An Approach to Deliver Quality Design by Designers Themselves

- How to Leverage Test Engineer's Viewpoints -

Kaoru Odachi
Beckman Coulter K.K.
Mishima, Japan
kodachi@beckman.com

Michiyo Chiba
SQC Inc.
Tokyo, Japan
mchiba@sqc.co.jp

Noriko Iizumi
Hitachi High-Technologies Corp.
Ibaraki, Japan
Iizumi-noriko@naka.hitachi-hitech.com

Takafumi Tadokoro
Yamatake Corp.
Kanagawa, Japan
Tadokoro-takafumi@jp.yamatake.com

Abstract

In many software develoment projects, most defects are detected in the testing phase. The defects created in the design phase can cause additional time and cost for reworks, as well as the threat to potential business opportunities. Although review is well-known to address this issue and various approaches for reviews with the test engineer's viewpoints, it is not yet easy to practice the review process effectively, due to such reasons as no independent testing organization exists or as the barriers between organizations or stakeholders.

With our strong beliefs, we set our primal goal to leverage the test engineer's viewpoints to prevent missing specifications so that the designers can improve the quality of the specifications by themselves. We carried out two experiments to observe what viewpoints the designers are missing; (1) the test engineer designs the function specification as a designer, and (2) the designer designs test specification as a test engineer using the function specification written by themselves or their own colleagues.

This paper reports the observed results of these experiments and the effectiveness of parallel design to understand the mind-set of designers.

1. Introduction

Many bugs are found in the testing phase in software development, most of which are created in the design phase. They cause extra time for fixings and could be a threat to potential business opportunities. For embedded software, the bugs found after shipment may result in product recalls with major impact on society. Review is well-known to improve the software quality in the design phase, and recent studies report various approaches to leverage the test engineer’s point of view in review meetings. However, in many embedded software development projects, it is quite common that only fewer resources are available and the designers take the responsibility for both designing and testing.

Based on the fact, we came to think about an approach where the designers can improve and deliver design quality by themselves. The true quality is delivered only if the design quality is merged into the function specifications. As one of the characteristics of embedded software development, and in most embedded software development projects, functions are hardly developed from the scratch; they are designed and developed as derivatives, and we know the product quality depends on the quality of the function specifications.

Our study last year shows that 40% of the bugs detected while designing the requirements. With this result, we discussed the possible an approach to improve the quality of requirement specifications by preventing “ambiguous specifications,” “specification errors,” and “missing specification.” [1] This year, we have both designer and test engineer in our team, we further discussed the possible an approach to deliver design quality by designers themselves with test engineer’s viewpoints. What we questioned ourselves was why designers could not find what test engineers could find while writing test specifications.

2. Goal

We set our goal to develop approaches for the designers to improve the design quality by themselves, especially with test engineer's point of view.

3. The Quality of Function Specifications

Design qualities are determined by those of function specifications. That is because function specifications are closely related to later phases of designs such as detailed design, implementation, and testing. They are to give information and share the image of what is to be designed, to implement, and what the test object should be. For the correct information, the design quality should be included in the function specification, which we focused on. In this paper, the definition of "function specifications" represents the documentations generated in the activities "1.6.4 Software Requirements" and "1.6.5 Software System Design," which are defined in the common frame, SLCP-JCF [2].

For the quality of function specifications, "Recommended Practice for Software Requirements Specifications" ("quality characteristics" hereafter) from IEEE 830-1998 [3] is available as a useful reference. First of all, we considered what events may occur if each quality characteristic lacks in function specifications. We, then, rated the rework risks into three levels according to the difficulty of defect detection or fixing. Table 1 describes the ratings for rework risks.

Table1. Events and Rework Risks for Lack of Quality Characteristics in Software Requirements Specifications (IEEE 830-1998 [3])

Quality Characteristics	Brief Description	Events	Difficulty: Detect	Difficulty: Fixing
Correctness	Every requirement stated in the requirements specification is one that the software shall meet.	May develop incorrect software or execute incorrect test	**Middle**	**Hard**
Unambiguousness	Every requirement stated in the requirements specification has only one interpretation.	May develop incorrect software or execute incorrect tests	Easy	Middle
Completeness	All significant requirements from customers and users should be acknowledged and treated.	May lack functions Functions neither to be developed nor tested	**Hard**	**Hard**
Consistency	No subset of individual requirements described in requirements specification conflict.	May develop incorrect software or execute incorrect tests	Easy	Middle
Importance Ranking	Each requirement in requirements specification has an identifier to indicate either the importance or stability of the particular requirement.	May affect the work order but does not lead to major problem.	Easy	Easy
Verifiability	Every requirement stated in requirements specification is verifiable.	Develop testable software Time-consuming, inadequate test execution	Easy	Middle
Modifiability	The structure and style of requirements specification are such that any changes to the requirements can be made easily, completely, and consistently.	Affect the change efforts.	Hard	Middle
Traceability	The origin of each of requirements is clear; it facilitates the referencing of each requirement in future development or enhancement documentation.	Same as above.	Easy	Easy

The quality characteristic that marks harder difficulty in both Defects and Fixing has higher risks for reworks, the table above indicates that lack of "Correctness" and "Completeness" have higher risks for reworks.

- Correctness: Written mistakes are at least written and visible; thus it could be detected and corrected by other parties at a high rate. However, in case that one cannot evaluate the conformance for true requirements, the error may be overlooked and increase reworks and fixings.
- Completeness: Missing specification, including unexpected events, is hardly detected for its missing existence. Fixings will be more and harder.
- Unambiguous and Consistency: Could be noticed out in coding or test executions; thus they have less risk for reworks.
- Importance Ranking: Does not have much impact for reworks.
- Verifiability: Leads less reworks. Helps to find defects in early stage of the process, such as W-model described later.
- Modifiability and Traceability: Lead less reworks using frameworks such as process and/or environment.

Also, the bug analysis on our project shows the missing specifications is the primary cause for bugs, which belongs to "Completeness" in Table1. With these results, we assured ourselves that "Completeness" is responsible for the quality of function specifications.

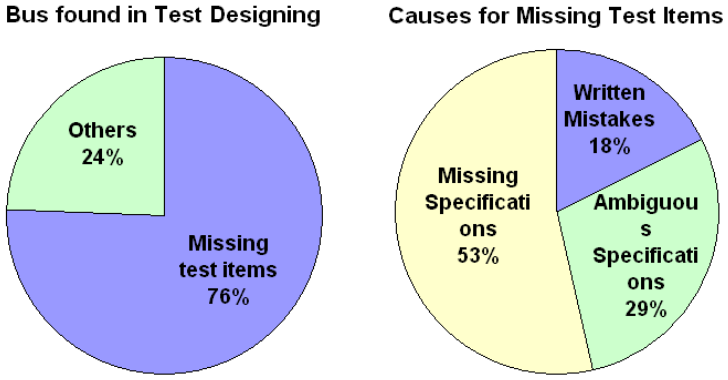

Figure1. Bug Cause Analysis

4. Test Engineer's Viewpoint

Many of recent studies report various approaches are employed to elaborate qualities by leveraging the viewpoint of test engineers in review meetings. That is because test engineers see defects in function specifications when test specifications are designed. In this section, we would like to identify what are the test engineer's viewpoints and the difference from that of designer's. An earlier study [4] introduces the new approach to leverage the test engineer's viewpoint in reviews. It analyzes each functional or non-functional requirement specification using a certain chart, which has cells for the object, actor, context, event, input, behavior, result, and severity for each requirement. By filling in each cell will automatically check the testability for each requirement specification. There was a slight difference in the total number of defects found between the existing reviews and the new approach, however, the study emphasizes the effectiveness in finding the defects with unambiguity. Another earlier study [1] reports the effectiveness of parallel designing of function specifications and test specifications by a single person to recognize errors and improve the quality in the design phase. In this study, a single person used a new process and a tool to do parallel designing of function specification and test specifications, and its case study was reported. It also took advantage of the testability point of view, which is known as W-model [5] but not widely adopted to projects.

Usually, the designers write function specifications paying attention "not to make mistakes and/or not to miss specifications." At this moment, they are focusing on how to describe (design) what are in their minds, it is difficult to notice their misunderstandings or false assumption. In contract, they focus on "finding errors while designing test specification," expecting there are "mistakes or missing specifications" with the test engineer's point of view. In other

words, it is not just about the testability; it has to do with something else. To clarify this, we carried out two experiments below:

(1) Test Engineer writes function specifications: Supposing that the test engineer can write function specifications with both viewpoints as a designer and test engineer.

(2) Designer tries parallel designing of function specifications and test specifications: Supposing that the designer notices the lacking viewpoints.

5. How to Leverage Test Engineer's Viewpoints

We discovered that there may be difference in the mind when writing function specification from writing test specification. To identify what viewpoints the designers are lacking, we carried out two experiments; (1) the test designer designs the function specifications as the designer, and (2) the designer designs test specifications using the function specifications that other designers wrote as an input.

5.1. Function Specifications Written by Test Engineer

This experiment is based on our hypothesis that if the test engineer can find defects in function specification, the test engineer's viewpoints should be automatically incorporated when they write function specifications, resulting in the specification with better quality. The objects for this experiment are to:

(1) Write function specifications with test engineer's viewpoints

(2) Observe and understand the designer's mind-set

(3) Search for the true cause for missing specifications

For the experiment, we used a paper shredder, of which functions are fairly easy to imagine. To make it more realistic, we did not prepare any specification format and limited the time. After finishing writing the function specifications, the test engineer wrote the test specifications with the self-written specifications and later verified if the test engineer's viewpoints were successfully incorporated as we expected.

Contrary to our expectation, there were missing specifications such as the inputs other than operating conditions, infrequent use cases, combination of multiple functions and/or operations, and abnormal cases. This explains that the mind-set of test engineer while wiring the function specifications is somewhat similar to that of the designers.

It also means the test engineer is primarily focusing on "how it functions correctly." In addition, we noticed several use of ambiguous words such as "etc.," which many test engineers are less fond of. Also, limited time might have caused the missing specifications; it seemed that only the minimal specifications were documented to meet the deadline. Table2 summarizes the comments from the test engineer on the experiment.

Table2. Comments in Writing Function Specifications

No.	Comments	Description
1	Difficult to visualize / image the whole functions	Only vague images are inspired for the structures and behaviors for the whole functions.
2	Insufficient skills	Cannot decide description format. Tricky drawing tool. Poor text representation.
3	Unconfident on deciding specifications	Tend to have subjective point of view. Not too sure about how the functions should be.
4	Duplicate thoughts in self-reviewing	Thoughts get confused and duplicated after self-reviewing the same part over again.
5	Confused with functions' complexity	Too much consideration to make sure not to miss specifications. Scope widens too much, resulting in scattered and confused thoughts.
6	Narrowing mind scope focusing on	Much focus on "how it functions," less focus on "how the

	"how it functions"	function is used (scenario)."
7	"Normal" patterns only	Mainly focus on "normal" conditions to function correctly, not "non-normal" conditions for abnormal stoppage.
8	Use of ambiguous wordings	Less hesitate to use ambiguous words; inadequate time to identify all possible functions.

The experiment proved that it is difficult to automatically supplement the test engineer's viewpoints when writing specifications. Even so, the test engineer actually found defects in self-written specifications later, it means the mind has been switched over when designing test cases. As Beizer wrote [6], test engineer's attitude is naturally skeptical, and they naturally and unconsciously think about how to destroy the test objects. This is exactly the attitude of the test engineer when designing tests based on the function specifications. We began to think that if the designers could refresh their minds after writing the function specifications and conduct reviews with "skeptical attitude," they might be able to prevent missing test cases.

5.2. Parallel Designing Based on W-model

An Earlier study [7] reports the effectiveness of parallel designing of function specifications and test specifications by a single person. And, likewise, our study reports [1] the problem-solving approach for "ambiguous specifications" using parallel designing. This experiment is based on our hypothesis that if the designer can find incorrect specifications and defects by designing function specifications and test specification in parallel according to W-model, the designers can recognize the viewpoints they tend to lack. W-model is a process model where testing is performed in parallel with development from earlier stage. The best benefit of adopting W-model is that the defects caused by missing specifications, ambiguous expressions, and inconsistency can be found and fixed without actually running tests.

However, in order for us to introduce the parallel designing to the project, we needed to consider the following concerns from the team members:

- It may improve quality, but may increase time and resources as well
- If it really reduce reworks -- unknown effectiveness and value

<<Promotion Tactics>>

To address these concerns, we used the tactics below and promoted adoption of the W-model:

(1) The promoters practice W-model themselves: Design the tests based on their self-written function specifications and find defects later.

(2) Share defect information and the value of defect detection, or ask questions each other, for example:
   - Does this function specification give sufficient information for implementation?
   - Is this function specification testable? Does it cause design reworks while running tests?

   After having shared the information above with the members, design the tests based on the function specifications written by other designers, and realize that the defects are found.

(3) Conduct a retrospective, discuss about the defects found and keep it in practice.

<<Adopting W-model>>

We decided to adopt the parallel designing to the design specification that is the derivative of "software system design" and "software detailed design."

As mentioned in 5.1, the viewpoints of test engineer help find defects easier. To observe this, we had another experiment on the designers to "design tests" by themselves in the following order:

(1) Focus on identifying fairly complete requirements, document common requirements only, not spend much time on defect detection because it happens later in designing test.
   (Reason) To switch the mind-set to test engineer's viewpoint before it gets fixed to that of designer's.

(2) The designers design tests, based on the function specifications written by other designers.
   (Reason) Using self-written specifications as inputs would not switch the designer's viewpoint.

(3) Using the design support tool, ensure to reflect the defects found in designing tests.

(Reason) The tool adds a link between the test specifications and function specifications, allowing changes being made in the specifications to be instantly reflected.

(4) Complete the test specifications and function specifications at the same time.

(Reason) Fixing defects found when designing tests is counted within the same phase, not in reworks.

<<Result and Effect>>

Tables3 shows the result of the experiment. By adopting the parallel designing, we found average of 50 defects per specification. The number of missing test items for missing specifications was significantly reduced from 15 to 0. Accordingly, rework reduced by 40%.

Also, all project members valued the parallel designing, especially for preventing the missing specifications. Before adopting the parallel designing, defects regarding to missing process interaction and/or data display order were commonly found. However, after adopting the design support tool, the corresponding parts in the function specifications and test specifications were shown side by side. In this way, the designers could focus on writing the specification in details, create the test cases for functions, and it helped to prevent bugs in earlier stage.

Table3. Defects Found in Function Specifications in Lower Phase
Before / After Adopting Parallel Designing

	Missing Test Cases: Missing Specifications	Missing Test Cases: Ambiguous Specifications	Test Case Errors: Written Mistakes in Specifications	Total
Before	15	8	5	28
After	0	3	0	3

Based on the experiment (parallel designing of function specification and test specifications), we would like to summarize the viewpoints that the designers tend to lack when writing function specifications.

When writing function specifications, designers are highly focused on transforming requirements into functions and "normal" cases. Also, paying close attention to functions narrows their mind scope, which may cause them to overlook the interactions between functions. The designers are knowledgeable about the development object, and, with their code of value, they falsely assume that others know what they know. This false assumption leads "intentionally unwritten specifications" or "unintentionally missing specifications," resulting in the lack of detailed description for processing (inadequate error conditions / error handlings), or lack of detailed interactions between functions.

As Beizer [6] and Myers [8] say, we realized that reviewing self-written function specification is not effective since the designers have little doubt in what they wrote. This means that the designer and the test engineer have different viewpoints from each other, it is rather difficult for a single person to design both function specifications and test specifications. We think the following factors prevented "missing specifications" in parallel designing along with W-model:

- Recognition: By removing misunderstanding or false assumption, the designers found defects (ambiguous specifications, specifications errors, and missing specifications) by designing tests based on the function specifications others wrote.
- Learning Testability: Before adopting W-model, all designers thought what they wrote was sufficient. However, they learned and realized the testability of what other designers wrote. This changed their attitude; it is not a direct factor yet plays very important part.

6. Discussion

In this section, from these experiments, we would like to discuss the viewpoints that the designers tend to lack. In addition to that, we would like to discuss the approaches to improve the design quality by designers themselves, as well as how to incorporate it to the function specifications.

6.1. Leveraging the Test Engineer's Viewpoints

First of all, we discussed how the designers can leverage the test engineer's viewpoints. According to the earlier study [4], the test engineer's viewpoints represent the testability that is to explain "how to test." With the test engineer's viewpoints, we discovered that designers tend to have "false assumption." Because of this, the area that the designer is writing is the entire scope for them, and it would not expand any further. However, if one person played the both role as the designer and the test engineer, as our experiment proved, reviewing the function specifications with skeptical attitude is effective to find missing specifications, after completing the specifications and refreshing the mind-set. In other words, in order to improve the design quality by designers themselves and reflect the quality to the function specifications, removing false assumption and conducting reviews with skeptical attitude are quite effective.

In our experiments, we had both designer and test engineer, but, in practical, the projects rarely have enough resources as such. Even so, more defects can be found if one can play a double role. For playing a double role, we found the followings particularly effective:

(1) After completing function specifications, take some time before designing tests

(2) Design tests in different order from writing function specifications

6.2. Benefits of Parallel Designing

As previously described, the experiments show that "normal" cases are mainly written, and the designers have false assumption that causes "intentionally unwritten specification." Namely "the designers unconsciously set their own scope" for what is to be developed. The entire picture is vague and invisible when writing function specifications, and it finally emerges when the software is developed and when they began to consider the user operations or abnormal cases. Figure 2 depicts the image of the entire scope. When the designers are writing specifications, their mind scope is set and limited to the area of their knowledge. Later, as the software is implemented and becomes visible, the area of unknown knowledge expands. Recognizing and acknowledging the unknown knowledge area is the key to the approach for the designers to improve the design quality and reflect the quality into the function specifications.

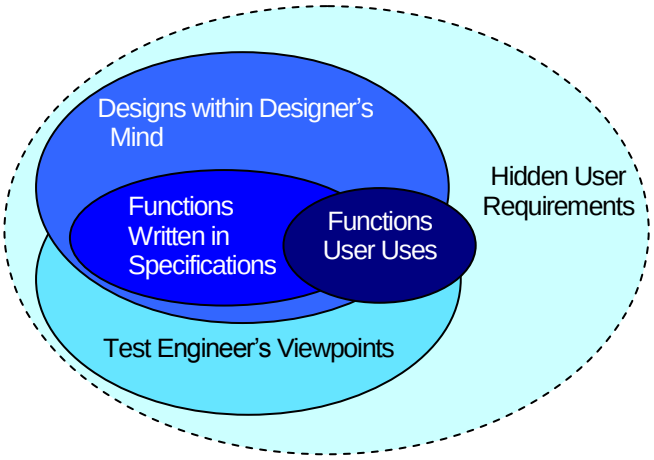

Figure 2. Designer's Mind Scope and Entire Picture of Software

The parallel designing is effective to recognize the mind scope of the designers, the review activity for the self-written design specifications on behalf of test engineer is also effective. The parallel designing makes the designers to recognize the mind scope that the designers set unconsciously. Also, we can always learn from the past. "Previous Bugs" [9] provides various field defect patters that are hardly recognized internally, and this may help widen the mind scope of designers.

6.3. Measuring Quality of Function Specifications

The effectiveness of the parallel designing to prevent missing specifications in functions specifications is shown indirectly, based on the analysis for the bug cause distribution of completed software. This proves, and we would like to conclude, that adopting the parallel designing to embedded software development makes difference to prevent missing specifications in function specifications. A research, which is from the quality of requirement specifications by the requirement working group [10], proposes the approach to measure document quality. This introduces some metrics for "the quality of requirement specifications itself," based on the guidelines in IEEE 830-1998 [3]. This approach, however,

requires numbers of relevant items after specification reviews. The parameters for these formulas are “the total number of requirements.” They are based on what is already written and not considering “what is not written,” which deeply related to “Completeness.” As was expected, it is quite challenging to measure completeness without having the entire picture.

7. Conclusion

Our experiment proved that it would be difficult to automatically complement by test engineers developing function specifications for designers, which indicates the necessity of a way or a system to deliberately supplement the viewpoint of test engineers. Another experiment of ours showed it would be very effective for a system to supplement the viewpoint of test engineers if designers develop test specifications as test engineers using function specifications created by themselves or their colleagues. In this way, designers will be able to have the viewpoint of test engineers and detect defects in function specifications. Designers themselves will be able to elaborate the quality of function specifications.

However, even the designers recognize their false assumption and their limited mind scope, the level of description in function specifications depends on the project, member, or the product to be developed. We would like to continue our further study for better approaches with such change factors.

References

[1] Quality Improvement in Embedded Software Development, Software Quality Profession accomplishment report 2008, Union of Japanese Scientists and Engineer

[2] Software Life Cycle Processes-Japan Common Frame 2007 Second Edition, Information-technology Promotion Agency, Japan Software Engineering Center, Ohmsha

[3] IEEE Recommended Practice for Software Requirement Specifications, IEEE Std 830-1998

[4] A proposal for the defects detection method of Software Requirement Specifications with a point of view of software testing engineer, Software Quality Profession accomplishment report 2008, Union of Japanese Scientists and Engineers

[5] The W-MODEL - Strengthening the Bond Between Development and Test、 Andreas Spillner、STAReast 2002

[6] Boris Beizer, *Software Testing Techniques, Second Edition*, Nikkei BP Publishing Center, 1994

[7] *Process Improvement Navigation Guide - Best Practice*, Information-technology Promotion Agency, Japan Software Engineering Center, Ohmsha, 2008

[8] The Glenford J. Myers, Rewrite and updated by Tom Badgett and Todd M, Thomas, with Corey Sandler,, *Art of Software Testing Second Edition*, Kindai Kagaku sha Co.,Ltd., 2006

[9] Method of effective leveraging previous bugs for engineers of embedded software, Software Quality Profession accomplishment report 2008, Union of Japanese Scientists and Engineers

[10] Quality of Requirements Specifications Accomplishment Report, Information Processing Society of Japan, Special-Interest Group Software Engineering, Requirement Engineering Working Group, Jan. 24, 2007