Chapter 4

QUALITY PRODUCT DERIVATION

This chapter presents the methodology followed for quality product derivation, which is done based on the weights of assets, relationships between the assets and by calculating the critical path. Two algorithms are proposed one is Quality Driven Product Derivation (QDPD) and the other Composition Algorithm (CA) for the above purpose. The best product based on requirements specified is finalized by taking results based on weights and also based on the critical path. A prototype is built to demonstrate the proof of concept. The application is tested with Dr. School product line. Results revealed that the proposed framework and the underlying algorithms are able to support configuration management and quality product derivation. Results of the application are evaluated with ground truth to establish the utility of the proposed framework.

4.1 PRODUCT DERIVATION

Product derivation for SPL plays a vital role in the success of product line. As customer needs change frequently, SPL is subject to changes accordingly. In fact, the customer needs are to be focused in real time. There is a dependency between assets of the product line. Changes made to assets have ripple effect propagated to other products. As customer needs are given paramount importance it is essential to anticipate and have required quality attributes to be considered depending on the requirements of the customer by considering change management strategies. SPL needs to consider variability that is the basis for having a product line [124]. Moreover, the presence of distributed technologies and ever growing service-oriented distributed applications in the real world force the SPL development to have refinements in its configuration management and quality product derivation. As the complexity grows in SPL there needs to have a framework that can cater to handle the growing complexity. In this context, derivation of a new product by using assets in SPL is a dynamic process. This process has to consider quality attributes besides the desired functionalities for a new product. When desired functionalities are known it is
possible to determine the needed artifacts. There might be a number of artifacts and their versions that can satisfy needs. However, it is important to have best possible product. For this reason, quality attributes are considered to filter out different product compositions in order to arrive at the best product. This chapter shows the empirical study made on the product derivation and the results that are obtained using weights of artifacts and the weights pertaining to relationships among them to have critical path while deriving new products. Critical path analysis is one of the important aspects that are involved in the determination of final product from a set of product compositions. The critical path analysis and usage of weights are incorporated in the proposed algorithms of this chapter. The following subsection throws light into the present state of the art of product derivation.

4.2 STATE-OF-THE-ART

This section provides current academic thinking on product derivation in software product line. Siegmund et al. [21] studied the product derivation process in SPL. They used non-functional requirements like quality and memory usage while deriving a new product. They tried to exploit non-functional requirements in order to optimize products. They used both non-functional properties and user-required properties for deriving a new product with high quality. There are different kinds of non-functional properties. They are known as inferred properties, directly assigned properties, and runtime properties. The directly assigned properties are the properties that can be treated as features either fully or partially. The inferred properties are properties that are measured as products that can help in inferring properties or they are measured in isolation. The third category of properties known as runtime properties are identified at runtime. The runtime properties are power consumption, memory and performance. Optimization of the product line is achieved by measuring runtime and inferred properties and mapping directly assigned properties to the corresponding product line model.

Guana and Correal [61] studied on quality-driven approach for software configuration management and model-driven strategy for deriving a new product. However, the new product derivation was based on the quality needs of customers
with respect to new products in the product line. While deriving new products, these researchers followed a selection strategy that could derive a product based on customer needs. Their approach in deriving product is a domain-centric approach which exhibited encouraging possibilities of choosing best artifacts that can meet the customer needs while new product is derived. In the same fashion, Deelstra et al. [4] focused on developing a framework for the purpose of product derivation in product lines. They understood that there was much work to be done with respect to product derivation methods as little was found in the literature. First of all, they studied issues involved in product derivation and then proposed a framework to overcome those issues in product lines while deriving new products. They tested their approach by closely working with two companies such as Thales Netherland B.V, and Robert Bosch GubH and deriving new products through their strategies.

Rabiser et al. [1] focused on the key activities involved in product derivation. Two tools such as Pro-PD and DOPLER proposed by them. The key activities involved include translating customer requirements, coverage analysis, creating product-specific requirements, implementing scope requirements, creating test cases that are product-specific, allocating requirements and create guidelines for decision makers. Tawhid and Petriu [51] explored the concept of model transformation in SPL through product model derivation. The product model derivation is the process of mapping between features present in the feature model to the actual implementations in the SPL.

4.3 PROPOSED METHODOLOGY

In the proposed framework, the SPL configuration management and product derivation are two indispensable parts; Figure 4.1 shows more details of them. The proposed framework is generic in nature and it can be applied to any kind of SPL. SPL configuration activity encompasses custom asset management, core asset management, products and SPLs. A single SPL contains a set of products and artifacts. Each product has core and custom assets associated with it. Assets and products are subjected to changes from time to time-based on the requirements of customers. These changes are to be tracked as part of SPL configuration management.
Towards this end, an algorithm is proposed. The algorithm computes weights for core and custom assets. Weights are also calculated for the relationships of assets in which they are involved.

When a new asset needs to be configured with different versions that show evolution, those different versions are assigned weights based on their possible importance in the product line. Both core and custom assets are reusable and configurable components in an SPL. This will facilitate to derive new products easily.

Figure 4.1: Details of SPL Configuration Management & Product Derivation
In Figure 4.1 it is evident that configuration of SPL and product derivation are illustrated. The configuration and version management is done for all products in the product line. They are also done for core assets and custom assets. More details on configuration management are found in Chapter 3. The product derivation has many activities involved. They include user inputs, weight computation, and critical path analysis, deriving best product, customer verification and feedback. The overview of all operations is described here. Later in this chapter algorithms are proposed to implement the proof of concept. User inputs are related to features and quality needs. Then, for all artifacts that come under the purview of the user needs are considered for weight computation. Once weight is computed different permutations that satisfy the requirements are created. Afterward, the critical path analysis is used to further refine the product derivation process. This will result in finding the best product. Customers who bought the product can use the product. Then it is called product in use. Once the product is thoroughly used by the customer, customer gives feedback which can help the product line development team to improve the product line based on feedback.

In a customer-centric environment, new product derivation is based on needs of customers. The procedure to derive new products is described here.

- Customer needs are analyzed and mapped to core and custom assets.
- If required, building a new core asset or custom asset is considered.
- Weights are computed for all the core and custom assets that are mapped to the interpreted user inputs. Weights are calculated using existing formula which is explained in section 4.4.1.
- All assets considered are used to form a network of nodes.
- Critical path analysis is carried out on the network to ensure that, certain nodes form part of the output.
- Besides, different permutations and combinations are made that meet the requirements of new product with certain degree of difference.
- Features like loose coupling and tight cohesiveness are considered in the process.
Finally, the best permutation is considered for product derivation. The derived product is delivered to customer. The best permutation is computed by considering the highest value obtained, which is the summation of weights of the core and custom assets identified.

The customer uses product and gives feedback. Based on the feedback, it is possible to repeat the cycle again to enhance the product further.

Products thus derived will be in the arsenal of the SPL considered.

4.4 QUALITY DRIVEN PRODUCT DERIVATION

In order to realize successful product derivation and improving SPL configuration management and product derivation, an algorithm named quality driven product derivation is proposed. This algorithm is used to arrive at quality product derivation. Both the algorithms are involved in the final derivation of product. The final product which is picked from a set of product compositions that meet the requirements of the new product is essentially a quality product. More details including the algorithms are provided here.

<table>
<thead>
<tr>
<th>Algorithm: Quality Driven Product Derivation (QDPD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs: Quality (Q) and feature (F) requirements</td>
</tr>
<tr>
<td>Outputs: New Product Derived (P)</td>
</tr>
</tbody>
</table>

1. Start
2. Initialize \(CA\) with available core assets
3. Initialize \(CusA\) with available custom assets
4. Initialize \(FA\)
5. For Each \(ca\in CA\) and \(CusA\)
   a. IF \(ca\) has \(Q\) and \(F\)
i. Add ca to FA

b. END IF

6. End For

7. Use FA in different propositions

8. Invoke Composition Analysis Algorithm

9. Determine the Final Composition of FA based on weights

10. Determine the Final Composition of FA based on weights

11. Derive New Product P

12. End

Figure 4.2: Proposed algorithm for deriving new product for SPL

In the algorithm “Quality” is a numeric item. The parameters considered are the memory, speed and security. For memory and speed numeric values are obtained from users and for security low, medium and high are taken from user and appropriate legend points are used to get numeric values. The legend points are taken as 1, 3, 5 for low, medium and high respectively.

The algorithm provides step by step procedure right from taking inputs to producing outputs. Moreover, the algorithm invokes another algorithm known as composition analysis algorithm. When multiple products result with different composition of artifacts, they are to be filtered out by using weights, relationships and critical path analysis. The algorithm presented in Figure 4.2 provides a systematic approach to derive a new product based on the needs of customers. It is the basis for the prototype application implemented to demonstrate the proof of concept. The algorithm takes input as feature and quality attributes. Then it has an iterative process that can pick different combinations of core and custom assets in the confines of given criteria. Both core and custom assets are involved in the product compositions. They are in fact used in order to have final product. The final product derivation is
subjected to quality and functional requirements desired in the newly derived product. Purely based on the weights different compositions are produced.

However, the composition analysis algorithm can have discrimination among different compositions and finally derive a new product that meets the requirements and also exhibits high quality in terms of satisfying needs beyond thresholds if any. For instance, an artifact is associated with certain weight which may correspond to its memory consumption. When customers have quality needs and they expect certain memory usage constraint, the proposed system can handle this situation. The situation is handled using the underlying algorithms adapted into the proposed framework.

The two algorithms presented earlier in this chapter work in tandem with each other. They work together in order to derive a high-quality product. When a new product is derived, its quality features are to meet the needs besides the functional requirements. However, when there are multiple product compositions that meet the needs, further analysis is made in order to get the best one from different candidate product compositions.

4.4.1 Composition Analysis Algorithm

For every proposition, score is computed and used for identifying best quality composition which becomes the final product derived. The score for each proposition is computed by adding the weights of the corresponding core and custom assets which are suitable for the given requirements. For example, as shown in Table 4.3 the score for each composition is termed as value. Value for the first proposition is computed as 1.7 which is the summation of 0.1+0.5+0.6+0.5. In order to achieve this, the algorithm makes use of another algorithm presented in Figure 4.3. The other algorithm is known as composition analysis algorithm which is meant for finding the best composition in the candidate list. The candidate list is verified by the algorithm and produce best quality product.

The composition analysis algorithm has three phases. In the first phase, it computes weights for all assets. The weights computed to all assets can provide a starting point to discriminate different versions of artifacts. When an artifact is available in different versions, it is essential to have known how to select the best one.
It is possible only through the criteria for selection. It is, in this case, two fold. First, the functional needs and the second the quality attributes. In the second phase, some of the compositions are chosen as best candidates. In the final phase, the algorithm employs critical path analysis for choosing the best composition which becomes the final product derived. Such product not only satisfies the client needs but also contain quality artifacts.

As shown in the composition algorithm there are three phases in which it works.

Following is the equation for calculating weights.

\[ W(f_a) = \sum_{f_v \in Bf_a} \frac{W(f_a)}{L_f(v)} \]  

Where \( W(f_a) \) refers to weight of the feature \( f \) in asset \( a \). The variable \( f_a \) represents a feature of the asset \( a \). The variable \( f_v \) represents other feature of asset \( a \). \( Bf_a \) represents all features of the artifact \( a \). \( L_f(v) \) represents the relationship between the features of the artifact \( a \).

Procedure followed to assign weights to the features in the assets is explained below.

1) Assign the weights to all attributes which can occur within an instance (simplification: consider weight of all attributes as 0.1).

2) Divide relationships into three categories i.e., association, inheritance and aggregation and calculate the weights to these relationships.

The following method is adopted for calculating the weight of relationships between the features of artifacts. For instance, 0.8 is assigned to inheritance relationship while 0.9 is associated with aggregation relationship [124]. The association of the relationships is computed as follows.

Weight from A→B = Number of messages from A→B / Total number of possible messages from A→B
3) Calculate the total asset weight for each individual asset in both core and custom assets using the equation 4.1.

For example in Table 4.1 weight of the asset A(2.3.1)=(0.1/0.9+0.1/0.5+0.1/0.5)=0.5

The final weight of an asset is computed as the sum of weight of all features of the asset. The possible score computed is used further to reduce errors in product derivation besides improving quality of product derivation. When quality products are derived, it reflects the true dynamics of the proposed framework and the underlying algorithms. The framework is used to guide the configuration management and product derivation. In this chapter, focus is on the product derivation. When the framework is used to find out the best possible product with different common features and variability obtained from user inputs, it is challenging to find out best compositions. Moreover, it is another challenging task to find the best out of different compositions.

**Algorithm:** Composition Analysis Algorithm

**Input:** Features of Assets $FA$ in different propositions

**Output:** Best composition based on weights/critical path analysis

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Start</td>
</tr>
<tr>
<td>2</td>
<td>Initialize $FA'$</td>
</tr>
</tbody>
</table>

**Computing Weights for Assets**

3. For each $fa$ in $FA$

   Extract $FV$ from $fa$

   For each $f$ in $FV$

   Compute weight of $f$ (equation 4.1) $S$  // $S$ represents the weight of an asset which is nothing but the sum of weights of all features/

   Associate weight to $fa$

   Add $fa$ to $FA'$
Finding the Best Composition Based on Weights

5 Initialize $H$ for holding highest score composition

6 Assign count of $FA'$ to $c$

7 Initialize $i$ to 0

8 For $i = 1$ to $c - 1$

   If $i^{th}$ composition has higher score than $i+1^{th}$ Then

   $H = i^{th}$ composition

   Else

   $H = i+1^{th}$ composition

   End If

9 End For

Finding the Best Composition based on Critical Path Analysis

10 For each $fa'$ in $FA'$

CriticalPath($i$)= weight of $i^{th}$ asset * relationship weight between $i^{th}$ and $i+1$

   Update $FA'$

11 End for

12 For $i = 1$ to $c - 1$

   If $i^{th}$ critical path of $FA'$ is higher than $i+1^{th}$ Then

   $H = i^{th}$ composition

   Else $H = i+1^{th}$ composition

   End If

9 End For

Figure 4.3: Composition Analysis Algorithm
In the first phase, weights are computed for assets. In the second phase, the best composition is identified and selected based on weights. In the third phase, the best composition is found based on the critical path analysis. The best permutation is considered by considering the highest value obtained based on weights of the core and custom assets identified and also different product compositions made of core and custom assets based on critical path analysis.

4.5 MATHEMATICAL MODEL FOR THE PROPOSED METHOD

In this section, a conceptual framework is provided. Configuration assets involved in the framework include concrete artifacts, variability models and software architecture. The product builder of SPL can pick different components in order to derive new product. Based on the quality features four assets are chosen and they are described here.

1. Let $S_A : \{sa.1, sa.2, \ldots, sa.n\}$ represent architecture of SPL where $sa.k$ is considered to be abstract component.
2. Let $C_C : \{cc.1, cc.2, \ldots, cc.n\}$ represent components that are nothing but core assets and $cc.k$ being the concrete component of SPL.
3. Let $V_E : \{vp.1, vp.2, \ldots, vp.n\}$ represent variability of SPL and $vp.k$ is considered to be a variation point.
4. Let $Q_S : \{qs.1, qs.2, \ldots, qs.n\}$ represent quality features of the product and $qs.i$ is one of the quality scenarios that needs to be in the derived product.

$Y: C_C \times V_E \rightarrow X$ is considered to be a realization relation. It is also considered as a property of concrete component named $cc.k$. This property can realize $vp.k$ which is a variation point. Similarly, $\alpha: C_C \times S_A \rightarrow X$ is an implementation relation which a property of $c.i$ which is a concrete component and realizes $a.i$ which is an abstract component. This is according to the architecture of SPL. The following are the two important relations that denote realization abstract components.
Let $\lambda: C_C \times S_A \rightarrow Q_S$ is the relation that denotes the realization of an architectural abstract component $sa.k \in A$ with corresponding realized component $cc.k \in C_c$, $a(cc.k, sa.k) \Rightarrow Q_S$. It does mean that the product's expected quality scenarios such as $qs.k \in Q_S$ are fulfilled by $a(cc.k, sa.k)$.

Let $K \subseteq C_C$ represent subset of components out of the core assets, and in similar fashion, let $W \subseteq S_A$ represent abstract components of SPL architecture. Let $S : \{V_E, K, W\}$ is a product's production plan where $(\forall vp)vp \in V_E : (\exists c, sa : cc \in K, sa \in W : \gamma(cc.vp) \wedge a(cc, sa))$.

From these two relations, it is understood that with respect to all variation points existed in $V_E$, one tuple is existed and denoted as $(cc, sa)$. Here $cc$ can realize $(cc, vp)$ which is a variation point. Here abstract component $(cc, sa)$ is implemented by $cc$.

Figure 4.4: Architecture, realization and implementation to fulfill quality requirements

Set definition for architecture, components and variability (a), relations representing formal realization and implementation (b), and relation that represents implementation which fulfills quality requirements
As shown in Figure 4.4 (a) there is set definition for different aspects such as variability, components and architecture. Figure 4.4(b) shows the relationship between formal realizations besides the implementation. Figure 4.4(c) shows the implementation relation which fulfills quality requirements of product derivation. As SPL makes use of quality attributes to derive new product those elements are explicitly included and represented by S.

\[(\forall vp, cc_o, sa_o) vp \in V_E, cc_o \in K, sa_o \in W : (\exists cc, sa \mid cc, sa \in K, sa \in W)
\]

\[:(cc_o, vp) \land \alpha(cc_o, sa_o) \land (\lambda(cc, sa)) \] is false

It does mean that all components cannot fulfil the requirements of the new product to be derived. Therefore let \( SQ : \{V_E, K, W, Q_S\} \) be production plan with required components \( Q_K \subseteq C_c \).

\[(\forall vp, cc_o, sa_o) \forall vp \in V_E, cc_o \in Q_K, sa_o \in W : (\exists cc, sa \mid cc, sa \in Q_K, sa, sa \in W)
\]

\[:(cc_o, vp) \land \alpha(cc_o, sa_o) \land (\lambda(cc, sa)) \] is true

S and SQ are compared here. SQ represents a collection of components that are compatible with a variation point besides satisfying quality attributes Q and realizing the architecture. Therefore the elements in K are filtered so as to conform to QK. For further optimization SP : \{(cc.i, vp.k), (cc.i, sa.j), q.i \in Q_S\}. For given q.i in Qs (cc.i, sa.j) is considered which reflects realization and implementation by identifying a set of sensitive points to achieve required quality standards.

**4.6 CASE STUDY**

Dr. School product line software comes under School Management System wherein it is proprietary software developed for education and healthcare domains. The source of the software is from Software Solutions Pvt Ltd Hyderabad. Its use case involves health care facilities provided by doctors at schools to ensure a healthy school environment. Doctor examines students and makes their profiles which are used further for ensuring well-informed decisions on the health care of patients. The SPL is in distributed environment that helps health care personnel and school personnel to work together. Health promotion camps and periodical health checkups
are conducted at schools to ensure that students can focus more on their studies. Figure 4.5 shows the components of the Dr School SPL.

Dr. School SPL has various components involved. They include GUI for intuitive interface, Main Controller for coordinating persistence, medical profile creation, medical camps, health promotion and health checkups. Persistence Manager takes care of storing and retrieving data permanently. Local Persistence and Remote Persistence are the two variants of Persistence that take care of persistence at local and global level. The local level persistence is related to the local storage media which is nothing but an RDBMS which is based on relational model which was proposed by Dr. E. F. Codd. Since RDBMS is the most stable model, it is widely used in the world for storage purposes. The global persistence management refers to the fact that persistence can be made in a remote system also. In other words, it is possible to store data in a remote server. The remote server might be located in the same building or across the continents. Thus the ability to store in a remote server can lead to more useful storage with reliability. Moreover, this can lead to have cloud services in future as well.

When data is stored on multiple servers, it is possible to have distributed processing. Stating differently it is possible to have a two-phase commit protocol when data is stored in multiple servers. The two-phase commit protocol enables distributed transactions. Distributed transaction is a set of activities carried out in multiple distributed servers. When all operations are carried out successfully then the transaction is said to be completed. When any one of the operations fails, it has to rollback the whole transaction. Thus the two phase commit is used in this case study in order to have efficient transactions over multiple servers. The reason behind this is that when multiple domains like healthcare and education are involved, it is necessary to have ability to have transition in two or more servers. In this sense, there needs to be distributed transactions in place.
There shall not be any direct connection between the components Main Controller and Media Controller. However, stakeholders can interact with GUI which uses Media Controller that takes care of handling different media like audio, video and images in communication and storage. Different media can help in having diversified storage and communication procedures that can cater to the needs of all stakeholders involved in the product line. CommunicationManager takes care of different types of communication between healthcare and school personnel. They include chatting, voice calls, e-mails and SMS. Chatting helps in providing textual
messages that can convey messages effectively. In the same fashion, voice calls can be made so as to convey message with audio enabled. The e-mails are alternative communication channel which is more effective as there is possibility of sending matter and also attachments. This can help in even sharing cases as required. All these components are reusable components that can be part of an SPL and they can involve in product derivation. The components when reused, it is possible to have productivity in product line in terms of reducing time and effort. It also can influence the budget and time to market feature of product line.

Figure 4.6: The feature model of Dr. School SPL

As shown in Figure 4.6, it is visible that the feature model of Dr. School is presented in terms of features. “Alternative” means either of the features should be used. Similarly, “Or” means either of or all the features can be used based on the requirements.

This can show the scope and external variability. The features are alternative features, mandatory features and optional features. There are three communication alternatives and two persistence alternatives such as local database and remote database. In the same fashion, there are four alternatives for communications. They are text, audio, video and image. The information inputs can be of different types. The text mode is simple and used as and when the stakeholders need communication in terms of passing messages. The audio communication is needed when two parties are to communicate orally and understand qualitatively. The video communication is required when there needs to be video related to healthcare and that has to be transmitted between the important stakeholders in the case. And finally the image
communication is also important when medical images are to be transferred between the stakeholders. Thus both the domains get benefited with different input formats supported. Or, mandatory and alternative kinds of notations are used in the feature model. The feature model shows different features and the relationships among them as well. The feature model is represented as ontology model in our work which is shown in Chapter 6.

**Variability Model**

![Variability Model Diagram]

**Architectural Model**

![Architectural Model Diagram]

**Concrete Components**

Figure 4.7: The variability model of Dr. School SPL

As shown in Figure 4.7, the SPL configuration assets along with their relationships are shown. There are abstract components in the architectural model while the concrete models are found in the process of realization. The variability model is required to know variability and variability points are the basis for making decisions for deriving new products in the SPL. There is the concept of abstract component and concrete component. Abstract component contains abstract functions without implementation details. The concrete components are the components that can be used in the SPL. These components are used in the product line. The components can also be related with the reusable technology known as Component Object Model (COM). The component technology helps reusability in SPL. The
architectural model and variability model are presented. The main difference is that the architectural model contains abstract components and they are realized in the other model. Moreover the variability model takes care of variability points that can help in producing components with specific functionality. Ultimately the variability is realized in the form of concrete components.

Figure 4.8: The feature model with security and quality scenarios

As shown in Figure 4.8, there are quality scenarios associated with Dr. School product line. It also has security requirements to be fulfilled. The persistence manager is responsible for taking care of local and remote database components for persistence. The implementation prototype is shown in Figure 4.9.

As presented in Figure 4.9, the proposed framework is realized with a prototype application which is built using Java programming language. The application demonstrates the functionality of the framework along with the two algorithms implemented for SPL configuration management and new product derivation. The application focuses on two functionalities. They are known as SPL configuration management and derivation of a new quality product. These two are
quite related in the SPL perspective. When configuration management is done effectively, it can have its impact on the accurate and quality product derivation. When quality product is derived, there will be success in product line.

In Figure 4.9, Dr. School product line is considered. In which there are five different products are considered named as P1, P2, P3, P4 and P5. Product P1 is elaborated in the figure which contains core asset database and custom asset database. In the core assets A,B with various versions are available and in custom assets P, Q with various versions as shown are available. Based on the requirements product is derived. A product is made up of multiple artifacts. Each product is in fact realized by combining multiple artifacts that exhibit commonality and certain variability. When variability is known well, it is possible to make well-informed decisions in the product line with respect to product derivation. With respect to Dr. School product line, there might be a number of schools and number of doctors with different requirements. However, there are certain common amenities to be provided by the product line. Therefore, the common things are identified and artifacts are built. The variability leads to having new products. Each school associated with the product line having its own requirements with more commonality and less variability. Such requirements can be handled with ease using the proposed approach for product derivation.

The product derivation process can help organization which renders services pertaining to Dr. School to have high quality products and that cater to the needs of potential collaborators. There is collaboration between the entities from both education and healthcare domains. At the same time, different schools can have different additional needs that are not required by other schools. Thus variability points are identified and used in the product derivation. Therefore it is essential to have clear picture of commonality and variability in the given product line in order to justify the product derivation. When new product is derived, there should be reason why that product is needed. Unless the new product shows variability it is not recommended to have product derivation. Product derivation, therefore, is used to have new product that exhibits certain new features and they are not available in different products in the product family.
Figure 4.9: Overview of prototype application

As shown in Figure 4.9 it is evident that the prototype application built to demonstrate the proof of concept. It shows Dr. School product line and corresponding products. Core assets and custom assets are also configured. The version control is implemented. Each artifact has a version which contains three parts. They are major version, minor version and micro version. The major version reflects an artifact that is subjected to major changes. The minor version and micro version shows relatively small changes in the artifact. However, the three parts convey meaning to the users of the application. Besides versioning the artifacts are identified by unique names. The application obtains the complete state of an SPL from repository and presents with
GUI intuitively. Moreover, it can help in product derivation process. In fact, the algorithms are implemented as underlying functionalities in the application.

The prototype application is built using Java programming language. It is object oriented language which has features such as Write Once and Runs Anywhere (WORA). It does mean that the prototype application runs on any platform provided JVM for that platform. The application as both Graphical User Interface (GUI) and functionality. The GUI is made using Java.swing package. This package contains lightweight components that can be reused to have user interface for the application. The swing API is also part of Java Foundation Classes (JFC). It does mean that JFC is the recommended API for GUI application development. In the prototype, JFC classes are used in order to have intuitive interface. The swing package and its threading concept are used in order to implement functionality. Therefore the prototype has two things. They are user interface and functionality. The user interface is made using Swing API in Java while the functionality is implemented using algorithms proposed in this chapter. The algorithms are implemented again using Java programming language. The data of SPL are stored in repositories provided by Relational Database Management System (RDBMS).

4.7 RESULTS AND DISCUSSION

Experiments are made on SPL configuration management and new product derivation. Towards this end, the prototype application is used. The application and underlying framework were implemented in such a way that it works for any SPL. Therefore it is generic in nature and not confined to any domain. Its configuration management involves versioning of core assets, custom assets, products besides computing weights and finding weight of the relationships among components. Experiments made with Dr. School product line revealed that the proposed framework is realized for achieving flexible configuration management of SPL and quality product derivation. The experimental results are presented as follows. The results show the usefulness of the framework and prototype application in deriving quality products in different SPLs. The results of Dr. School SPL are presented in this chapter.
As shown in Table 4.1 there are different core assets that have been configured by the proposed application. Every asset is associated with versions. Weights are computed using equation 4.1. For example in Table 4.1

weight of the asset A(2.3.1) = \( \frac{0.1}{0.1/0.9 + 0.1/0.5 + 0.1/0.5} = 0.5 \)

The weights are used in order to have best product derivation later. Four core assets namely A, B, C and D are presented with different versions available.

As shown in Table 4.2 there are different custom assets that have been configured by the proposed application. Every custom asset is associated with versions. Each asset is associated with weights which are computed using equation
4.1. Weights are used later for best product derivation. Four assets named P, Q, R, and S are presented with different versions available.

Table 4.2 – Custom assets available with different versions

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1.1.1(0.1)</td>
<td>Q1.1.1(0.5)</td>
<td>R1.1.1(0.3)</td>
<td>S1.1.1(0.6)</td>
</tr>
<tr>
<td>P1.2.2(0.2)</td>
<td>Q1.2.2(0.3)</td>
<td>R1.2.5(0.2)</td>
<td>S1.2.1(0.3)</td>
</tr>
<tr>
<td>P1.2.3(0.6)</td>
<td>Q1.3.3(0.3)</td>
<td>R1.3.6(0.3)</td>
<td>S1.2.2(0.2)</td>
</tr>
<tr>
<td>P2.2.5(0.6)</td>
<td>Q1.4.2(0.2)</td>
<td>R1.4.2(0.6)</td>
<td>S1.3.3(0.1)</td>
</tr>
<tr>
<td>P2.3.6(0.5)</td>
<td>Q2.1.2(0.1)</td>
<td>R2.2.5(0.2)</td>
<td>S2.3.5(0.2)</td>
</tr>
<tr>
<td>P2.3.6(0.3)</td>
<td>Q2.2.3(0.2)</td>
<td>R2.3.5(0.3)</td>
<td>S2.3.6(0.3)</td>
</tr>
<tr>
<td>P3.1.2(0.1)</td>
<td>Q3.1.2(0.6)</td>
<td>R2.4.1(0.6)</td>
<td>S3.1.1(0.6)</td>
</tr>
<tr>
<td>P3.2.2(0.2)</td>
<td>Q3.2.3(0.3)</td>
<td>R3.1.1(0.3)</td>
<td>S3.1.2(0.6)</td>
</tr>
<tr>
<td>P3.3.1(0.5)</td>
<td>Q3.3.5(0.2)</td>
<td>R3.2.5(0.2)</td>
<td>S3.2.5(0.2)</td>
</tr>
</tbody>
</table>

As shown in Table 4.3 there are different product compositions that are made up of core and custom assets and the weights associated with assets. Each product composition has four assets that include both core and custom ones. The weights of all assets are computed and a value is derived as presented in the table. This value determines the quality of the product. More value refers to more suitable or quality product based on the needs of customer. The first row in the table reflects high value for weight which is said to have high quality.
Table 4.3 - Different product compositions made of core and custom assets based on weight of assets

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>P</td>
<td>Q</td>
<td>value</td>
<td></td>
</tr>
<tr>
<td>A3.1.2(0.1)</td>
<td>B1.1.1(0.5)</td>
<td>P2.2.5(0.6)</td>
<td>Q1.1.1(0.5)</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>A3.2.5(0.2)</td>
<td>B1.2.5(0.3)</td>
<td>P2.3.6(0.5)</td>
<td>Q1.2.2(0.3)</td>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td>A3.3.1(0.5)</td>
<td>B1.3.2(0.3)</td>
<td>P2.3.6(0.3)</td>
<td>Q1.3.3(0.3)</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>A3.3.2(0.6)</td>
<td>B1.4.1(0.2)</td>
<td>P2.3.7(0.4)</td>
<td>Q1.4.2(0.2)</td>
<td>1.4</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4 - Different product compositions made of core and custom assets based on Critical path analysis

<table>
<thead>
<tr>
<th></th>
<th>Wt</th>
<th></th>
<th>Wt</th>
<th></th>
<th>Wt</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>B</td>
<td></td>
<td></td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3.1.2(0.1)</td>
<td>0.5</td>
<td>B1.1.1(0.5)</td>
<td>0.3</td>
<td>P2.2.5(0.6)</td>
<td>0.8</td>
<td>Q1.1.1(0.5)</td>
<td>1.18</td>
<td></td>
</tr>
<tr>
<td>A3.2.5(0.2)</td>
<td>0.6</td>
<td>B1.2.5(0.3)</td>
<td>0.5</td>
<td>P2.3.6(0.5)</td>
<td>0.8</td>
<td>Q1.2.2(0.3)</td>
<td>0.97</td>
<td></td>
</tr>
<tr>
<td>A3.3.1(0.5)</td>
<td>0.4</td>
<td>B1.3.2(0.3)</td>
<td>0.2</td>
<td>P2.3.6(0.3)</td>
<td>0.8</td>
<td>Q1.3.3(0.3)</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>A3.3.2(0.6)</td>
<td>0.3</td>
<td>B1.4.1(0.2)</td>
<td>0.6</td>
<td>P2.3.7(0.4)</td>
<td>0.8</td>
<td>Q1.4.2(0.2)</td>
<td>0.82</td>
<td></td>
</tr>
</tbody>
</table>

As seen in Table 4.4, many candidate propositions are presented. The weights in the column-2 of Table - 4.4 are the weights of the relationship between the assets. This weight is calculated based on the type of relationship between the assets. For example weight 0.8 is assigned to the link between P and Q which is an inheritance relationship. The links between A and B ,B and P are association relationships and the weight is computed using the following formula

\[
\text{Weight from } A \rightarrow B = \frac{\text{Number of messages from } A \rightarrow B}{\text{Total number of possible messages from } A \rightarrow B}
\]
Weight from A→B = 1/2 = 0.5
Weight from P→Q = 2/6 = 0.3

Features like time complexity, space complexity and security are considered for each asset beyond calculating the weight. Based on the user requirements, appropriate assets are selected and when more than one combination of assets is available, the best product is derived based on critical path and also the highest weight as mentioned in the algorithm. However, the best product based on the critical path analysis and weights is found by Composition Analysis (CA) algorithm. Then the final product is highlighted and returned. Table 4.3 shows best products based on weights while Table 4.4 highlights the final product that is selected based on critical path analysis. The critical path is the longest path in a network. Hence from the above table, the first row that is the combination of version A3.1.2 and B1.1.1 of core assets and P2.2.5 and Q1.1.1 of custom assets is identified as the best product for the given requirements. The value based on critical path analysis is computed as summation of product of the weight of the asset and the weight of the link for all the assets. For example, the final value in Table 4.4 of the first row is calculated as 
\[(0.1 \times 0.5) + (0.5 \times 0.3) + (0.6 \times 0.8) + 0.5 = 1.18\].
The sum of all weights of all artifacts identified in a product composition constitutes a value which refers to the quality of the product. Quantitative analysis of the weights is done in order to have best product derivation. Quantitative analysis leads to finding the product composition which can satisfy customer needs as much as possible. The other compositions are finally discarded and the best one is used for the purpose of product derivation.

Results shown in Figure 4.10 are based on the information presented in Table 4.3. The horizontal axis represents products while the vertical axis represents weights. Results revealed that the product to the left in the graph is best. It is based on the weights associated with the underlying artifacts of newly derived product. Results are obtained by applying both the algorithms presented earlier in this chapter.
The results presented in Figure 4.11 are based on the information presented in Table 4.4. The horizontal axis represents products while the vertical axis represents value of critical path. The product considered in this experiment was Dr. School product line which is a school management system that takes care of education and medical domains. The four products shown in the Figures 4.10 and 4.11 are the
different combinations of core and custom assets for the product Dr. School for the given requirements.

As shown in Figures 4.10 and 4.11 the first combination i.e., a product with the combination of A (3.1.2), B (1.1.1), P (2.2.5) and Q (1.1.1) is the best product for the given requirements based on weights and also the critical path. Critical path analysis is the technique used in many domains like effort estimation, time estimation etc. The longest path in the network (with assets as nodes and the relationship between the assets as the links) is called the critical path. The assets forming the critical path are termed as critical assets. Hence, the assets involved in forming the critical path are considered and those assets are finalized for the best product derivation.

The results were obtained after execution of both the algorithms presented in this chapter. The value associated in the form of critical path is the basis for making decisions. More value indicates more weight with respect to critical path. And more value refers to high quality as well. Thus final decisions made using weights and critical path ensure that best quality is derived using the proposed approach.

**SUMMARY**

In the recent past, SPLE has attracted significant research. As the software development involves complexity due to distributed technologies, SPL configuration management assumed important. SPL configuration management and quality product derivation are challenging optimization problems considered in this thesis. SPL configuration management is actually a paradigm shift from that of traditional configuration approaches. With SPL development cost and time are reduced. However, there is increased complexity in the SPL as it deals with plethora of core and custom assets with commonalities and variability. In the presence of diversified customer needs the SPL configuration management and product derivation needs to deal with quality and performance attributes. Product derivation process needs to be effective so as to fulfill the needs of customers. In this chapter, a generic framework is presented which caters to effective SPL configuration management and quality product derivation. Two algorithms are proposed and implemented to identify best possible compositions while deriving new product. In the iterative process, the
algorithms finally choose the best product that satisfies customer needs. The algorithms exploit the weight associated core and custom assets and the weights of the relationships between the assets.

The critical path is the longest path arrived at from the start to the end of an activity based on the requirements of functional and quality attributes the best composition arrives, which considers low coupling and high cohesiveness. When cohesiveness is within artifacts and less coupled nature between artifacts, this is the ideal situation where the product line success will be more. Critical path analysis is thus improving the quality of product to be derived. When high-quality products are derived with customer centric approach, the success of product line increases the derived products meet the increased needs of customers with certain variability. Dr. School product line is used to demonstrate the proof of concept. Results of the application are in conformance with the ground truth evaluation of the proposed framework and underlying algorithms. The third part of the framework that is variability aware design patterns is implemented and tested for optimization of configuration management and quality product derivation in the next chapter.

******