An Approach for Resource Version Control for Evolutionary Software Product Line

1Deokyoon Ko, 2*Suntae Kim, 1Sooyong Park

1Dept. of Computer Science & Engineering, Sogang University, Seoul, South Korea
{maniara, sypark}@sogang.ac.kr
2Dept. of Computer Engineering, Kangwon National University, Samcheok Si, South Korea,
stkim@kangwon.ac.kr

1. Introduction

Software Product line(SPL) is aimed at maximizing reusability by classifying commonality and variability through software product family[1]. In order to establish SPL, many participants including domain experts and software developers should be involved, and they create diverse resources. As these artifacts are concurrently modified by several participants throughout the project, it causes artifacts to be inconsistent and unmaintainable. It is mainly attributed to the lack of systematic approach for version controls for the product line resources in consideration with SPL and its product evolutions.

There have been some work for managing evolutionary SPL (see, [3][4]). Mitschke et al. presented an approach to resource versioning in SPL, and how to propagate resource changes into related resources[3]. However, they did not handle issues in evolutionary perspective. While Gurp and C. Prehofer suggested how to manage versions of SPL according to changes of products derived from SPL[4], they did not tackle diverse resources in developing SPL and its products.

This paper presented an approach to maintaining versions of resources in SPL. It starts with defining SPL resources including a feature model, domain model, source code and so forth. Then, we propose version identifiers to cover SPL development and maintenance phase. Particularly, resource versioning in the maintenance phase is further discussed by detailed scenarios covering SPL and product evolutions with keeping consistency between SPL and its derived products.

2. Background

In this section, we introduce general steps and their artifacts in order to establish SPL. We proposed the SPL development process in our previous study[2] as shown in Fig. 1. It begins with feature modeling for analyzing common and variable features, and then analyzes domain and design elements to realize the features with UML. All design model classes are realized into classes called feature code which is the code
segment that implements one or more features. It is surrounded by the pair of the FEATURE_CODE and END_FEATURE_CODE comments. Also the comments includes implementing feature list. A developer should define Feature traceability (FT) for maintaining a traceability between features and elements of each artifact, enabling one to instantiate products by commenting the feature code out according to the feature configuration.

Figure 1. SPL Development Process and Artifacts

3. Resource version control for evolutionary SPL

In this section, we propose a version identifier of each resource for SPL. In SPL, we argue that the key considerations for versioning resources are version of SPL, versions of software products derived from the SPL, and features of the SPL and software products. The version identifiers for SPL, products and its features are summarized in Table 1. As software products are derived from a SPL by feature configuration, product version identifiers contains its base SPL version identifier. According to the product version, versioning of product features includes that of product line features.

Table 2. Version Identifiers for Resource of SPL and Products

<table>
<thead>
<tr>
<th>Versioning Resource</th>
<th>Version Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPL(S)</td>
<td>S Name</td>
</tr>
<tr>
<td>Product(P)</td>
<td>S Name</td>
</tr>
<tr>
<td>SPL Feature(SF)</td>
<td>S Name</td>
</tr>
<tr>
<td>Product Feature(PF)</td>
<td>S Name</td>
</tr>
</tbody>
</table>

Fig. 2 shows examples of resource versions suggested in Table 1. Due to the space limit, it only shows a feature model, design model and source code for SPL. In
versioning resources, a feature in a feature model is considered as the first class. All resources that have a feature traceability with a specific feature conform the version of the feature. This is because that a feature plays a key role for product configuration. Based on the suggested versioning approach, we discuss its feasibility with the following three scenarios in developing and maintaining SPL as well as software products derived from the SPL.

Scenario 1: Versioning in Developing SPL

For versioning resources, two principles should be kept throughout SPL development. First, any change of resources having a feature traceability with a feature causes to increase version of the feature. Second, increasing version of a leaf node feature leads increasing versions of parent node features, so that it eventually make to increase the product line version. Fig. 3 exemplifies the two cases. When updating related resources with feature Child 2, it makes its version increase from PL|1.0-C2|1.0 to PL|1.0-C2|1.1 (see the dotted line in the figure). This version increase leads increase of Root feature version and product line feature into PL|1.1-R|1.1. Increasing version of product line is propagated into all features’ product line version.

![Figure 2. Tagging feature code into resources](image)

![Figure 3. Evolution of feature](image)

Scenario 2: Versioning in SPL Evolution

After developing SPL, SPL instantiates diverse software products according to the feature configuration, which is a starting point of SPL maintenance. In SPL...
maintenance, SPL and product evolutions are commonly occurred by adding new features or updating existing features on demand. In this situation, resource versions play a central role to control system configuration management so that it can reduce rework and increase productivity. Fig. 4a shows a case of SPL evolution case, and its resource versioning. Initially, we assume that SPL $A|1.0$ instantiated Product P1 which version is $A|1.0(P1|1.0)$ (see, ①). After evolving SPL with $A|1.1$, the SPL instantiates the product P1 again so that the product version becomes $A|1.1 (P1|1.0)$ (see, ③). At this time, product version $(P1|1.0)$ starts with the initial version 1.0 because it is derived from a newer version of SPL, and it causes to ignore previous product specific changes. When a product maintainer needs to apply the previous product specific changes, he/she should update the newly instantiated product resources.

**Scenario 3: Versioning in Product Evolution**

Product evolution is typically occurred by localization which are not accepted from SPL maintenance boards. Fig. 4b shows how to maintain resource version in the case of product evolution. Modifying resources in software products causes to increase product specific version. In the figure, a version $Product P1$ has been increased into $A|1.0 (P1|1.1)$ after updating $product P1$. Once the product are updated in their scope, product maintainers suggests product line change board control to decide if the change should be applied into the SPL. If they accepts the product specific changes into SPL, it makes SPL to increase their version.

![Figure 4. Versioning in SPL and Product Evolution](image)

**4 Conclusion**

This paper has been presented an approach to maintaining resource versions of software product line in evolutionary perspective. We first presented version identity for SPL and principles of version increase of related resources. In addition, we showed three scenarios to manipulate resource versions. As future work, we are planning to tool support to enable resource versioning in evolving software product line as well as software product.
An Approach for Resource Version Control for Evolutionary Software Product Line

References


* Corresponding Author: Suntae Kim, Ph.D.

Acknowledgements.
This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology. (2012M3C4A7033348)