A Systematic Process to Design Product Line Architecture*

Soo Dong Kim, Soo Ho Chang, and Hyun Jung La

Department of Computer Science, Soongsil University,
1-1 Sangdo-Dong, Dongjak-Ku, Seoul, Korea 156-743
sdkim@comp.ssu.ac.kr, {shchang, hjla}@otlab.ssu.ac.kr

Abstract. Product Line Engineering is being accepted as a representative software reuse methodology by using core assets and product line architecture is known as a key element of core assets. However, current research on product line engineering has room to provide specific and detailed guidelines of designing product line architectures and reflecting variability in the architecture. In this paper, we present a reference model and a process to design the architecture with detailed instructions. Especially architectural variability is codified by describing decision model representing variation.

1 Introduction

Product Line Engineering (PLE) has been widely accepted as a representative software reuse methodology using core assets. Architecture plays a key role in scoping applications and it defines overall structures for applications. A core asset in PLE provides a framework for developing various products in the product line. As a key element of core assets, product line architecture (PLA) should also be generic to be applied to various products.

Although processes or methods to design PLA have been suggested in various research works, there is a large room for improvement, providing specific and detailed process of defining PLA and reflecting architectural variability. Especially, how the essential elements of architecture design such as driver, view, and styles can be applied to PLA should be specified in detail.

In this paper, we first present a reference model of PLA and propose a systematic process to design PLA. Each activity of the process is elaborated with detailed instructions. In addition, architectural variability is codified by describing decision model representing variation points.

2 Related Works

Bosch proposes a design method for system family software architectures [1]. When designing family architecture, architects design architecture based on archetype that is

* This work was supported by Korea Research Foundation Grant. (KRF-2004-005-D00172)

O. Gervasi et al. (Eds.): ICCSA 2005, LNCS 3480, pp. 46 – 56, 2005.

core abstractions of the system, assess architecture for quality requirements using scenarios, and then transform quality requirements to functionality to improve the quality attributes of architecture. For transformation, system family architecture may require achieving variable requirements, optionality of parts of the architecture, and conflicts between components as well as imposing architectural style, architectural pattern, and design pattern.

QADA is a method standing for Quality-Driven Architecture Design Analysis method[2]. The method consists of activities; *Requirements engineering, Conceptual architecture design and analysis, Concrete architecture design and analysis*. In conceptual architecture analysis, analysis and representation of variability is focused using variation point description and product-line pattern.

Ceron and his colleagues propose processes for developing reference architecture and deriving the architecture as well as architectural meta-model[3]. The meta-model appends architectural variability to P1471[4]. The process for developing reference architecture consists of three activities; *Scoping, Choosing architectural style, Providing variability*. Applying functional and non-functional features into architecture, this work also points out conflict problems of variability, a stability of common requirements and architectural style in reference architecture as well.

Thiel suggests a process framework that supports the design of high-quality product family architectures, called QUASAR [5]. QUASAR is organized with three workflows; *Preparation*, *Modeling* and *Evaluation* that analyze the achievement of architectural qualities. To integrate variability with PLA design, this work gives guidelines for documenting variability about where variation points are in architectural views, how to instantiate them, and resolution rules.

These related works define more or less implicitly what is included in PLA and suggest overall process for designing PLA. Especially, they mention needs to represent and document variability in designing PLA. Hence, we can make more practical design process by supplementing detail instructions. To enhance importance of designing variability, we can classify types of the architectural variability and represent variability more concretely by using architectural decision model.

3 Meta-model of Product Line Architecture

Based on our survey, we now present our meta-model of PLA by taking the common elements of the related works and refining them as in **Fig. 1**.

Elements of PLA are distinguished into abstract elements and concrete elements. The abstract one is conceptual elements to which PLA should conform or refer, whereas the concrete one is elements which constitute PLA as physical parts. From the meta-model, we specify each element as followings.

Architectural View: Based on the requirements and PL analysis model derived from the requirements, we choose perspectives to illuminate PL, called as a *view* shown in the figure [6]. Although many kinds of view types are proposed, there is no standard on architectural view [7]. However, we choose the three kinds of view, Module View, C&C View and Allocation View, as specialized view types of PL architectural view since they are generally accepted [2][3][6].

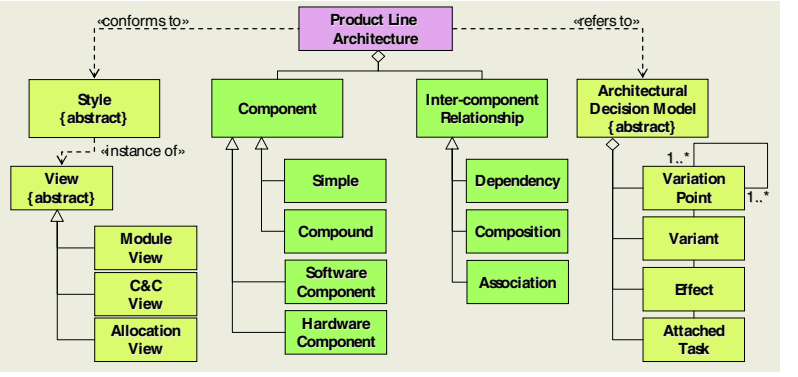

Fig. 1. Meta-model of PLA

Architectural Style: An architectural *style* is a specialization of elements and relation types [6] and helps simplify the architecting process [8]. From the notion, architecture design begins with choosing most appropriate architectural styles and components and inter-component relationships in architecture are more or less directly derived from the selected styles. Therefore, it is fair to state that the components and relationships effectively implement functional and nonfunctional requirements within architectural styles. Architectural styles can be defined as abstract elements to which architecture conforms rather than constituents of PLA. A style is a partial instance of a view, and a set of several architecture styles can realize a view [6][9].

Component and Inter-component Relationship: PLA consists of *Components* and *Inter-component Relationships*. Components implement functional and non-functional requirements. While functional requirements are directly designed, non-functional requirements may derive additional functional requirements which realize the quality attributes and are implemented into the components. A component is specialized into simple and compound components as their composition relationships. A component is also specialized into software and hardware component.

Inter-component relationship may have several stereotypes depending on architectural views by which the relationships are represented. In the meta-model, the relationship is specified as dependency, composition, or association. Generally a dependency is for message passing between components, composition is for relationships between simple and compound components, and an association is for persistent relationships between hardware components.

Architectural Decision Model (ADM): Since decision model is specification of variability in PL, it is not a design element only for PLA but a reference from which variability is designed into PLA. Hence, the relationship between PLA and ADM is shown as *refers to,* as in **Fig. 1**. We call the decision model capturing architectural variability *Architectural Decision Model (ADM)*. *Variation Point*, *Variant*, *Effect*, and *Attached Task* are constituent to the architectural decision model [10][11].

To elaborate architectural variability, we propose following candidate variants types for architectural variation points;

- **Architectural Style:** Since a style represents part of architecture, architecture may contain a set of styles. Besides, architecture should be stable [3]. Therefore, a variation point of an architectural style set may have a few style variations. That is, a few styles in the set may vary from product to product such as optional or alternative.
- **Component:** Variations of components in architecture can be classified to optional and alternative. Optional variability is for the case in which a component is used or not. Alternative is for the case in which another component can be substituted for one component. Note that variations may be occurred in components such as logics, workflows, or data [12]. However it is not architectural variability but intra-component variability. Therefore descriptions of the intra-component variability are out of the scope of this paper.
- **Relationship:** Within the one style, message passing among components may be changed by applications. We define that variation occurs in inter-component relationship. Note that this variation point should be distinguished from architectural style variability.

4 Process and Instructions

We now present a process to design PLA in **Fig. 2**. The process consists of five activities and each activity has its detailed steps.

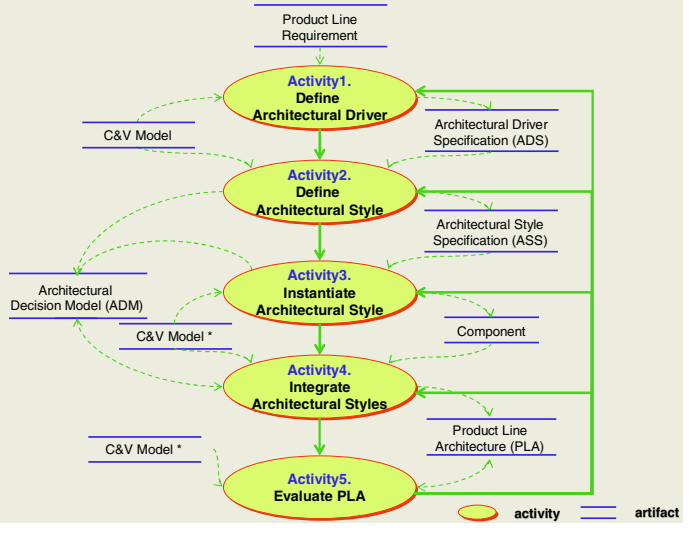

Fig. 2. Process and Artifacts for PLA Design

Since these activities are included in the phase for PLA design, domain analysis is preceded and component design is followed. PLA design phase begins with analysis model delivered from domain analysis phase and carries over a PLA to component design phase.

The five activities are Define Architectural Driver, Define Architectural Style, Instantiate Architectural Style, Integrate Architectural Styles and Evaluate Architecture. Each activity is decomposed by steps and provides instructions and templates with the steps.

4.1 Activity 1. Define PL Architectural Driver

Overview: This activity is to derive a set of PL architectural drivers for a product line. An architectural driver is a requirement which has influence on the design of architecture [1][13]. Therefore, acquiring right architectural drivers is an essential prerequisite for well-designed architecture. In this activity, both common and variable drivers are identified since there can be variation on product architectures.

Input and Output: As shown in **Fig. 3**, the input to this activity is both product line requirement specifications (PRS) and C&V model. A PRS specifies functional and nonfunctional requirement. Each type of requirement for variation can be mandatory, alternative or optional. C&V model is an analysis model of PRS, and the model specifies both common and variable requirements in systematic way. Examples of C&V model are feature analysis [14].

We assume that PRS is available before this process is applied. That is functional and nonfunctional requirements are separately defined and the PRS is relatively complete and consistent. If not, techniques of requirement engineering [15] can be applied to derive a high quality PRS. We also assume that C&V modeling has been completed before this process.

The output artifact of this activity is Architectural Driver Specification (ADS) which specifies architectural drivers and their priorities.

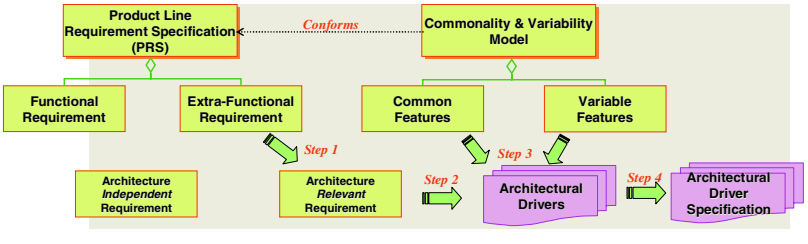

Fig. 3. Information Flow from PL Requirement to PLA

Instruction: This activity is carried out in four steps as in **Fig. 3**. The first step is to extract architecture-relevant requirements from PRS. As shown in **Fig. 3**, not all nonfunctional requirements are architecture-relevant. Hence, we need to extract nonfunctional requirements that have some impact on the design of PLA. In general, quality attributes and constrains are architecture-relevant. For example, reliability as a quality attribute may be applied into architecture as data mirroring.

The second step is to define architectural drivers from the architecture-relevant requirements. That is, architecture-relevant requirements are analyzed and itemized as architectural drivers. Each architecture driver is given a name for further references in subsequent activities.

The third step is to classify architectural drivers according to the common and variable features from C&V model. An architectural driver itself can be variable in two forms; alternative and optional.

The fourth step is to prioritize drivers according to the commonality and significance. Architectural drivers are the main source for choosing architectural styles and therefore different drivers may yield different styles. When a product line has several drivers, resulting architectural styles may be complicated. As a logical way to resolve the possible complications of styles, we suggest to prioritize drivers using product line dependent criteria. Each driver is given with its *priority, name, description* and *variability type* which can be mandatory, optional, or alternative.

4.2 Activity 2. Define Architectural Styles

Overview: This activity is to derive architectural styles. By using architectural style that eases the design process by providing routine solution for recurring problems, we can reuse design and code, easily understand a system's organization, and gain insight into style-specific analysis of solution characteristics [16]. In this activity, architectural styles which satisfy with architectural drivers and make PLA effective are defined.

Input and Output: To define an appropriate architectural style, this activity requires ADS. One output is an *Architectural Style Specification (ASS)* which addresses architectural views, styles, and rationales that are shared by PL applications. The other output is a part of an ADM which specifies architectural variation on PL architectural style set. ADM describes all architectural variability and only part of ADM, *Style Set Variability*, is defined in this activity.

Instruction: This activity is to choose appropriate architectural styles according to derived architectural drivers and consists of three steps as followings.

The first step is to select architectural views by which PLA is illustrated. At least one view should be chosen from the three views [6]. Note that a view may focus on several architectural drivers and an architectural driver may also be realized in one or more views. With this step, a view list with architectural drivers is listed and ranked as priority of the drivers.

The second step is to choose architectural style for each view. We firstly explore and list candidate architectural styles for each architectural driver and then, decide architectural style as our strategies, project policies, or constraints. The rank of architectural drivers can affect resolving conflicts among architectural styles.

The third step is to specify ADM for variation on the architectural style set. Style set variability which is identified and specified in this step is stemmed from variations on architectural drivers. Different architectural drivers drive different architectural styles covering each driver as shown **Fig. 4**.

For one architectural view, several architectural drivers have their styles. Especially an architectural driver which has variation has one or more style depending on variation type. From the **Fig. 4**, we can extract architectural style set {style a, style b, …, *style i-1*|*style i-b*, ..}. The style set has variation on *style i*.

According to variation type of architectural driver in ADS, variants of style set variability are defined as variable driver and its style. Variation type is equally

transformed from variation type of ADS. Effect and task can be specified in this step, and further refined in *refining overlapped area step in activity 4*.

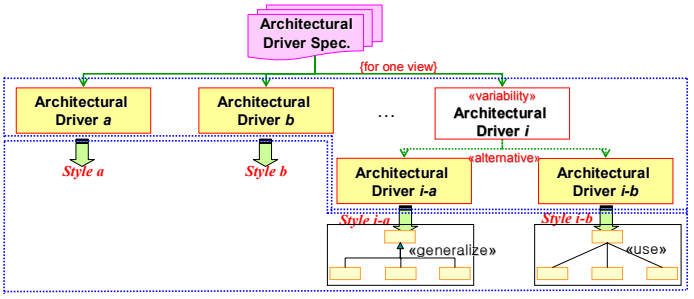

Fig. 4. Architectural Style Set Variability of PLA

4.3 Activity 3. Instantiate Architectural Style

Overview: This activity is to realize ASS and ADM. Architectural styles are represented by architectural units and their relationships. In this activity, a specification of architectural style set is transformed into concrete parts of architecture. During style instantiation, variation on a style is also applied into the instantiated styles.

Input and Output: The input to this activity is both an ASS and an ADM which are defined in activity 2. The output artifact of this activity is an embodied architectural style set which are represented by actual components and their relationships. ADM is also refined as appending *Architectural Style Variability*.

Instruction: This activity is carried out in three steps. The first step is to extract component. Types of components in architecture can be divided into types; software and hardware components. Software components are applied into logical view such as module view and process view such as C&C view. To extract the component, we may use a clustering method in [17]. Hardware components are represented in physical view such as allocation view and the component may be server, DBMS, and other hardware units. To extract these physical components, we may use strategic constraints.

The second step is to apply the extracted components into architectural style. For one architectural style, we arrange the extracted components and then elicit relationships among arranged components. Depending on types of components, dependency, composition, or association can be applied with specific stereo-types.

The third step is to append architectural style variability to ADM. Architectural style variability is discovered in component or inter-component relationship. **Fig. 5** shows an example of architectural style variability on a share-date style of Sale Domain.

During instantiating architectural style, one component may not be used or replaced with other component by one application. In addition one relationship between components may be omitted or changed depending on applications. These variations may be represented in ADM.

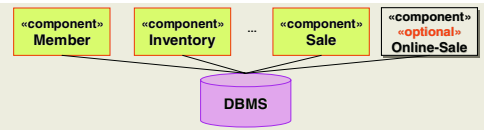

Fig. 5. Architectural Style Variability of PLA

4.4 Activity 4. Integrate Architectural Styles

Overview: This activity is to finalize PLA by combining instantiated architectural styles. Individual instantiated architectural style is a part of PLA, so it should be populated into whole PLA. During arranging several styles, overlapped area among styles should also be recognized and resolved in this activity. In addition, ADM is more refined by *Effects* and *Tasks* describing propagation of architectural variability.

Input and Output: The input to this activity is an instantiated architectural style set where styles may have architectural variability. The output artifact of this activity is PLA in which the whole range of architectural elements is represented in terms of components and their relationships on chosen views.

Instruction: This activity is carried out in two steps. The first step is to gather the instantiated styles on a same view. Since components in different styles may have different granularity, it is needed to normalize components in different styles into same-grained components in this step.

The second step is to link styles and refine overlapped areas among styles. Some components may be included in several instantiated styles and other components may be embedded in compound components which are included in other styles. From these cases, we define overlapped area which contains some components and their relations included in two or more styles. The overlapped area is distinguished two types; one is area having architectural variation and the other is area not having variation. In the case of overlapped area having variation, architectural variability should be handled more carefully. By resolving overlapped area and applying variability into the area, overlapped component may be refined and relations may be modified. **Fig. 6** shows an example of overlapped area during integrating instantiated styles.

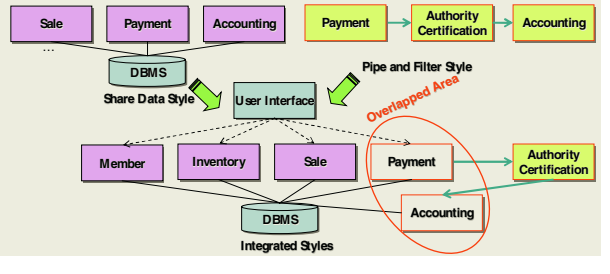

Fig. 6. An Example of Integrated Styles and Overlapped Area

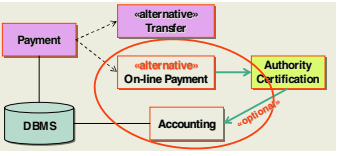

Fig. 7. An Example of Overlapped Area having Variability

Fig. 7 represents a resolution of overlapped area. In this case, *payment component* is refined *Transfer* and *On-line payment* components as alternatives. For its propagation, the relationship between *Authority Certification* component and *Accounting* component is refined as optional relationship. The refinement is also appended to ADM in this step

4.5 Activity 5. Validate PLA

Overview: This activity is to evaluate PLA with several check lists and decide whether PLA should be refined or finalized. As criteria of items in the check list, the process is returned to prior activities or closed. In this activity, we propose a check list which support to validate PLA and instruction using the check list.

Input and Output: The input to this activity is PLA defined through activity 1 to 4, C&V model, and a predefined check list. The output of this activity is evaluated PLA and result indicating process direction.

Instruction: First of all, we define a check list to evaluate PLA for this activity as shown **Table 1**.

Table 1. Check List to Evaluate PLA

Artifact	Check Point
Architectural Driver	Do the architectural drivers meet non-function requirement?
	Are derived architectural drivers used for designing PLA?
	Is the priority of driver right?
	Are variable architectural drivers defined based on adequate criteria?
Architectural Style	Are selected views satisfied with all architectural drivers?
	Are all of drivers applied into styles?
	Isn't a driver unnecessarily applied into several styles?
	Is the selected style efficient?
Instantiated Style	Do extracted components cover PL requirements?
	Are extracted components economic?
PLA	Is identified overlapped area right?
	Does the overlapped area resolve effectively?
Architectural Decision Model	Are all variations of drivers delegated to adequate variation point?
	Is all variants necessary?
	Are all variability propagations in overlapped area detected?
	Are all variation points and variants consistent through the artifacts?

Check Points in the check list are classified with artifact of each activity. Depending on activity goal, check points emphasize completeness, accuracy, efficiency, or conciseness. In architectural driver, check point focuses on completeness, accuracy of architectural driver for PL requirements. Check points of architectural style are efficiency of extracted styles for architectural driver, the point of instantiated style is completeness for quality attribute and functional requirements, and the point of PLA is its suitability for integrated style set. For ADM, completeness and efficiency are indicated. Based on the list, we may decide whether PLA design should be refined or finalized.

5 Concluding Remarks

The architecture of core asset should be generic to be applied to various products. Therefore, it is an essential element of core assets. We presented a reference model of PLA and proposed a systematic process having 5 activities. Each activity of the process was elaborated with detailed instructions and artifact templates. We also identified how the architectural variability is traced to elements of decision model.

We showed how architectural elements such as driver, views and styles can be applied to PLA. Using the proposed process, one can design a high quality PLA supporting architectural variability as well as architectural commonality.

References

[1] Bosch, J. Design and Use of Software Architectures, Addison-Wesley, 2000.
[2] Matinlassi, M., Niemela, E., and Dobrica, L., "Quality-driven architecture design and quality analysis method : A revolutionary initiation approach to a product line architecture," VTT Technical Research Center of Finland, ESPOO2002, 2002.
[3] Ceron, R., et. al., "Architectural Modeling in Product Family Context," *proceeding of EWAS, LNCS 3047*, Springer-Verlag Berlin Heidelberg, 2004.
[4] IEEE Recommended Practice for Architectural Description of Software-Intensive Systems (IEEE Standard P1471); IEEE Architecture Working Group (AWG); 2000.
[5] Thiel, S., and Hein, A., "Systematic Integration of Variability into Product Line Architecture Design," *proceeding of SPLC2, LNCS 2379*, Springer-Verlag Berlin Heidelberg, 2002.
[6] Clements, P., et al., Documenting Software Architectures Views and Beyond, Addison-Wesley, 2003.
[7] Woods, E., "Experiences Using Viewpoints for Information Systems Architecture: An Industrial Experience Report," *proceeding of EWSA 2004, LNCS 3047*, Springer-Verlag Berlin Heidelberg, 2004.
[8] Heineman, G., and council, W., *Component-Based Software Engineering*, Addison Wesley, 2001.
[9] Bass, L., Clements, P., Kazman, R., *Software Architecture in Practice*, Addison-Wesley, 2003.
[10] Sinnema, M., et al., "COVAMOF: A framework for Modeling Variability in Software Product Family," *proceeding of SPLC 2004, LNCS 3154*, Springer-Verlag Berlin Heidelberg, 2004.

[11] Kim, S., Chang, S., and Chang, C., "A Systematic Method to Instantiate Core Assets in Product Line Engineering," *Proceedings of APSEC 2004*, Nov. 2004.
[12] Kim, S., Her, J., and Chang, S., "A theoretical foundation of variability, in component-based development," Journal of Systems and Software, To Appear.
[13] America, P., et al., "Scenario-Based Decision Making for Architectural Variability in Product Families," *proceeding of SPLC 2004, LNCS 3154*, Springer-Verlag Berlin Heidelberg, 2004.
[14] Choi, S., Chang, S, and Kim, S., "A Systematic Methodology for Developing Component Frameworks," *LNCS 2984*, *Proceedings of the 7th FASE*, 2004.
[15] Lauesen, S., Software Requirements Styles and Techniques, Addison-Wesley, 2002.
[16] Garlan, D., Allen, R., Ockerbloom, J., "Exploiting Style in Architectural Design Environments," *Proceedings of SIGSOFT'94*, Foundations of Software Engineering, pp. 175-188, 1994.
[17] Kim, S., Chang, S., "A Systematic Method to Identify Component," *Proceedings of APSEC 2004*, Nov. 2004.