7th INTERNATIONAL MULTIDISCIPLINARY CONFERENCE
Baia Mare, Romania, May 17-18, 2007
ISSN-1224-3264

MODERN CONCEPTS IN PRODUCT DATA MANAGEMENT

Adrian Petrovan[1], Cezar Toader[2], Cristinel Costea[3]

[1]Assistant, [2]Professor PhD, [3]Assistant,
North University of Baia Mare, 62/A Dr. V. Babes Str., RO-430083, Romania

Abstract: *The increasing importance of better customization of industrial products has led to development of configurable products. They allow companies to provide product families with a large number of variants. Description and management of a large product variety within a single data model is challenging and requires a solid conceptual basis. In this paper, we discuss the major concepts of configurable products taking into account both the evolution of the schema and the instances.*
Keywords: *Product Data Management (PDM), Configurable Products*

1. INTRODUCTION

Product families, product configuration and product data management are research areas with great deal of practical importance for traditional products, i.e., mechanical and electronic products. In software engineering, product families have also become an active area of research as the numbers of variants of a software product have in many occasions increased due to various requirements from different markets, hardware platforms, customer specific individualization, and so on.

A *configurable product,* or a *product family,* is such that each *product individual* is adapted to the *requirements* of a particular customer order on the basis of a predefined *configuration model,* which describes the set of legal *product variants*. One challenge is to find the correct concepts for modeling configurable products so that the descriptions can be kept up to date as the product evolves. The challenge has not been adequately answered by the research in product configuration, product modeling standards or commercial product data management (PDM) systems [1].

From the product modeler's viewpoint, modeling of configurable products involves the following levels:

- *The basic concepts* of the chosen data modeling method. For example, 'class', 'attribute', 'classification', 'inheritance' and so on.
- *The product modeling concepts* described using the basic concepts. Examples include 'product', 'has-part relation', 'optional part' and 'alternative parts' and conditions, such as 'incompatibility of components'.

- *Product family descriptions* formed using the product modeling concepts. These descriptions are also called *configuration models*.
- Finally, *product individuals* are instantiated according to the configuration model.

Firstly, configuration modeling does not fit nicely into the data modeling view for various reasons. Firstly, configuration models need new concepts, such as 'has-part relation with alternative parts', not available as traditional data modeling concepts. Secondly, as companies develop their products, product family descriptions are constantly changed. Product family descriptions, however, conceptually correspond to a traditional schema. In traditional database schema evolution, the existing data is typically converted to reflect the changes in the schema. Thirdly, individuals of configurable products have long lifetimes and histories of their own. More importantly, they do evolve independently of the schema since customers may change their product individuals as they please. This is a crucial difference to traditional databases since the evolution of individuals is not reflected in the schema and consequently, the individuals do not necessarily conform to the schema.

2. CONFIGURABLE TRADITIONAL PRODUCTS

Configurable products clearly separate between the process of designing a product family and the process of generating a product individual according to the product configuration model. This places configurable products in between mass products and one-of-a-kind products by enabling customer specific adaptation without losing all the economical benefits of mass-products. Knowledge based systems for configuration tasks, product *configurators,* have recently become an important application of artificial intelligence techniques for companies selling products adapted to customer needs [3].

The purpose of a configurator is to allow managing the configuration models and support the configuration task. Up to now, product configurator systems have been mostly implemented as commercial sales configurators, as parts of ERP systems or as company specific developed solutions. These configurators work well for products that exclusively consist of pre-designed components. However, in some cases, products consist of a mix of pre-designed, parametric and modifiable components.

Parametric components are pre-defined to some extent, but the key parameter values must be determined before they can be manufactured and assembled.

A modifiable component has pre-defined functionality, but the design is not solved in detail because of the large variations in customer demands when it comes to that part of product's functionality, e.g. inlet and outlet connector design for high pressure reducing

valves used in the process industry. In order to define a modifiable component, an engineering task must be done during the sales-delivery process.

The existing configurators provide limited support for such products. For this reason, the definition of configuration should be extended in order to support configuration of products containing pre-designed, parametric and modifiable components.

3. PRODUCT CONFIGURATION MODELING CONCEPTS

The difficulties in managing configuration models are widely recognized, the vast majority of the models ignore all aspects related to the evolution of configuration models and configurations.

Version is a concept for describing the evolution of products. Versions typically represent the evolution of a *generic object* that, in turn, has a set of versions. A generic object is sometimes also called a *generic version* or *generic instance*. There can be two kinds of *component references* to versions: *statically bound* and *dynamically bound*. A statically bound reference specifies explicitly a component and one of its versions. A dynamic component reference, i.e., a component reference to a generic object, can be bound to a specific version at various points in time. The idea is that in a given context, one of the versions from the set is a representative for the generic object, that is, the one to which dynamic references are bound. In the following, we utilize the version set approach, but for both the schema and the individuals in parallel.

Schema evolution is a relevant issue for databases. The approaches to schema evolution can be classified to *filtering, persistent screening* and *conversion*. In filtering, a database manager needs to provide filters so that an operation defined on a particular *type version* can be applied to any instance of any version of the type [2].

4. PDM PRODUCT CONFIGURATION CONCEPT

PDM systems were aimed at supporting all design engineering work, but have had weak support for product configuration. Improved capabilities for product configuration support have been introduced in most PDM systems.

During the product development process, the PDM system is used to manage product data in different phases. At the same time, PDM enables concurrent development of products and product configuration models. A PDM integrated product configurator further provides support for version management of product configuration models. It enables the user to re-use old product configuration models and retrieve old product variants. This is important for

maintenance and delivery of spare parts. The configuration logic stored in a PDM integrated product configurator is typically parameters, options and constraints (explicit and case tables). For simple product configuration models, there is no need of specialist programming skills. A graphic representation makes the configuration modeling task easy [2].

For more complex products a programming effort still needs to be done. That enables the system administrator to create more advanced product configuration models. It also supports integration with external applications such as parameter calculation programs, publishing programs, constant table sheets, visualization tools etc. This can be important for companies which already have assemble-to-order configurators but need further product customization during the sales-to-delivery process. In this case, a PDM based product configurator can first execute an external assemble-to order configurator and then proceed to a customization process for the automatically generated solution.

5. CONCLUSIONS

For traditional products, various methods exist that allow modeling product families essentially with the concepts with some approach-specific variations. These concepts for modeling the variety of traditional products seem to suit modeling software product families as well. Utilizing the concepts from traditional product families and adapting them for representing the architecture and variation of software product families, we believe, would lead to concise and manageable models. In addition, such models would open a way to using the AI methods developed in the field of product configuration to support the generation of product variants on the basis of the models.

Generally, it is rather safe to say that methods for modeling and management of variation of traditional products are more advanced than that of software product families. Nevertheless, two major areas were identified for transferring results from software product families to traditional configurable products.

REFERENCES

1. Mannisto, T., *A conceptual modeling approach to product families and their evolution*, Doctoral thesis, Helsinki University of Technology, 2000.
2. Mannisto, T., Peltonen, H., Sulonen, R., *View to product configuration knowledge modeling and evolution*. In Configuration-papers from the 1996 AAAI Fall Symposium, AAAI Press, 1996.
3. Tiihonen, J., Soininen, T., *Product Configurators – Information system Support for Configurable Products*, TAI Research Centre and Laboratory of information Processing Science, Product Data Management Group, Helsinki University of Technology, 1997.