IJCSN International Journal of Computer Science and Network, Volume 4, Issue 1, February 2015
ISSN (Online) : 2277-5420 | www.IJCSN.org
Impact Factor: 0.417

Variabilities in Ontology-Driven and Rule-Based System for Management and Pricing of Family of Products

[1] I. P. Oladoja, [2] O. S. Adewale, [3] A. T. Adesuyi, [4] T.Fakoya

[1,2,3,4] Department of Computer Science, Federal University of Technology Akure, Nigeria.

Abstract - The increasing amount of variability in software systems led to a situation where the complexity of variability management becomes a primary concern during software development. Variability has been recognized as the key to systematic and successful reuse. In family-based approaches, like software product families, variability is a means to handle the inevitable differences amongst systems in the family while exploiting the commonalities and thus increases the reusability of software. The present research provided a model for creating variability of family of products of mobile phones, using Ontology Web Language semantic technology, to address the need of minimizing the cost of a new software development. The model developed allowed the reuse of components to assemble a new product. In this research, Phone X family of products was considered and, an ontology was built for it using Resource Description Framework schema. The variability design was implemented using ASP.Net and C# programming language. The result showed that the model was able to produce phones, each with different components options.

Keywords - *Variability, Ontology, Family of Products, Pricing, Mobile phones.*

1. Introduction

Designing and developing product families have been well recognized as an effective means to accommodate an increasing product variety across diverse market niches variety by reusing Many companies are investing in product family development practices in order to provide sufficient variety to the market while maintaining the economies of scale and scope within their manufacturing capabilities (Robertson and Ulrich, 1998).The family of products is characterized by two concepts namely variability and commonality. The variability gathers all the properties that are different within the members of the family while the commonality gathers all the properties that are true for all members of the family (Weiss and Lai, 1999). In fact variability asks to be identified, but mechanisms for the variability management have to be done. Even if the Family of products is a new paradigm, the management of variability is not a new problem and several techniques of conception and programming allows managing it. The particularity of the variability in the family of products is that the variability must be specified explicitly and it is part of the Family of products architecture. In the family of products the variability has two dimensions which are Time and Space (Bosch et al., 2001). The variation in time represents the variation of one proven elements in a firm's activities and offerings, product family design can offer a multitude of benefits including reduction in development risks and system complexity, improved ability to upgrade products, and enhanced flexibility and responsiveness of manufacturing processes (Sawhney, 1998). (Meyer and Utterback, 1993; Sundgren, 1999).

In addition to leveraging the cost of delivering product from one version to another while the variation in the space represents the variation between products of the same family. The variation points can be used to model different level of abstraction; for example the concept can be used to distinguish between product capabilities, the operating environment, domain technologies and implementation techniques. Prior research has addressed the various key issues related to variability identification and modeling. Feature-Oriented Domain Analysis (FODA) focuses on the systematic discovery and exploitation of commonality and variability in related software systems (Kang et al., 1999). FODA is primarily done to identify distinct features in the domain. These features depict common as well as different aspects of a domain. FODA uses the techniques of generalization/specialization, parameterization and aggregation to model variations. FODA is aimed at analyzing and modeling the various types of features required in the system. In this study, we aimed at minimizing the cost of a new software

development for mobile phone production via the use of the components of existing software to assemble new mobile phone products. All the products within a family are characterized by some common features but also by their differences (called variance); hence the need to define basic elements that make a product different from another, which from a customer perspective, could be some features or tools that are optional. Thus, this paper provides a model for creating variability of mobile phones products using ontology web language (OWL) semantic technology. The proposed system will be useful in product prototyping.

2. Related Studies

Kannan and Balasubramaniam (2003) worked on Ontology-based Support for Variability Management in Product and Service Families. They emphasized that Product/service family engineering, which encourages the development of a common product platform, plays a key role in facilitating large-scale and planned reuse in the development of customized products. They discovered that Customers prefer products and services that are specifically tailored to suit their needs, but are no more expensive than generic products. Product and service development firms are increasingly using family-based development approach to satisfy these conflicting requirements of distinctiveness of the products/services and at low costs. Their belief was that ontology can best be suited as solution for this problem. Their objective was the use of ontologies in variability management for software families. They developed an ontology that catalogues the different concepts associated with variability.

The ontology was used to define the elements characterizing the knowledge elements necessary for managing variability in product/service families. They also developed a knowledge management system (KMS) integrated with an ontology development tool to facilitate knowledge capture and retrieval for variability management. Methontology was the methodology used for their ontology development due to the suitability of the intermediate representations prescribed by the method to model variability, and also the usability and simplicity makes it suitable for use by designers of service families. The ontology developed can only be used for software families and cannot be reuse for other product/service families, and also does not support automatic inferences. Vincent and Darijus (2007) developed an Ontology-Centric approach for flexible configuration and pricing of product families. They stated in their research that mass customization is the new paradigm that replaces mass production, which is no longer suitable for today's turbulent markets, growing product variety, and opportunities for commerce. Their goal is to minimize the cost of a new software development, instead of developing the software from scratch; the components are reused to assemble a new product. They created ontology and rules-based system for variability management in product family development using semantic technologies. Semantic power of OWL was used to support the management and the pricing of product families. The pricing calculation uses the rules engine developed by Det NorstkeVeritas (DNV) Software. This approach enables the system to separate the product family architecture from its usage. Their approach has shown some shortcomings of using semantic technology. First the execution speed is low, which can be a problem in a system that interacts directly with components as DNV BRIX or DNV Rule Engine. In Asikainen (2005)the research was based on the methods for modeling the variability in software product families. Variability was define in his research as the ability of a system to be efficiently extended, changed, customized or configured for use in a particular context.

The goal of the research is to define a solid conceptual basis for modeling the variability in software product families, and to provide the concepts of formal semantics in such a way that reasoning on the models is possible using existing inference tools. Asikainen made emphasis that there is a growing demand for variability of software, and a significant research interest in the topic, as exemplified by the workshops and special issues devoted to it. He said further that products that incorporate variability are useful for various purposes: for example, such products can address multiple user segments, allow price categorization, support various hardware platforms and operating systems, offer different sets of features for different needs, and cover different market areas with different languages, legislation, and market structure. Addressing these concerns without variability would be very difficult, if not impossible.

In Mohsen Asadi, Dragan Gasevic, Yair Wand and Marek Hatala (2012) worked on Deriving Variability Pattern in Software Product Lines by Ontological Considerations. They stated that, Variability modeling is widely used in software product line engineering to support reusability. Specifically, it is used in the derivation of concrete software products from a reusable solution within a family of products and that to help manage variability, several modeling languages have been proposed for representing variability within a family of products. In their research, they investigated the use of ontological theories for theoretical analysis of variability modeling languages. An ontological theory defines constructs required for describing the structure and processes of the world in general. Therefore, in this paper, they analyzed *essential* variability of products in a software product line by investigating variability between real-world domains they

are intended to represent. More specifically, they employed concepts from Bunge's ontology as the theoretical framework for identifying possible variability patterns among products based on real-world domains variability. Afterward, they developed a theoretical framework for variability and apply the framework for evaluating expressiveness of variability languages. Based on their analysis, they concluded that feature models and Orthogonal Variability Model (OVM) have the same representational expressiveness for modeling conceptual variability but both languages has lack of variability completeness as they do not have any construct for representing order in g variability. Conclusively, majority of the reviewed work used UML to model their product, and UML lacks explication of relationship among component object of product. More so, their product makeup was done in a way they cannot be easily reused to make another similar product. Therefore, this paper provides a way that the development of a new product from similar products could be generated from the component features of existing family products.

3. System Design

In the present study, the family of products under consideration is Phones X. Phone X products have different types of phones and all these phones have some basic common features and differences. The differences in them are what we are considering as the variability in the products. The phone ontology is divided into two parts: components and style of phone. The components of a phone are further broken down into three groups: hardware, software and pattern. Note that, for example, the software group can be divided into operating system, operation support, protocol, application and so on. Phone style refers to the exterior description, such as shape, color and so on. Here we present the main terminologies of the ontologies associated with phone components. The concepts of phones will be divided into six parts, namely, model, hardware, software, shape, standard and color.The top-down approach is adopted in defining Classes under each concept, and sub-classes can be added until further expansion is not possible. For example, the shape of a mobile phone is divided into four sub-classes which are bar, slide, flip and rotary.

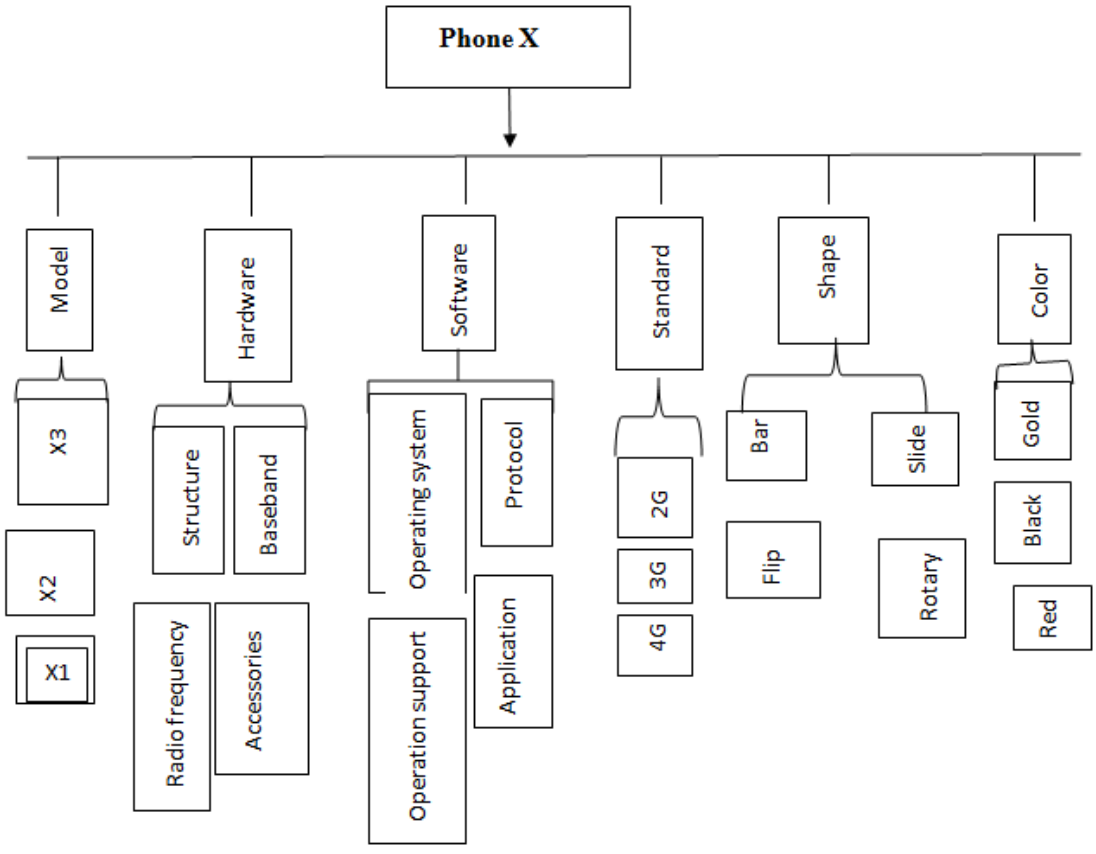

Figure 1: A simple hierarchical graph of Phone X ontology concepts

3.1 System Components

The system has two or more users and these users interact with the system using different components. The goal of this section is to present the different components and how each component fit all together. The system has four components:

i. Web site. The sales person interacts through the web site and it allows creating offers and generating a paper version of an offer. An offer contains a set of packages that have been priced.
ii. Ontology. The role of the ontology is to be the operational data for the web site. The web site uses the ontology as a database component. The ontology encodes all constraints of the system. The ontology is also used by the other components.
iii. Rules editor. This component is used to create the pricing rules for the packages.
iv. Packages management tools. This component aims to support the family of products engineering. The domain engineering is supported by an ontology editor whereas the product engineering is supported by a configurator tool. The ontology editor is to manage the family architecture (i.e. manage the ontology). The configurator enables to create, delete and edit packages. It uses the ontology as basis for all the operations it makes, but it also edits the ontology creating instances of class.

The product variability ontology was developed and the system was built around the product variability ontology (PVO). An administrator creates an operational data based on the product variability ontology. The product (Phones X) family is made up of family members such as X1, X2, X3, etc. The product has features (mandatory and optional) and prices which are represented in equation (3.1). Users will then interact with the web application component to perform analysis or query on the operational data in order to find a product and calculate its price while the package management tool will use the ontology. In the package management tools, the steps to be applied on the ontology to obtain a family architecture are encoded. The inherent constraints will be encoded in the ontology as follows:

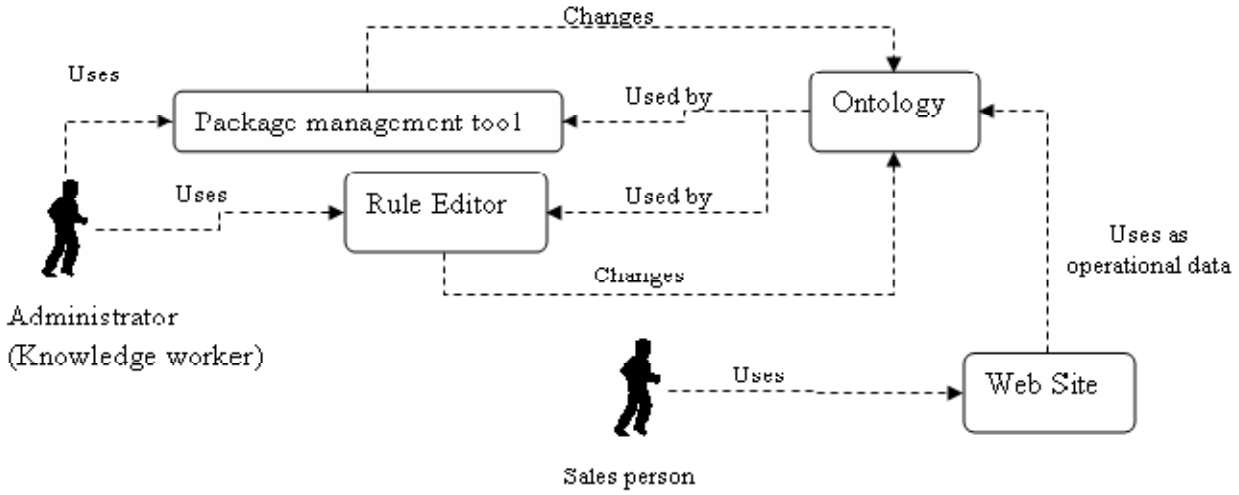

Figure 2: The Application's components use (Dedeban, 2007).

3.2 Products Composition Management

The management of the products composition includes:

i. Create a new product.
ii. Delete a product.
iii. Change a product.

3.2.1 Create a New Product

To create a product, we used the (configurator model by Vincent Dedeban, 2007) idea of a configurator using feature model. Configuration activities consist of making decisions about the desired product based on the products features. The configuration tool was used to ensure a consistent, complete and correct solution. In our approach the features, packages, and program (that will be called "Tool" in the rest of the document) are represented in the model below.

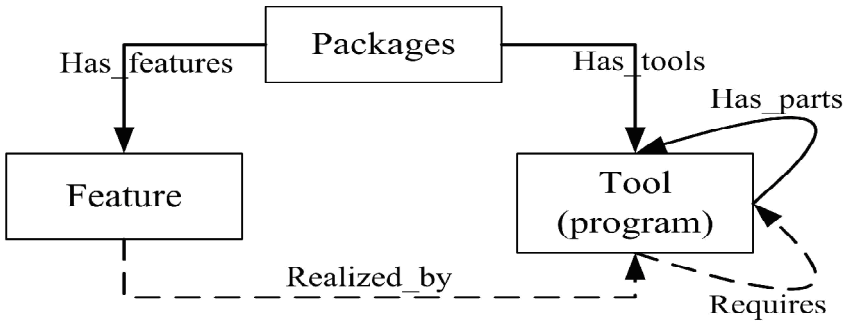

Figure 3: Configurator model

The realized by relation is a further bidirectional dependency that expresses that features are supported by Tools in an n-to m mapping: one feature can be supported by one or more artifacts or the other way round one or more features can be realized by one artifact. This relation is useful to check the model consistency. In fact the Tools that are in the composition of a product realize a set of features. This set has to be the same as the set of feature defined by the relations *Has_ Features*. In other words a product is described by features that are supported by the tools that compose the products.

3.2.2 Delete or Change an Existing Product

When deleting an existing product some offers that include the deleted product can become inconsistent with the data. It is the same problem when changing an existing product (Noy et al., 2004).

3.3 Feature Model

A feature model consists of a feature diagram and other associated information (such as rationale, constraints and dependency rules). A feature diagram (see figure4) provides a graphical tree-like notation that shows the hierarchical organization of features. Features are prominent and distinctive user visible characteristics of a system. Systems in a domain shared common features and also differ in certain features.

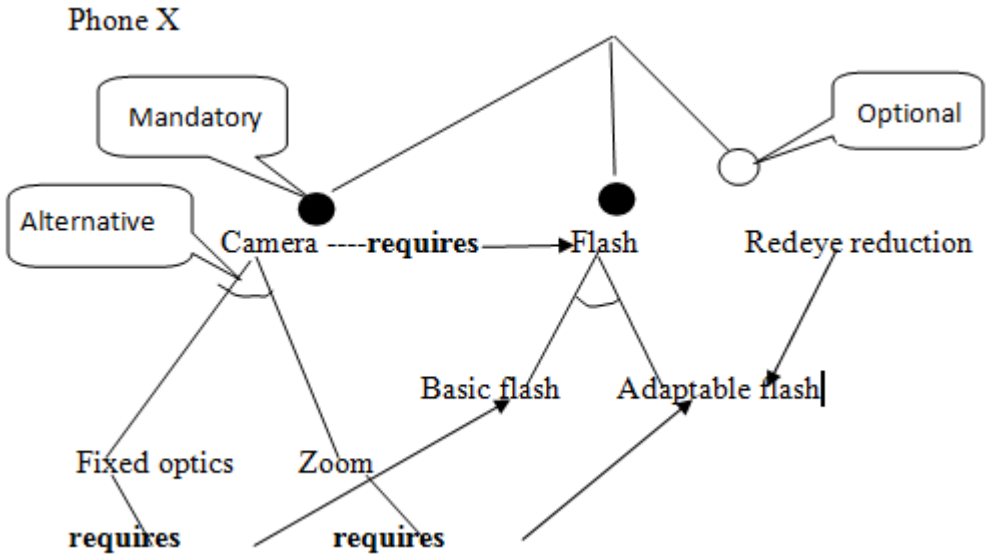

Figure 4: Feature model with constraints

3.3.1 Owl Representations of Features

We can use Ontology Web Language (OWL) to represent all the features (see the OWL codes below). For example, we can represent "F is a mandatory sub-feature of P" with the following OWL Description Logic (DL) axiom:

<owl: Class rdf: about="P"/>
<owl: on Property rdf: resource="#hasF">
<owl: someValuesFromrdf: resource="#F">
</owl: Restriction>
</owl: Class>

and represent"F is an optional sub-feature of P" with the following OWL DL axiom:
<owl: Class rdf: about="F"/>
<owl: Restriction>
<owl: on Property rdf: resource="#hasP">
<owl: someValuesFromrdf: resource="#P">
</owl: Restriction>
</owl: Class>

3.4 Application Model for Variability in Product Family

Let P_k represents the product family with members such as Phone X1, Phone X2, Phone X3, and Phone X4 with different features represented in Fe and their prices Pr given in equation (3.1).

$$P_k = \{Fe, Pr\} \quad 3.1$$

Products are made according to features. In order words combination of features makeup a product and each distinct feature has a particular price; this implies that the price of a complete whole product is the sum of the prices of the features that makes up the product. We can then say that a product's price is features based.

For any product P_k, there are mandatory features and optional features such that,

$$Fe = \{F_{mandatory} \cup F_{optional}\} \quad 3.2$$

Let $fe_i \subseteq F_{mandatory}$ *be specific mandatory features of the product*, $i = 1,2,3, \dots, n$

while $fe_j \subseteq F_{optional}$ *be optional features of the product*, $j = n+1, n+2, \dots, n+m$ *and* $Pr = Price$.

Therefore;

$$Price = \left(\sum_{i=1}^{n} C(fe_i, P_k) + \sum_{j=n+1}^{n+m} C(fe_j, P_k) \right) \forall fe_j \in \propto \; and \; fe_i \in \omega \quad 3.3$$

where ω is the compulsory set of features and $\propto \neq \{\emptyset\}$
where C is the function that returns the price or cost of the mandatory features.

Note: that, there exist at least one or more optional features in a particular brand of product in order to distinguish it from other brands.

4. Implementation

Rules express business knowledge. The rule system enables administrators to express their domain knowledge and store them in a rule base which is used by the website component. The rule system consists of an editor for editing rules and application interfaces as well as a runtime environment for rule execution. A rule is a conditional statement in a general form:
If <premise> then <consequent>

The rules are logically chained because predicate defined as the consequent in one rule is used as premise in another rule. The rule below define that if the Network type is 4G then the price A is 200 and the consequent CalcA is true. If a Rule uses CalcA in the premise then the premise of this rule will be considered true.
<rule>
<comment></comment>
<rulename>4G Network</rulename>
<vardec>
<varname>Network</varname>

```
<vartype>string</vartype>
</vardec>
<vardec>
<varname>a</varname>
<vartype>double</vartype>
</vardec>
<source></source>
<priority>0</priority>
<premise>
<preactions />
<atomic>?Network=="4G";</atomic>
<trueactions>
<action>?a:=200</action>
</trueactions>
<falseactions />
</premise>
<consequent>
<named>
<name>CalcA</name>
<argument>?network</argument>
<return>?a</return>
</named>
</consequent>
</rule>
```

5. Conclusion

This research work showed some direct benefits of adopting semantic web technologies to manage and price product families and this was made possible by representing feature models as OWL axioms.The consistency of the family architecture can be checked automatically and this enabled the configuration tool to create only consistent packages. The problem of redundancy among highly similar families was solved using rules to infer the best family taxonomy and using the inheritance mechanism. It seems that the semantic technologies really helped to handle the family of products concept by the family architectures consistency and by improving the coherence of the new product created by the product engineering phase. This research work has introduce variability concept into family of product; which satisfied the need to define components of a product that differentiated it from the other; which also enhanced centralization of pricing of family of product. This research also rendered ontological approach to variability of family of product on rule based, such that a product is made-up of components, and each component has a unique price. Therefore, the price of a product is the summation of prices of its make-up components. Conclusively, this work presented variability in price of family of product as a function of differences in components that make-up a particular product. It provided an easier method of producing product from an existing one and not from scratch.

References

[1] AsikainenTimo, (2006), Methods for modeling the variability in software product families. PhD Thesis, Helsinki university of Technology.

[2] Bosch et al, (2001), "Variability Issues in Software Product Lines," presented at Proceedings of the Fourth International Workshop on Product Family Engineering PFE-4, Bilbao, Spain.

[3] Clements.P. and Northrop.L. (2002), Software Product Lines: Practices and Patterns. Upper Saddle River, NJ: Addison-Wesley.

[4] Dedeban.V. and Strasunskas.D. (2007), An ontology-centric approach for flexible configuration and pricing of product families.

[5] Eisenecker.W.U and Czarnecki.(2000), Generative Programming: Methods, Tools, andApplications. ACM Press/Addison-Wesley Publishing Co, New York, 832p

[6] Jiao, J., & Tseng, M. M. (1999). A methodology of developing product family architecture for mass customization.*Journal of Intelligent Manufacturing, 10*(1), 3–20.

[7] Kang.K.C.,Cohon S. G., Hess J. A., Novak W. E., and A. S. Peterson, (1999) "Feature-Oriented Domain Analysis (FODA): Feasibility Study," Software Engineering Institute, Carnegie Mellon University, Pittsburgh CMU/SEI-90-TR-21.

[8] Kannan Mohan and Balasubramaniam Ramesh,(2003), Ontology-based Support for Variability Management in Product and Service Families,Proceedings of the 36th Hawaii International Conference on System Sciences.

[9] Meyer, M., &Utterback, J. (1993).The product family and the dynamics of core capability.*SloanManagement Review*, Springer 1993, 29–47.

[10] Mohsen .A., Dragan .G., Yair .W., and Marek .H., (2012), Deriving Variability Pattern in Software Product Lines by Ontological Considerations.

[11] Noy.N.F, and Musen.M.A,(2004), Ontology versioning in an ontology management framework. IEEE Intelligent Systems 19 (4), pages 6–13, [cited at p. 46]

[12] Pine.J. (1993), Mass Customization: *The new Frontier in Business Competition*. Boston, MA: Harvard Business School Press.

[13] Robertson, D., & Ulrich, K. (1998).Planning product platforms.Sloan Management Review*, 39*(4), 19–31

[14] Sawhney, M. S. (1998). Leveraged high-variety strategies: From portfolio thinking to platform thinking. *Journal of the Academy of Marketing Science, 26*(1), 54–61

[15] Sundgren, N. (1999). Introducing interface management in product family development.*Journal of Production Innovation Management, 16*(1), 40–51.

[16] Weiss M.D. and Robert Lai. C.T.(1999). Software Product-Line Engineering : A Family-Based Software Development Process. Addison-Wisley, [cited at p. 9]