The evaluation knowledge of standard software asset using The Seven Samurai framework

Nobuhide Kobayashi\textsuperscript{a,b}, Hikari Yamada\textsuperscript{a}, Hiroyuki Utsunomiya\textsuperscript{a,b}, Shuji Morisaki\textsuperscript{b}, Shuichiro Yamamoto\textsuperscript{b}

\textsuperscript{a}DENSO CREATE INC. 3-1-1 Sakae Naka-ku, Nagoya Aichi 460-0008 Japan
\textsuperscript{b}Nagoya University Furo-cho Chikusa-ku, Nagoya Aichi 464-8601 Japan

Abstract

The knowledge which is needed on automotive software development, increases significantly according to large-scale, complexity of automotive software. Therefore, it is very difficult for an engineer to understand the whole software development. This paper introduces a way of constructing a meta-model, which visualizes the knowledge of expert engineers, based on The Seven Samurai framework. It can solve issues of system development by considering the seven types of elements which are defined in this framework. And then, its name was defined based on the famous Japanese cinema. Additionally, this paper shows the evaluation results of applying the meta-model to the evaluation of actual standard software assets in the product line, and then the effectiveness of the proposed approach is confirmed based on the results.

Keywords: The Seven Samurai framework; product line; variants; meta-model; NFR framework

1. Introduction

Efficient development methods of automotive software are needed to support next-generation system developments such as the self-driving system in the automotive industry. Product line is one of the good method for improved productivity. But actual development projects have to keep standard software assets which have appropriate variants in order to achieve a significantly effective product line. However, according to large-scale, complexity of automotive software, it is difficult for an engineer to understand all activities of a software development. Therefore, an actual development project always needs to depend on the knowledge of expert engineers. This paper shows the standard meta-model which is related to activities of an automotive software development based on The Seven Samurai framework. And then, it explains the method of extracting variants on product line based on the meta-model. Finally, it
proposes the method of evaluating standard software assets based on variants extracted as criteria. Section 2 describes the related work of activities of automotive software development. And then, section 3 explains the meta-model extracted using The Seven Samurai framework, section 4 defines the evaluation method of standard software assets. And then, section 5 describes the case study of the proposed method for a standard software asset, section 6 discusses the effectiveness of the method. Finally, section 7 concludes this paper and shows future work.

2. Related work

In the automotive industry, EAST-ADL and AUTOSAR are standardized as description language for work product developed by activities of an automotive software development. These standards have been adopted to actual development projects of mainly European companies. EAST-ADL standardizes all design factors and their relations on activities of an automotive software development. Also, it supports factors related with safety cases required when we correspond to ISO26262. While, AUTOSAR standardizes specifications related with software platform which provides common functions needed by all automotive software. It then standardizes specifications related with software component which realizes application control depended on requirements of each products. Finally, it standardizes process and toolchain based on these specifications. However, the amount of specifications is enormous and the details of its descriptions are very complex. Therefore, it is difficult to extract only related information to variants which are evaluation criteria for standard software assets, from these standards. Additionally, a notation of variants and various design methods using EAST-ADL and AUTOSAR are proposed. However, a clear view which visualizes all activities of automotive software development has not been discussed. The lifecycle of product line based development has been discussed, including examples of automotive companies. However, activities of automotive software development can not be visualized based on reference models such as The Seven Samurai framework. And then, modeling of design factors related with product line have been discussed, but have not been adopted to automotive software development.

3. Meta-model of an automotive software development

This section explains the meta-model which is visualized activities of an automotive software development by using The Seven Samurai framework. It is the simple framework which is consisted of the only seven type of elements, therefore it is easy for engineers to understand the visualized activities and their relations. And then, this meta-model describes the details for the only following systems. Problem (P1) describes that dangerous driving causes accidents, because the consideration of these issues is very important on an automotive software development. And then, Intervention system (S2) is detailed according to P1. Other systems are described by the granularity that can be reused on a general automotive software development. Finally, Fig. 1 shows the whole of this meta-model.

3.1. Context system (S1)

S1 describes the causes of Problem (P1). In automotive software development, S1 correspond to ”Environment” such as climate and ”User”. As the usage of an actual project, S1 can be used for various analysis by embodying factors of S1 based on features of automotive software. For example, ”User” is embodied as a driver, and then it has language as the attribute. This result can be used for extracting variants of product line based on the difference of drivers.

3.2. Intervention system (S2)

Requirements for automotive software are classified into functional requirements related to the benefit of users and non functional requirements related to the compliance on the realization of functional requirements. S2 describes necessary functional requirements to solve P1, and it is embodied as Vehicle status indication, Dangerous drive warning and Vehicle status monitoring in this paper. When we describes more details of functional requirements on actual projects, it is possible to adequately embody S2 according to factors related to S2 on the meta-model. For example,
Vehicle status monitoring is related to monitor Collaborating system (S5). According to factors of S5, it can be embodied such as SensorActuator status monitoring and so on. Finally, the above mentioned non functional requirements are described in S3 and S5 later.

3.3. Realization system (S3)

S3 describes necessary resources and constraints to realize functional requirements defined on S2. First, the former is embodied as Developers, Process, Method, Design tool based on PMTE\textsuperscript{16}. And then, the latter is embodied as Regulations which need to be complied by an automotive software product. Furthermore, non functional requirements are classified into the above mentioned constraints like regulations and attributes of functional requirements such as processing speed\textsuperscript{15}. Attributes of functional requirements are explained on S5 later.

3.4. Deployed system (S4)

S4 describes factors related with the structure of automotive software. Additionally, factors are extracted as the following.

- S2 is realized by Application component.
- Microcontroller and Other ECU (Electronic Control Unit) on S5 are controlled by Software platform
As the above mentioned, it is possible to design adequate software structure which can cover influences of variants by decomposing factors corresponding to each factor in other related systems. Finally, this software structure is compliant with the AUTOSAR software structure (refer to Fig. 2).

3.5. Collaborating system (S5)

S5 describes the targets controlled by S4 in order to realize S2. Concretely, it is composed of Microcontroller, SensorActuator, and Other ECU. S4 is embedded in Microcontroller, and the SensorActuator is controlled by S4. Finally, Other ECU means the communicating target of S4. Additionally, the above mentioned non functional requirement related to ”attributes of functional requirements” is described as attributes of this system’s factors.

3.6. Sustainment system (S6)

S6 describes necessary resources to maintain S4. And then, the classification of resources are the same as S3.

3.7. Competing system (S7)

S7 describes alternative software components within the development of S4. Software platform will be expected to be exchangeable according to changes of S3 on an AUTOSAR compliant development. Therefore, S7 is embodied as Other software platform.

3.8. Modified context system (S1’)

S1’ is composed of S4, S5 and Problem (P2). P2 describes additional issues caused when S2 is deployed to S4.

4. Evaluation method of a standard software asset

This section explains the method of extracting variants as the effectiveness criteria of a standard software asset. It is difficult to define the effectiveness criteria which convince various stakeholders. However, the proposed method can extract variants as the criteria based on the meta-model which is defined in section 3. Therefore, if stakeholders have approved the meta-model, they can be convinced the criteria. Additionally, this section also shows the method of evaluating a standard software asset by using extracted variants.
4.1. Extraction of criteria

This section shows the way of extracting criteria based on the meta-model of an automotive software development. The following explains the step of extracting criteria.

Step1: Extract factors of other systems related with factors of Deployed system (S4).
Step2: Create the matrix of extracted factors on Step1 in the horizontal rows, and factors of S4 in the vertical columns (referred to as the evaluation matrix later).
Step3: Check corresponding cells if each row factor is related to each column factor within the evaluation matrix. Table 1 shows that the evaluation matrix is created according to this section.

Table 1. The evaluation matrix.

<table>
<thead>
<tr>
<th>Evaluation factors</th>
<th>Application component</th>
<th>SensorActuator component</th>
<th>Software platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intervention system (S2)</td>
<td>Vehicle status indication</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dangerous drive warning</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vehicle status monitoring</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Realization system (S3)</td>
<td>Design tool</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Collaborating system (S5)</td>
<td>SensorActuator</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other ECU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Microcontroller</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sustain system (S6)</td>
<td>Diagnostic tool</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Competing system (S7)</td>
<td>Other software platform</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2. Evaluation of the effectiveness

This section explains the method of evaluating the effectiveness of a standard software asset using the evaluation matrix created in section 4.1. Additionally, this paper adopts weighted SIG (Softgoal Interdependency Graphs) as the evaluation view of effectiveness\(^{17,18}\). SIG is the notation which can visualize a goal tree of non functional requirements, and relations between a goal tree and operations (ex. candidate of software architecture)

Step1: Define the top goal as "Compliance for the standard software architecture".
Step2: Decompose to sub Softgoals according to factors in the row of the evaluation matrix. Finally, the bottom of the Softgoal tree are corresponded to variants of the standard software architecture.
Step3: Define the weight for each sub Softgoals. The total of the weight needs to be 1.0.
Step4: Correspond the evaluation target to an operation of SIG. If the change at the bottom of Softgoal can be covered by the only marked factor in the column direction of the evaluation matrix, the relation between the bottom of Softgoal and the operation is defined as a positive relation (+1). If not, the relation is defined as a negative relation (-1).
Step5: Calculate the evaluation result \(A_w(g)\) as the effectiveness of a standard software asset using the following expression(Child(g) is a set of subgoals). \(P_w\) defines priority weight for the decompositions of Softgoals. \(C_w\) defines the contribution weight between the parent and child Softgoals.

\[
A_w(g) = \sum_{h \in \text{Child}(g)} P_w(h) \times C_w(h) \times A_w(h)
\]
5. Evaluation example for a standard software asset

5.1. The way of confirmation for the effectiveness

This section explains the way of confirmation for the effectiveness of our proposal based on the case study. In this case study, to evaluate the two software shown in Fig. 3. (a) shown in the figure is not considered reusability, Application and SensorActuator component is implemented for each actuator. Therefore, the application controls which realizes functional requirements and the actuator controls are implemented as the monolithic software components. After that, a software component gets the necessary sensor values from Microcontroller in each software component. Finally, Communication which is developed as its own communication module is adopted to cover the change of communication with Other ECU (includes the communication with Diagnostic tool). Incidentally, (b) is the software which is considered reusability, the changes related with Microcontroller, Sensor and Actuator can be covered. However, each software are compliant with proprietary interfaces, they are not compliant with AUTOSAR which is the standards in the automotive software industry.

Finally, this paper can not consider the further decomposition of Intervention system (S2), because the meta-model of an automotive software development relates to Intervention system (S2) and each factor of Deployed system (S4). Therefore, this case study is set as the following precondition.

- Intervention system (S2) is not changed.
- The target software has a lot of occasions to be embedded with various automotive systems.

5.2. The result of this experiment

Fig.4 and Fig.5 shows the weighted SIGs of target (a) and target (b), respectively. Both have the negative relations for Design tool and Software platform, because they adopt the proprietary interfaces and are not compliant with AUTOSAR. Then, (a) has the negative relations for Softgoals except Other ECU and Diagnostic tool, because it is implemented as the monolithic software except the communication function.

Finally, Table 2 and Table 3 shows the evaluation results calculated based on the above-mentioned weighted SIGs. The evaluation value of (a) is -0.44, and the value of (b) is 0.11. This result shows that (b) is more effective than (a) on the effectiveness as the standard software asset, and it is consistent with the expectation of the proposed method. Because (b) is considered reusability, (a) is not considered. Additionally, this case study is set in the precondition defined in the section 5.1. Therefore, the weights related with S5 and S7 are defined as more important factors.
Fig. 4. Evaluating the impact of the target system (a).

Table 2. Tabular evaluation of the target system (a).

<table>
<thead>
<tr>
<th>SIG decomposition structure</th>
<th>Target system (a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compliance for the standard software architecture</td>
<td>1/9</td>
</tr>
<tr>
<td>Compliance for Intervention system (S2)</td>
<td>1/3</td>
</tr>
<tr>
<td>Compliance for Realization system (S3)</td>
<td>1/3</td>
</tr>
<tr>
<td>Compliance for Collaborating system (S5)</td>
<td>1/3</td>
</tr>
<tr>
<td>Compliance for Sustainment system (S6)</td>
<td>1/3</td>
</tr>
<tr>
<td>Compliance for Competing system (S7)</td>
<td>1/3</td>
</tr>
</tbody>
</table>

| Compliance for Intervention system (S2) | 1/3   |
| Compliance for Realization system (S3) | 1/3   |
| Compliance for Collaborating system (S5) | 1/3   |
| Compliance for Sustainment system (S6) | 1/3   |
| Compliance for Competing system (S7) | 1/3   |

-0.44  
-1.00  
1  
-1.00  
0.00  
1  
1.00  
-1.00  

6. Discussion

The experiment result of Section 5.2 showed that the proposal of this paper could quantitatively evaluate the effectiveness of a standard software asset. Also it showed that the meta-model of an automotive software development was useful for extracting criteria of various evaluation. The knowledge related with activities of an automotive software development will be more difficult to be shared according to large-scale, complexity of automotive software. From this result, the Seven Samurai framework was confirmed to be effective for sharing the knowledge. This framework
is consisted of only seven factors, and it is easier to learn than huge standards such as EAST-ADL, AUTOSAR. Therefore, it is expected to easily apply to actual development projects.

However, this paper can not consider more decomposition of Intervention system (S2) which is affected by frequent changes on actual automotive software developments. From now on, it is necessary to evaluate more case studies included in the detail of S2.
7. Conclusion

This paper introduced that the knowledge of an automotive software development can be visualized as the meta-model based on the Seven Samurai framework. It could then propose the evaluation method of a standard software asset as one of methods which can be invented by using this meta-model. Additionally, the experiment result for the two softwares (which have a difference to span on influence for changes) proved the effectiveness of the proposed method. It could also show the quantitative value as the reusability of automotive software is calculated by using the weighted SIG.

Future work includes more experimental evaluation of the proposed meta-model based on the Seven Samurai framework. In particular, it is important to apply to actual automotive software developments.

Acknowledgements

We are particularly grateful for the assistance given by co-workers at DENSO CREATE INC..

References

5. The AUTOSAR development partnership, . AUTOSAR System Template Release 4.2.2. 2015.