Extending ATAM to Assess Product Line Architecture

Taeho Kim, In-young Ko, Sung-won Kang and Dan-hyung Lee
Information and Communications University, Daejeon, Korea
{taeho, iko, kangsw, danlee}@icu.ac.kr

Abstract
Software architecture is a core asset for any organization that develops software-intensive systems. Unsuitable architecture can precipitate disaster because the architecture determines the structure of the project. To prevent this issue, we have to evaluate software architecture. Current evaluation methods focus on single product architecture, not product line architectures and hardly considered the characteristics of product lines, such as variation points. To resolve this issue, we need to create a new evaluation method or extend current architecture evaluation methods to evaluate software product line architecture. This paper will focus on the latter and provide several methods to evaluate product line architecture. The first extended method identifies four common quality attributes of the product line architecture. The second one figures out architecture views. The third develops tags to indicate the variation points, and the last one repeats ATAM phases to validate the product architecture using Extended ATAM.

1. Introduction
The purpose of evaluating the architecture of a software system is to design the best architecture to validate the quality requirements, improve stakeholder communication and eliminate system risk. Currently, most evaluation methods focus on single product architecture, not product line architecture because finished products derived from single product architecture. There are few evaluation methods for software architecture in product line engineering. In addition, there are some problems to assess product line architecture. First, existing methods for product line assessment cannot be applied directly to product line architecture [4], and it is hard to assess product line characteristics in detail with present evaluation methods [11]. Second, individual product architecture may require a different prioritization of quality goals than product line architecture [6]. Third, the validation of product line software architecture is inherently more complex than that of software architecture for single systems. [9]. A key concern with a product line intended to span several decades has been the adequacy of the architecture’s modifiability in implementing required variability in future systems. Current methods make it hard to evaluate architecture’s modifiability [10]. Lastly, product line architecture supports different sets of goals that are specific to different products, but these goals are often conflicting. [12]. Product line architecture needs to consider three things to resolve above issues. The first is identifying and supporting variation points. The second is considering common quality attributes for product line architecture such as reusability. The last one is evaluating the architecture for product line suitability. To assess product line architecture, any evaluation method should satisfy all of the above. The current methods cannot do this because it focuses on characteristics of single product. This paper suggests a solution to resolve these problems. The remainder of the paper is organized as follows: Section 2 presents what ATAM is. Section 3 develops an extension of ATAM. Section 4 describes the steps of Extended ATAM, Section 5 presents and discusses the case study, and Section 6 concludes. My main motivation is to develop an evaluation method for architecture design in product line engineering that will ultimately lead to increased product quality and development while reducing cost and time-to-market. Characteristic

2. Architecture Tradeoff Analysis Method (ATAM)
ATAM is the successor of SAAM, and it is suitable for analyzing multiple quality attributes. ATAM is one of the best-known. It helps identify the architectural approaches used and, by means of scenarios, exposes areas of risk as a result. These steps are preceded by a brief presentation of the architecture and its business goals.
2.1 Current ATAM Steps

ATAM originally has 2 phases, with 6 steps in the first phase and 3 steps in the second. During phase 1, the evaluation team meets with the project team to gather and analyze information for about a day. During phase 2, the stakeholders join with the project team, and analysis continues for two days, typically. In the first step, the evaluation manager presents what ATAM is and what the purpose of ATAM is. The evaluation team members must understand the context for the business drivers. The project’s architect presents the system’s most important functions and business goals and the architectural drivers from step 2. In step 3, the architect gives details of the architecture, such as quality requirement and the architectural design. After getting the highest-ranked scenarios from step 5, the evaluation team analyzes the architectural approaches. This paper will focus on steps 4, 5, and 6 of the first phase; though all steps of both phases require some modification.

3. An extension of ATAM

Current ATAM evaluation methods focus on the evaluation of single product architectures [11], even though ATAM has been used successfully to assess product architecture in a variety of domains [6]. There are several studies regarding evaluating product line architecture using ATAM. However, most of them evaluated a single product and hardly considered the characteristics of product lines, such as variation points.

This paper provides several methods to evaluate product line architecture, added to the original ATAM approach, resulting in EATAM. The first extended method identifies four common quality attributes of the product line architecture. The second one figures out architecture views based on those four quality attributes. The third develops tags to indicate the variation points among the four quality attributes, and the last one repeats ATAM phases to validate the product architecture using EATAM.

3.1 The two stages of evaluating product line architecture and product architecture

Since ATAM can evaluate only one product, it is not suitable for product line architecture as is. Therefore, the evaluation method must be extended in two ways, first, by evaluating product line architecture and second, by evaluating multiple products. Figure 2 shows the structure for extended ATAM (EATAM). As you can see, we have added an extra stage to ATAM. Extended ATAM has one stage to evaluate product line architecture and another stage to evaluate product architecture. On the right side of the table, we can see product architecture for several products which were derived from product line architecture. Each product is evaluated separately. If we were to integrate the 2 stages, we would lose some of the flexibility of a two-stage process. For example, sometimes we need to add additional quality requirements and functions, but in that case, the product architecture would not have the same quality attributes and requirements among its products. To resolve this issue, Extended ATAM has two stages. Software architecture for a product line must focus on essential qualities. Therefore, the first stage has four common quality attributes for product line architecture. Modifiability is the ability to assemble various elements in various combinations. Portability is an ability to use various Hardware elements. Scalability is an ability to support getting smaller. Extensibility is an ability to add features to the product family. However, additional quality attributes can be added, such as performance and security. Then, domain and feature models are analyzed to determine variation points.

Figure 1. Current ATAM Steps

Figure 2. The extended ATAM
3.1 Focusing on common quality attributes of product line architecture

Software architecture for a product line must focus on maximizing four qualities. The first is modifiability, which is the ability to assemble various elements in various combinations. The second is portability, the ability to use various Hardware (HW) elements. The third is scalability or the ability to manage variability in size and features. The last is extensibility, which is the ability to add features to the product family. All of the above are essential and unique quality attributes when product line architecture is created. Therefore, we also should consider these four qualities attributes if product line architecture is evaluated.

It is important to consider quality attributes in product and product architecture. There are essential quality attributes in product line architecture such as extensibility. The first stage, as shown, has four common quality attributes to evaluate product line architecture, but additional quality attributes can be added, such as performance and security. Then, domain and application models are analyzed to get variation points and quality attributes.

3.2 Analyzing variation points using domain analysis and feature modeling

A conceptual view is related to commonality and variability, and a realization view is needed to see modifiability and portability, which are important quality attributes of product line architecture. Product architecture encompasses all quality attributes, but product line architecture includes only 2 views regarding a product line’s quality attributes. Therefore, product line needs to concentrate on architecture views in order to get quality attribute scenarios because architectural views which are derived from requirements are the actual model which depicts requirements more in detail.

Finally, the conceptual view is analyzed to get variant points, includes commonality and variability of the product line, and the realization view is also analyzed to get the characteristics of product line architecture based on essential quality attributes of product line architecture.

3.3 Defining the PLUC tag approach to quality attributes

Product Line architecture must identify and support the variation points among its family members. Extended ATAM handles this issue by using extended Product Line Use Cases (PLUC) tags because the numerous variations of product line architecture can be presented more explicitly and can be added or changed more easily by using tags.

When an evaluation method is used to assess product line architecture, variant points of the product line must be considered in order to generate exact quality attribute scenarios. Variability and commonality are derived from Use Cases. Use Cases are a really useful way to describe the functional requirements of a system and to create a feature model of the product line. Most evaluation methods do not specifically reflect the variation points of a product line’s artifacts. In addition, the quality attributes of product line architecture can be changed according to a product’s purpose.

The product line can be recognized as a set of products with common characteristics. It is possible to relocate from the product line to the product architecture level which describes a single product. It is difficult to demonstrate all variation points when the quality attributes of product line architecture are changed, but it is unnecessary to reflect all of them anyway. Therefore, the proposed extension adds variability to the four quality attributes based on a tag that denotes the variation points of the product line requirements. For this reason, tags are incorporated into the four quality attributes in order to identify quality attribute scenarios and identify specific variations. This extension will be used on ATAM. The tags can be of four kinds.

- Optional
  Optional features are those features that need to be provided by only some members of the product line.

- Alternative
  Two or more features may be alternatives to each other, where only one of them can be provided in any given member of the software product line.

- Parameterized
  A parameterized feature is a feature that defines a product line parameter whose value needs to be defined at system configuration time.

- Prerequisite
  A feature may depend on another feature; the feature it depends on is called a prerequisite feature.

- Mutually inclusive
  If two features are always needed together, then the features are considered mutually inclusive features.

Tags also will be used in ATAM in order to generate a quality attribute utility tree.
3.4 Creating product line scenarios of PLUCs for derivation of variation points

Domain and feature models are analyzed to get variation points. Since variation points are rather complex, they are expressed as tag types and then abbreviated to different types of product line scenarios. These product line scenarios are used to create a quality utility tree. It is difficult to manage all variation points because there are a lot of them. We can easily express all variation points as extended PLUC tags easily. The numerous variations points of product line architecture can be expressed more simply and can be added or changed more easily by tags. Product line architecture has not only many variation points (e.g., Language), but variations (e.g., Italian, English, Korean) as well. Table 1 shows a sample of product line scenarios using tag expression.

Table 1. A sample of product line scenarios using tag expression

<table>
<thead>
<tr>
<th>Product Line scenario expression</th>
<th>Quality Attribute</th>
<th>Architecture Attribute</th>
<th>Generic Product Scenario</th>
<th>Product Line Scenario</th>
<th>I</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>M0_SS1</td>
<td>Feature upgrade on Display</td>
<td>PLA</td>
<td>[GPS, MO] on display layout and beep sound in &lt; 3 person-weeks</td>
<td>M0_SS1-1</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>M0_SS2</td>
<td>PLA</td>
<td>M0_SS6-10</td>
<td>[GPS, MO] on display layout and beep sound in &gt; 3 person-weeks</td>
<td>M0_SS6-10</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>M0_SS3</td>
<td>PLA</td>
<td>M0_SS11-18</td>
<td>[GPS, MO] on display layout and beep sound in &lt; 2 person-weeks</td>
<td>M0_SS11-18</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>M0_SS4</td>
<td>PLA</td>
<td>M0_SS1</td>
<td>[GPS, MO] on display layout and beep sound in &gt; 2 person-weeks</td>
<td>M0_SS1</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>M0_SS5</td>
<td>PLA</td>
<td>M0_SS20</td>
<td>[GPS, MO] Add the ability to display controlled and make beep sound via remote control in &lt; 2 person-weeks</td>
<td>M0_SS20</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>M0_SS6</td>
<td>PLA</td>
<td>M0_SS12-3</td>
<td>[GPS, MO] Add the ability to display controlled and make beep sound via remote control in &gt; 2 person-weeks</td>
<td>M0_SS12-3</td>
<td>H</td>
<td>L</td>
</tr>
</tbody>
</table>

As you can see, M0_SS1, which is one of the product line scenarios in the product line scenario field in Table 2, includes tag expressions, and the tag expression has a lot of meanings. Table 3 shows the details of M0_SS1.

Table 3. Tag expression of Product Line variability

<table>
<thead>
<tr>
<th>Variability Scenario</th>
<th>MO_SS1</th>
<th>Tag expression</th>
<th>Modifiability</th>
<th>Priority</th>
<th>Feature Category</th>
<th>Variant Points</th>
<th>Prerequisite</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>OTI=1 &amp; (OTI=1) &amp; (RTI=1)</td>
<td>MO</td>
<td>(H</td>
<td>L)</td>
<td>Optional</td>
<td>Multi-line display</td>
<td>Multi-tone enabled</td>
</tr>
<tr>
<td></td>
<td>MO_SS1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SS1</td>
</tr>
<tr>
<td></td>
<td>MO_SS2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MO_SS3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.5 Evaluating product line architecture using variability scenario

Since variation points are rather complex, they need to be expressed as tag types and then abbreviated to different types of product line scenarios. After that, the product line scenarios are applied to the quality attribute utility tree as shown Table 2. The product line scenarios are based on generic product scenario which has several product line scenarios.

3.6 Assessing Product Line Architecture using Extended ATAM

There are two evaluation stages in Extended ATAM. The first stage focuses on product line architecture and the other focuses on product architecture. It makes more sense to evaluate product line architecture before evaluating product architecture. The characteristics of
product line architecture need to be gathered to evaluate product line architecture.

When evaluating product line architecture, the evaluation team needs to gather and analyze customer requirements in order to identify quality attributes. After completing the evaluation of product line architecture, the evaluator must find high priority quality attributes in the product line architecture and look for those attributes in product architectures and make any modifications necessary to satisfy both customer requirements and the priorities of the product line architecture.

3.7 Assessing Product Architecture using Extended ATAM

Since the product line architecture is evaluated in Stage 1, its architectural approach is described. The results from Stage 1 are used in Stage 2 to assess product architectures. The product line scenarios are applied to product architecture. If we were to integrate both Stage 1 and Stage 2, we would lose some of the flexibility of a two-stage process.

4. The Steps of EATAM

Extended ATAM has two stages. The first stage is to evaluate product line architecture, and the second one is to evaluate product architecture.

4.1 Stage I: Product Line Architecture Evaluation

These are detailed steps of EATAM evaluation. Steps 4, 5 and 6 of original ATAM are extended to evaluate product line architecture. This then becomes Stage 1 of EATAM. ATAM originally has 2 phases, with 6 steps in the first stage and 3 steps in the second. This paper will focus on steps 4, 5, 6 of the first phase, though all steps of both phases require some modification.

4.1.1 Present the EATAM

The evaluation team presents an overview of the Extended ATAM and its activity.

4.1.2 Present the business drivers

The evaluation team describes the product line architecture’s business drivers and business needs, the commonality and variability of product line architecture. Project team has to present and describe the quality goals.

4.1.3 Present architecture

The project team has to present the overall product line architecture.

4.1.4 Identify architectural approaches

The evaluation team analyzes the product line architecture documentation and the architect’s presentation in step 3 and explicitly identifies the patterns and approaches used in the commonality and variability of product line architecture. They simply catalog the patterns and approaches of product line architecture.

4.1.5 Generate quality attribute scenarios

The evaluation team identifies Quality attribute specifications based on four common quality attributes and then analyzes feature modeling and domain analysis. They gather variation points using feature modeling and domain analysis. Then, domain and application models are analyzed to get variation points. They define the variation points of quality attribute specification and common quality attributes using PLUC tags. After that, they create product line scenarios of PLUCs for derivation of variation points.

4.1.6 Analyze architectural approaches

The evaluation team analyzes specific product line scenarios of high-priority generic scenarios from Step 5 and evaluates product line architecture using product line scenarios. After that, they analyze how the architecture is associated with variation points and how the associated design decisions affect quality attributes.

As you can see in figure 4, each step in Extended ATAM consists of several activities.

Figure 3 shows the steps of product line architecture which is tailored to ATAM evaluation. As I mentioned, this stage focuses on steps 4, 5, 6 of the first phase, though all steps of both phases require some modification.
Each activity includes particular inputs and generates its own outputs as shown figure 5. Each input and output is numbered, starting with A1, A2, etc. We will show the step-by-step approach to using inputs and generating outputs in each activity.

**Figure 5.** The activity of step 5 to assess PLA

### 5. Case Study: The Microwave oven product line

#### 5.1 Overview

This chapter presents the results of Extending the Architecture Trade-off Analysis method (ATAM) to assessing product line architecture. This method has been validated through a case study involving microwave oven software in the appliance domain.

#### 5.2 Scope, requirements

The microwave oven product line is a popular one of the home appliance domain. The microwave oven includes two kinds of product, basic and premium. The two products offer basic products and several options. The basic microwave oven system provides functions for cooking food, setting and displaying time. The basic oven supports a one-line display, and the other one has a multi-line display for a time-of-day clock.

### 5.3 Stage I: Product Line Architecture Evaluation

We present the results of the Extended ATAM evaluation. We define the variation points of modifiability using extended PLUC tags.

#### 5.3.1 Analyze feature modeling and domain analysis

We should be able to analyze application and domain to get variation points. The product line commonality includes the Cook food function, and display and set time functions are variation points in this product line. Display time includes display language, which is a variation point. The variants of display language are Italian, Korean and Japanese.
5.3.2 Gather variation points using feature modeling and domain

The requirements of the microwave oven are all determined from the problem description. Every microwave oven has a display to show the time and warning messages. It can present either one or two lines. Korean is the default display language, and the others are alternates.

5.3.3 Define the variation points of quality attributes using PLUC tags

We should be able to define the variation points of quality attributes using PLUC tags. Table 4 shows modifiability of PLUC tags from Microwave oven’s quality attribute. The ability to features on the display LCD is one refinement attribute.

Table 4. PLUC tags of derivation from Microwave oven’s quality attribute

<table>
<thead>
<tr>
<th>Quality Attribute</th>
<th>Attribute Refinement</th>
<th>Scope</th>
<th>Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifiability</td>
<td></td>
<td>LCD</td>
<td></td>
</tr>
<tr>
<td>LCD</td>
<td>Feature on display</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LCD</td>
<td>Feature on display</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.3.4 Create variability tags of PLUCs for derivation of variation points

In the previous section, the variation points are connected via Boolean operators. In this way, a variability scenario is generated. One of the product line scenarios includes several things, which are tag expressions, quality attributes, priorities, feature categories, variation points, prerequisites, generic scenarios and specific scenarios. Table 5 shows the detail of “M0_SS1_1”.

Table 5. Tag expression of M0_SS1_1

<table>
<thead>
<tr>
<th>Variability Scenario</th>
<th>Tag expression</th>
<th>Quality attribute</th>
<th>Priority</th>
<th>Feature Category</th>
<th>Variation points</th>
<th>Prerequisite</th>
<th>Scenario</th>
<th>Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Modifiability</td>
<td>M0</td>
<td>Optional</td>
<td>Multi-line display</td>
<td>Multi-line must be enabled</td>
<td>Upgrade the display time layout to different types of products</td>
<td>OT1==0, OT2==1 &amp; (OT1== 1) &amp; (OT2==0)</td>
</tr>
</tbody>
</table>

5.3.5 Attach product line scenarios to scenario based evaluation approach

We can see the two kinds of scenarios, which are generic product scenarios and their product line scenarios. Table 6 shows the tabular form of the utility tree for the microwave oven.

Table 6. Tabular form of the utility tree for the microwave oven

<table>
<thead>
<tr>
<th>Qualitative Attribute</th>
<th>Architecture Tag</th>
<th>Generic Product Scenarios</th>
<th>Product line Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifiability</td>
<td></td>
<td>PLA</td>
<td>M0_SS1_1</td>
</tr>
<tr>
<td>LCD</td>
<td></td>
<td>PLA</td>
<td>M0_SS1_1</td>
</tr>
<tr>
<td>LCD</td>
<td></td>
<td>PLA</td>
<td>M0_SS1_1</td>
</tr>
<tr>
<td>LCD</td>
<td></td>
<td>PLA</td>
<td>M0_SS1_1</td>
</tr>
<tr>
<td>LCD</td>
<td></td>
<td>PLA</td>
<td>M0_SS1_1</td>
</tr>
</tbody>
</table>

5.4 Stage II: Product Architecture Evaluation

We present the assessment of a single product. Some of the architectural approaches and quality attributes that can be identified in the product are: This product has the same quality attributes as M0_SS1_1 tag in Stage 1. Table 7 shows the architecture approach description for microwave ovens. The activity of approach description is the same as ATAM because it is used for a single product.

Table 7. Architecture approach description for microwave oven

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Architecture</th>
<th>ATAM Product’s Scenario 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Attribute: Modifiability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scenario: When developing multiple display with high sound, it takes a week and 3 person. Before displaying all the test, all the options must be enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Environment: Operational</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response: switch to function normally</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Architectural decisions: Trade-off</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 Ben: Modifiability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reason: architectural decisions increase the modifiability since more errors can be found in a test phase. However, performance is decreased due to the additional error checking it makes now</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3 Ben: Using more checking is required since operation and memory issues. Can be critical issue in microwave oven</td>
</tr>
</tbody>
</table>

5.5 The result of Case Study

We analyzed feature modeling and domain analysis and gathered variation points using feature modeling and domain. Then, we defined the variation points of common quality attributes and created product line tags of extended PLUCs for derivation of variation points. It is difficult to create a utility tree with all variation points of product line architecture in ATAM. However, extended ATAM resolves this issue by using product line scenarios which are derived from extended PLUC tags. Even when variation points are changed or added, etc, PLA can still be evaluated.
6. Conclusions and future work

It is important to assess software architecture in the development of quality software products. Evaluating architecture is a big activity for product line development because it reuses of the software architecture. This research has focused on current evaluation methods, and especially concentrates on ATAM for architecture evaluation and analyzed the characteristics of product line architecture.

In this paper, we proposed a method to analyze the specific requirements and quality attributes of product line architecture assessment and extend ATAM using them. Then, we apply extended ATAM to the appliance domain. This paper focuses on steps 4, 5, 6 of the first phase, though all steps of both phases require some modification. Extended ATAM allows evaluation of the architecture of an entire product line, rather than single product.

We applied EATAM to the electrical appliances field. We expect that this evaluation method will enhance not only the efficiency of the process but ultimately, the performance of the products themselves.

Figure 7 shows the 9 steps of the EATAM and presents how steps contribute to the outputs that the EATAM delivers after an evaluation. "***" means that this step is a primary contributor to the output; "**" means that the step is a secondary contributor.

Future work will delve more deeply into further modification of the entire process. More future work will extend CBAM (Cost Benefic Analysis Method) for architecture design in product line engineering.

7. References


[7] Hassan Gomaa, Foreword by Ivar Jacobson, “Designing Software Product Lines With UML From use Cases To Pattern-Based Software Architectures”


[10] Analysis of a software product line architecture : an experience report
