Concepts for a Product Line Knowledge Base & Variability

Roland Laqua

Fraunhofer Institute for Experimental Software Engineering (IESE),
Sauerwiesen 6, D-67661 Kaiserslautern, Germany
Tel.: +49 (0)6301 707 162
laqua@iese.fhg.de

Abstract. This paper presents the basic ideas of constructing a product line content & knowledge base on top of arbitrary configuration management system. Its need stems from the fact that evolving product lines arise a number of significant challenges, which can only be mastered with sound information base. The complexity of a product line engineering and its sheer number of additional artifacts and methods aggravates the intellectual control of a product line and its change over time. Management activities like product design, building of road maps or effort estimation become more difficult than for single product engineering and ask for more sophisticated information. We think that the problems can be solved with the explicit depiction of product line content & knowledge. Only then we can understand and control how a product is constituted from single software artifacts, what are their dependencies and relations, and how changes on shared assets affect the products in a product line. The product line content & knowledge base is a prerequisite for product line management services and processes, which makes it possible to manage the evolution of a product line and to control its consistency and completeness.

1 Introduction

Someone once said that, "a product line without a content & knowledge base is like a country without a map". However interesting the experience of driving from A to B without a map might be, there can be no doubt that when the goal is to arrive at a destination as quickly as possible, a kind of map must be.

Similarly, if you are looking at a particular product in a product line the knowledge how the product is derived and where it is located in the content base is indispensable. Unfortunately, we are facing the situation that up to now there is neither a concept nor a system that provides this kind of map that guarantees the intellectual control of the products and their adhering assets in a product line.

Perhaps it is not very surprising that in the early years of product line engineering a product line content & knowledge base was not a metaphor. Today the situation is quite different; the sheer number of product line methods & principles (for example [BFK+99], [WL99], [KCH+90]) in the last years makes accurate and advanced knowledge base techniques of major importance. One answer to this problem in the realm of transportation is GPS (Global positioning system), in the realm of product line engineering it is a product line content & knowledge base.

Up until now there is no equivalent to the traditional single product configuration description like a make-file in the world of product line engineering. True enough; people mark their assets in configuration management systems as parts of a certain product in a product line, but the resulting configuration remains firmly in the paradigm of single product development. The world of product line engineering is quite different, here the distinction between individual products as isolated artifacts vanishes and the requirement is for a product line content & knowledge base to span multiple products, to cover a vast pool of relations and dependencies between them, and to control their change over time.

The problem has been recognized in configuration management for several years, but the methodology used to address it – variant branching, module building, or family oriented configuration descriptions - have only solved parts of the problem. The main purpose of variant branching is to support maintenance activities like corrective changes for eliminating errors or perfective changes for improving the quality of a product. Misunderstanding of this concept is misusing variant branching for defining products and is intermixing unconsciously maintenance structures with product structures. The product and maintenance variants are enmeshed in a clew that violates the principle of separation of concerns and hence is very hard to oversee. Modules in configuration management systems can help structuring a product by encapsulating components. Unfortunately, many CM systems do neither relate the modules nor version the modules (examples the concurrent version system (CVS), source integrity from MKS). Ergo the impact of module changes cannot be calculated in space and time what makes it impossible to control the evolution of a product line. Family descriptions like for example PROTEUS [TGC95 (zeller)] that maps variabilities in PCL are static, of low abstraction, must be adapted manually to changes and do not reflect the dynamics of a product line.

That is why a new technology is called for, and product line content & knowledge management provides an approach that marries and extends the best of several worlds. It uses existing configuration management systems for content management, extends them for a product line knowledge base, and introduces a more abstract, top down and language independent description of product lines. It is our firm conviction that product line content & knowledge base will become indispensable for tomorrows product line engineering like maps for travelers. And once they have become omnipresent, they will represent the standard GPS of product line engineering.

In this paper, we explore and concentrate on the ingredients of a content & knowledge base for product line engineering. Section 2 shows the core ideas for a content & knowledge base and sketches the essentials for a product line management system on top of it. Section 3 presents the currently available technology that can be used to transform the ideas in a real product line oriented content & knowledge base. In section 4 an abstract metamodel for the product line knowledge structure is presented that will be extended by variability mechanism introduced in section 5. Section 6 continues key structure of the technical realization and major visualization techniques. Section 7 summarizes the benefits of the product line content & knowledge base and section 8 closes with a short outlook about the current status and future activities.

2 The Idea

The basic idea is to manage product lines and product complexity based on explicit Product Line Content & Knowledge. The essential steps and considerations of our approach can be described as follows:

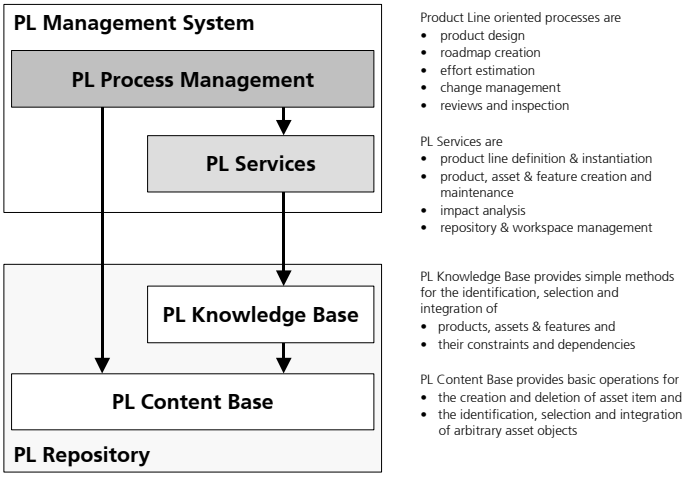


Fig. 1. Product Line Content & Knowledge Management System Layers

Product Line Repository (PLR). The product line repository consists of a formal content & knowledge model and a content & knowledge base that contain the instances of the model according to the formal model.

- **Product Content & Knowledge Model.** Content & knowledge are formalized in a model, which describes the structure, i.e. the constituting element types of a PL plus the knowledge about the internal structure of the elements and the relations/dependencies between them. A minimal model contains the basic PL-concepts: products, assets, features and decisions (see section 4). This model is customizable, i.e. it can be used to derive a company- or approach-specific PL-model like [WL99] or [BFK+99]. Although minimal it is sufficient to build up a PL-repository, which is independent from a given approach and can be used to derive a physical PL.
- **Product Line Content & Knowledge Base.** For a concrete product line the repository contains product instances according to the formal model. It consists of two parts, first the content base that is a physical storage for all product line artifacts, which are required to build a product, e.g. code, specifications, analysis & design documents. Second a product line knowledge base (PLKB) that contains the knowledge about the artifacts in the content base.

Product Line Management System (PLMS). The repository represents only a simple data-storage that provides primitive operations that deal with repository-items. The management of a PL requires more sophisticated services that support the consistent evolution of the PL, project management activities like cost- and effort estimations, or optimizations on the product line. We suggest to base these services on the primitive operations provided by the PLR and to implement them as a layer on top on

the PLR (see Fig. 1). Due to its focus on PL-management issues we call it product line management system (PLMS).

Implementation. We suggest implementing the PLR and PLMS as a "Product Line based extension of CM", i.e. on top of existing CM systems. The PLMS provides basic services for the definition and creation of a product line, the creation and maintenance of products, assets and features, plus impact analysis, and repository and workspace management. PL processes like effort estimation will be built up on top of them.

We would like to mention that the basic principle - managing PLs on the basis of explicit PL content & knowledge - is already being applied successfully in manufacturing. For example using a sales system, a customer can configure a PC in its components & features according to his requirements like CPU (feature: clock rate = 400MhZ), hard-drive (feature: capacity = 40 GB) and memory (features: capacity = 256MB, access time = 8,5 ns). This is only possible because the sales system has access to explicit PL line content & knowledge, so that it can tell that certain features are incompatible or what a single feature will cost. Now this kind of information is exactly what we also need in PL management. So why should we do without this successful principle - the important question is: can we ever manage PLs without it ?
However, we are absolutely aware that we can only transfer the basic ideas that are really applicable to software products. Actually, we will see in the further sections that the details of the approach are quite specific for software.

We have to mention again that this paper will concentrate on the product line content & knowledge base and does not explore the PL MS.

3 Ingredients of the Approach

We take reuse literally and therefore we identified all ingredients from state-of-the-art software-technologies & engineering that are required or helpful to make our approach work in detail. The main elements are:

Technical infrastructure. The technical infrastructure that is related to the PL content & knowledge repository are the CM systems that provide the basis for the repository and the integrated developer environments (IDE) that must be integrated in a product line management system

- **CM-systems** provide the basis for managing items in time. They offer the basic versioning services and hence form the bottom layer for the implementation of our approach where products and PLs are managed in time and space.
- **IDEs** are used to develop single products. In our approach they will keep this role without change. The only difference is that single products are instantiated with the help of the PLMS and the IDEs will access a product of a PL like a singe product.

Engineering techniques. Most essential for PL are the variability concepts. Our relevant engineering techniques for their integration in a PL content & knowledge base are feature logic and weaving technologies.

- **Features & Feature Logic** are fundamental concepts in PL-Engineering [CE00] that we will use throughout the PLR & PLMS. Features are user valued functions,

e.g. "validate the password of a user" [PF02] that characterize a product, and feature logic is used for assigning values to features and the definition how the different features are combined in a single product [ZEL97]. This indicates that feature logic plays an important role in product instantiation.

- **Variabilities & Variation Points** are PL-concepts that describe the variability in the PL and how it is resolved in the product-instantiation process [SVA00] [WG02].
- **Knowledge Representation** techniques are required for the formalization and modeling of the PL knowledge base. Actually there are already international standards available like the W3C standard "Topic Maps" that deal with this subject.
- **Weaving Techniques** are evolving techniques like frames [BAS97] or aspect-orientation [KHH+01] that are already successfully used to weave variabilities into software artifacts. We think that they can be used to solve specific details in our approach, e.g. the implementation of single-product instantiation.

Visualization infrastructure. We think that management issues and daily PL-activities can only be executed efficiently when they are supported visually. Hence visualization will play an important role in the realization of our approach. Visualization concepts describe which PL-elements, -processes or -relations need to be visualized and how they can be presented in a meaningful way. Also visualization technology (e.g. web technology) will be required for the realization.

4 Product Line Knowledge Base

The PL knowledge base realizes the fundamental concepts for visualizing a PL, controlling its evolution and navigating through it. Today neither CM systems nor IDEs are providing the necessary capabilities for navigating in a PL or analyzing commonalities and variabilities between the different products. The minimal set of structural elements for representing the PL knowledge we found so far are products, assets, asset elements, features and decisions. More complex structures like the product line models in PuLSE [BFK+99] or FAST [WL99] can be derived from these basic structures by typing and defining relationship between the assets.

Products. Products are the most fundamental elements of a product line. They are the uppermost elements in a top down view on it. Products are configured by a set of assets that can be shared between different products (see Fig. 2, Fig. 3).

Assets. Assets are self-contained software artifacts that can be versioned and are used for creation of products. They can exist in different structural variants different from the maintenance variants in the PL content base (Fig. 3).

Asset Elements. Asset elements are the structural elements, which constitute an asset. For example, a function definition inside of a java source code file represents an asset element. Asset elements will be used for integrating variability on finer granularity level in order to increase the flexibility and expressiveness of PL knowledge base but they are not managed explicitly inside of the content base (Fig. 3).

Features. Features are describing user as well as system functional requirements for asset and asset elements and are orthogonal to them because changing a feature does

not affect assets, asset elements nor relations between them and vice versa. Features will be assigned by tags or similar techniques to a certain asset or asset element (see Fig. 2)

Decisions. Decisions in the context of a PL knowledge base are fixing the functional requirements that have to be fulfilled for a certain product. That means they resolve the features that will be set for the product.

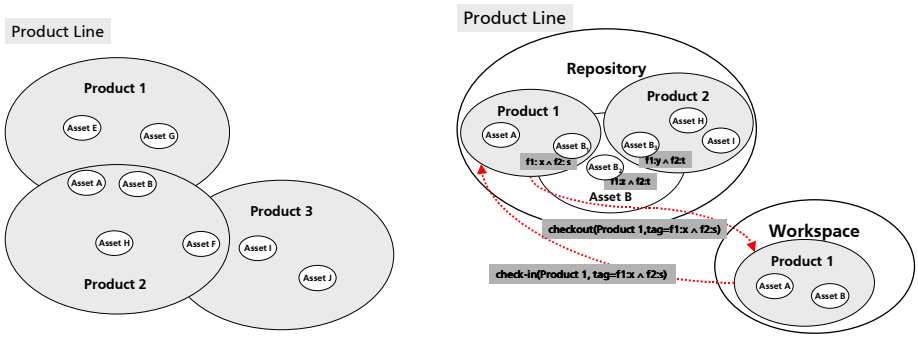


Fig. 2. Products, Assets & Features

A product line contains an arbitrary number of products that will be constructed by at least one asset. The asset can be contained on the uppermost level of the repository or in another asset of type container. Assets as well as asset elements can be related by an asset relation that is either contained directly in the product line or again inside of an asset of type container.

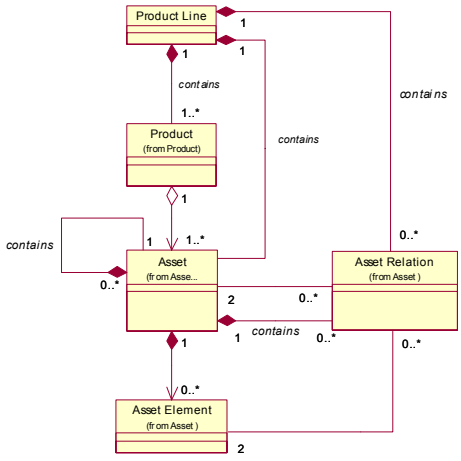


Fig. 3. Product Line Model

5 Product Line Variablity

Our PL knowledge base model (see Fig. 5) offers three different levels for integrating variability these are the product, asset, and asset element level. Variability on the product level just means that there can exist alternative products in the product line. In our knowledge model this case is out of scope, because it does not support reuse directly. On the asset level variability means that a certain asset can be shared between different products but not necessarily by all products. In the model (see Fig. 5) this is shown by the inheritance of an asset or asset relation from a variation point that indicates the variability. Exactly this is different from single product development where no information about variable parts is available. Beside the situation that an asset can be optional there can also exist alternative forms of an asset (see Fig. 4). Variability on the asset element level means that the internal structure is variable. In the structural model an asset containing variant asset elements becomes a generic asset by definition. Again variant asset elements are derived from variant point to indicate variability.

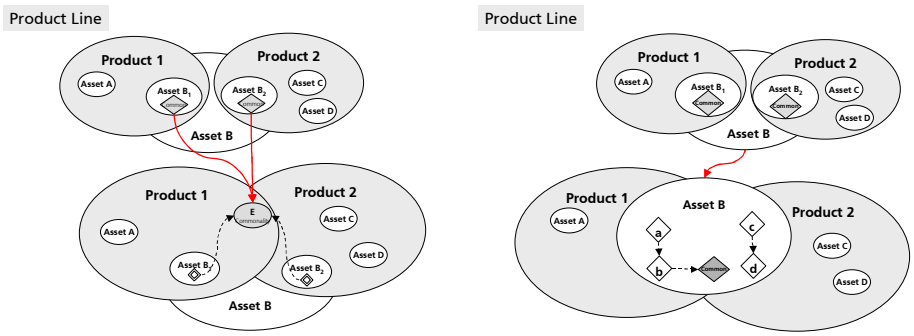


Fig. 4. Asset & Asset Element Variability

The basic elements for integrating variability in our PL knowledge base are assets, asset relation, generic assets, variation point and variant asset elements.

- **Variation Point.** A variation point is a place in the PL knowledge base where variability will be integrated in the products line. Variable elements like an asset, asset relation, variant asset element are inherited from variation point in order to indicate that this element can be optional, it can exist an alternative, or values must be set for resolving the variability. In principle we are providing three different variability types in our PL knowledge model namely options, alternatives and range.
- **Variant Asset Element.** A variant asset element is derived from asset element and variation point. Like the asset element the variant asset elements are the finest granular elements that can be managed by the PL knowledge base. They are provided for structuring the content of an asset in order to increase the flexibility and expressiveness of the PL knowledge base. In contrast to an asset the asset elements are not versionable and they reside always in a generic asset.
- **Generic Asset.** Generic assets are derived from assets and differ from them by containing variant asset elements. For creating a product the generic assets must be instantiated what means that the variant asset elements must be resolved.

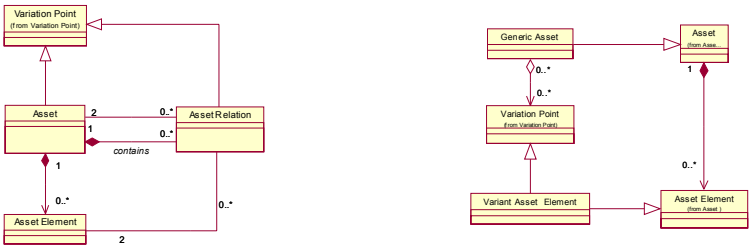


Fig. 5. Asset, Asset Relation & Asset Element Variability

The depicted PL knowledge model (Fig. 5) shows the variability on the asset and asset element level. The variability handling for asset and asset element is different. Whereas variant asset elements depict explicitly the variability on the asset element level., asset can always be variant elements because they are derived from variation point and hence there is no explicit model element for variant assets. The reason is that in a workspace assets are no longer variant elements.

Despite the pure existence of variant assets and asset elements we have to consider that variation can change over time. For example, variant asset can be transformed into generic assets or non-variant assets and vice versa (see Fig. 4).

Generic Asset/ Asset Transformation. Generic assets will often contain common parts that will be changed in one asset variant. These changes must be reintegrated manually in other asset variants in order to keep the variant asset consistent. Therefore, it is advisable to separate this common part in a new and commonly used asset. This decreases the effect of common change in variant assets, and hence increases the reusability.

Asset /Generic Asset Transformation. In some cases it is neither sensible nor possible to separate common parts in new assets. In that case a generic assets can help to indicate variant parts as variant asset elements. Now, like for the variant asset/asset transformation common part have to be changed only once and not for variant assets. Nevertheless, in contrast to variant asset/asset transformation there is major drawback, if a common part is changing, then the generic asset has to be re-instantiated for all products using it.

6 Benefits

The services that can be implemented based on our approach contribute to important issues in product-line evolution. We distinguish between management activities that do not affect the direct change of products and pure development activities for the modification and maintenance of products.

Management. The most relevant management activities that will be supported by our approach are project management activities, customer management and product design.

- **Project Management.** Project management parameters like estimations of effort, cost and time can be answered now quite simple. For example, we can easily identify important effort/cost drivers in our PL using an impact analysis (implemented in the PLMS) that tells us which features have a big change impact on the other assets in the PL. We think that "diving" into architectural details or - even more horrific - code details and doing estimations on that base is neither an efficient approach nor can it really answer the important questions.
- **Customer Management & Supply chain.** PL content & knowledge is now available for all participants in a customer oriented software production process. This has the potential to improve drastically the whole supply-chain between customer, marketing, project management, and development. For example, feature knowledge is now explicitly available for marketing and customers. When a customer requires a feature change, the related project effort is not a just a fantasy guess, but knowing the impact, it is estimated on a sound base. Hence the developers in the project have a realistic chance to realize the feature change within the planned effort. And - most important – the customer gets the product with the required change in the promised time.
- **Product design.** Our approach also helps to focus on designing a certain product like for example designing an automobile. Before a prototype or even the first release of a product will be manufactured, the product line management system virtually visualizes products and their constituting assets, features, decisions and relations between them in graphical view. This allows to analyze, evaluate and improve product characteristics in advance, before the product itself will be instantiated and certain assets will be modified or developed completely new.

Development. The method supports a top down as well as a bottom up view on a product line. The bottom up view shows the different assets and their versions that are provided by the configuration management system, whereas the top down view depicts the different products, features and decisions inside of the PL content & knowledge base.

- **Product Evolution.** All kinds of development activities, e.g. asset changes are supported by specific services, which are implemented in the PLMS. Using the impact analysis a developer can calculate the impact of a planned asset change. This provides him with the exact information which other assets (including products) are affected by this change. Depending on the kind of change (e.g. an interface) the PLMS can also resolve automatically part (if not all) of the required update operations in the asset base. This kind of support leads to a well-defined, consistent and controllable evolution of the PL.
- **Product Creation.** Our approach also supports the creation of new products. The creation process consists of such steps as deciding which assets of the PL a product consists of and which features and respective feature-values the product has. As all the required information is an integral part of our PLR, the instantiation process, i.e. building the physical single product, can be executed automatically. This kind of intuitive and hence easy product creation and instantiation by configuring a new product from an overall PL-asset base using explicit PL-content & knowledge corresponds to the configuration of manufactured products that we described earlier in the idea section.

7 Outlook

The depicted approach covers all artifacts in the PL, integrates existing PL concepts, focuses on the relevant processes and issues during PL evolution, and provides a logical and consistent set of layers plus services that support these processes.
We think that this is the first part but proper foundation for the realization of tools that support PL creation and evolution in practice. However, this is neither the end nor beginning of the end but the end of beginning to design a sound PL management system that starts with the construction of a sound PL content & knowledge base. There is still a lot of work to do when we look at the details, which have to be elaborated and cleared in each layer. Furthermore, we suspect that many questions will only arise at that point in time when such a tool is available and is used for the productive evolution of SPLs.

References

[BAS97] Basset, Paul G. Framing Software Reuse. Lessons learned from the Real World. Yourdon Press, 1997.

[BFK+99] J. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, and J.-M. Debaud. *PuLSE: A Methodology to Develop Software Product Line*s, in the Proceedings of the Symposium on Software Reuse (SSR'99), May 1999

[CE00] Czarnecki, C., Eisenecker U. Generative Programming: Methods, Tools and Applications. Addison-Wesley, 2000.

[EST95] Estublier, J., editor. Software Configuration Management: selected papers/ICSE SCM-4 and SCM-5 workshops, volume 1005 of Lecture Notes in Computer Science, Seattle, Washington, Springer-Verlag, 1995 .

[KCH+90] Kang, K., Cohen, S., Hess, J., Nowak, W., Perterson, S. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report, CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1990.

[KHH+01] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold W.G. An Overview of AspectJ. In: Object-Oriented Programming, 15th European Conference, ECOOP 2001 – Proceedings, pp. 327-353. Springer, 2001.

[PF02] Palmer, Steven R., Felsing, John M. A Practical Guide to Feature-Driven Development. Prentice Hall, 2002.

[SVA00] Svahnberg, M. Variability in Evolving Software Product Lines. PhD at the Univerity of Kalrskrona/Ronneby, Sweden, 2000.

[TGC95] Tryggeseth, E., Gulla, B., Conradi, R.. Modeling systems with variability using the PROTEUS configuration language. In Estublier [EST95], pages 216-240.

[WG02] Webber, D., Gomaa, H. Modeling Variability with the Variation Point Model. In: Software Reuse: Methods, Techniques, and Tools, 7th International Conference, ICSR-7 Proceedings, Springer, 2002.

[WL99] Weiss, D.M., Lai, C.T.R.. Software Product Line Engineering: A family-based software development process, Addison-Wesley, MA, 1999.

[ZEL97] Zeller, A. Configuration Management with Version Sets, A Unified Software Versioning Model and its Applications. PhD at the University of Braunschweig, Germany, 1997.