Proceedings of IDETC/CIE 2005
ASME 2005 International Design Engineering Technical Conferences
& Computers and Information in Engineering Conference
September 24-28, 2005, Long Beach, California USA

DETC2005-84617

PRODUCT STRUCTURE MODELLING FOR THE MADE-TO-ORDER ENVIRONMENT

Qianfu Ni
Queensland University of Technology
2 George Street, Brisbane, Q4001,
Australia
q2.ni@qut.edu.au

Prasad KDV Yarlaggada
Queensland University of Technology
2 George Street, Brisbane, Q4001
Australia
y.prasad@qut.edu.au

Wen Feng Lu
Singapore-MIT Alliance Fellow
Manufacturing Technology
71 Nanyang Drive, Singapore
wflu@SIMTech.a-star.edu.sg

ABSTRACT

Product structure is the key information widely used by various business activities performed at different departments and different stages. In the made-to-order environment, product structure representation becomes more complicated because each product can have many variants with slightly different constitutions to fulfill different customer requirements. In such a context, product structure management comes to two interrelated functions: family structure management and variant structure management. At the same time, these two functions need to be seamlessly integrated to ensure the consistency of a family structure and its variant structure. From the business process perspective, throughout the entire product lifecycle, different business activities look at product structure with different purposes. Some activities are carried out based on variants and deem individual variants as different products and some need to be performed based on an entire family. As such, it is imperative to develop a product structure model that is capable of flexibly representing product families and product variants to serve up different processes in a product lifecycle. In this paper, a product structure model based on a master-variant pattern is proposed. The model can explicitly represent common characteristics of a family and particular characteristics of individual variants. Moreover, the variant structure representation is built on the top of the family structure representation. As a result, it provides an effective means to synchronize two types of structures. It also makes product family and variant concepts transparent to various business processes so that effective support can be provided to processes integration in the made-to-order environment.

INTRODUCTION

To provide customers with tailored products faster, better and cheaper, manufacturers have shifted their production mode to mass customization to take advantage of mass production for small batch-size production and start to organize the business operations from the process viewpoint to improve the enterprise-wide performance [1]. In the context of mass customization, a product initially consists of a common base and modularized functional subsystems to form a customization platform. The common base and commonly demanded functional subsystems can be made in the volume production for product configuration and other subsystems are made based on customer orders. As such, after the initial stage, the design activities are minimized to the determination of a configuration and design some special features for individual customers [2]. The fashion in which the manufacture is operated is called made-to-order.

As product is becoming one of the most significant assets in current enterprises to pursue competitive advantages, the effective management of product throughout the entire product lifecycle is becoming much more important than ever before. Product structure is a hierarchical tree representing the classification of parts that compose a product and the interrelationships of the parts. It is critical and widely used by various business processes [3]. At present, research in the area of product structure, namely product modeling, generally attends to structure and represent detailed data that is related to a single product, and many product structure models and associated management systems are specifically developed for the purpose of the design management. At present, models considering product family and capable of supporting the entire product lifecycle rarely exists [4]. However, it is obvious that mass customization has far-reaching influence on many organizational functions, such as sales and marketing, product engineering, and manufacturing [5]. Moreover, different functional departments in an enterprise may have different requirements to the product representation for different purposes. Some business activities should be carried out based on a family while others may only be applicable to individual variants. Therefore, a product structure model for such an environment is different from traditional product models in that product data has to be related to both a family of products and a particular product variant in a single context without data redundancy. For the integration purpose, various needs from the upstream and downstream of the product

lifecycle also have to be considered and supported. This paper presents a product structure model that integrates the family representation and variant representation to support different business processes in the product lifecycle

Nowadays, customization is the only method of tailoring an enterprise system for a particular company. Therefore, deployment cycle could be long and implementation cost could be very high. As enterprise systems are complex in nature, the failure rate of the implementation of enterprise systems is also quite high. These are the great barriers for enterprises, intending to adopt an enterprise system, to take action for deployment of the system. It is essential to offer enterprise systems with high flexibility and configurability so that rapid deployment can be achieved. To address this need, we focus our research on studying enterprise system flexibility and configurability and developing semantic model-driven enterprise system architecture. In this research, a system for business process integration in the made-to-order environment is developed to verify and demonstrate the efficiency of semantic model-driven method and architecture. The presented product structure model is one of key models embedded in the prototype system and plays a critical role of enabling the integration of business processes.

RELATED WORK REVIEW

Due to the importance and complexity, the design process management has received much attention in manufacturing companies. In the past decade, different Product Data Management (PDM) systems have been adopted by many large companies to manage the product design process [6] and, currently, the popularity of PDM systems is still steadily increasing [7]. One of the typical functions offered by PDM systems is to manage the product structure [8]. However, few available PDM systems are powerful enough to manage the product structure for mass customization because of the weakness of the product structure model at the representation of product family [9]. Apart from this, as the PDM framework is particularly defined for the design management, product structure models underneath the PDM systems lack of the capability to support the integration with other business processes, such as customer order management, planning and production, purchasing and inventory management [10, 11].

Some researches are reported on modeling the product structure to represent product family. Sudarsan [12] presented a product information modeling framework based on three models: Open Assembly Model (OAM), Design-Analysis Integration model (DAIM) and Product Family Evolution Model (PFEM). The PFEM model consists of three sub-models: family, evolution, and evolution rationale. Overall, the framework is developed with intention to: (1) capture product, design rationale, assembly, and tolerance information from the earliest conceptual design stage—where designers deal with the function and performance of products—to the full lifecycle; (2) facilitate the semantic interoperability of next-generation CAD/CAE/CAM systems; and (3) capture the evolution of products and product families. The main purpose of the PFEM model is for the evaluation of product families. It does not stress the effective representations of common characteristics of a family and particular characteristics of a variant. Jiao [5] reported a product structure model to represent product family in the mass customization context. The model consists of three views: functional view, technical view and structural view. The functional view focuses on the classification of diverse functional features of product portfolio for customer recognition. The technical view attends to represent building blocks to leverage the organization of the design repository. The structural view is to represent the topological structure of building blocks and configuration rules to guide the end product configuration. Our understanding is that the model are helpful for companies to shift from the individual product development to family based design by providing a systematic method for establishing building block repository and configuration rules. It lacks of the capability to support the design process management in the made-to-order-environment. Fujita [13] proposed a product structure representation by decomposing a product into different subsystems. By employing entity relationships to represent the topological structure of subsystems and attributes to represent the association possibilities of subsystems, the model puts its focus on maximizing product varieties using minimum building blocks to achieve optimized a customization platform. Zhang [2] presented a general data model to effectively represent the product structure by considering product families, and a behavior model to represent functional requirements of product structure. The model attends to maximize information sharing among different stakeholders, such as customers and supplier. The model aggregates family representation and variant representation together. Therefore, the model can not effectively support process integration because business activities and relationship between product and other information entities need to clearly differentiate families and variants. Janitza [9] also proposed a product model for mass customization by incorporating product decomposition and part specification into one model. This model focuses on providing a highly flexible product model specification for the product designer and simpler configuration for the customer. The family representation and variant representation has not received enough attention and the synchronization of two representations is not addressed.

This paper develops a product structure model that offers a clear boundary between family representation and variant representation. At the same time, the model is capable of effectively synchronize a family structure and its variant structures. The model also considers the needs of process integration.

REQUIREMENTS ANALYSIS

In a made-to-order environment, the product must be customizable [14]. Ultimately, product variants are realized by part variants that can be achieved in different ways, such as parameter variant, material variant and connection mechanism variant [2]. From the structural viewpoint, a product variant can be customized by:

- Add-in subsystems. Additional features are embedded into a standalone subsystem. The subsystems can be added into the initial product when required by customers [15]. For example, the anti-lock braking function is often provided by an independent subsystem that can be installed into cars for individual customers.
- Alternative subsystems. A function can be provided by different optional subsystems with different features

[2]. For instance, the audio functions can be provided by a cassette player with a built-in radio receiver by default. The default audio subsystem can be replaced by a CD player or a video player with a radio receiving function if requested by customers;

- Removable subsystems, a functional subsystem can be removed from common configuration for customers with limited budget or other rquirements. For example, a camera is usually sold with a lens. However, if requested, customers can only buy a camera body, especially in the professional camera market.

A product model for product families should be capable of representing product family structures, variant structures and constituent components, including part variants and subassembly variants. At the same time, the family structure should be able to serve as a template to constrain the definition of variant structures so that the consistency of a family structure and its variant structure can be synchronized.

In addition to design process, other business processes throughout the product lifecycle also utilize the product structure [3, 8]. In different processes, people with different concerns may look at the product structure from different prospective. For example, customers need information about optional configurations and corresponding prices. Internally, designers need to overview the family spectrum to configure products based on customer requirements or to enhance the customizability while process planners need variant structures to work out process plans. Therefore, the product model should provide the flexibility to transparently represent the product family and variants in different ways.

For the process integration purpose, differentiation of product family and variant representations are critical. Throughout the product lifecycle, instances of different product families, part families, product variants and part variants are accumulatively created, manipulated and associated by various business activities to support or drive continuous activities [16]. Product structure model and other business class models act as a blueprint for managing data consistency as well as integrity, and governing the evolution of associations between business objects. To achieve the success of process integration, it is essential to provide an explicit family and variant representations because activities in business processes may only be eligible to families or variants. The associations between product and other business objects may only be able to be established based on families or variants. Different stakeholders with different purposes are only interested in families or variants.

An effective product structure model should be able to leverage the existing industrial practice. For example, in general, all variants in a family share a unique family id and name and each variant may also have a secondary unique identity and name. Apart from id and name, values of some attributes also can keep constant for all variants to represent family features while other attribute values can be different from variant to variant. Therefore, a mechanism is needed to easily maintain the consistency of common attributes and enable uncommon attributes to have different values to characterize individual variants.

MASTER-VARIANT PATTERN

To make the model capable of effectively representing the common features of a family and special features of different variants, a master-variant pattern, as shown in Figure 1, is adopted as a fundamental technique for establishing the product structure model. In the model, the interfaces *IMaster* and *IVariant* are modeled to represent common properties and behaviors that all families and variants should have respectively. The interface *IMVLink* defines common properties and behaviors of associations between masters and variants. The classes *Master*, *Variant*, and *MVLink* are default implementations of the three interfaces respectively. The classes that inherit the class *Master* or directly implement the interface *IMaster* are enforced to comply with the principles defined by the master-variant pattern. The cardinalities of the association between *IMaster* and *IVariant* imply that one master can have one or unlimited variants and a variant should have and only can have one master. In other words, a master and its variants exist interdependently. Though it can have multiple variants, a master can not exist without a variant. In turn, a variant also can not exist without a master. It has to be pointed out that attributes common to all variants should be defined in master classes, which are the classes that directly or indirectly implement the interface *IMaster*. Uncommon attributes should be modeled in variant classes, referred to as the classes which directly or indirectly implement the interface *IVariant*. In this pattern, *IMaster* is an abstract for grouping variants of a family and represents the common characteristics while *IVariant* represents the special characteristics of individual variants.

In the model, the attribute *handle* in each class is defined for a system to internally manage associations and references. The attributes *id* and *name* are defined to uniquely identify individual families. The attribute *version* is used to differentiate variants in a family. The model implies that all variants can share the same id and name. Meanwhile, it also allows each variant to have a special name by defining the attribute *variantName* in the class *Variant*. The attribute *version* in the class *Variant* enables to manage each variant as a version, which is the typical practice in the made-to-order environment.

The master-variant pattern offers three main advantages: 1) it provides a clear boundary between the family representation and the variant representation. At the same time, it offers the capability to maintain the data integrity; 2) it is capable of representing common characteristics of families and specific characteristics of individual variants; and 3) it can flexibly meet different requirements of different business processes. Masters or variants can be explicitly used as inputs to a business process. Associated entities can be explicitly linked to masters or variants. For example, process plans should be linked to variants while assembly specifications of a functional subsystem applicable to all variants of a family should be attached to the master.

PRODUCT STRUCTURE MODEL

Based on n the master-variant pattern, the product structure model shown in Figure 2 is developed. In addition to an essential function of representing product composition and interrelationships among parts and subassemblies, the model is

different from a traditional one in that the family concept is incorporated. As shown in the model, product, part and subassembly are represented by three groups of classes respectively: *Product*, *ProductVariant*, and *ProductMVLink*, *Part*, *PartVariant*, and *PartMVLink* as well as *Subassembly*, *SubassemblyVariant* and *SubassemblyMVLink*. The model introduces the family concept to product, part and subassembly because product, part and subassembly are represented based on the master-variant pattern. The classes *Product*, *Part* and *Subassembly* represents product families, part families and subassembly families respectively while the classes *ProductVariant*, *PartVariant* and *SubassemblyVariant* represent product variants, part variants and subassembly variants. A special case where a master only has one variant linked indicates that the product, part or subassembly is a normal product, part or subassembly.

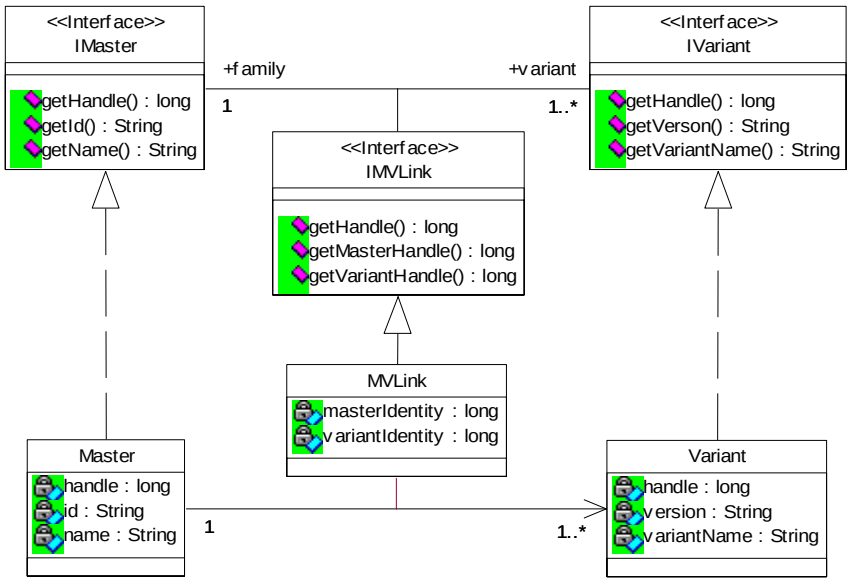

Figure 1: Master-Variant Pattern

Family Structure

For clarity, the family structure model is taken out from Figure 2 and shown in Figure 3. In the model, aggregation associations between *Product* and *Part*, *Subassembly* as well as *StandardPart* imply that a product can consist of non-standard parts, subassemblies and standard parts. Besides, *Subassembly* has aggregation associations with itself, *Part* and *StandardPart*. These associations indicate that a subassembly can constitute other subassemblies, non standard parts and standard parts. Therefore, in a whole, a product can constitute parts and subassemblies that are organized a hierarchical tree structure. As shown in Figure 2, *Part* and *Subassembly* are master classes as they implement the interface *IMaster*. According to the master-variant pattern, a master class represents a family rather than a concrete product. Hence, the model shown in Figure 3 only reflects what part families, subassembly families and standard parts are involved in a product family and where the part families, subassembly families and standard parts are positioned in the hierarchical structure. It does not manage concrete information about which variants in part families and subassembly families are involved in a product variant. However, based on the master-variant link, all part variants and subassembly variants are clearly reflected. Therefore, the family model provides an overall view of a product family about product variants and all optional part variants and subassembly variants. The overview is named a product family spectrum [10].

Figure 4 shows the sample spectrum view of a simplified car family based on the developed model. The family structure shows that a car family, represented by *Car:Product* (*Car: Product* is a UML notation denoting that the object named Car is an instance of the class *Product*), can consist of an audio subsystem, represented by *Audio:Subassembly*, and an engine, represented by *Engine:Part* (assume that a engine is a component). Further, a audio subassembly consists of a radio subsystem, represented by *Radio:StandardPart*, and a media player, represented by *MediaPlayer:Subassembly*. Based on the master-variant link, the spectrum can provide information about what variants that the audio family, engine family and media player family have respectively. It indicates that three types of engines with different rated powers and three types of audio subsystems, which are cassette player, CD player and video player, are available for selection.

The spectrum can effectively assist designers to configure products for customers, amend design to reorganize existing functions into configurable subsystems, design new alternative subsystems, or innovate new functional subsystems to enhance customizability of a family. It also provides helpful information for customers to configure products during the preparation of orders.

Variant Structure

In addition to the family structure management, the variant structure management is a basic requirement to a product structure model for mass customization as well. A variant structure clearly reflects what part variants and subassembly variants are used to form a particular product variant. At the same time, the model should be capable of enforcing the consistency of the family structure and variant

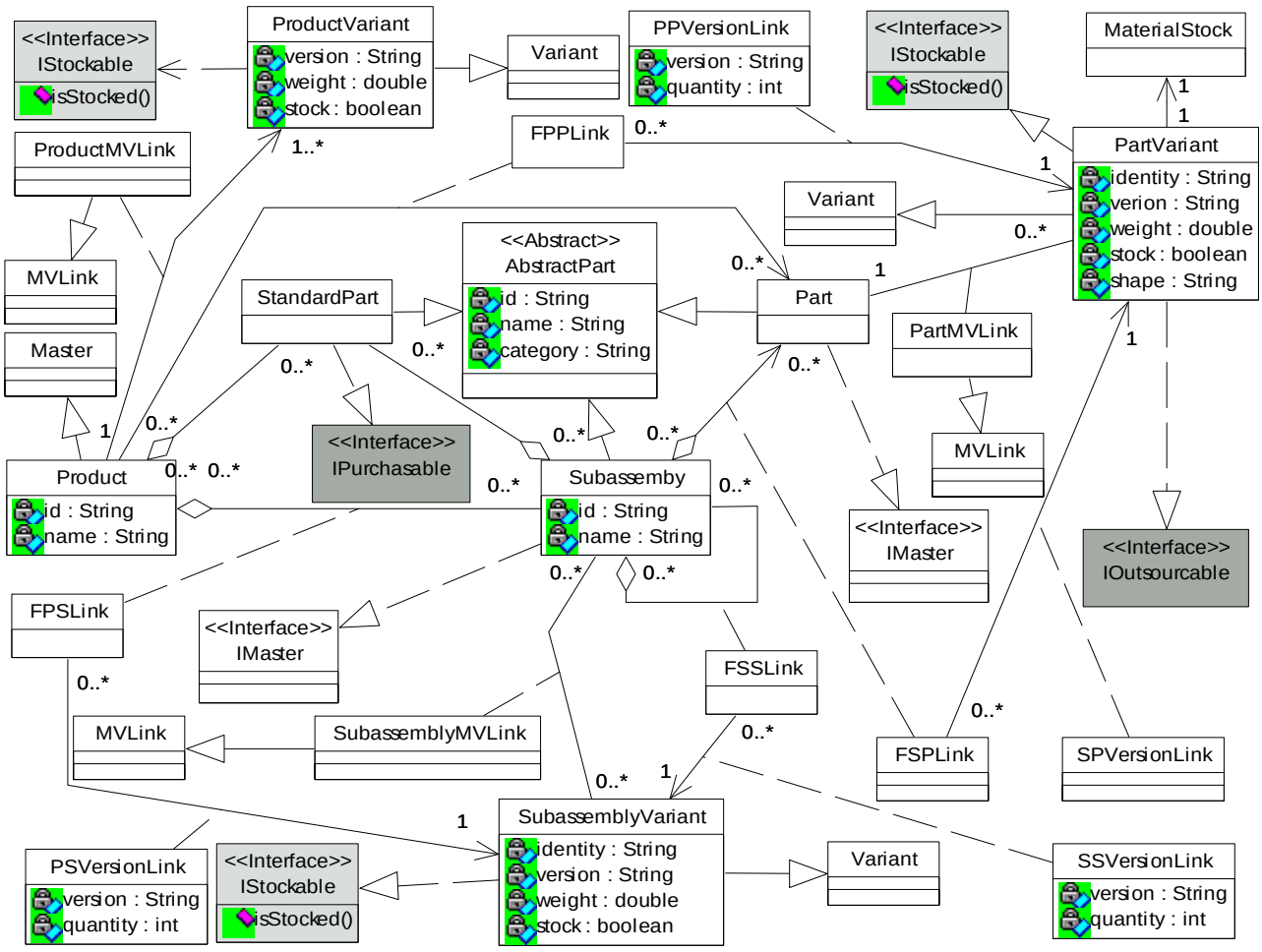

Figure 2: Product structure model

structures. To achieve this goal, the variant structure model is built on the top of the family structure model. As shown in Figure 2, *FPPLink* and *FPSLink* respectively represent associations of a product family with a part family and a subassembly family, and *FSSLink* represents association of a subassembly family with other subassembly families. To further represent variant structures, three association classes, i.e. *PPVersionLink*, *PSVersionLink* and *SSVersionLink*, are defined to associate *FPPLink* with *PartVaraint*, *FPSLink* with *SubassemblyVariant* and *FSSLink* with *SubassemblyVariant*. *PPVersionLink*, *PSVersionLink* and *SSVersionLink* are called version links. A key attribute in the version links is *version*. The value of this attribute indicates which product variant or subassembly variant the associated variant is used for.

To explain the variant structure model, the relationships between a car variant and engine variants are taken as an example. As shown in Figure 5, the car family has three variants, i.e. *CarA*, *CarB* and *CarC*, and the engine family also has three variants, which are *Engine1.8*, *Engine2.0* and *Engine2.2*. Based on the family structure model discussed above, *Car* and *Engine* is associated through *CarEngineLink*, which is an instance of *FPPLink* (the association class of the product family and the part family). As mentioned above, this association is not capable of providing information about which engine variant is used for *CarA*, *CarB* and *CarC* respectively. Based on the family structure model, to reflect the associations between the engine variants and the car variants, three version link instances are introduced, i.e. *EngineVersionLink1*, *EngineVersionLink2* and *EngineVersionLink3* to associate *Engine1.8*, *Engine2.0* and *Engine2.2* with *CarEngineLink* respectively. The attribute *version* in the version link classes plays the role of defining which car variant each associated engine variant is used for. It can be seen from Figure 5 that *Engine1.8* is used for *CarA* as the value of the attribute *version* of *EngineVersionLink1* is *CAR.A*, which is same as that of the attribute *version* of *CarA*. In addition, multiple associations between *CarEngineLink* and an engine variant imply that the engine variant is used by multiple car variants.

Compared to the variant structure model that directly associates variants of different part families and subassembly family with product variants, a significant advantage of this model is that the family structure model and the variant structure model are integrated. As a result, product variant structures can be well controlled by the corresponding product family structure. For example, in Figure 4, if the engine family was not associated with the car family, *CarEngineLink* will

not exist. Consequently, no engine variants could be associated with any product variants. Therefore, this model is capable of synchronizing the family structure and the variant structures. This feature is very significant in case that multiple families are managed in one company and there exist multiple subsystems that provide same functions and are not exchangeable to the families. For instance, two engine families are maintained for the two car families respectively without exchangeability. During the product configuration, this model can effectively prevent from selecting incompatible variants based on the family structure. Another advantage of the model is that product variants and part variants can have their own version naming conventions. It is capable of managing the part families shared by multiple product families.

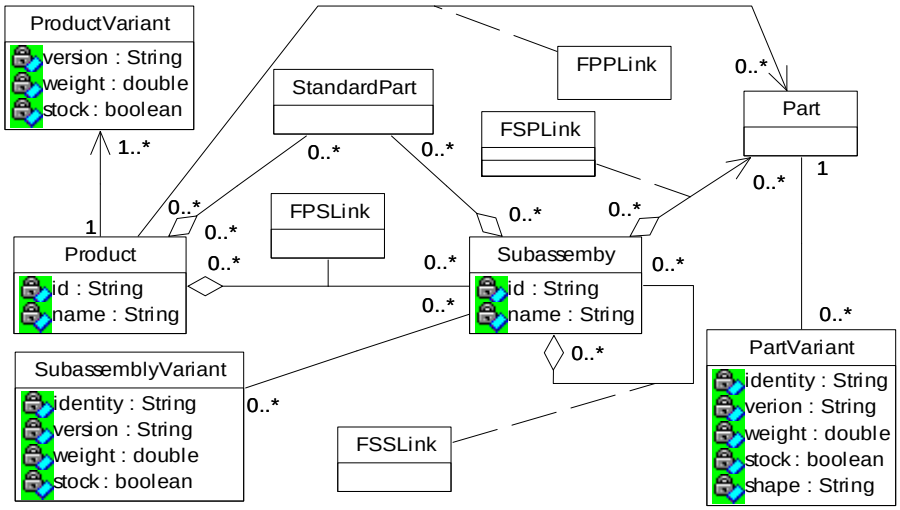

Figure 3: Family structure model

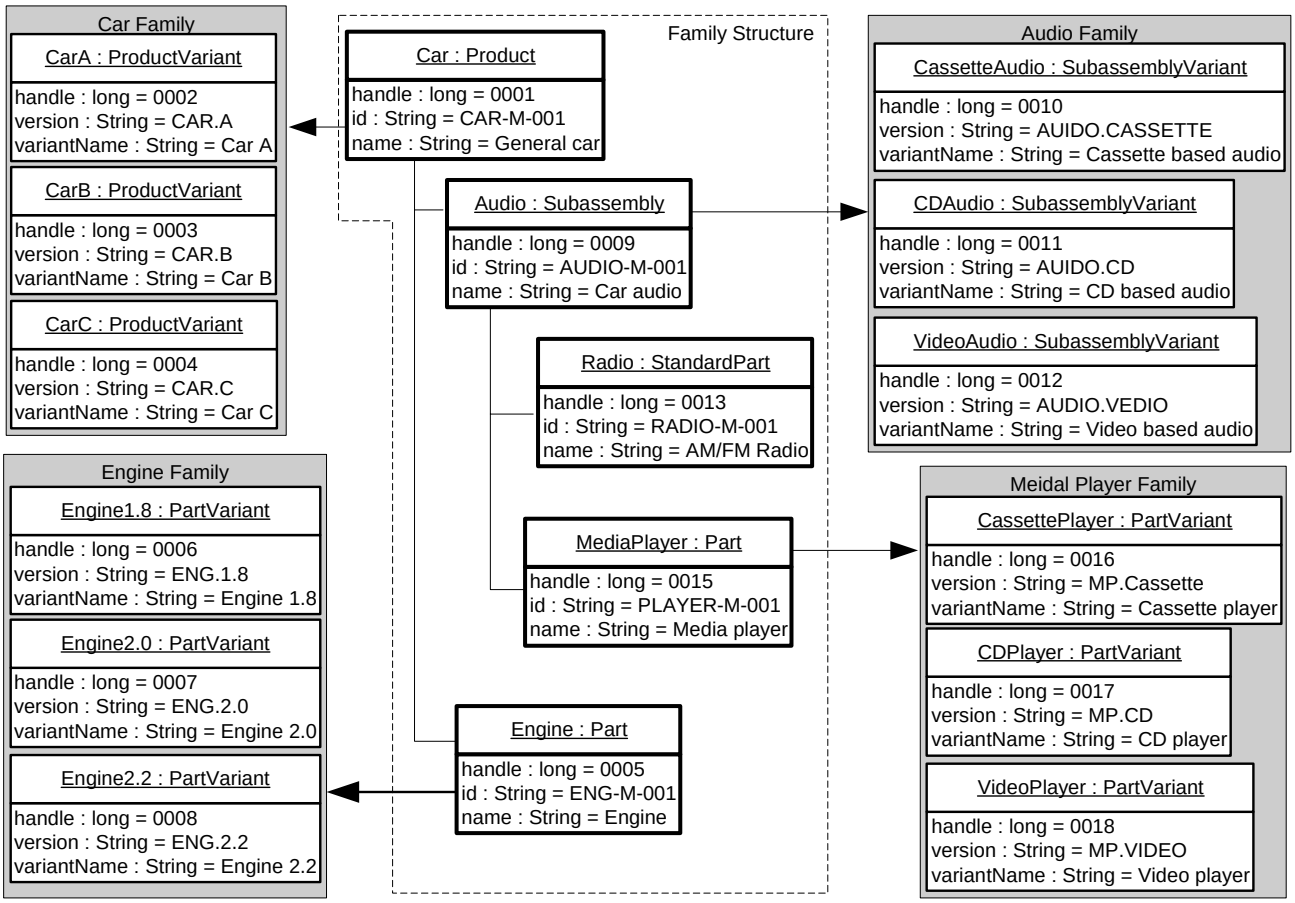

Figure 4: A simplified car family spectrum

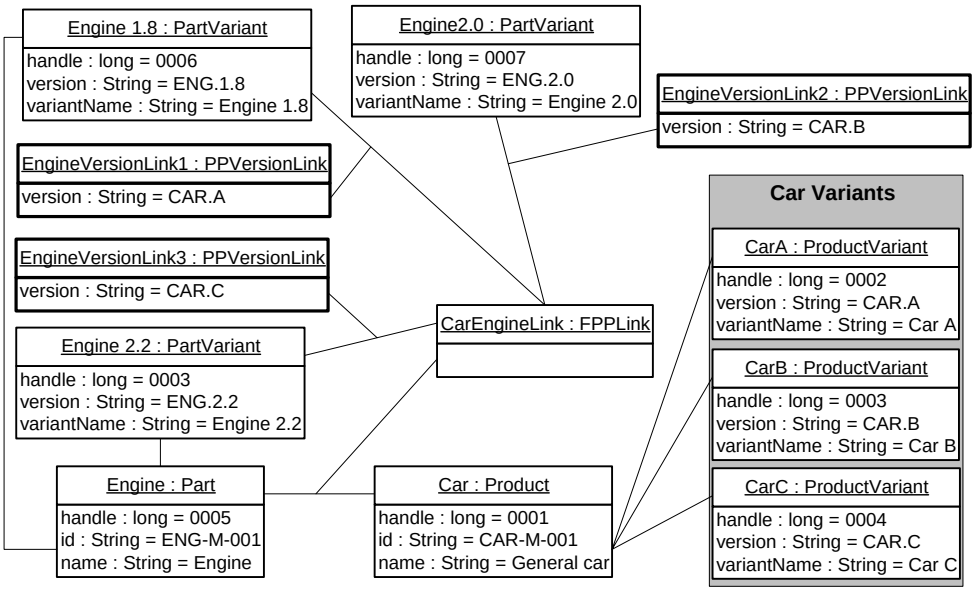

Figure 5: A simplified car family spectrum

BUSINESS PROCESS INTEGRATION SUPPORT

As shown in Figure 2, the model differentiates standard parts, represented by *StandardPart*, and non-standard parts, represented by *Part*. *Part* implements *IMaster* and is further associated with *PartVariant*, a subclass of *Variant*, according the master-variant pattern to represent part variants. There are two main reasons for differentiating non-standard parts and standard parts: 1) the family concept is inapplicable to standard parts; 2) the processes that non-standard parts go through are different from those for standard parts. Standard parts are purchased from the market and managed in the inventory. However, the non-standard parts may go through various processes, such as the production process if they are made internally or the outsourcing process if they are made by partners, and the inventory management process if they are made to stock.

The interfaces *IStockable*, *IPurchasable* and *IOutsourcable* are modeled to enforce the implementing classes to comply with the processing rules of stock management, purchasing management and outsourcing management. The implementation of the interface *IStockable* by variant classes, i.e. *ProductVariant*, *PartVariant* and *SubassemblyVariant*, implies: 1) common parts, subassemblies and even products are allowed to be made to stock; and 2) it enables made-to-order and made-to-stock decision to be made at the variant level, which means that, in a part or subassembly family, the variants commonly demanded can be made to stock while the ones only demanded by a few of customers are particularly made when ordered. It can be seen that the model is capable of leverage the main objective of the mass customization to take advantage of the volume production to deliver tailored products for customers through configuration.

Figure 6 shows the material model associated with *Part*. *PartMaterial* is modeled to represent the chemical and physical properties while *Shape* is defined to describe the shapes and sizes according to the standards in the market. The association of *PartMaterial* and *Shape* represents material stocks that can be purchased from the market and stocked in the inventory. By associating materials to part variants, the model enables part variants to be derived by using different materials. It also leverages the centralized resource management and the integration between design and the resource management.

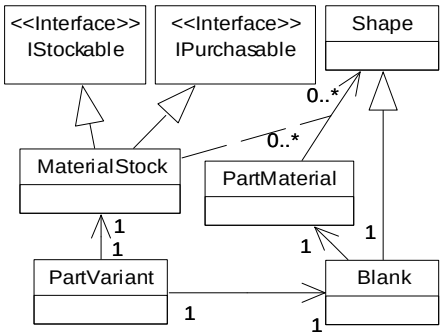

Figure 6: Integration with other processes

PROTOTYPE

A prototype system, named Collaborative Business Solution (CBS), has been developed to demonstrate the efficiency of the developed model. The CBS attends to provide functions for project management, product configuration management and inventory management to support the business activities in the made-to-order companies. Multi-tier architecture shown in Figure 7 is developed based on the web technology, which provides a light-weight and operating system independent platform for users to search, browse, retrieve and manipulate information disseminated and shared remotely [18]. The apache web server (version 2.0.8) is adopted as the web server and Tomcat (version 4.1) is selected as the JSP (Java ServerPage) engine. The kernel in the architecture is the CBS server, which is organized into two parts: system services and application services. The system services provide functions for security management and system administration.

The applications services are further organized into three layers: foundation layer, functional layer and domain layer.

On the foundation layer, the entity service is responsible for managing information entities by considering the data integrity. According to the principle of the master-variant pattern, a master at least has one variant associated and no variant can exist without a master. Therefore, the service provides the capability of ensuring the information integrity based on the following rules:

1) When initializing a new family, the service automatically instantiates a master and a variant, and associates them together;
2) When adding a new variant, the service ensures that a new variant is associated with an existing master and prevents a redundant master from being created;
3) When removing the last variant of master, the master is removed automatically.

The relationship service provides functions for managing entity relationships and navigating information based on relationships. In the relationship management, the service figures out relationships should exist between masters or variants and ensures appropriate entities are associated. The persistence service acts as a gateway of accessing database. While storing information entities, it maps information entities to corresponding tables. While retrieving information from database, it gathers information from database and converts the information to appropriate objects. The three foundation services works together and attends to make the master-variant concept transparent to other services.

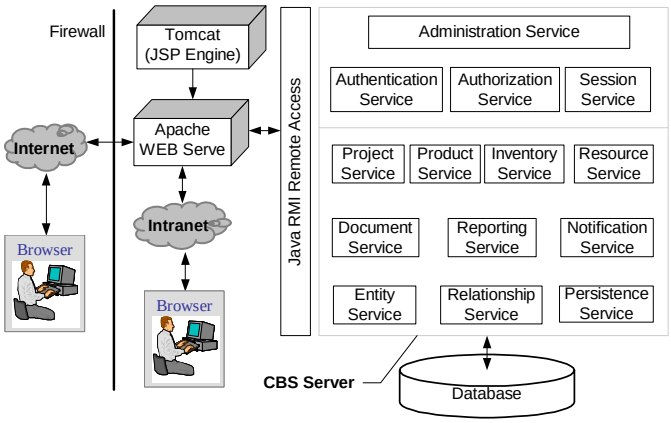

Figure 7: Architecture of the prototype system

Built on the top of foundation layer, the functional layer provides particular functionality for managing documents and generating reports. The document service is responsible for associating various documents, such as engineering drawings and product specifications, with products and parts. The main function of the service is to wraps documents as binary objects. Then it requests the relationship service to associate the document with an object – a document owner. Reports are a special type of documents. They provide an effective way for information exchange and analysis. This service provides a template based method to generate various reports. In this method, report templates are used to define the look and feels of reports, such border, cell font and alignment. A series of configuration files are employed to define information to be rendered into reports and positions of each piece of information. A report generator gathers manipulate information according to the configurations, and finally incorporates the information into a report based on the template. By cooperating with the document service, reports can be associated with other objects, such as projects, products or parts. In the implementation, Microsoft Spreadsheets are adopted to define report templates and the open Java APIs from Apache for manipulating various files based upon the Microsoft's OLE 2 Compound Document format [19] is adopted to develop the service.

The domain services are developed to provide functions to integrate and manage business processes based on the product structure model developed. Functions for project management include project initialization, project schedule and progress tracking. In the made-to-order environment, there exist two types of projects, i.e. internal projects and external projects. Internal projects are initialized to manage family design and plan the production of common parts, subassembly and functional subsystems. External projects are created based on customer orders to fulfill customer requirements by cooperating with the inventory service and the resource service. In general, internal projects are managed based on family structures while external projects work on variant structures. The product service provides the capability to manage product family structures, variant structures, part families, subassembly families and a standard part library to assist product configuration, process planning and workshop task generation. Figure 8 illustrates a family structure view with two families, i.e. a car family and a truck family. Car variants and truck variants are achieved by alternative engines and audio subsystems. The inventory service manages stocks of common parts and commonly demanded variants according to safety levels. The resource service manages capacities and capabilities of resources, such as machines, materials, designers and operators to support design task management, process planning and workshop task management.

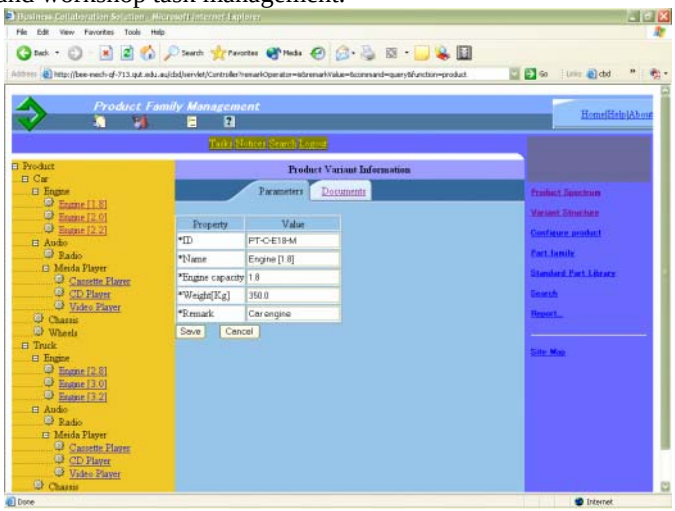

Figure 7: A sample view of product spectrums

CONCLUSION

In manufacturing companies, product structure is the output information of product design. It is critical information for integration of business processes, such as project management, production planning and workshop task generation [18]. In the made-to-order environment, a product is

referred to as a family with a number of variants. In such a context, family structure and variant structure need to be explicitly represented to characterize common characteristics and particular characteristics of individual variants from the business process perspective. Moreover, two representations should be seamlessly integrated to maintain the consistency of the two types of structures. As such, it is a significant industrial need to develop a product structure model that is capable of representing the family structure and variant structures in an effectively way. To address this need, this paper presents a master-variant based product structure model. In the model, a master represents common characteristics of a product family and a variant represents particular characteristics of a product variant. The family structure representation is realized by associating product masters, part masters and subassembly masters. The variant structure representation is built on the top of family structure representation. Therefore, this model is capable of maintaining a clear boundary between product family structures and variant structures. At the same time, it enables the seamless integration of the product family structure management and the variant structure management so that the consistency of a family structure and its variant structures can be easily maintained.

This model overcomes the shortages of product structure models for representing a single product, which are commonly employed by current PDM systems. As traditional product models can not explicitly represent product families and variants, many efforts are needed to customize existing PDM systems for a made-to-order environment. From the business process perspective, it also limits the capability of a PDM system to support process integration. The proposed model can flexibly provide different views of product structures for different processes and effectively support process integrations. A prototype system has been developed to demonstrate the model efficiency and the capability of supporting process integration..

REFERENCES

1. Ni, Q.F., Ming, X.G. and Lu, W.F., 2003, "Computer-Supported Collaborative Environment for Distributed Product Development," Proceedings of the International Conference on Agile Manufacturing-Advances in Agile Manufacturing.
2. Zhang, W.J., 1999, "Information modelling for made-to-order virtual enterprise manufacturing systems," Computer-Aided Design, Vol. 31, No. 10, pp.611-619.
3. He, W., Ni, Q.F., Ming, X.G. and Lu, W.F., 2004, "Product Structure Management for Enterprise Business Processes in Product Lifecycle," The proceedings of 11th ISPE International Conference on Concurrent Engineering, July, Beijing.
4. Jiao, J.X., Tseng, Mitchell, Duftty, Vincent G. and Lin, Fu Hua, 1998, "Product family modeling for mass customization," Computers ind. Engng, Vol. 35, No. 3-4, pp.495-498.
5. Du, X.F., Jiao, J.X. and Tseng, Mitchell M., 2000, "Architecture of Product Family for Mass Customization," Proceedings of the 2000 IEEE International Conference on Management of Innovation and Technology, 2000.
6. CIMdata, 1998, "Product Data Management: the definition, an introduction to concept, Benefits, and Terminology," CIMdata, Inc., USA.
7. Xu, X. W. and Liu, T., 2003, "A web-enabled PDM system in a collaborative design environment," Robotics and Computer Integrated Manufacturing, Vol. 19, No. 4, pp.315-328.
8. Eynarda, B., Galleta, T., Nowaka, P. and Roucoules, L., 2004, "UML based specifications of PDM product structure and workflow," Computers in Industry, Vol. 55, No. 3, pp.301-316.
9. Janitza, D., Lacher, M., Maurer, M., Pulm, U. and Rudolf, H., 2003, "A product model for mass-customisation products," Lecture Notes in Computer Science, Vol. 2774, pp.1023-1029.
10. Hameri, A. and Nihtila, J., 1998, "Product data management—exploratory study on state-of-the-art in one-of-a-kind industry," Computers in Industry, Vol. 35, No. 3, pp.195-206.
11. He, W., Ni, Q. F. and Lee, B. H., 2003, "Enterprise Business Information Management System based on PDM Framework," The proceedings of 2003 IEEE International Conference on Systems, Man & Cybernetics, Washington, D.C., USA.
12. Sudarsan, R., Fenves, S.J., Sriram, R.D. and Wang, F., 2005, "A product information modeling framework for product lifecycle management," Computer Aided-Design, in-press.
13. Fujita, K., 2002, "Product variety optimization under modular architecture," Computers in Industry, Vol. 34, No. 12, pp.953--965.
14. Da Silveira, G., Borenstein, D. and Fogliatto, F. S., 2001, "Mass customization: literature review and research directions," International Journal of Production Economics, Vol. 72, No. 1, pp.1-13.
15. Salvador, F. and Forza, C., 2004, "Configuring products to address the customization responsiveness squeeze: A survey of management issues and opportunities," International Journal of Production Economics, Vol. 91, No. 3, pp.273-291.
16. Forza, Cipriano and Salvador, Fabrizio, 2002, "Managing for variety in the order acquisition and fulfillment process: The contribution of product configuration systems," Int. J. Production Economics, Vol. 76, No. 1, pp.87-98
17. Li, W. D., Ong, S. K., and Nee, A. Y. C., 2005, "A Web-based process planning optimization system for distributed design," Computer-Aided Design, In Press.
18. Jakarta POI, "Java API to Access Microsoft Format Files," http://jakarta.apache.org/poi.
19. Van der Aalst, W. M. P., 1999, "On the automatic generation of workflow processes based on product structures," Computers in Industry, Vol. 39, No. 2, pp.97-111.