ABSTRACT
Software Product Line (SPL) is an effective approach in software reuse where core assets can be shared among the members of the product line with an explicit treatment of variability. However, variability can be done by adapting this core asset to meet the changing needs of customer thus further highlights the needs to integrate requirements to architecture. We have proposed an evaluation framework based on issues inherent in SPL and integrating requirement to architecture. Using this evaluation framework, we evaluate six core assets development approaches in a systematic and consistent manner and reflect on current investigations and open issues that provide foundation for further research in this area. This paper presents an investigation and highlights on the desired criteria to provide better means to integrate the requirement to architecture. The evaluation results shows there are still room for improvements in the existing approaches towards a better integration between requirements to architecture.

General Terms
Documentation, Design

Keywords
Requirement, architecture, software product lines, variability management

1. INTRODUCTION
In Software Product Line (SPL) reuse occur with the use of core assets [1-3]. With core assets, overlaps among members of the family can be leverage by merging common parts core asset and at the same time managing its variabilities. With variability, the same core asset in the product line can be reused by members of the product line. 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 requirement and precisely describing them [4, 5].

However, transitions of variability between higher level of abstraction to realization level still remain vague due to the difficulty in relating variability at the requirement (analysis) and realization (implementation) level [6-8]. Furthermore, the transformation between requirement specifications to software architecture is a creative task that is not much supported by the current software development [9-11]. Therefore, it is compulsory to define, identify and represent the variations systematically and explicitly at requirement level [12, 13].

Systematic variability management research challenge can be organized into three main areas: Designing variability; Using Variability and Evolving variability [14]. In this research our focus is on the second research area where the use of variability designed in an earlier phase of the lifecycle and where the management of variability and variation points remains a major challenge due to the numerous feature interactions and variation points to represent the variability in different level of abstractions in software development[15].

This paper highlights the issues in requirement to architecture concern in core asset development approaches in SPL and evaluates six core assets development approaches with proposed evaluation framework criteria. The remainder of this paper is organized as follows: In section 2, this paper discusses on the systematic variability management in Software Product Line (SPL). Section 3 discusses the fundamental issues in both SPL and requirement to architecture concern. Section 4 discusses related works on core asset SPL development approach and its relation with requirement to architecture concern. In section 5, criteria for evaluation framework is proposed and is used to evaluate related works. Section 6 discuss on the evaluation. Lastly, section 7 describe the conclusion and future and work.

2. FUNDAMENTAL ISSUES FOR SPL AND REQUIREMENT TO ARCHITECTURE CONCERN
In this paper, we highlighted three important issues that are central in SPL and also in the requirement to architecture consideration. The three issues are:-

2.1 Variability as a First Class Representation
In SPL development, variability must be considered explicitly as a first class representation [7, 16] and the
most explicit representation for variability is via meta-
model [17]. Variability representation in the form of
meta-model is needed in order to provide a unifying
framework for multiple modeling views with different
notations. It also contains the product line meta-classes
and its relationships to enable consistency checking and
essential for tool support as it represents a schema for a
product line repository, which stores the artifacts
developed as a result of product line engineering.

2.2 Systematic Variability Management
Due to the creative task in the transformation between
requirement specification to software architecture, the
quality in architecture and design task heavily dependent
on the skills and cognitive capabilities of developers [9,
11, 18]. Therefore, building architecture which satisfies
the software requirements is an ad hoc, informal and
unsystematic process. Systematic variability
management must be incorporated to integrate
requirement to architecture in a well defined repeatable
process. Figure 1 shows the integration of process from
requirement to architecture where the ellipse shows the
process and the rectangle shows the artifacts produce by
each processes.

2.3 Integration of Requirement to
Architecture
Requirement and architecture could not be separated as
highlighted in [18, 19]. The integration is further
highlighted in [19] where software architecture is
defined as containing stakeholders-need statements and
also rationale that the architecture implementation fulfill
the needs of the stakeholder.

3. RELATED WORKS
Related works on transition from requirement to
architecture has been done in [9, 20]. The authors in [9]
compares, classifies and evaluates the suitability of
Architecture Description Languages (ADL), Goal-Based
Approach, Problem Frames, Use Case Maps, Model
Bridging, Rule Based Decision Making, Architecting
Requirements, Object Oriented Transition and Weaving
Requirements and Architecture Processes for
transitioning against sixteen evaluation criteria’s.

However, all the approaches are applied in single system
development only and the discussion is not fully
revolving around the evaluation criteria. Nevertheless we
have use two suggestions by the authors for our
evaluation criteria:

i) To study on multiple viewpoints in
covering partial requirements for
architecture transformation.

ii) To examine the mapping of requirement
properties and architectural structures for
the transition.

Other than [9], there are evaluations done for SPL as in
[20]. SPL development approaches such as FeatuRSEB,
KembrA and Functionality-based Architectural Design
(FAD). The evaluation is based on derivation of
architectural components from requirement and also
on mapping mechanism from the different phases. The
evaluation criteria proposed in this paper is basically
related with the issues highlighted in Section 2.

3. EVALUATION FRAMEWORK
FOR VARIABILITY MANAGEMENT
CONSIDERATION
Our evaluation criteria are chosen based on the
three issues highlighted from Section 2. Table 3.1
shows the issues and related criterion.

<table>
<thead>
<tr>
<th>Table 3.1 Evaluation framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criterion</td>
</tr>
<tr>
<td>Variability Representation</td>
</tr>
<tr>
<td>Formal if the representation is in the form of metamodel, Semi-Formal if representation is in modeling language such as UML and Arbitrary if there are no standard modeling language.</td>
</tr>
<tr>
<td>Systematic variability management</td>
</tr>
<tr>
<td>Integration of requirement to architecture</td>
</tr>
<tr>
<td>Requirement Viewpoints</td>
</tr>
</tbody>
</table>
5. EVALUATION ON CORE ASSETS DEVELOPMENT APPROACHES
We have chosen six core assets development approaches where these approaches concentrate on developing domain requirement and domain architecture as core assets. We have chosen the Product Line UML Based Software Engineering Environment (PLUSEE) from the work of [17, 23], Frame Based from the work of [4, 24-26], Domain Requirement Asset Manager (DREAM) from the work of [13, 27, 28], Feature Oriented Reuse Method (FORM) from the work of [29, 30], the work from Savolainen [7, 31] and Drama from the work of [32, 33]. The following subsections show the evaluation of core asset development approaches for each criterion:

5.1 Variability Representation
Among all the approaches, only PLUSEE and DREAM have supported their approaches with an explicit metamodel representation. PLUSEE encompasses a multiple-view meta-model to unify SPL representation requirement, analysis and architecture phase where each phase has its own view of metamodel. DREAM approach comprehensively use metamodel in Domain Requirement and Domain Architecture and also produce Traceability metamodel to relate between the two phases.

In Framed Based, there is no metamodel specified in this approach, however they propose an augmented structured system analysis and design graphical notation to model variant requirement. In other work, frame based method have been used for managing domain requirement with UML as well [26]. In Drama, there are also a model in the form of class diagram to show the relationship between goal, scenario and also variability.

For approaches such as Savolainen and FORM there are no specified metamodel for these approaches. Both of these approaches use arbitrary notation such as tree-like structure in Requirement Definition Hierarchy method by Savolainen and also feature model by FORM.

5.2 Variability management process
Dream is the only approach which has explicit variability management in its elicitation, specification and design process. In DREAM, the integration process of elicitation, specification and design is based on its matrix based technique in analyzing commonality and variability of the most atomic requirement (primitive requirement) in legacy systems [13]. The analysis is carried out in specification process in order to generate use case diagram and design process in order to develop component based domain architecture.

Other approaches such as PLUSEE, Frame Based, FORM, Savolainen and Drama have concentrated on either one or two processes in requirement to architecture integration. Drama concentrates on elicitation of logical components using Goal and Scenario based technique and also using quantitative analysis to construct domain architectures without any elaboration on specification process. PLUSEE concentrates on integration of requirement and analysis process however the design process is not elaborated. Framed Based approach concentrate on the specification of requirement and also developing frame based Interface Description Language to represent architecture specification with no elaboration on elicitation process. FORM does not support relationships between requirements and architecture explicitly [32]. Definition Hierarchy method in Savolainen approach concentrate on elicitation and analysis of requirement but do not elaborate on their proposed architectural assets.

5.3 Requirement to architecture dependency
An explicit requirement to architecture dependency has been specified in Dream, Savolainen and Drama approaches. Dream uses variability and commonality analysis matrix in order to derive architectural design. Trace relationships is used to map between domain requirement and domain architecture. Definition hierarchy method is used in Savolainen as a method to identify architectural drivers [7]. Rules are specified in natural language to map between requirement, feature and architectural assets. In Drama, goal and scenario based domain requirement analysis is used to identify basic architectural units. Mapping between requirements to architecture is based on the transformation of goal into logical component and the scenario which describes how the logical components work.

Approaches which have implicit dependency in requirement to architecture such as PLUSEE, Frame based and FORM do implement mapping but to certain phases only. PLUSEE use feature dependency for mapping between requirement and analysis views, but the mapping to architectural view is not elaborated. In Frame based approach dependency is at requirement level only and there is no mapping specified for requirements to architecture. FORM also has an implicit requirement to architecture dependency.

5.4 Requirement Viewpoints
Almost all of the approaches have requirement viewpoint except for Savolainen where this approach only concentrates on capturing, structuring, analysis and documentation of requirements. Approaches such as PLUSEE, Dream and Frame-based concentrate on
functional requirements where on the one hand PLUSEE and Dream, represent functional requirements with use case. On the other hand, Frame based approach represents functional requirements with structured system analysis and design modeling notation such as Data Flow Model and Entity Relationship Model.

FORM has a different technique to represent requirement viewpoints where the authors use features as the means for stakeholders to communicate their requirements. Thus, product features identified in the marketing and product plan is organized into feature model. Functional requirements are captured in a set of models such as use case model and an object model. Another different technique is by Drama where viewpoints related to requirements are the Abstraction View and Quality View. Abstraction View consists of top down and bottom up view. Bottom up view enables the identification of initial goal requirement and top down view validates the initial goal and refines it into sub-goals and scenario. In Quality View, functional requirements is mapped into quality attributes [33].

5.5 Architecture Viewpoints
PLUSEE, Framed Based, Form and Drama have both structural and architectural viewpoints in their approach. Though having the same viewpoints, they have different models in their views. PLUSEE represents static (structural) modeling view with a class model and represents dynamic (behavioral) view with collaboration and statechart model. Framed Based approach represents Static or structural modeling view with component and connector diagram and behavioral view is supported with state transition diagram. FORM defines its architecture in three different viewpoints (subsystem, process and module) and this can correspond to both structural and behavioral views in architecture. In Drama, Structure View represents system’s boundary and structural relationship between entities in a particular context while Function View captures the interactions between various components [33].

Other approach such as Dream supports only Structural View where component model describes domain architecture. In Savolainen, there is no viewpoint and also modeling notation to specify architectural concentration.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>PLUSEE</th>
<th>Framed Based</th>
<th>FORM</th>
<th>FODA</th>
<th>Savolainen</th>
<th>Drama</th>
</tr>
</thead>
<tbody>
<tr>
<td>Criterion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variability Rep.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Formal</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Semi Forma</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Arbitrary</td>
<td></td>
<td>/</td>
<td>/</td>
<td>/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variability Mgmt.</td>
<td></td>
<td>/</td>
<td>/</td>
<td>/</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1 Summary of Evaluation on Core Assets Development Approaches

6. RESULTS AND DISCUSSIONS
From an initial observation of the evaluation, it can be seen that formal representation for variability representation has been proposed by only a few approaches. The same goes to the explicit process for variability management and also for requirement to architecture dependency.

An explicit link between the requirement analysis/management discipline and the architectural development is essential in order to utilize the variation point that has been placed in software architecture [34]. In order to integrate or relate the variability in requirement to architecture some approaches use the same paradigm throughout the different abstraction level of requirement to architecture.

From the evaluation, most researchers concentrate on feature dependency for the link between requirement to architecture where Dream and Drama are the exception. Dream concentrates on the most atomic requirement while Drama concentrates on goal and scenario driven analysis to derive architectural components. Certain researcher use arbitrary representation in order to transfer the variability as in the approach of Savolainen but we foresee this as not as understandable as using UML notation language which have a standardize language for software developers.

However, majority of the approaches have multiple viewpoints in covering partial requirements for architecture transformation. Based on the evaluation framework, Dream fulfills most of the criteria of evaluation compared to other approaches. Nevertheless, its matrix based implementation in the same domain of e-business travel legacy system can be further evaluated on multiple domain SPL such as accounting system which is
applied in manufacturing, education and plantation domain.

7. CONCLUSION AND FUTURE WORK
The paper has compared six core asset development approaches for building domain requirements and domain architectures to a specially developed evaluation framework. The list of criteria is not exhaustive and it is based on three issues inherent in SPL development and also requirement to architecture consideration. The comparative evaluation framework helps researchers to identify the strength and weakness of current approach and consequently discover the opportunities of improvement to be addressed in the next proposed approach. Our future work is to propose a metamodel for the integration of product line requirements to product line architecture in a multiple domain SPL.

8. REFERENCES


