Process improvement using XDDP
- Application of XDDP to the Car Navigation System -

Eiji Nakai
DENSO TECHNO Co.,Ltd
Abstract

1. Introduction
2. Problems in Conventional Development Process
3. What is XDDP?
4. Case 1. Application to General Project
5. Case 2. Application to PWAT(*)
6. Conclusions

(*) PWAT : Project Without Accumulated Technical information
Introduction

• Current situation in car navigation software development
  – Number of functions and size of software are increasing rapidly
  – Many software variations are needed
  – Higher quality is required
  – Development period is getting shorter
• Most of developments of navigation software are “enhancement-based development”
  – Develop new product based on existing product
    • Add new functions
    • Make improvements of existing functions
Overview of Conventional Development

• Use V-model
  – Development process to develop new software product

• “Addition” and “Change” are contained in enhancement-based development
  – “Addition”
    • Add new functions to base software
  – “Change”
    • Change existing functions in base software
Problems in Conventional Development

• It is difficult to identify all change points
  – One change point causes other changes
  – Adding a new function affects existing functions

• Source code becomes complicated by changes
  – Source code maintenance is not easy

• Design policy or change background is not always described clearly in the documents
  – Implementation of the change depends on engineers’ knowledge

Problems are caused by “Change”
Causes of Problems

1. Change points are scattered in base documents
   - It is difficult to detect related changes
   - Review does not have much effect with such documents

2. Some engineers change the base source code without enough analysis
   - It leads to degrading other functions
   - The change points are not always appropriate

Details Conventional Process

Development Process is not suitable for “Change”
Further Problems

- PWAT (Project Without Accumulated Technical information)
  - No engineers have enough information about base software in the project

Problems of PWAT
- Quality in investigation of base software depends on engineers’ experience and intuitions
  ⇒  • Investigation is insufficient in coverage
      • It takes too much time to analyze the base software
      • Documents on the result of investigation are insufficient

Conventional Investigation does not meet PWAT
What is XDDP?

- XDDP is a software development process focused on Changes

- Advantages
  - Change information is arranged and described properly for the development
  - All related changes are detected through the development process
Features of XDDP

(1) Two independent processes
   “Addition” and “Change”

(2) Specification techniques
   USDM specification description
   ( *) Universal Specification Describing Manner

(3) Documents focused on change
   “Change Requirement Specifications”
   “Traceability Matrix (TM)”
   “Change Design Documents”
(1) Two independent processes

- “Addition” and “Change”
  
  - Addition
    
    • The process is the same as new development process
  
  - Change
    
    • Make new documents described about only changes
    
    • Additional Function is treated as one change
    
    • Change the source code after all change points are identified

Detect and describe all change points
(2) Specification techniques

- **USDM Format**

1. Describe requirements and specifications structurally
   - “Requirement” series of behaviors
   - “Specification” concrete behaviors to realize requirement

2. Clarify the reason
   - The reason is necessary to detect proper change points

3. Describe change points by “before/after” in “Change Requirement Specifications”

### Table: Specifications

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Reason</th>
<th>Requirement</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req.1</td>
<td></td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Req.1-1</td>
<td>Branch</td>
<td>Req.1-2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Requirement</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reason</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Comment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speciaiton</td>
<td></td>
<td>Speciaiton</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;Group A&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Req.1-1-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Req.1-1-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speciaiton</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;Group B&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Req.1-1-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Req.1-1-4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Speciaiton</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&lt;Group C&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Req.1-2-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Req.1-2-2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**USDM prevents missing change points**
(3) Documents focused on change

- Describe all change points in three types of documents before changing source code

[1] Change Requirement Specifications
  - What and Why should we change?

  - Where should we change?

[3] Change Design Documents
  - How should we change?
### Change Requirement Specifications

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Reason</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req.1</td>
<td>During recovery from brownout condition, navi plays from the beginning of the song. This specification is changed to play from the last position at low voltage detection.</td>
<td>*brownoutrcondition navi transfers to brownout condition by low voltage detection. Software is reseted at this timing. If navi has already created the song list, it can play from the last position at the time of recovering from brownout condition. So it’s decided to unify action regardless of the timing.</td>
</tr>
</tbody>
</table>

#### <Change the process of checking backup data>

<table>
<thead>
<tr>
<th>Branch Requirement</th>
<th>Requirement</th>
<th>Reason</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req.1-1</td>
<td>As a result of integrity check backup data, change the return value from “disabled”, “enabled” to “disabled”, “Enabled” and “valid only for size”.</td>
<td>Because the size in backup data is valid, the device is considered to same at the time of low voltage detection, and can play from the last position.</td>
<td>Consistency check backup data, the size of the devices (total size and free size) to compare. If size information is consistent with the backup data, it is determined the same as last connected device.</td>
</tr>
</tbody>
</table>

#### <Change the size of device information acquisition process>

<table>
<thead>
<tr>
<th>Specification</th>
<th>Requirement</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req.1-1-1</td>
<td>Add the definition of “valid only for size”.</td>
<td></td>
</tr>
<tr>
<td>Req.1-1-2</td>
<td>Change the condition for clearing the backup data from not “OK” to not “valid backup” and “valid only for size”.</td>
<td></td>
</tr>
</tbody>
</table>

#### <Change the size information comparison process>

<table>
<thead>
<tr>
<th>Specification</th>
<th>Requirement</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req.2-1-1</td>
<td>Backup data check results are “successful”. Otherwise, navi create track list.</td>
<td></td>
</tr>
<tr>
<td>Req.2-1-2</td>
<td>Song list is “not created successfully”, navi create track list.</td>
<td></td>
</tr>
</tbody>
</table>

---

This information is the exclusive property of DENSO CORPORATION. Without their consent, it may not be reproduced or given to third parties.
[Example] Change Requirement Specifications + TM (2)

Whole Picture of Change Requirement Specifications and TM

Change Requirement Specifications

Source File

Where we should change
# Example] Change Design Documents

<table>
<thead>
<tr>
<th>Project</th>
<th>XXX</th>
<th>Date</th>
<th>31/08/2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Name/Task Name</td>
<td>SourceFileA/ Module A</td>
<td>Author</td>
<td>XXX</td>
</tr>
<tr>
<td>Change Requirement Specification</td>
<td>Change the condition for clearing the backup data from not “OK” to not “valid backup” and “valid only for size”.</td>
<td>Modifier</td>
<td>XXX</td>
</tr>
<tr>
<td>#Req.1-1-2</td>
<td></td>
<td>estimate lines</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>estimate time</td>
<td>1H</td>
</tr>
<tr>
<td></td>
<td></td>
<td>change lines</td>
<td>actual time</td>
</tr>
</tbody>
</table>

- **Policy of modify**
  - Nothing special.

- **Change about structure of data**
  - No Change.

- **Change about structure of function call**
  - The return value of FunctionB() becomes to “valid”, “size only valid”, and “error”.

- **Change about out of function**

<table>
<thead>
<tr>
<th>Item#</th>
<th>Change points</th>
<th>estimate lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Add the definition of “valid backup” and “valid only for size”.</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Change about function**

<table>
<thead>
<tr>
<th>function</th>
<th>Change points</th>
</tr>
</thead>
<tbody>
<tr>
<td>FunctionA()</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chage point</th>
<th>Item#</th>
<th>Change points</th>
<th>estimate time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Change the timing of finishing process from not “OK” to not “valid backup” as the return value of FunctionB().</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
We applied XDDP to our navigation software development

Case 1. General Project

Case 2. PWAT (2 types)
Case1. General Project

• Applied project
  – Prototype development
    • Development period: 2 months
    • Change size: 1,000 LOC (Line of Code)

• Result
  – Defects were decreased (from 2 to 0) in QA test
  – Productivity was increased 1.26 times

![Comparison of Productivity]
Analysis of Result

- Man-hours distribution

Conventional Process

- Many defects were detected in test
- Design, coding, and test were repeated
- Man-hours could be decreased although much time was spent in design

XDDP

- Most of time was spent in design
- Coding was finished at once

Change points were identified properly in XDDP
Summary of Case 1

- Overall productivity can be improved even if we invest much time in identifying all change points
  - Hasty change of source code causes more work hours
  - Concrete change points (change specifications) make time in changing source code shorter
## Case2. PWAT

- **Applied project**

<table>
<thead>
<tr>
<th></th>
<th>PWAT(A)</th>
<th>PWAT(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Process</td>
<td>X-PWAT(A)</td>
<td>X-PWAT(B)</td>
</tr>
<tr>
<td>Target</td>
<td>Middleware (Audio Control)</td>
<td>Middleware (Voice Recognition)</td>
</tr>
<tr>
<td>Change Size [LOC]</td>
<td>300</td>
<td>500</td>
</tr>
<tr>
<td>Base Size [LOC]</td>
<td>26,000</td>
<td>9,500</td>
</tr>
<tr>
<td>Period [month]</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Situation</td>
<td>Outsourced company developed base software</td>
<td>Other company developed base software</td>
</tr>
</tbody>
</table>

**Apply XDDP to these two PWAT**
Application Policy

• Policy
  – Investigation process should be designed in accordance with the knowledge level of engineer

• Reason
  – Investigation process is not defined in XDDP
  – Situation of project are not always the same

Design investigation Process by types of PWAT
Design of Investigation Process (1)

- Procedure
  1. Select software engineering techniques for investigations by types of PWAT
  2. Incorporate these techniques into XDDP
     - Define outcomes of investigation as input to make “Change Requirement Specifications”

- PWAT(A) / PWAT(B)
  - PWAT(A) : No information about the source code
    - Engineer can imagine function behavior
  - PWAT(B) : PWAT(A) + No knowledge of the domain
    - Engineer are not familiar with functions
Design of Investigation Process (2)

- Outcomes of investigation
  - PWAT(A) : Process flow
  - PWAT(B) : DFD, AFD, Sequence Diagram
### Application to PWAT(A)

<table>
<thead>
<tr>
<th></th>
<th>PWAT(A)</th>
<th>PWAT(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>X-PWAT(A)</td>
<td>X-PWAT(B)</td>
</tr>
<tr>
<td>Process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target</td>
<td>Middleware (Audio Control)</td>
<td>Middleware (Voice Recognition)</td>
</tr>
<tr>
<td>Change Size [LOC]</td>
<td>300</td>
<td>500</td>
</tr>
<tr>
<td>Base Size [LOC]</td>
<td>26,000</td>
<td>9,500</td>
</tr>
<tr>
<td>Period [month]</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Situation</td>
<td>Outsourced company developed base software</td>
<td>Other company developed base software</td>
</tr>
</tbody>
</table>
**Example in PWAT(A)**

1. **Create process flow**
   - Function of creating list
   - Process of checking backup data
   - Invalid backup data
   - Process of getting backup data
   - Process of creating track list
   - Process of saving song playback position
   - Saving the result of creating song list
   - End

2. **Extract groups by each process**
   - Branch Requirement
     - Req.2-1: The timing of creating song list is determined by combination the results of checking the backup data and the results of creating the track list.
   - Specification
     - Req.2-1-1: Backup data check results are “successful”. Otherwise, navi create track list.
     - Req.2-1-2: Song list is “not created successfully”, navi create track list.

3. **Describe requirements and specifications corresponding under each groups**

---

**Detect Change Points by Process Flow**
Result of PWAT(A)

• Process
  – Compare X-PWAT(A) to simple XDDP

• Result

  **Quality**

  - Defect detection rate was decreased to about one-half
  - Productivity was almost the same
# Application for PWAT(B)

<table>
<thead>
<tr>
<th></th>
<th>PWAT(A)</th>
<th>PWAT(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>X-PWAT(A)</td>
<td>X-PWAT(B)</td>
</tr>
<tr>
<td>Process</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target</td>
<td>Middleware (Audio Control)</td>
<td>Middleware (Voice Recognition)</td>
</tr>
<tr>
<td>Change Size [LOC]</td>
<td>300</td>
<td>500</td>
</tr>
<tr>
<td>Base Size [LOC]</td>
<td>26,000</td>
<td>9,500</td>
</tr>
<tr>
<td>Period [month]</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Situation</td>
<td>Outsourced company developed base software</td>
<td>Other company developed base software</td>
</tr>
</tbody>
</table>
Identity the Change Process by DFD

Example in PWAT(B)

1. Make DFD about function related Change Requirement

Change Requirement Specification

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Req.1</th>
<th>Add UTF-16 in the transferring character code when navi creates **** from *** file, in addition to ISO 81 and SJIS and UTF-8.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>*** system's standard character code is UTF-16, so this module must transfer from other character code to UTF-16.</td>
<td></td>
</tr>
</tbody>
</table>

2. Specify related process of Change Requirement

3. Extract the process as group

4. Lower level DFD

5. Extract Branch Change Requirement

Identity the Change Process by DFD
**Example in PWAT(B)**

1. **Specify objects affected by change**

2. **Make Sequence Diagram**

3. **Extract change specifications**

### Change Requirement Specifications

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Req.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reason</td>
<td>Add UTF-16 in the transferring character code when navi creates **** from *** file, in addition to ISO_81 and SJIS and UTF-8.</td>
</tr>
<tr>
<td>Reason</td>
<td>*** system's standard character code is UTF-16, so this module must transfer from other character code</td>
</tr>
</tbody>
</table>

### Change designation of user setting used in **** function

### Change the process of reading out file

### Change the designation of transferring character code at readed

#### Add the process of transferring **** data to UTF-16 before the process of ****.

#### Add UTF-16 in the transferring character code when navi creates **** from *** file, in addition to ISO_81 and SJIS and UTF-8.

### Add the process of transferring **** data to UTF-16 before the process of ****.

#### Add the process of definition and initialization of variables used in **** function before it is called.

#### After Req1.5.1.1, get the data size of *****, in case it is transferred to UTF-16.

#### Transfer the *** data to UTF-16, and reserve it to the area acquired at
Result of PWAT(B)

• Process
  – Compare X-PWAT(B) to X-PWAT(A)

• Result

- Defect detection rate in QA Test was improved
- Productivity was improved by minimum necessary investigation
Summary of Case 2

• It is important to design investigation process in XDDP
  – X-PWAT(A) was not effective to PWAT(B)
  – XDDP will be more effective by designing investigation process

• It is necessary to detect and describe specifications properly
  – In PWAT, it is difficult to understand all the base source code
  – We should make efforts to detect change specifications and describe them. That makes reviews more effective
Conclusion

• We applied XDDP to car navigation software development
  – Quality and productivity were improved
  – Overall productivity can be improved even if we spend much time in identifying all change points

• We confirmed XDDP could be applied to PWAT
  – We improved investigation process by designing the process in accordance with the knowledge level of engineer
  – The effectiveness was confirmed in our projects