An Agile Development Method for Multiple Product Lines of Automotive Software Systems

Kengo Hayashi*
DENSO CORPORATION
kengo_hayashi@denso.co.jp

Mikio Aoyama
Nanzan University
mikio.aoyama@nifty.com
Scenario

1. Background
2. Challenges
3. Approaches
4. Twofold Iterative Process
5. Process Design and Process Assets
6. Independence Analysis of Variability
7. Application and Effectiveness
8. Discussion and Future Works
9. Conclusions
1. Background
Parking Support System with Ultra Sonic Sensors

Product evolution is fast and expected to expand into many vehicle variations

Need to deal with both variability and agile evolvability concerns
Development Organization and Software Architecture

Collaboration between two divisions in the practice of SPLE

The Core Team Develop

- Change of Major Functions
- Improvement of Functionality
- Software Architecture Refinement
- Add Security Functions
- Change of Automotive Platform

The Derivative Team Develop

- Add Security Functions
- Change of Automotive Platform
- Improvement of Functionality
- Software Architecture Refinement

Legend
PD : Product Derivation
Evo : Product Line Evolution
◆ : Versions of Core Assets
○ : Core Product
● : Derivative Product

The derivative team concurrently develops along with MPLE
2. Challenges
SPLE (Software Product Line Engineering)

SPLE deals with diversity by separating development into:
- Domain engineering
- Application engineering

In practice
Some issues to be solved for SPLE
- Incomplete architecture
- Evolutionary change over multiple generations
- Lack of test automation

The cost of application engineering done not become 0
**MPLE (Multiple Software Product Line Engineering)**

The derivative team develops multiple products concurrently.

**Development Schedule**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Develop of Product C</strong></td>
<td><strong>Develop of D</strong></td>
<td><strong>Develop of F</strong></td>
<td><strong>Develop of H (ver.1)</strong></td>
<td><strong>Develop of I</strong></td>
</tr>
<tr>
<td><strong>Develop of H (ver.2)</strong></td>
<td><strong>Release of I</strong></td>
<td><strong>Release of J</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Portfolio Management**

- Overdelivery?
- Developer release?
- Early development?
- Add developer?
- Request for overtime?
- Productivity

**Portfolio management becomes more complicated, and risk increases**
Issues in Portfolio Management of MPLE

**Portfolio**

- **Developer Resources**
- **Core Assets**
- **Development Environment**

**Issues**

- It is painful to increase or decrease developers since domain skills are necessary
- It is confusing to modify the same asset at the same time
- A limitation of the resource of test environments causes a bottleneck of testing

Realize effective portfolio management for stable development
3. Approaches
Overview

**Issues**

- **Developer Resources**
  It is painful to increase or decrease developers since domain skills are necessary.

- **Core Assets**
  It is confusing to modify the same asset at the same time.

- **Environments**
  A limitation of the resource of test environments causes a bottleneck of testing.

**Approach**

**Scrum with multi projects**

- Grasp more accurately on:
  - Time-box
  - Story point

- Incrementally into subdivisions:
  - Feature Oriented

**Solution**

**Twofold Iterative Process Structure**

- **Process Design**
- **Process Assets**
- **Dependency Analysis of Variability**

**Introduce Scrum’s framework to enhance portfolio management**
Twofold Iterative Process Structure
Twofold Iterative Process Structure

<Conventional Agile Method>
Learn productivity and feedback to the plan with single iteration loop

<Twofold Approach>
Minor iteration loop within a project

Major iteration loop across the multiple projects

Two fold feedback within and across iterative multiple projects
Monitoring Development Size and Productivity

Integrated measured velocity = planning guidelines

Burndown chart of Product C
- Implementation is reduced by product D
- Increase / decrease from plan

Burndown chart of Product D
- Difference between start to end

Velocity graph integrating multiple product developments
- Stack consumption points of each sprint for all product development
- Productivity of development team = moving average of 7 sprints

Record of development size
- Refer records in the portfolio plan
5. Process Design and Process Assets
Process Design and Process Assets

The process can be iteratively reused over the engineering of multiple applications

Design and Reuse Processes as Process Assets:
- Tailoring result
- Work procedure
- Configuration of artifact

Designed Process maintains the learning effect across the projects
6. Dependency Analysis of Variability
### Dependency Analysis of Variability

Dividing method and order constraints of development determined by dependency of variation points

#### Analyze the structure of variability

#### Analyze the dependency of the set of variation points

#### Identify the order of divided development unit

#### Realize incremental development with less regression testing cost

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Variability structure</th>
<th>Constraints and dividing method</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>vp1 (\rightarrow) vp2</td>
<td>vp1 (\rightarrow) vp2</td>
</tr>
<tr>
<td></td>
<td>VPo (\rightarrow) vp2</td>
<td>VPo (\rightarrow) vp2</td>
</tr>
<tr>
<td>vp2 depends on vp1</td>
<td>vp1 (\rightarrow) vp1</td>
<td>Constraints (\rightarrow) vp1</td>
</tr>
<tr>
<td></td>
<td>VPo (\rightarrow) vp2</td>
<td>VPo (\rightarrow) vp2</td>
</tr>
<tr>
<td>vp1 and vp2 are inter-dependent</td>
<td>vp1 (\rightarrow) vp1</td>
<td>vp1 (\rightarrow) vp1</td>
</tr>
<tr>
<td></td>
<td>VPo (\rightarrow) vp2</td>
<td>VPo (\rightarrow) vp2</td>
</tr>
</tbody>
</table>
7. Application and Effectiveness
Application

The presenter as the leader of the development team

<table>
<thead>
<tr>
<th>Development duration</th>
<th>10 months</th>
</tr>
</thead>
<tbody>
<tr>
<td>A unit of sprint</td>
<td>2 weeks</td>
</tr>
<tr>
<td>Total number of sprints</td>
<td>22 sprints</td>
</tr>
<tr>
<td>Total number of projects</td>
<td>11</td>
</tr>
<tr>
<td>Size of unit (KLOC)</td>
<td>1 - 20</td>
</tr>
</tbody>
</table>

Development Period and Target Number of Projects

Statistics obtained from the actual projects
Stability of the Development

Measure the velocity (moving average of 7 sprints)
productivity is predictable = development is stable

The development is highly stable if items have iterativeness
Leveling the Test Effort and Usage of Test Environment

Comparison of test environment usage rate with similar scale development

<table>
<thead>
<tr>
<th></th>
<th>Conventional</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test days (Day)</td>
<td>12</td>
<td>20</td>
</tr>
<tr>
<td>Test hour (H)</td>
<td>77.75</td>
<td>78.32</td>
</tr>
<tr>
<td>Average (%)</td>
<td>80.99</td>
<td>48.95</td>
</tr>
<tr>
<td>Max (%)</td>
<td>162.50</td>
<td>106.25</td>
</tr>
</tbody>
</table>

Lowered usage rate and leveling the peak load
Higher Velocity and Better Manageability of Value Stream

Distribution of development period vs development effort for each development items
Lower development time/period indicates higher velocity
Lower SD indicates better manageability

<table>
<thead>
<tr>
<th></th>
<th>Conventional</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item Count</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>Dev. Time (H)</td>
<td>31.53</td>
<td>19.99</td>
</tr>
<tr>
<td>SD</td>
<td>25.49</td>
<td>13.02</td>
</tr>
<tr>
<td>Dev. Period (Day)</td>
<td>16.82</td>
<td>4.82</td>
</tr>
<tr>
<td>SD</td>
<td>7.21</td>
<td>2.15</td>
</tr>
</tbody>
</table>

SD: Standard Deviation

Reduced variations and improved velocity of value stream
8.
Discussion and Future Works
Discussion and Future Works

Q1. Has the portfolio management been strengthened?
A1. Yes. Stable productivity was obtained, the development scale was able to be grasped, and it become possible to keep updating the executable plan.

Q2. Is this approach the best?
A2. No, but Better. Automatic testing and a more ideal configuration system can realize simpler development.

Q3. Do not apply agile development for domain engineering?
A3. Domain engineering is easier to apply. However, it is necessary to care the architecture because the architecture is easy to erosion.

Q4. Is further improvement possible?
A4. Yes. In the future, we plan to develop an architecture accommodating concurrent development with domain engineering.
Conclusions
Conclusions

Goal

• Improvement of manageability in concurrent product development on MPLE

Solutions

• Twofold Iterative Process Structure
• Process Design and Process Assets
• Dependency Analysis of Variability

Benefits

• The development is highly stable if items have iterativeness
• Lowered usage rate and leveling the peak load
• Reduced variations and improved velocity of value stream

Future Works

• Develop an architecture accommodating concurrent development with domain engineering
About the Speakers

Mr. Kengo Hayashi is an architect and project manager of Advanced Safety Engineering Div., DENSO CORPORATION, Kariya, Japan. He has engaged in the development of car navigation software systems and advanced sensing software system. He is pursuing the doctoral program at Aoyama Laboratory, in the graduate school of software engineering, Nanzan University. His research interests include software management, software product line engineering, and agile development.

Dr. Mikio Aoyama is a professor of Dep. of Software Engineering, Nanzan University, Nagoya, Japan since 2001. His research interests include requirements engineering, software architecture, and machine learning for the applications in cloud/edge computing and automotive systems. Prior to joining academia, Dr. Aoyama has 15 years of experience in the development of large-scale real-time distributed systems at Fujitsu Limited. His paper “Agile Software Process and Its Experience” presented at ICSE ’98 is one of the earliest work on agile.
DENSO
Crafting the Core