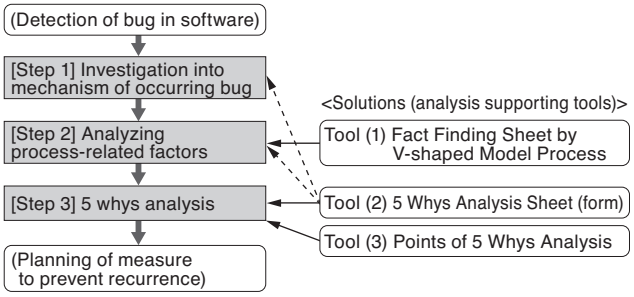

Fig. 4. Procedure for analyzing the root cause when a bug is found and solutions

We prepared Tool (2) "5 Whys Analysis Sheet" shown in **Fig. 5**, which is intended to enable analyzers to perform operations step by step according to a specified procedure. This sheet contains entry columns corresponding to a series of items that will be the outputs of individual steps. It is also designed to prevent the skipping of steps and enable personnel other than the analyzer to objectively check analysis results by filling in these columns.

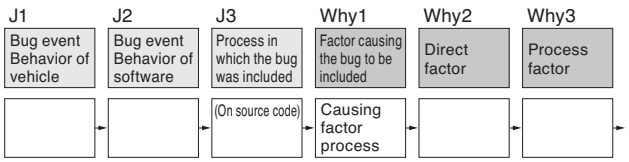

Fig. 5. 5 Whys Analysis Sheet (excerpted)

4-2 Fact finding along the development process of the V-shaped model

In this section, we describe a procedure for accurately grasping when the bug was included, narrowing down the "causing factor processes," and identifying the "factor causing the bug to be included."

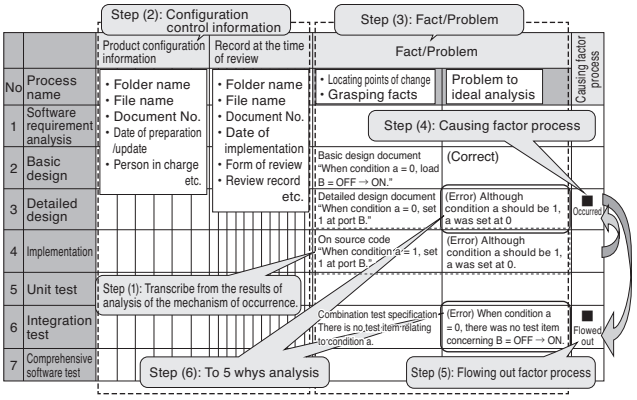

Fig. 6. Fact Finding Sheet by V-shaped Model Process

What is important in this procedure is to correctly investigate any and all facts. We prepared Tool (1) "Fact Finding Sheet by V-shaped Model Process" shown in **Fig. 6** to perform this investigation. This sheet arranges the names of development processes and investigation items in the form of a matrix, so that even inexperienced analyzers can investigate all necessary items without a waste of time and efforts simply by filling in the columns step by step. The specific investigation procedure using the Fact Finding Sheet by V-shaped Model Process is shown below.

(1) Based on the investigation into the mechanism of bug occurrence in **[Step 1]** in **Fig. 4**, enter the location where the bug was included and details in the Fact/Problem column for the coding process.
(2) By making reference to registered documents for the project, prepare products of each process when the bug concerned was included, and enter recorded information at the time of a review.
(3) By making reference to the products collected in (2) above and based on positive proof, check the details of the bug entered in (1) and whether there are related facts and errors, and enter them in the Fact/Problem column.
(4) Trace the upstream processes one by one from the process in which the bug entered in (1) was included by making reference to the information entered in the

Fact/Problem column. If the upstream processes up to the preceding process are correct and there is an error in the present process, determine and identify the present process as the causing factor process. In the example shown in **Fig. 6**, the error "condition a = 0" is included in the "detailed design" process judging from the information entered in the Fact/Problem column. However, since the preceding upstream "basic design" process is correct, the "detailed design" process is considered to be the causing factor process.

(5) Basically, derive the outflowing factor process assuming that the upstream process preceding the verification process (on the right side of the V) corresponding on the V-shaped model to the causing factor process is the outflowing factor process. In the example shown in **Fig. 6**, the upstream "Integration test" process preceding the "unit test" process, which corresponds to the "detailed design" process identified as the causing factor process in (4), is considered to be the outflowing factor process. In the V-shaped model, this concept is based on the principle that the bug should be detected in the upstream process preceding the outflowing (verification) process corresponding to the causing (detailed/implementation) process in which the bug was included. We prepared the flow of processes and that of products shown in **Fig. 7** so that even analyzers unfamiliar to this concept can identify the error according to the procedure. They help derive the outflowing factor process from the causing factor process.

(6) Set the information entered in the "Problem to ideal analysis" column in Why 1 of 5 whys analysis described in the section that follows.

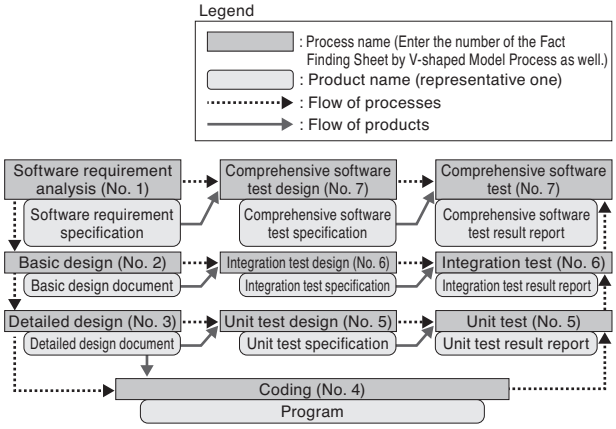

Fig. 7. Flows of processes and products of V-shaped model

With the procedure we have described thus far, you can standardize specific operations in the process in which bugs are included, which have not been subjected to in-depth fact finding and have relied on analyzers' intuition, and objectively judge analysis results with the aid of positive proof.

4-3 Performing 5 whys analysis

Next, perform 5 whys analysis using the 5 Whys Analysis Sheet shown in **Fig. 5** with Why 1 of the causing and outflowing factor processes derived in 4-2 as the starting point.

To derive the root cause, all factors must be investigated in depth in a regular, orderly manner. However, 5 whys analysis has been applied to limited cases in our software development sections, and well-established know-how has not been available. There are pieces of general literature describing know-how about 5 whys analysis, but many of them address samples targeted at problems relating to manufacturing equipment and are not of much help in analyzing software bugs.

To enable even inexperienced analyzers to conduct quality analysis, we compiled errors likely to occur in analysis applied to software development into Tool (3) "Points of 5 whys Analysis" as shown in Table 1, with samples representing experience in software added, by making reference to "Ten Rules on the Performance of 5 Whys Analysis" proposed in Literature (5).

Table 1. Points of 5 whys analysis (example)

No.	Point	Example in software development (×: bad example, ○: good example)
B4	Analyze latent bugs in the source project from the viewpoint of the present project.	(×) The system design of the source project contained the bug. → The factor causing the bug is unknown because it was included in the source project in the process of its development. (○) The system design of the source project contained the bug. → Although necessary products were not available at the time of the carry-over of the system design of the source project, difference verification was not conducted when applying it to the present project.
B8	Analyze the bug from the viewpoint of the mechanism, not as a personal matter.	(×) The person in charge was poor in skill and could not perform in-depth analysis. (○) Behavior when the error occurred was not thoroughly analyzed from the viewpoint of vehicle behavior. (○) The person in charge was assigned as design leader although he/she did not receive necessary education.

5. Application to the Development Process and the Effect of the Introduction of the Procedure

In July 2010, the analysis procedure described in Chapter 4 was finally adopted as a standard rule of the mass production development department after trial application.

Under the guidance and supervision of the software quality assurance (SQA) department, personnel in charge of respective development projects began to apply the procedure to analyze bugs.

According to data on software quality activities compiled by the SQA department at the end of the business year 2010, the lead time from the occurrence of a bug to the completion of root cause analysis was reduced by 40% from the 2009 level. The total lead time from the occurrence of a bug to process improvement was shortened to

approximately one-third of the 2009 level, and measures to prevent the recurrence of problems could be finalized in a shorter period of time.

To analyze the contribution of each approach described herein to software quality improvement, we conducted a questionnaire survey to the SQA department and rated the Tool (1) "Fact Finding Sheet by V-shaped Model Process," Tool (2) "5 Whys Analysis Sheet," and Tool (3) "Points of 5 Whys Analysis" in terms of "reduction in lead time" and "improvement in analysis quality." The results of the survey are shown in Fig. 8. We also discussed the effect of the introduction of the procedure, as well as the results of hearings from the SQA department separately held. The results of the discussion are described below.

"(1) Fact Finding Sheet by V-shaped Model Process" used for "analyzing process-related factors" in **[Step 2]** in 4-1 helps identify the process in which a bug was included based on objective fact verification. This proves that the Fact Finding Sheet enables analyzers to accurately grasp facts and is also effective in improving analysis quality and reducing the lead time (score: 3.4 points in both items). In addition, we received the following comment: "Although we have to take time to complete the Fact Finding Sheet, reanalysis due to insufficient fact verification can be reduced and, as a result, it helps reduce the lead time." This indicates that the improved quality of fact finding makes a contribution to activities to promote measures against the sources of problems.

However, we could not determine whether "(2) 5 Whys Analysis Sheet" and "(3) Points of 5 Whys Analysis" used in "5 whys analysis" in **[Step 3]** in 4-1 are effective in improving analysis quality at present although the former has some effect of reducing the lead time. This is probably because analyzers' experience and familiarity with 5 whys analysis greatly affect 5 whys analysis quality. For this reason, we consider it important to make good use of Tool (3) as an educational material and collect and organize past cases and further accumulate and utilize analysis know-how tailored to our software development.

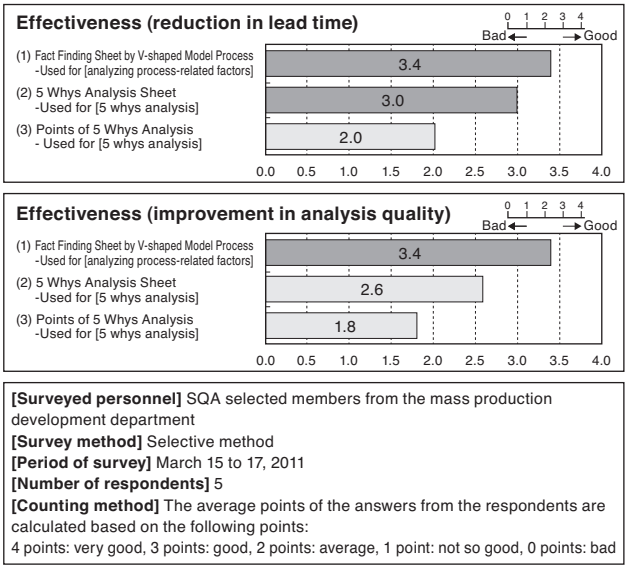

Fig. 8. Results of questionnaire to the SQA department

6. Conclusion

This paper described the 5 whys analysis procedure we developed for the purpose of efficiently conducting process improvement activities for automotive software that is required to offer a high level of quality, as well as the effect of its introduction.

We consider that our developed procedure is applicable not only to automotive software development adopting the V-shaped model but also widely to analysis of the root causes of problems that occur in the development process.

There is still plenty of room for improvement in analysis quality. To further improve analysis quality, we are determined to promote the use of the tools for education and the accumulation and use of analysis cases peculiar to our development process and intended products as know-how for 5 whys analysis.

References

(1) Software Engineering Research Center: "Key Practice of Capability Maturity Model, Ver. 1.1," CMU/SEI-93-TR-25, Carnegie Mellon University (1993).

(2) Satoshi Terakubo et al., "Construction of CMM Level 3 Software Development Process for Automotive ECUs," SEI Technical Review No. 166, pp. 45-50.

(3) Hitoshi Ogura: 'Introduction to Thorough Use of 5 Whys Analysis – Workshop Improvement Beginning from "Why?"' JIPM Solution (1997).

(4) Isao Hayakawa et al., Software Quality Symposium 2008, 'Applying "ask why five times" method on software development,' pp. 185-194.

(5) Hitoshi Ogura: "Make Full Use of 5 Whys Analysis! Complete Drill for Capturing 5 Whys Analysis," JIPM Solution (2002).

Contributors (The lead author is indicated by an asterisk (*).)

T. KATAOKA*

- Software Development Center, AutoNetworks Technologies, Ltd.
  Engaged in the development of in-vehicle software and establishment of its R&D environment.

K. FURUTO

- Manager, Software Development Center, AutoNetworks Technologies, Ltd.

T. MATSUMOTO

- General Manager, Software Development Center, AutoNetworks Technologies, Ltd.