Variability Integration at Requirements and Architecture Level in Software Product Line Engineering

Shahliza Abd Halim
Software Engineering Department
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
81310 Skudai, Johor.
shahliza@utm.my

Dayang Norhayati Abang Jawawi
Software Engineering Department
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
81310 Skudai, Johor.
dayang@utm.my

Safaai Deris
Software Engineering Department
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
81310 Skudai, Johor.
safaai@utm.my

ABSTRACT
Software reuse promised higher productivity and quality and reducing product delivery time. Nevertheless, this advantage could not be reaped without systematic software reuse. One of the significant approaches for systematic software reuse is Software Product Line (SPL). SPL considers related products in a domain in terms of its commonality (known as core asset), its differences (known as variability) and its derivation in order to build specific product in the product line. With first class consideration of variability, it enables effective reuse of artefacts thus assisting in an easier derivation of individual members in SPL compared to developing similar products as single system development. However the identification of variability at the architectural level requires the understanding of variability at the requirements level. Therefore, the integration and relations between variability in requirement and architecture level of the core asset must be explicitly defined for stakeholders to understand how designers realized product line variability and also for product derivation. This paper reports ongoing research in designing and implementing variability representation between different abstraction levels in SPL.

General Terms
Documentation, Design

Keywords
Requirements, architecture, software product lines, variability management, core assets

1. INTRODUCTION
In Software Product Line (SPL) reuse occurs with the use of core assets or reusable artefacts [1-3]. With core assets, overlaps among members of the family can be leverage by merging common parts of the core asset and at the same time by managing its variabilities. SPL has two main development processes, Domain Engineering and Application Engineering. Domain engineering involves creating and maintaining core assets (e.g., a product family architecture and a set of components). During application engineering, these reusable artifacts are used to build the products. In both phases, variability management plays an important role. During the first phase, domain engineers introduce all kinds of variability mechanisms (e.g., parameterization or abstraction) in the software artifacts, so that application engineers can make specific choices (extension, change, customization, or configuration) that suit the required functionality and quality of the product they are creating.

Reuse can be done by adapting this core asset to meet the changing needs of customers. Thus, the building of the most important core asset, the PL Architecture [1, 3] requires the understanding of variant requirements and precisely describing them [4, 5]. As a result, identification of variation points at architecture level requires the understanding of variability at the requirement level. Therefore, the integration and relations between variability in requirement and architecture level of the core assets must be explicitly defined for stakeholders to understand how designers realized product line variability and also for utilizing the variability (product derivation). In order to achieve this, we propose core assets which comprise of models supporting the front-end of SPL development – at requirements and architecture level. The models to represent the requirements level are the use case, feature model and natural language. At architectural level, component and connector model is used as a representation. These models will be extended in order to represent variability which is the most important features in SPL. Furthermore, traceability among the models will be determined to enable the derivation of product specific application at the application engineering process.

This paper highlights the ongoing research in order to relate variability at requirements to architecture level that concern core assets development approaches in SPL and also its reuse during application engineering. The remainder of this paper is organized as follows: In Section 2, the paper discusses briefly the state of the art of this research field and further highlights the objectives of this research. Section 3 emphasize on the methodology of this research. Section 4 discusses the
current work and preliminary results of this research Section 6 generally discuss the significance and novelty of this research. Lastly, section 7 discusses on the conclusion and recommendation for future work.

2. RESEARCH MOTIVATION AND OBJECTIVES

Software requirements express the needs and constraints placed on a software product which contributes to the solution of real-world problems [6]. Software architecture can be defined as concerning with the study of the structure of software including its topology, properties, constituent components and their relationships and patterns of combination.[7]. Software architecture has a more structured representation which constitutes module, components or even functions as compared to software requirements which are usually represented in a natural language form. In this research we focus on the variability integration between requirements and architectural level referring to the linking and mapping between variabilities at these two abstraction levels. Variability integration objective is to integrate variation points from requirements to architecture in order to have an efficient traceability from different abstraction level and also to leverage product specific derivation process in SPL.

Ability to understand requirements for SPL lies on the use of natural language for a comprehensible requirements statement and also the use of graphical models to give more syntax and semantic meaning to the requirements statement. There are several approaches which emphasize on natural language as a representation of requirements such as FORE, Variability Tags, MRAM/TRAM, Central Decision Model, Requirements Recycling, Feature Modelling and PLUSS [8]. However, the specification of requirements only in natural language is not adequate to obtain a complete and consistent description of requirements

Basic feature model is used to represent requirements for SPL graphically. Albeit the popular usage of feature model by various researches in SPL, feature model reportedly does not properly represent the semantic in requirements as reported in [9-12]. Furthermore, commonality and variability could not be objectively identified in feature model as it relies on developer’s intuition and domain expert’s experience for common and variable requirements determination.

In order to complement the use of feature model to represent requirements, there are methods which propose use case model to be used with feature model. Among the methods which use both of these models to represent variability at requirements level are FeatureRSEB[13], Features and Use Case for Embedded Systems [14], PLUSS [15] and Automated mappings between Use case diagrams and feature models [16] Amid the various approaches done by the researchers to use feature models and use case diagram together, there are few attempts from the researchers in order to map the requirements model to architectural model, specifically to component and connector diagram.

The use of component and connector diagram as a representation of architecture has been done by several researchers which address different levels of abstraction in SPL development. A recent account of such work is given by extraction of component and connector view from detailed UML diagrams [17], using component and connector architectures for developing object oriented design [18] and using incorporating variability in the component and connector view [19]. However based on our knowledge there are still no works which associate requirements model with component and connector model to date.

Apart from identifying the modelling elements to represent the core assets in Domain Engineering, there are also problems in deriving the product specific application from the product line in application engineering process. Deelstra, Sinnema and Bosch [20] has identified the problem as complexity to derive application specific product due to the very large numbers of variation points and interdependencies between them. Therefore, by modelling the integration and relations between variability in requirement and architecture level of the core assets, it helps the developers to explicitly define the choices available for deriving products from the core assets. Furthermore it helps the stakeholder to understand how designers realized product line variability and also helps in product derivation.

Based on the state of the art in integrating requirements to architecture, there are four important research questions that need to be answered by this research: How to represent variable requirements?; How the variable requirements can help to model variable architecture?; How to relate the variability in requirements to architecture?; and finally How does this relation helps in deriving product specific application from the product line?. From the research questions, we further identify four objectives to be fulfilled in our research:

i) To model generic representation for core assets at requirement to architecture level

ii) To create traceability to map between the representations

iii) To create technique for deriving architecture based on application specific requirements

iv) To verify and validate the generated artefacts.

3. RESEARCH METHODOLOGY

The main aim of this research is to propose a model driven engineering approach for the integration of variability at requirements and architectural level. The integration involves the traceability between different artefacts of these two abstraction levels which helps in the derivation of these artefacts at the application engineering level.

In this research we take a pragmatic viewpoint in which existing theories, methods, tools and techniques is combined to support our goal. Our research methodology
covers the development process for SPL known as domain engineering where core assets is built and application engineering where the derivation of specific product is done. Figure 3.1 shows the steps in our research which combine existing approaches and methods in order to fulfil our research objectives.

The details of the steps taken in order to integrate the variability between the different abstraction levels are as described in the following sections.

### 3.1 Domain Engineering Process

In domain engineering process, the identification of core assets involves the scoping and modelling of three models with variability information, use case model, feature model and component and connector model. Apart from scoping and modelling the models, traceability between the models have to be determined in order to enable the integration of the models together.

#### 3.1.1 Use case Modelling

Use case modelling is done with the help of product line scoping matrix which enable the creation of domain use case model. For the scoping method, we follow the work of [21] where matrix is used to determine the scope for SPL requirements. In order to do scoping of product line requirements, the applications related to the context of the product family is identified and the common and optional Primitive Requirements (PR) is determined based on [21].

The work in [21] also determines the domain use case model based on the matrix determined. Scoping based on matrix would produce a domain use case model where the common and variable use case where the PR is associated with the suitable use case. The common and optional use case will be shown via \(<\text{extend}>\) and \(<\text{include}>\) relationships.

#### 3.1.2 Feature Based Modelling

In feature based modelling, we follow the work of MRAM/TRAM method. MRAM/TRAM method use pattern in order to transform textual requirements into feature diagram. [22, 23]. In MRAM, product line requirements in textual format is organized into a set of trees (forest) using the parent-child dependency relationships[23]. In order to transform the textual requirements into feature diagram, the textual requirements have to be connected to architecture, the textual requirements is transformed into feature diagram with the help of pattern.

#### 3.1.3 Component Connector Modelling

In component and connector modelling, Requirements Model is transform into Component and Connector Model in Architecture level. However the mechanism for the transformation is yet to be determined.

### 3.1.4 Traceability between Models

In this research, traceability is not about how to generate the traceability link but is more onto the structure of traceability model to enable the designer or architect to record the information for further use in the derivation product derivation. There are structures in the form of metamodel proposed by [9, 24, 25] for using \(<\text{trace}>\) for traceability between artefacts however its usage for the requirements and architectural artefacts used in this research is yet to be determined.

### 3.2 Application Engineering Process

Application engineering process involves two main activities, the derivation of the three artefacts in core assets and also the validation and verification of the three generated artefacts.

#### 3.2.1 Derivation

Based on [26], there are basically five approaches for deriving product specific application. Among the approaches are Specification Approaches, Intelligent Build, domain specific languages and product generation, metamodeling and Frame Technology. However in this research we concentrate on model driven engineering for generating artefacts, we will concentrate on domain specific languages and product generation in order to derive product specific application.

#### 3.2.2 Validation and Verification

When a different models are used and the models represent different abstraction levels, there are possibility of inconsistency in the models occurring and the verification of the generated artefacts have to be done in order to determine the generated artefacts are correctly generated.
4. CURRENT WORK

We have completed the overall overview of the generic representation for core assets at requirements to architectural level as in [27]. Current work of this research is on developing requirements model based on case study with Library System taken from the paper from [27, 28]. In the paper, the authors have categorized the library systems into City Library (CL), University Library (UL) and Research Library (RL). In this paper we have further added two more types of libraries, Children (CL) and School Library (SL) to include more variations of the library systems.

4.1 Use case Modelling

We start with determining the common (C) and variable (P), Primitive Requirements (PR) based on [21] as shown in Table 4.1. Ratio and threshold to determine which PR are common or which PR are optional (P) are calculated based on [21]. Table 4.1 shows the common and variable PR.

<table>
<thead>
<tr>
<th>No</th>
<th>PR</th>
<th>CV</th>
<th>Context</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CL</td>
<td>UL</td>
</tr>
<tr>
<td>PR1</td>
<td>Loan</td>
<td>C</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PR2</td>
<td>Return</td>
<td>C</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

PRs are further classified into suitable domain use cases using PR-use case matrix as shown in Table 4.2.

<table>
<thead>
<tr>
<th>No</th>
<th>PR</th>
<th>CV</th>
<th>Use case name and CV Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Loan Material (C)</td>
</tr>
<tr>
<td>PR1</td>
<td>Loan</td>
<td>C</td>
<td>✔</td>
</tr>
<tr>
<td>PR2</td>
<td>Return</td>
<td>C</td>
<td>✔</td>
</tr>
<tr>
<td>PR3</td>
<td>Report Loss</td>
<td>C</td>
<td>✔</td>
</tr>
<tr>
<td>PR4</td>
<td>Overdue Control</td>
<td>C</td>
<td>✔</td>
</tr>
<tr>
<td>PR5</td>
<td>Item Reserve</td>
<td>P</td>
<td>✔</td>
</tr>
<tr>
<td>PR6</td>
<td>Item Suggestion</td>
<td>P</td>
<td>✔</td>
</tr>
</tbody>
</table>

4.2 Feature based Modelling

In Table 4.3, we associate textual requirements with a suitable use case.

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Related PR</th>
<th>CV Property</th>
<th>Requirements Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loan Material</td>
<td>PR1, PR2, PR3, PR4</td>
<td>C</td>
<td>PR1.1 There shall be facility to add loan books PR2.1 Users are allowed to loan @ number of books. PR2.2 There shall be facility to check late returned book. PR3.1 There shall be facility to report loss. PR4.1 There shall be facility to calculate the amount to be compensated.</td>
</tr>
<tr>
<td>Item Reserve</td>
<td>PR5, PR6</td>
<td>P</td>
<td>PR5.1 Items can be reserved via e-mails PR5.2 Items can be reserved via on-line systems PR5.3 Items can be reserved at due dates.</td>
</tr>
</tbody>
</table>
5. RESULTS AND DISCUSSION

From the proposed process of this research as noted in Figure 3.1, the expected contribution of this work is the integration of variability between requirements to architecture based on the natural language and graphical models used to represent the different abstraction levels. The integration covers the traceability between the three models to enable the derivation of the models in application engineering phase. From our knowledge, existing work do not use the combination of the three models identified in this research for integrating requirements and architecture. These three models can also be used for representing different viewpoints of stakeholders at requirements and architectural level.

For our current contribution based on our present work, we transform the use case diagram identified through the scoping process into feature diagram to enable stakeholder to choose the required features more clearly. The traceability between the use case, PR and textual requirements in natural language has been done. As a future work, we have to determine the traceability between both requirements model. Another work has to be done is the transformation between the requirements model and architectural model and the traceability between the models which represent different abstraction level.

6. CONCLUSION AND FUTURE WORK

In this paper we have presented the progress of our PhD work. Our goal is to provide a method that effectively enables the core assets identified in domain engineering to be derived in application engineering. Core assets identified during domain engineering involve two important abstraction level in SPL, the requirements and architectural level. In the near future, we plan on integrating the approaches and methods in an experimental tool that can be used in case studies of SPL.

ACKNOWLEDGMENTS

Our profound appreciation goes to Ministry of Higher Education (MOHE) and Universiti Teknologi Malaysia (UTM) for their financial support in this study.

7. REFERENCES


