Abstract

Software product line has introduced a systematic reuse of work already carried out during software development within an organization. It gives an opportunity not to build the software from scratch. UML has almost become industry standard for visual modeling of software project. The current form of UML does not directly support the concept of software product line process there is a need to introduce some new or extensions in the existing diagrams of UML so that they can model the software product line engineering process. In this paper we have proposed a conceptual framework for visual modeling of software product lines, by introducing diagrams and symbols to develop the diagrams and support the process.

1. Introduction

The concept of software product line, proposed by SEI, is a comprehensive model for an organization building applications, which are based on common architectures and other core assets [1]. Clements [2] defines the term software product line as a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way. Van der Linden [3] pointed out that in 1995, the Architectural Reasoning for Embedded Systems (ARES) project began in Europe to provide architecture support for developing product families. In the overview of another European project, Engineering Software Architecture, Processes and Platforms for System-Families (ESAPS), a system family is defined as a group of systems sharing a common, managed set of features that satisfy core needs of a domain. The main objectives of system families are to reduce development efforts and to handle the impact of growing system complexity. Ommering [4] introduced another term called “product population”, which is a collection of related systems based on similar technology but with many differences among them.

Clements et al. [5] report that software product line engineering is a growing software engineering sub-discipline, and organizations including Philips®, Hewlett-Packard®, Nokia®, Raytheon®, and Cummins® are using it to achieve extraordinary gains in productivity, marketing time, and product quality. Many organizations that deal in wide areas of operation, including consumer electronics, telecommunications, avionics and information technology, are using software product lines because they effectively utilize software assets. Hein et al
[6] aims that the objective of software product line is to address the specific needs of a given business. Krueger [7] considered that the objective of a software product line is to reduce the overall engineering effort required to produce a collection of similar systems by capitalizing on the commonality among the systems and by formally managing the variation among the systems. A software product line gives an excellent opportunity to establish a production facility for software products based on common architecture. It provides a means for the reuse of assets, thus reducing development time and the cost of software products to capture various market segments.

2. Research Motivations

Software product line has emerged as an attractive phenomenon within organizations dealing with software development process. Practically there is no direct way of modeling product line activities visually for better understanding of development process. Most of the research carried out emphasis on making extension in UML. But software product line handles multiple products with multiple core assets involved. So only introducing some extension in UML will not enable us to completely provide visual modeling and version control management, because inherently UML handle single product at one time. A product consists of multiple components and composition of the products describes how and which components are used to develop it. Similarly a component can be used in multiple products within a software product line. There is a need to define a strategy to represent the software product line activity in terms of relationship among the components and the products. Every component should be traceable with reference to its utilization and version history.

Similarly every product should completely describe its composition with reference to components. This strategy should work at the software product line level not only at the individual component and product level. This will help everybody in the development and management levels to get access this information.

In this paper we have proposed a framework for visual modeling of software product lines of core assets as well as products within software product line which will enable us to store the vital information about components and products with a relationship of their composition and utilization.

3. Visual Modeling of Software Product Line

Visual modeling in software development process has almost become vital activity after the vast utilization of UML. For visual modeling it is required to give a symbolic representation to the different process and then present those symbols in a collective form with interaction each other to represent an activity. We proposed the visual modeling of software product line in to three major diagrams i.e., Core Asset Diagram and Product Development Diagram and Product Composition Diagram. Table 1 represents various symbolic representations proposed to different potential core assets. Although one cannot limit the types of core assets, so we have proposed that later on different other core assets can be added in to the repository of symbolic representation as and when required by an organization.
<table>
<thead>
<tr>
<th>Symbols</th>
<th>Software Product Line</th>
<th>Product</th>
<th>&lt;&lt;Part of&gt;&gt;</th>
<th>&lt;&lt;Uses&gt;&gt;</th>
</tr>
</thead>
</table>

Table 1: Symbolic Representations of Core Assets.

Table 2: Symbols representation
The symbols represent the following:

- **Software Architecture**: represents the architecture of the software.
- **Budget Diagram**: 
- **Software Component**: 
- **COTS**: represents the Commercial of the Shelf components present in the repository.
- **Documentation and Specification**: any written documentation and specification about software.
- **Domain Diagram**: illustrates the domain diagram includes for example like feature diagram.
- **Requirement Diagram**: visual representation of the requirements, like data flow diagram etc.
- **Schedule**: shows the schedule of the software development.
- **Use Case Diagram**: represents the functional characteristics of the software visually.
- **Test Cases**: stores information about the testing of software.
- **Test Plan**: presents the testing strategy and plan for software.
- **Work Plan**: highlights the work breakdown structure for software project.

The Table 2 describes the symbolic representation of software product line, product, relationship and association.

- **<<Part of>>**: The part of relation describes the composition property of component and products in software product line.
- **<<Uses>>**: the uses relation describes the association property of products and component in the software product line.

### 3.1 Core Asset Diagram

Core asset diagram represents visually the different core assets present in the core asset repository. It also shows the evolution of core assets with different versions developed during the product development by tailoring the components according to new modified requirements. It represents the product developed with association and composition of various core assets in the product. The **<<Part of>>** relation describes the composition and parent object identification whereas **<<Uses>>** relation describes the utilization of core asset in a product. Figure 1 describes a simple core asset diagram. Figure 1 shows the initial repository of software product line and a resulting product. Following facts are presented in the diagram,

- All the Core Assets directly connected with Software Product Line symbol with a solid arrow pointing from Software Product Line Symbol to them indicate that they belong to the initial repository of Core Assets.
- The Core Asset, which points towards a Core Asset with a solid arrow, indicates that a tailoring has been performed to produce a new version of Core Asset to fit into requirements of the product.
- The Core Assets pointing towards the product symbol with dashed arrow indicates they are being used to develop the product.
- The product pointing towards a Core Asset with a solid arrow indicates the addition of new Core Asset as a result of product development.

### 3.2 Product Development Diagram

The product development diagram represents the tree structure of software product line development activity. It represents how different products evolution is happened and we can trace the various stages of development of different versions of a product. Figure 2 shows a software product line development activity. Initially three products are resulted and later on various different versions of those products are resulted. The forward and backward path provides us important information about product development and evolution as follows:

- **Forward Path**: we can understand the development activity of software product line and the resulted products.
- **Backward Path**: we can trace the parent product of each product in order to get information about various versions of products.
3.3 Software Product Line Composition Diagram

The software product line composition diagram shows the product development activity from Core Assets repository. It illustrates the core assets present in the core asset repository with their version information. It represents the development of product by using various versions of the core assets. Following information can be retrieved from Figure 3:

- The Core Assets are stored on the basis of categories for easy retrieval the categories are labeled as Catalog-1, Catalog-2 etc.
- The components labeled as BW-1, BW-2, BW-3, SA-1, SA-2, SA-3, P1, SM-1, SM-2, SC-1 and SC-2 constitutes the initial core asset repository.
- The nodes labeled, as Product 1, Product 2, Product 3, Product 4 are the products developed.
- The composition of the Product-1 is BW-11, SM-11, P11, SC-11 and SA-1.
- The later versions of that component BW-1 are BW-11 and BW-12.
- The products, Product-1, Product-2, Product-3 and Product-4 are developed from the product line.

4. Conclusion

Software product line is gaining popularity over period of time due to its impact on cost and quality of software products. Visual modeling has been successfully in use by software practitioners to better understand the problem. In this work we have presented a conceptual framework for the visual modeling of software product line development. Three major diagrams core assets diagram, product development and product composition diagrams are presented in this work. We are currently working on a tool to provide further support to this conceptual work.

5. References