Extractive Approach for Transit of Legacy Products to Software Product Line with Organizational, Process and Technical Factors

Seonghye Yoon, Soojin Park, Sooyong Park[†1]

Abstract: Extractive approach can be applied to reengineer legacy products built within project based or customized product development model into product line. Most of the existing research related to extractive approach either analyzes commonality and variability or focuses on architecture reengineering. However, when moving to a different development model, technical factors as well as related process and organizational factors must be considered in order to minimize conflict within the organization and prevent one time, event reengineering. This study proposes a model which considers all three factors and provides a sustainable transit to product line. This model suggests that feature model be verified based on analysis of code change logs and bug issues and conduct appropriate domain engineering scoping by taking organizational factors into consideration.

Keywords: product line, extractive approach, organizational factors, process factors, technical factors

1. Introduction

When legacy products built within project-based or customized product development model passes through the stable stage and enters the evolution stage, the product's flexibility and variability becomes taken into consideration. This is because despite the many commonalities of the products, the various versions cause the maintenance cost to go up. When products built using project-based development model produce the next product from one product, ad-hoc reuse is used if commonality is assessed to be high. From this point, co-management of each product ceases to occur. Also, products built using customized product development model have one product base and are manageable, whereas derived products have various product base and eventually unmanageable commonality arises among the derived products. These methods cause problems such as code duplication, code inconsistency, code being used in a context different from the one initially intended[1].

Software product line is a way to solve the various products and versions issues arising from mass customization of software which must satisfy such diverse customer needs[2]. This makes the common features within a family of product reusable by making them core assets and effectively manages inter-product variability[3], supporting time-to-market, providing flexibility, reducing cost for the planned changes[4]. However, compared to conventional, software product line has more burden of building early core asset and more providing flexibility for unplanned changes[5], thus when domain is immature or when early cost investment is difficulty, other ways are chosen. Converting legacy products built in this way to product line requires extractive approach[6]. Existing relevant research analyzes commonality and variability using only technical factors and suggests methods of reengineering the architecture or design based on that analysis[7][8]. However reengineering only considering the technical factors may fail due to decreased effectiveness compared to investment caused conflict with existing organization or technology or end up being only one time reengineering[9].

[†1] Sogang University

To solve these problems, this study proposes a model which applies the extractive approach but decides domain engineering scope for the given situation by taking into account organization, process, and technical factors, and supports it so that it can be converted in a sustainable way. This model verifies the coverage of feature model with analysis results based on technical and process factors. Then organizational factors are reflected to determine the domain engineering scope appropriate for reengineering.

In Section 2, this paper discusses the methods and example of software configuration management in a family of products built with project-based or customized product development model. Section 3 suggests a general process model for building core assets of products mentioned in section 2 and detailed plan for each step. Section 4 includes the conclusion and topics for future research.

2. Background

This section takes a look at the form of evolution and maintenance for legacy products built within project-based or customized product development model, and introduces a management using subversion, a software configuration management tool.

2.1 Evolution and Maintenance of the Legacy products

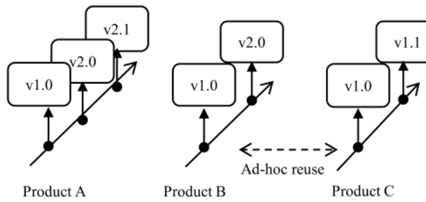


Figure 1 Products developed within the project-based development model [1]

Based on the development model, a way of evolution and maintenance differs for each product. Figure 1 shows the process in which a base-line is created and managed in ad-hoc reuse when a new product is created in a project-based development model. After Product A has been developed to v2.0, in order to develop a similar system, product B, code is copied and base-line is set using ad-hoc reuse, and code is modified for any other requirements not supported by product A.

Products developed within customized product development model are composed of product base which is the base-line of customizing and derived products which are customized from the product base. There is a tendency to ad-hoc reuse, similar to project-based development model, if the product base cannot flexibly support the customizing. Since all derived products include product base, it can be said that there are many commonalities, just as setting a base-line with ad-hoc reuse in project-based development model. Because commonality is already separated into product base, transit to product line is easy. Nevertheless, as derived products grow in number, additional analysis will be needed for commonalities occurring by chance.

2.2 Examples of Subversion Repository for the Legacy products

Figure 2 shows how products with base-line from the aforementioned ad-hoc reuse are managed with subversion, a software configuration management tool.

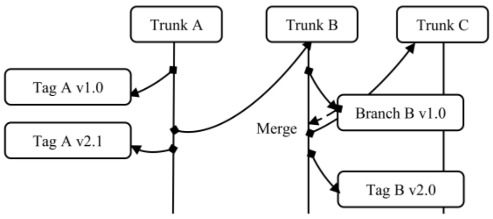


Figure 2 Configuration Management Products developed within the project-based development

Initially, Product A is managed with Trunk A. When Product B is created, branching may occur or Trunk B may be created. In case of branching, history management in comparison with Product A is possible, but because it is project-based, there is a weakness that responsibility cannot be divided clearly. For this reason, a separate trunk is often managed as in Figure 2.

Figure 3 shows the management of products developed within customized product development model. The locus of responsibility is stronger than in product base, thus a separate branch, rather than a separate trunk, is created and managed. A branch is created for each customer requiring customizing, and the said branch is managed until changes are merged to the product base. While branch is being maintained, if the product based is evolved, the incurred update cost is significant.

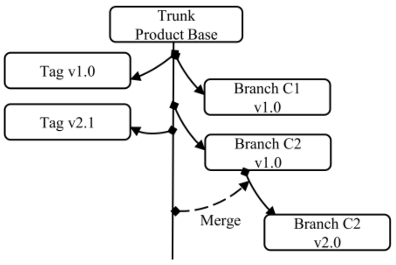


Figure 3 Configuration Management Products developed within the customized product development

Branch or trunk by ad-hoc reuse can trace features since they have the same base. Commonality or variability of features of each version can be determined through the feature change history. This plays an important role in extracting the feature during transit to product line.

3. Extractive Approach for Product Line

In order to reengineer legacy products with software configuration management similar to aforementioned products, process model based on extractive approach as in Figure 4 is suggested. Suggested model takes into consideration organizational, process and technical factors so that reengineering does not end as a one-time event and sustainable transit to product line takes place. Organizational factors relate to working conditions such as team building or domain knowledge. Process factors relate to development procedures such as code change or bugfix. Technical factors relate to software artifacts such as feature model, design model, or code.

First, legacy products or versions to be converted are selected. Products or versions of products are determined through analysis of currently actively derived versions or versions with continued maintenance plan. As certain versions are not maintained when products are evolved, if all versions are seen as target of analysis, then unnecessary features are analyzed and more than necessary efforts are used. Once the target of analysis is determined, reengineering to product line takes place through the following 6 steps.

3.1 Feature Extract

Feature model[10] is created based on experience of developing legacy products and needs analysis on the current market. If there is a domain export or sufficient domain knowledge, it is better to conduct the market analysis first. If that is not the case, however, then it is recommended that the legacy products analysis be conducted first.

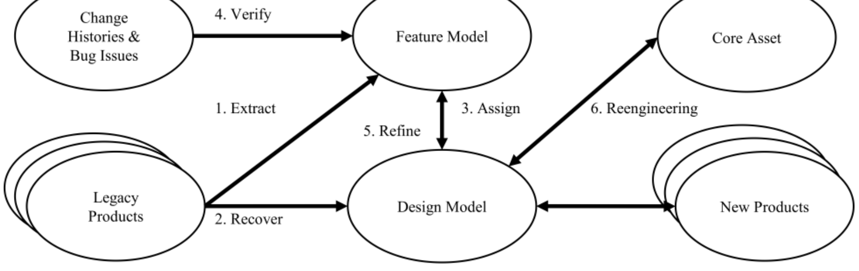


Figure 4 Extractive Approach for Sustainable Product Line

3.2 Design Model Recovery

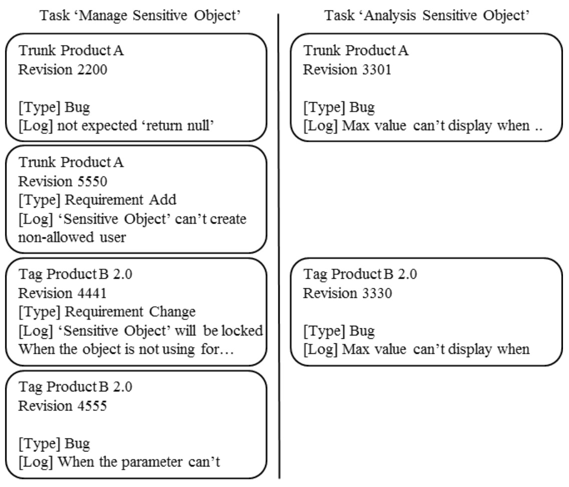
In the case that ad-hoc reuse is used as base-line or there is same product base, the products' design models are similar. Recovery is done by product, and the differences are merged and recovered as one design model.

3.3 Feature and Design Assignment

Created feature model and the relationship in elements of design model are linked. Their relationship is very important as the relation key in analyzing the coverage of the legacy products features of the feature model and the change history of each feature in the subsequent steps. Update will be continuously while passing through each step.

3.4 Feature Model Verification with Process and Technical Factors

This stage verified whether the feature model can cover all of the features of the legacy products and whether the commonality and variability were properly analyzed.



```
Task 'Manage Sensitive Object'

Trunk Product A
Revision 2200

[Type] Bug
[Log] not expected 'return null'

Trunk Product A
Revision 5550
[Type] Requirement Add
[Log] 'Sensitive Object' can't create
non-allowed user

Tag Product B 2.0
Revision 4441
[Type] Requirement Change
[Log] 'Sensitive Object' will be locked
When the object is not using for…

Tag Product B 2.0
Revision 4555

[Type] Bug
[Log] When the parameter can't

Task 'Analysis Sensitive Object'

Trunk Product A
Revision 3301

[Type] Bug
[Log] Max value can't display when ..

Tag Product B 2.0
Revision 3330

[Type] Bug
[Log] Max value can't display when
```

Figure 5 Change history using configuration management and bug tracking system

Figure 5 is an example of a change history created from recomposition of database management products's configuration management system log and bug tracking system issues for verification. Feature is composed of several tasks which implement the feature. A task is a transaction composed of a path from when a call starts to when it ends. When analyzing change log, tasks are distinguished from the combination of upper path which calls the modified function and a lower path which the modified function calls. The type of change is compared to related issue of bug tracking system and classified as *Bug/ Refactoring/ Requirement Add/ Requirement Change/ Requirement Delete*. *Bug* and *Refactoring* type are not included in analysis at this step since they are not factors which influence change in features. Commonality and variability are analyzed and recommended based on the *Requirement Add, Requirement Change* or *Requirement Delete* type that occur for each product and version. Feature model is verified and refined by referring to the recommended commonality and variability.

Figure 6 is an example which refines the feature model using the change history in Figure 5. Based on the change history of *Requirement Add* type occurring in Product A and change history of *Requirement Change* type occurring in Product B, it was recommended that 'Access control' is being provided for Product A and 'Lock' for Product B. Based on this, feature model is refined. As change history analysis is based on changes in code, there is a limit in feature abstraction. Therefore, feature can be recommended by it is difficult to determine the feature's abstraction level and provide automatic refine.

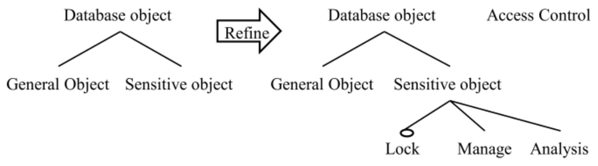


Figure 6 Example of Refining Feature Model

3.5 Design Model Refinement with Organizational Factors

Design models of legacy products must be refined into design models for applying core assets. For this, the scope of core asset must be set. The main factors for determining the threshold of core asset scoping are organizational factors, and threshold decides whether each task is included in the core asset depending on its stability and changeability. Table 1 describes the items which must be taken into consideration for stability and changeability in order to be a core asset. Low stability signifies high probability of a bug, so stability should be raised through review or inspection. High changeability signifies that there are still frequent changes occurring in the feature, meaning that additional analysis of the features is necessary. High stability and low changeability signifies a good candidate for reuse and transit to core asset is easy.

Table 1 Things to Consider for Stability and Changeability for Core Assets Candidates

Stability	Changeability	Things to Consider
High	Low	Good to reuse
High	High	Analyze feature
Low	High	Review, Analyze feature
Low	Low	Review

When stability and changeability is calculated for each feature, core asset is scoped and reflected in the design model based on factors changing from other development model to product line such as the team's work distribution in terms of organization, organization's management skills, and presence of domain expert, team's level of product line engineering knowledge.

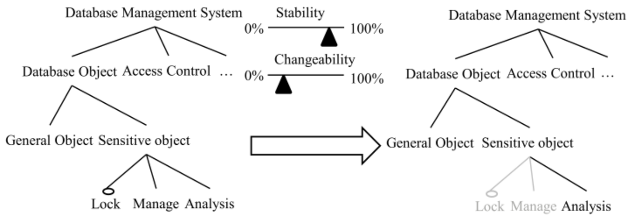


Figure 7 Example of Core Asset Scoping

Figure 7 is an example of situation where members of domain engineering team is not sufficient and there is not enough time to reflect the structure in the next release so core asset is built on to the level where stability is guaranteed and features are clear. For example, if 'Lock' feature has high changeability and low stability, and 'Manage' feature has low stability, then appropriate factors should be put on hold in the current organizational factors until the next reengineering period. Finally, features can be core assets excepted gray features considering the current organizational factors.

3.6 Reengineering to Product Line Assets

Features which belong to the core assets in 3.5 are reengineered, made into core assets, and reflected in products.

When the products built through reengineering enter the stable stage, new core asset scope is determined and reengineering takes place applying the same process to the application engineering.

4. Conclusion

As legacy products are maintained and evolved, the versions which must be managed become various and accordingly, many problems occur. To solve these problems, extractive approach is used to transit to product line. Existing research focuses on transit to product line considering technical factors. However, in order to increase the success rate of reengineering, organizational and process factors must be taken into account. To that end, stability and changeability of core asset candidates must be measured with process and technical factors. Scoping of core asset was adjusted with the level of accommodation through organizational factors. This alleviates the rejection of suddenly changing development model while making sustainable reengineering to product line possible.

This study provides a summary process. In the future, research will be conducted on extraction of commonality and variability through change history analysis, measurement of stability and changeability, and adjusting of threshold.

Reference

1) W. Codenie, N. González-Deleito, J. Deleu, V. Blagojević, P. Kuvaja and J. Similä, "Managing Flexibility and Variability: a Road to Competitive Advantage," in *Applied Software Product Line Engineering*. CRC Press, 2010.

2) C. Krueger, "Easing the transition to software mass customization," *Proceedings of the 4th Workshop on Software Product-Family Engineering*, Springer, 2002.

3) D. L. Parnas, "On the design and development of program families," *IEEE Transaction Software Eng*, vol. 2, 1976.

4) K. Schmid and M. Verlage, "The economic impact of product line adoption and evolution," *IEEE Software*, vol. 19, 2002.

5) K. Känsälä, "Good-Enough Software Process in Nokia," *Lecture Notes in Computer Science*, vol. 3009, Springer, 2004.

6) P.C. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*, SEI Series in Software Engineering, Addison–Wesley, 2001.

7) H. Lee, H. Choi, K. C. Kang, D. Kim, and Z. Lee. "Experience report on using a domain model-based extractive approach to software product line asset development," *ICSR '09: Proc. of the 11th Intl. Conf. on Software Reuse*, Springer-Verlag, 2009.

8) V. Alves, P. Matos Jr., L. Cole, P. Borba, G. Ramalho, "Extracting and Evolving Mobile Games Product Lines," *Proceedings of Ninth International Software Product Line Conference*, 2005.

9) J. Bergey, D. Smith, S. Tilley, N. Weiderman and S. Woods, "Why Reengineering Projects Fail," *Technical Report CMU/SEI-99-TR-010*, SEI/CMU, 1999.

10) K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak and A. S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," *Technical Report CMU/SEI-90-TR-21*, SEI/CMU, 1990.

Acknowledgments This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2012-(C1090-1231-0008)).