5th World Congress for Software Quality – Shanghai, China – November 2011

Process Improvement using XDDP

- Application of XDDP to the Car Navigation System -

Keiji Kobata
DENSO E&TS TRAINING CENTER CORPORATION
Ohbu, Japan
keiji_kobata@denso.co.jp

Eiji Nakai
DENSO TECHNO CORPORATION
Kariya, Japan
eiji_nakai@denso.co.jp

Takahiro Tsuda
DENSO CORPORATION
Kairya, Japan
takahiro_tsuda@denso.co.jp

Abstract

In this paper, we describe the experience to improve quality and productivity in car navigation system development using XDDP (eXtreme Derivative Development Process). Maintenance process is used to develop new software product. Defects and problems are caused by insufficient analysis of base software. Therefore we applied XDDP to our developments. XDDP is a rational and practical process for enhancement-based development. Productivity was improved by 26% and no defects were detected in QA tests. Then we applied XDDP to the developments where engineers lack information about the base source code. With software engineering techniques, quality and productivity were improved dramatically compared to the case when XDDP alone was used.

1. Introduction

The source code of navigation system is expanding in size and getting more complicated due to the increase of navigation functions. High quality and on time delivery are required in navigation system because it is installed to automobiles.

Most software projects for car navigation systems in our company are classified as enhancement-based development. This type of development is to develop a new software product by adding, changing or deleting functionality to an existing software product. In a large-scale software enhancement project, it is very difficult to identify all change points in the source code completely and estimate the impacts of the changes with high accuracy.

The projects where engineers lack information about the base software are increasing recently. In such projects, it is much more difficult to extract change points and estimate the impact. We need more time to achieve required quality and productivity.

In our developments, the base documents are modified to enhance the source code with maintenance process [1]. The information about changes is added to the base documents and the impact of the changes is estimated with them. It is tough to identify change points and estimate the impact correctly.

Therefore we applied XDDP to our developments. XDDP is a development process specialized in enhancement-based development. It is a well-known process in Japan. The feature of XDDP is that the documents only about change points are made and reviewed in the development. From these documents, change points can be identified and the impact of changes can be estimated.

First, we applied to a general project in navigation system and confirmed the effect of XDDP in quality and productivity. We analyzed the result and showed the effect in detail. Second, using the information we had got from the experience, we applied XDDP to the projects where engineers in charge lacked information about base software. We suggested two techniques in XDDP to make requirement specifications with base source code. These two techniques are used in accordance with knowledge level of engineers. By these techniques, quality and productivity were improved compared to the case when XDDP alone was used.

The constitution of the paper is as follows. In chapter 2, problems are discussed in our development process. In chapter 3, we introduce a summary of XDDP and USDM (Universal Specification Describing Manner). In chapter 4, the results of applying XDDP to a general project are shown in quality and productivity. And we apply XDDP to the projects where engineers lack information about the base software. We suggest two techniques to make requirement specifications and discuss the effect of them. Chapter 5 is a conclusion of this paper.

2. Problems in Our Development Process

Navigation systems have been developed using maintenance process to modify base source code. In the process, we make requirement specifications and design documents to identify change points and estimate the impacts of changes.

Figure 1 shows our development process. When new functions are added, engineers make requirement specifications and design documents in new development process. For changes of existing functionality, we use maintenance process. Change points in requirements are added to the base requirement specifications to make requirement specifications. And change points in requirement specifications are added to the base design documents to make design documents as well.

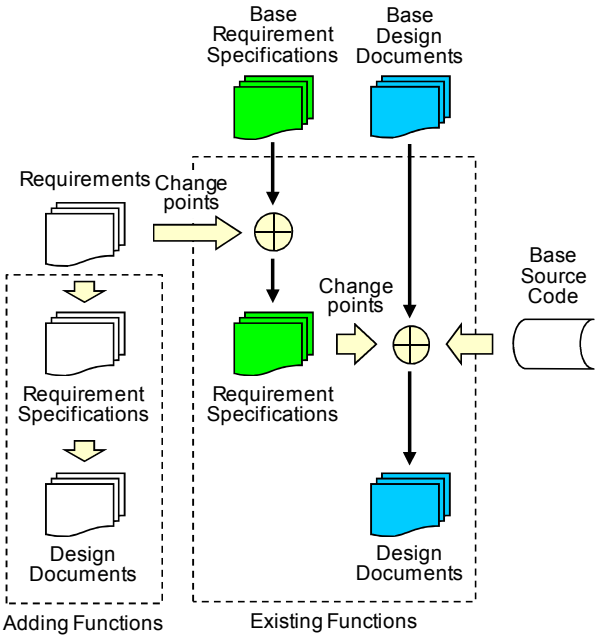

Figure 1. Conventional Process

It is very difficult to specify other change points and estimate the impact from modified base documents. One change in base source code leads to other changes in other components of different functions. Especially in high performance embedded system such as navigation system, the added source codes affect so many functions that we have difficulty in identifying change points and estimating the impact. So we have a high risk that problems and defects will occur, even if one change point (several source codes) is modified in base source code.

We also have a problem that it is not clear where we should write the change points caused by adding functions in base source code. The change points can't be added to base requirement specifications easily and it seems strange that change points are described in requirement specifications for adding functions. In many cases, they are missing in documents and serious problems and defects can occur.

In our developments, the documents about adding functions are reviewed so much, but changes of base software tend to be left to the engineers in charge. In other words, identifying change points and estimating the impact entirely depend on the engineers' skill. Some engineers change the source code as soon as they find a change point in the source code. The reason is that they can't extract change information from base software and we don't have a proper process for extracting change information.

Therefore we applied XDDP to our developments in order to solve these problems.

3. Overview of XDDP

3.1 XDDP [2][3][4]

XDDP, developed by a Japanese consultant, Yoshio Shimizu in 2007, is an enhancement-based development process. The feature of XDDP is that documents made in XDDP are all about change information about base software. We must make quite new type of documents only about changes. XDDP consists of two independent processes to make the documents easily; one is for adding functions "addition process" and the other is for changing base source code "change process".

Figure 2 shows the process of XDDP. In addition process, we make "requirement specifications on adding functions" about new tasks or functions. Then design documents are made from "requirement specifications on adding functions". The process is the same as new development process.

In change process, "change requirement specifications" about changes in base source code are made from base documents, base source code and "requirement specifications on adding functions". "Change

requirement specifications" are about changes, additions and deletions in base source code. Then "Traceability Matrix (TM)" is made to specify where change points are modified in the base source code. TM is a matrix of change specifications and the base source code. And we make "change design documents" about how to modify the base source code regarding the change points in the TM.

In XDDP, change specifications are extracted from change requirements in "change requirement specifications", analyzing related documents and base source code. Change specifications contain changes made to the base source code where new functionality is added in addition to changes and deletions in existing functionality. All change points in base source code are extracted and described in "change requirement specifications".

When a new module for new functions is developed, we make "requirement specifications on adding functions" about the module. To add the module to the base source code, we need to change base source code. The change points in accepting the module in base source code are described in "change requirement specifications". In this way, we describe all change points in the documents and review them.

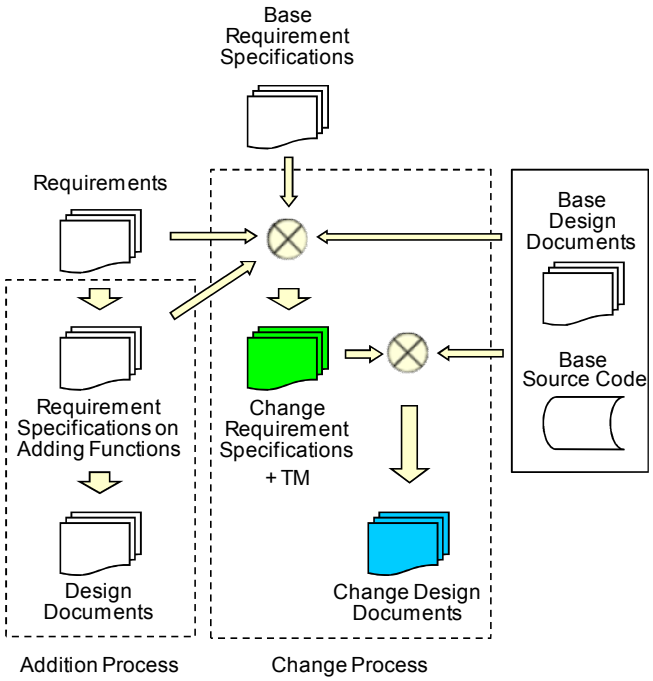

Figure 2. XDDP

3.2 USDM [5]

"Requirement specifications on adding functions" and "change requirement specifications" are written with USDM. USDM, developed by Yoshio Shimizu in 2005, is an effective manner to make requirement specifications. Requirements and specifications are shown structurally to prevent missing specifications.

Figure 3 shows USDM format. Requirements are described as series of function behaviors, performance and constraints of system. And specifications are extracted from the requirements and written under the requirements. When the scope of a requirement is large, the requirement can be divided into some branch requirements. We describe branch requirements under the requirement. Specifications can be classified into some groups to extract them effectively when specifications increase.

"Requirement specifications on adding function" are written in the same way as normal requirement specifications. In "change requirement specifications", we describe only change points in requirements and specifications.

Requirement	Req.1	Requirement	
	Reason	Backgrounds or Objectives	
	Comment		
	Branch Requirement	Req.1-1	Branch Rquirement
		Reason	
		Comment	
	Specificaiton	<Group A>	
		Req.1-1-1	Specification
		Req.1-1-2	
		<Group B>	
		Req.1-1-3	
		Req.1-1-4	

Figure 3. USDM Format

4. Application of XDDP to Developments

4.1 General Project in Navigation System [6]

We applied XDDP to our general development and verified the effect of XDDP. The development period was two months. The changed source code was about 1KLOC (Kilo Line of Code). The engineer in charge had been well informed about the source code and the navigation functions.

The productivity was increased by 26% when XDDP was used. The number of defects in QA test decreased to zero by XDDP, although we had detected some defects before. We define the productivity by dividing total development time into the size of total changed source code.

The man–hour distribution of the developments is shown in Figure 4. The upper one is the result of general development with conventional process and the lower one is with XDDP. Comparing with these results, XDDP is very effective in our developments.

Analyzing the distribution in detail, designs and tests are repeating in the conventional process. It is caused by the defects due to the insufficient analysis of the base software. The required quality is achieved by repeating designs and tests. In XDDP, design reviews are effectively executed and the coding time decreases to the half of conventional process. In the eighth week, code reviews are executed. The development is finished without repeated designs and tests.

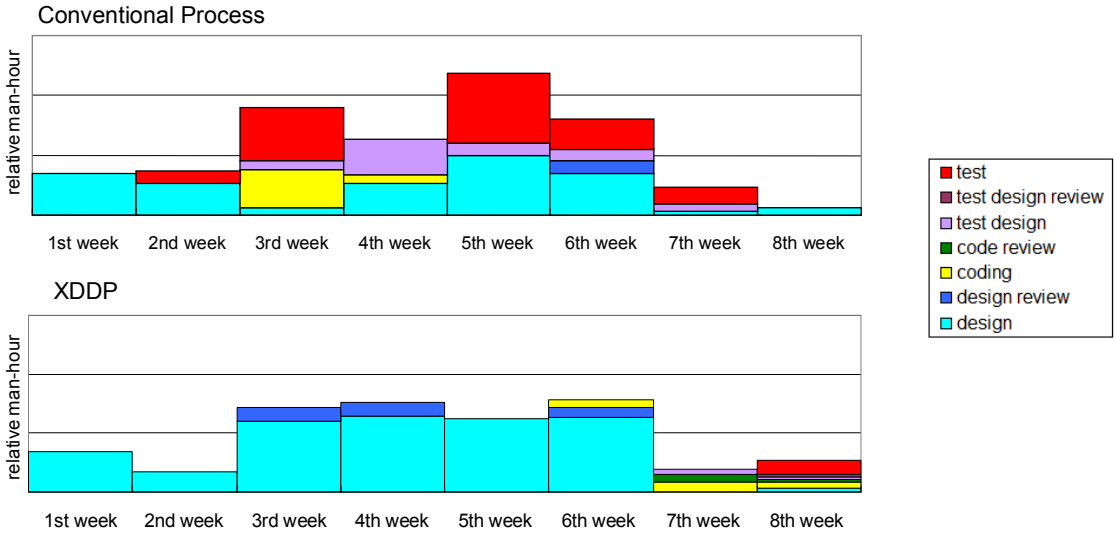

Figure 4. Comparison of the Man-Hour Distributions before and after XDDP Application

4.2 Project without Accumulated Technical Information (PWAT)

Next we applied XDDP to the developments where engineers in charge lack the information about the base source code. It becomes more difficult to estimate the impact of changes.

We call such a type of project “Project without Accumulated Technical information” (PWAT). In the general project (4.1), it is supposed that engineers with the information about the base source code are in charge of a development. Developments in PWAT are increasing recently. There is a high possibility that a development will become PWAT when the base source code is in the following situation.

- source code changed in outsourced companies
- source code developed in other companies
- source code no engineer understands in detail

Furthermore there are increasing cases where engineers don't have enough knowledge of the development domain. Using the source code developed in other companies, sometimes no documents are prepared in advance. Then PWAT is classified into following two types in accordance with engineer's knowledge level and we applied XDDP to each PWAT.

A : no information only about the source code

B : no information about the source code and the development domain

In applying XDDP to PWAT, the problem is how to analyze the base source code and identify change specifications correctly. We suggest improved XDDP to solve the problem by incorporating software engineering techniques into XDDP. We define improved XDDP for each PWAT as X-PWAT(A) and X-PWAT(B).

In later chapters, to confirm the effect of each X-PWAT, we compare the results according to Figure 5. In PWAT(A), rst2 (X-PWAT(A)) is compared to rst1 (XDDP).In PWAT(B), rst4 (X-PWAT(B)) is compared to rst3 (X-PWAT(A)).

		Process		
		XDDP	X-PWAT(A)	X-PWAT(B)
Project	PWAT(A)	rst1	rst2	-
	PWAT(B)	-	rst3	rst4

Figure 5. Result Patterns of Project and Process

4.3 PWAT (A) : no information only about the source code [7]

The quality of change requirement specifications can't be secured easily in PWAT(A) because of the engineers who lack the information about the source code. It means that the change requirement specifications depend on the engineers' skills, experience and understanding of the development domain. Required quality and productivity aren't achieved in PWAT(A) when XDDP alone is used.

Then we suggest that the structure of change requirement specifications corresponds to the processing flow of the base source code (Figure 6). When XDDP is applied to PWAT(A), the processing flow that covers change points is made from base documents and base source code. Next we extract change requirements and specifications corresponding to each processing in the flow. XDDP with this process (making a processing flow and extracting information from the flow) is defined as X-PWAT(A).

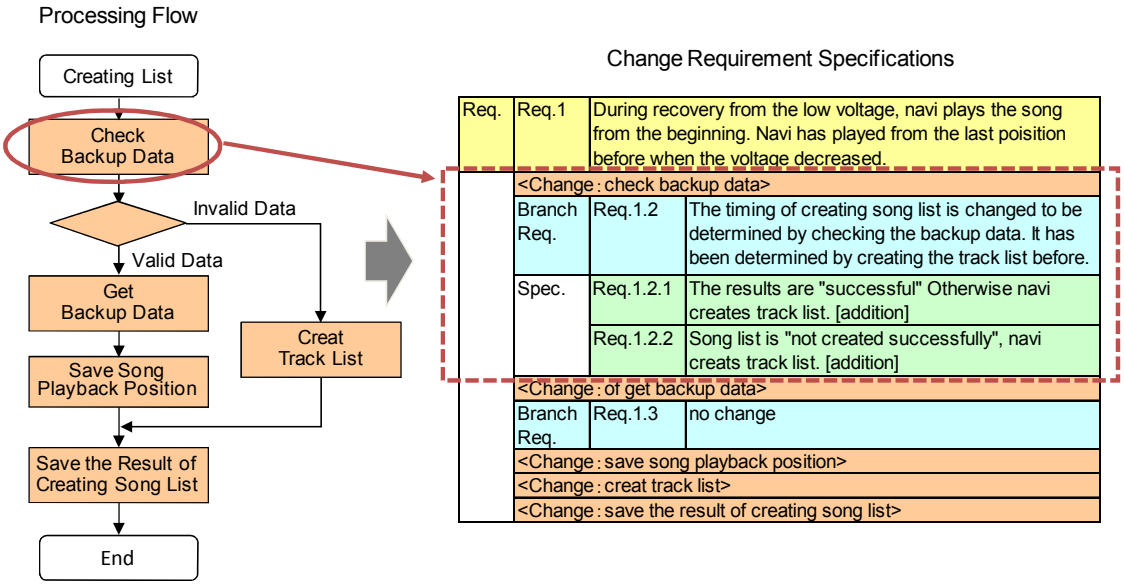

Figure 6. X-PWAT(A)

We applied X-PWAT(A) to a development in PWAT(A). The size of total base source code was 2.6 KLOC and the changed source code was about 0.3 KLOC. The development period was one month. The engineer in the company took over the outsourced development. The engineer lacked information about the source code but had the knowledge of the development domain.

Figure 7 shows the results of application of XDDP and X-PWAT(A) to PWAT (A). The productivity is almost the same. But the defect detection rate is decreased to the half of XDDP by X-PWAT(A). The rate is decreased in both review and test. It proves change specifications are extracted more correctly. X-PWAT(A) with processing flow is effective in making change requirement specifications.

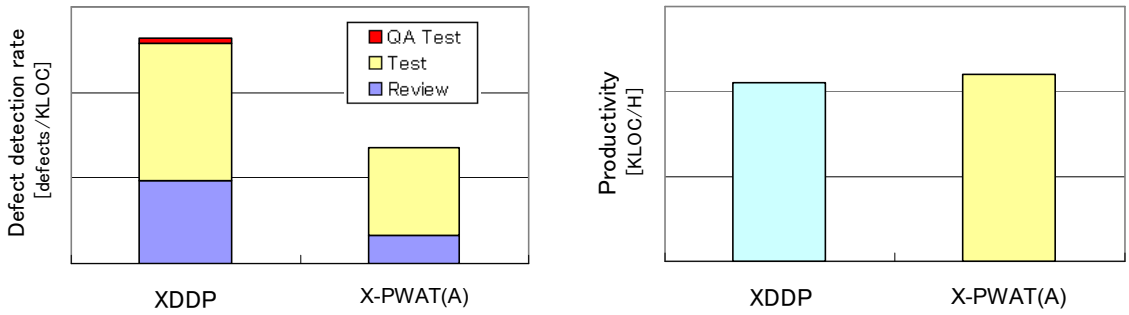

Figure 7. Effectiveness of X-PWAT(A)

PWAT (B) : no information about the source code and the domain [8]

We applied XDDP to a development in PWAT(B) [7]. In PWAT(B), change requirement specifications are made mainly from the base source code because we don't have adequate documents about the base software. The quality of change specifications depends on the engineers' intuition, experience and skills. The coverage of change requirements is narrow and it is insufficient to analyze the change points and estimate the impact.

We applied X-PWAT(A) to PWAT(B) using information we had got in PWAT(A). However, quality and productivity were not improved as much as we had expected. Because of the engineer who doesn't understand even the overview of base software, it is difficult to make processing flow directly from the base source code. As a result, the processing flow is not enough to extract change requirements and change specifications.

Therefore we suggest the technique to analyze the base source code systematically with DFD (Data Flow Diagram) [9], AFD (Architecture Flow Diagram) [10] and sequence diagrams to make change requirement specifications (Figure 8). XDDP with this process (analyzing the source code with the design techniques) is defined as X-PWAT(B).

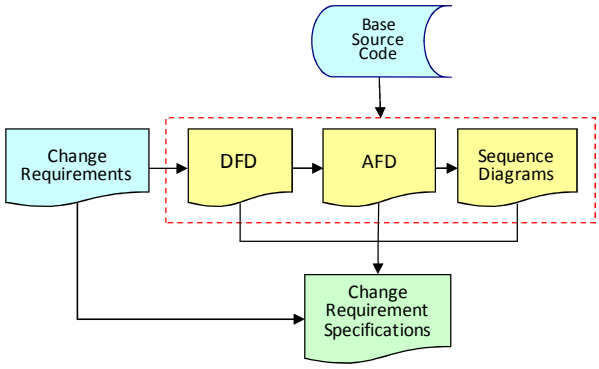

Figure 8. Overview of X-PWAT(B)

Figure 9 shows X-PWAT(B). We make DFD from the base source code related to a change requirement and identify the process affected by the change requirement. Then we make detailed DFD about the process and find out the process which should be modified by the change requirement. The branch change requirement is extracted from the process. Thus we extract change requirements by corresponding to DFD in the structure of change requirements in USDM. Then, we describe the internal structure of the process precisely with AFD and identify the object affected by the branch change requirement. By turning attention to the object, change specifications are extracted from the sequence diagrams made from AFD.

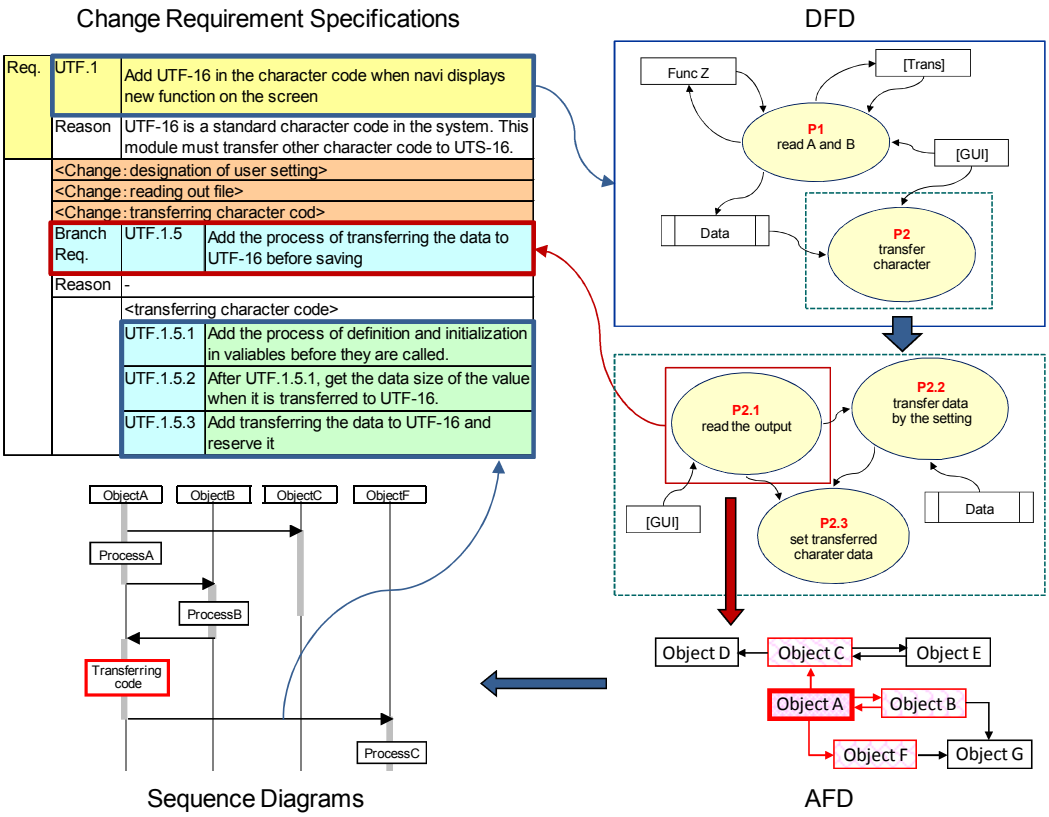

Figure 9. Extracting Change Specifications in X-PWAT(B)

We applied X-PWAT(B) to a development in PWAT(B). The size of total base source code was 9.5 KLOC and the changed source code was about 0.5 KLOC. The development period was three months. The engineer in charge took over the source code developed in another company. The engineer didn't have any information about the source code and the development domain at all.

Figure 10 shows the results of application of X-PWAT(A) and X-PWAT(B) to PWAT(B). The rate of defect detection is decreased dramatically in tests and the number of defects is zero in QA test in X-PWAT(B). Productivity is improved because defects in test are decreased by extracting change specifications properly. It shows that X-PWAT(B) with DFD, AFD and sequence diagrams is more effective than X-PWAT(A) in PWAT(B).

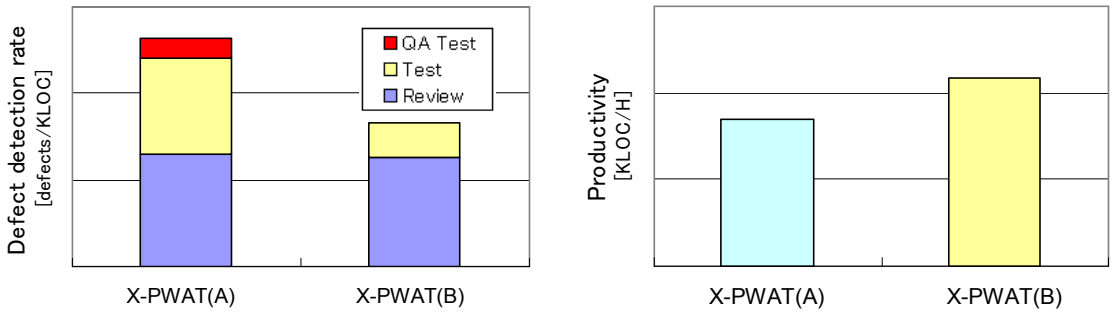

Figure 10. Effectiveness of X-PWAT(B)

5. Conclusions

We applied XDDP to software developments of car navigation system in our company. Productivity and quality were improved compared to the conventional developments. We've got the information about XDDP from the experience. Then we applied XDDP to developments in PWAT. We suggested two techniques to make change requirement specifications in accordance with the knowledge level of the engineers. They were very effective in quality and productivity in each PWAT. Thus, XDDP is an effective process in our car navigation system developments and productivity and quality can be improved in various situations.

References

[1] ISO/IEC 14764:2006(E) IEEE Std 14764-2006.

[2] The JUSE (Union of Japanese Scientists and Engineers) website [Online], available at http://www.juse-sqip.jp/vol10/qualityone_01.html (accessed on 2011/5).

[3] The JUSE website [Online], available at http://www.juse-sqip.jp/vol11/quailtyone_01.html (accessed on 2011/8).

[4] Yoshio Shimizu, Process Improvement in Enhanced Development, Gijutsu-Hyohron co., Ltd.,2005 [in Japanese].

[5] Yoshio Shimizu, Technique for Specifying and Describing Requirements, Gijutsu-Hyohron co., Ltd.,2007 [in Japanese].

[6] Yoshiyuki Kato, Software Process Improvement using XDDP Process, *Software Quality Profession 2008 (SQiP2008)*, 2008.

[7] Eiji Nakai, Application of XDDP to a project without accumulated technical information in past development, *SQiP2009*, 2009.

[8] Takahiro Tsuda, Technique to make change requirements specification using a design method, *SQiP2010*, 2010.

[9] Tom Demarco, Structured Analysis and System Specification, Prentice Hall, 1979.

[10] Derek J. Hatley, Imtiaz A. Pirbhai, Strategies for Real Time System Specification, Dorset House Publishing Co Inc.,U.S, 1988.