Data Modeling for an Integrated PLM and CSP Framework for Configuration Design

H. Ling[1], W. J. Zhang[2]*, J. X. Li[2] and Helen Xie[3]

[1] Fu-Dan University
[2] University of Saskatchewan, wjz485@mail.usask.ca
[3] Integrated Manufacturing Technologies Institute, NRC
(*Corresponding author)

ABSTRACT

Configuration design is an important strategy for achieving rapid mass customization in product development. Constraint satisfaction programming (CSP) is known to provide a general tool for configuring a product (i.e., configuration design in this case). Product life cycle management (PLM) is an important strategy for management of products. Integration of these two strategies is a very promising concept towards an effective computer support for life cycle configuration design. This paper discusses the data modeling issues for an integrated PLM and CSP framework for configuration design. These issues, not well addressed in the general engineering design literature, include (1) inter-relations among design requirement, product structures, and design knowledge, and (2) the concept of structural and semi-structural data and their integration.

Keywords: Configuration design, Data modeling, Semi-structural data, Structural data.

1. INTRODUCTION

The trend of today's manufacture industry is changing from mass production to mass customization. The companies who win the markets are those who can deliver highly customized products with the fastest speed. As such, the production development changes from "stock-to-order", to "assemble-to-order", and/or to "engineer-to-order". Stock-to-order (STO) refers to a manufacturing situation where a whole product is available in the manufacturer's inventory. Assemble-to-order (ATO) refers to a manufacturing situation where the product architecture is known, and the product design means to determine instances in conformity with the product architecture to meet the customer requirement. ATO differs from STO in that in the case of former, instances of components may not be available to the manufacturer which directly communicates with a customer, and they are usually supplied by other manufacturers. ATO is often integrated with the order, supply, and manufacturing systems, so once a product is configured, and production and delivery can follow immediately. Usually, the engineering work is not required for ATO. However, some manufacturing works may be needed when a product is assembled. Engineer-to-order (ETO) deals with problems where not all components are ready to use; some may need to be designed and then fabricated specifically to meet the customer requirement. In order to be quick and agile in response to volatile dynamics of markets, manufacturers should pursue ATO as much as possible. Realization of ATO demands configuring a product from a set of "available" parts satisfying customer needs, which is called product configuration design (or configuration for short).

This paper concerns the development of an effective computer aided system for configuration design. The system is called Configurator for short. Elsewhere a concept which integrates the product life cycle management (PLM) and constraint satisfaction programming (CSP) [Li, 2004] and a methodology employs the CSP for configuration design for a broad area of applications [Zhang et al., 2004] were proposed by the authors. In this paper, specifically, issues of data modeling for the realization of the PLM-CSP framework are discussed. Generally speaking, the current literature has not provided satisfaction solutions to this issue, though product data modeling has been studies for decades. Specifically, there appears to be no study reported on data modeling which is tied to CSP. A more detailed discussion on related work will be presented later in this paper.

The reminder of the paper is organized as follows. Section 2 presents an example to be used for illustration in the subsequent discussions. Section 3 briefly discusses the PLM-CSP framework, which provides a context and a

requirement for data modeling. Section 4 presents data models that are needed for implementing the Configurator which is based on the PLM-CSP framework. Section 5 discusses the related work. Section 6 presents conclusions.

2. CONFIGURATION DESIGN: AN EXAMPLE

The elevator system is taken as an example; see Fig. 1. Fig. 1 is an assembly (part-whole) diagram which represents the elevator architecture [Yost, 1996]. We only considered 16 components (shaded in Fig. 1) for the purpose of this research. For each component, there could be several instances. The selection of these instances in order to achieve some desired performance at the whole system level makes sense for the elevator design to be a configuration design problem.

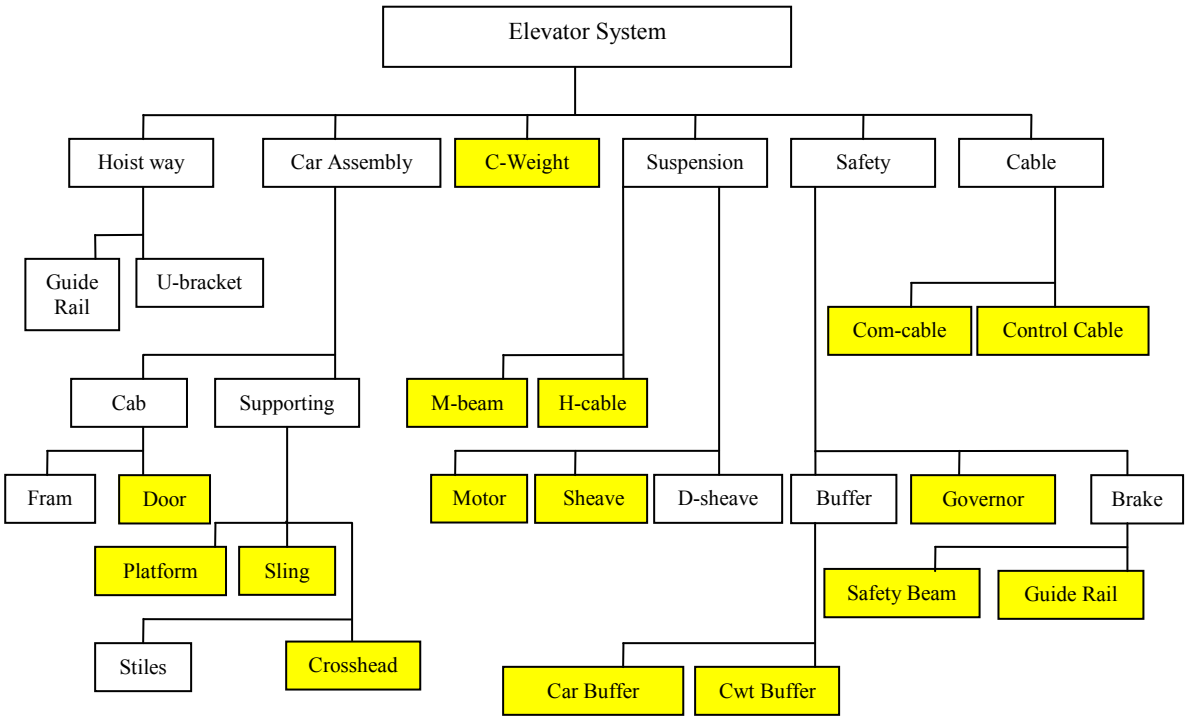

Fig. 1. System decomposition of the elevator system (shaded items may be considered in this paper)

3. INTEGRATED PLM AND CSP: A GENERAL FRAMEWORK

In [Li, 2004] we proposed to integrate PLM and CSP. PLM has three roles in this connection. The first role is that PLM facilitates the engineering activities in the course of configuring a product; see Fig. 2a. Knowledge stored in the format of CSP is meaningful only in the sense of various constraints, while PLM promises to store all information and knowledge about a product [Sinha, 2004]. For instance, the background information concerning why a particular part is interfacing with other parts, or why a particular color is not available for the interior of an elevator will be stored in PLM, but certainly not in CSP. The second role of PLM is that PLM provides a semantic rich expression such that it becomes a meta-knowledge representation for CSP; see Fig. 2b. For example, for a part, say a platform in the elevator system, there are two attributes, size and weight. The knowledge representation at the CSP level may go to the attribute level, where the operand in the CSP expression may have the form (e.g., the platform), such as 'P01#.weight' and 'P03#.size,', where 'P01#' and 'P03#' are the component identities of the platform and the safety system, respectively, and 'weight' and 'size' are attributes or features of these components, respectively. To the CSP, 'P01#' and 'P03#' do not make any sense because there is no expression clause that asserts any semantics for them. Furthermore, the fact that two pieces of information, i.e., 'P01#.colour' and 'P01#.size', are related to a same product, or represent two features of a same product, is missing at the CSP level. However, at the PLM level, there are

representations to assert semantics to the instance 'P01#' (for example) through the schema – instance mechanism, and to assert that the weight and the size are two attributes of the platform. Last, since PLM promises to contain complete information regarding a product under configuration, PLM can be a source to produce heuristic knowledge for improving efficiency in searching solutions to a CSP problem. In the next section, we present data modeling issues for building a Configurator based on the PLM-CSP framework.

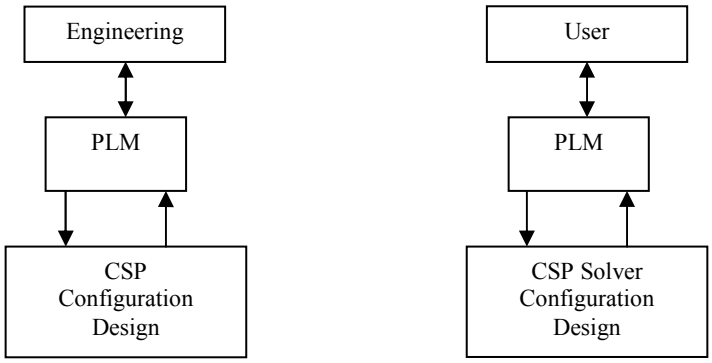

Fig. 2. Integration framework

4. DATA MODELS

The data models should capture: (i) design requirement, (ii) product configuration (product assembly or architecture and product connectivity), and (iii) design knowledge. Furthermore, for all these data models there is an issue: what data fall into the structural data category and what else data fall into the semi-structural data category. The structural data is one that conform to a particular data model framework (e.g., relational, etc.), while the semi-structural data does not. Typically, the semi-structural data appears like note or document. The data model should eventually integrate both categories of data. In the following these models and the issue are discussed.

4.1 A Data Model for Requirement

The design requirement includes: (1) the function, (2) the constraint, and (3) the wish [Zhang, 1994]. The examples of the *functional* requirement are: the speed of the elevator must be greater than 300 feet/min; the capacity of the elevator must be 1000 lbs, etc. The example of the *constraint* requirement is: the color of the interior must be red. The *wish* requirement includes the statements which represent the customer's desires: (i) Quality: as good as possible; (ii) Cost: as low as possible; and (iii) Time: as short as possible. Note that the wish may not necessarily be achieved, which differs from the function that has to be fulfilled and from the constraint that has to be subjected to.

A data model for the requirement is shown in Fig. 3. Specifically, Fig. 3 represents the semantics that the design requirement is associated with the function, the constraint, and the wish. The multiplicity indicated on the diagram shows that there must be at least one function requirement; yet there may be none constraint or none wish requirement. It is noted that in Fig. 3 the customer who proposes the requirement is also shown. This is intended to track the requirement change initiated by the customer. It is the customer who keeps the rationale for a particular piece of requirement. So when the requirement is changed by a customer (say A), the customer (say B) who is associated with the one to be changed may need to be communicated. For instance, a message is posted to customer A: "The requirement you suggested is to be changed by customer B." If A and B are the same person, such a message with its underlying mechanism is useful for customer requirement management, especially requirement history recording. The detail regarding the customer requirement management refers to [Brown, 2000].

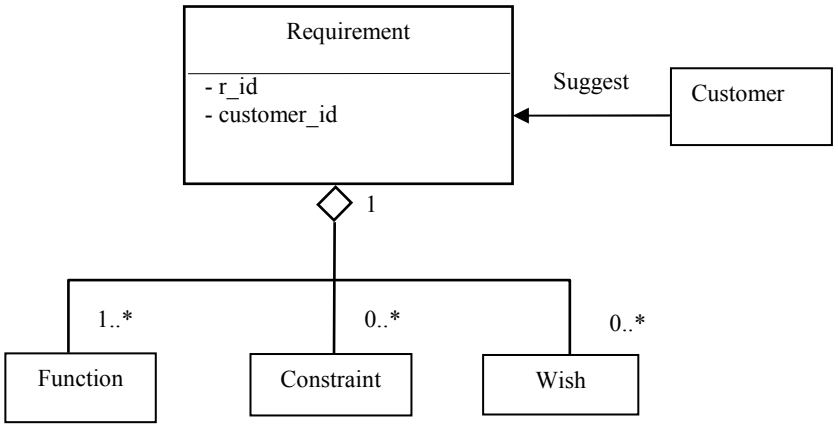

Fig. 3. A data model for the requirement

Fig. 4 represents the function requirement. There are two ways to specify the function requirement (see Fig. 4, F1 and F2, respectively). One way (F1) is, for example, the capacity of an elevator is 3000 lb. The other way (F2) is, for example, the capacity of an elevator increases from 3000 lb to 5000 lb. In the data model for the function, the attribute 'function_word' means all the features used to describe a particular product. For example, for the elevator system, the function words are: 'capacity', 'platform.weight', 'platform.size', etc. The syntax 'platform.weight' has the following meaning. The first word 'platform' stands for the class, and the second attribute 'color' stands for the attribute of the class 'platform'. The generalization of this syntax is self-explanatory and applies to the rest of discussion in this paper. The attributes 'operator', 'quantity', 'from_quantity', and 'to_quantity' are self-explanatory with respect to their corresponding classes (F1, F2). The two examples of the function requirement, as discussed before, can then be expressed in the form of instances as follows: Instance of F1: <F001#, 'Capacity', '>', 3000>; Instance of F2:<F002#, 'Capacity', 3000, 5000>, where the symbol '>' means greater than.

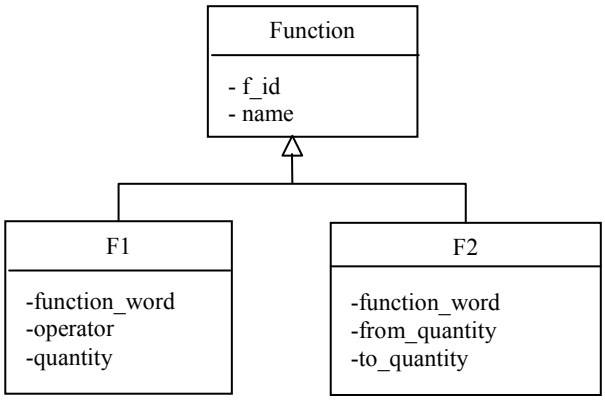

Fig. 4. A data model for the function

The example of the constraint, regarding the color of the interior of the car in the elevator system, can be represented as: Instance of constraint: <C001#, Interior.color, 'Red'>, where the term 'Interior.color' refers to the color of the interior of the elevator, and it is a product feature. Fig. 5b is a data model for the wish requirement.

The wish may be applied to the whole product, or a set of components. An example of the wish would be: the cost of an elevator system should be the lowest. The data representation of the wish requirement can be expressed in the form of instance as follows: < W001#, Cost, Lowest, {component 1, component 2, …, component n} >, where the 'W001' stands for an identity of a wish; 'cost' is a behavior of the product; 'lowest' represents the degree of the wish in the customer's mind. In the bracket '{… …}', all concerned components of a product are listed. When a whole product system (e.g., the elevator system) is described for a certain behavior or property, simply put the name of the product in the brackets.

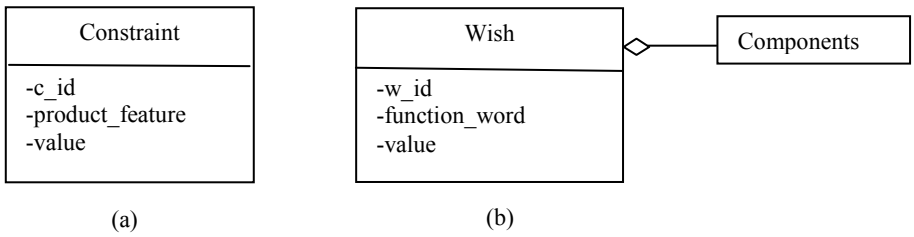

Fig. 5. Data models for the constraint and the wish

4.2 A Data Model for Product Configuration

A product configuration is viewed as a network of connections among a set of components. Such a view of product is also called connectivity view. A product configuration may also be viewed as a set of sub-systems or components based on the function decomposition at varying levels. Such a view of product is also called assembly view. For example, the elevator system is decomposed into (see Fig. 1): (a) Hoist way assembly, (b) Car assembly, (c) C-weight, (d) Suspension, (e) Safety, and (f) Cable. The Hoist way assembly is further decomposed into the components: guide rail and U-bracket, and the car assembly is further decomposed into the subsystems: car and supporting system.

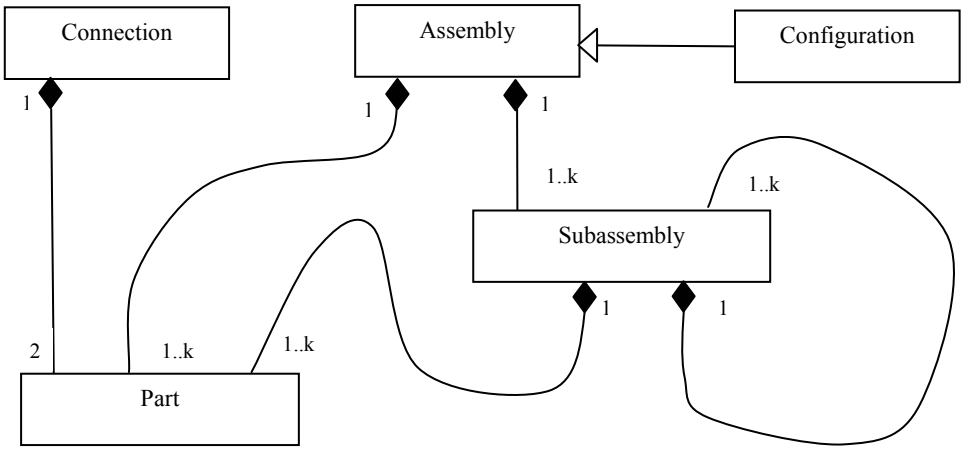

Fig. 6. Data model for the assembly

A data model which captures the above semantics is shown in Fig. 6. In this figure, it is further remarked that the line which connects the class 'subsystem' to itself express that the level of decomposition of subsystems may be more than one and varying. Note that Fig. 6 also represents the connectivity view of product configuration. To make the model more general, the connectivity can be viewed as a set of pair relationships [Zhang and Van der werff, 1993]. For example, a product has four objects that are connected, as shown in Fig. 7.

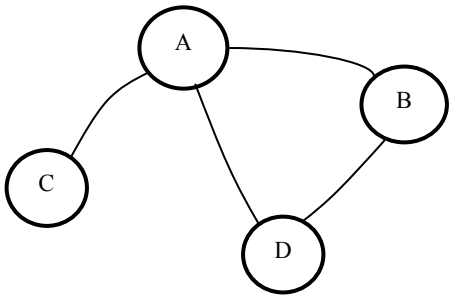

Fig. 7. Pair relationships for the general connectivity representation

This connectivity view can be expressed by a list of pair relationships as follows: <A, B>, <A, C>, <A, D>, and <B, D>. Further, there are various types of connections in the product architecture, for example, the shaft-and-hole, face-to-face against, etc. Fig. 8 represents this semantics. Specifically, for the shaft-hole connection type, a set of attributes that describe it is shown in Fig. 9, where the attributes of the class 'Shaft_Hole_Connection' are self-explanatory.

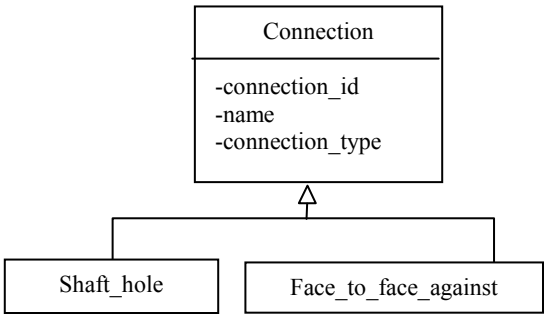

Fig. 8. A data model for the connection

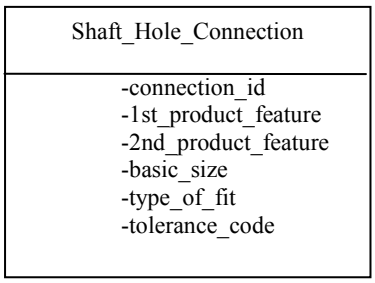

Fig. 9. A Data model for the Shaft-hole connection type

There are linkages between the requirement data model and the product configuration data model. This is the key to make sense of integrated function, behavior, and structure model [Zhang et al., 1997]. In particular, the domains of the attribute 'function_word' and the attribute 'product_feature' should be included by the domain of the product "dictionary" and "ontology" (PDO). The PDO contains the following information: (i) Structural features of components, of subsystems, and of systems, (ii) Behavioral features of components, of subsystems, and of systems, and (iii) Their semantics. For example, in the PDO, there may be the following definitions:

001#: Platform.size: the size of the platform;
002# : Interior.color: the color of interior;
003# : Capacity: the capacity of a system;
004# : Cost: the effort which is converted into money for making products, subsystems, and components.

In the above expressions, the left numbers are the identifiers of attributes or features. The first two examples are the structural feature, while the last two are the behavioral feature.

4.3 A Data Model for Configuration Design Knowledge

Design knowledge is about knowledge for synthesis; that is, it answers various "how-to-achieve" questions. Knowledge for synthesis of configurations is, for example, the colour of an elevator interior should be red when the door of the elevator is open vertically, when the door is opened vertically the size of the platform must be '2.5B' or '4B', etc. Generalizing these examples shows that knowledge for configuration design is naturally expressed by production rules. The rules can be further expressed in a declarative manner by table form; see Fig. 10. Further generalizing the table form representation can lead to the constraint representation which is mostly in the binary or ternary form. It is noted that the constraint here differs from the constraint requirement discussed before. Here, a constraint makes sense when a corresponding relationship must be maintained. For instance, the color of the interior of an elevator is related to the color of the door of the elevator, specifically the red of the interior matches the white of the door. When such a relationship needs to be maintained, a constraint is then built upon the color of the interior and the color of the door. Note that the relationship and the constraint are viewed interchangeably hereafter.

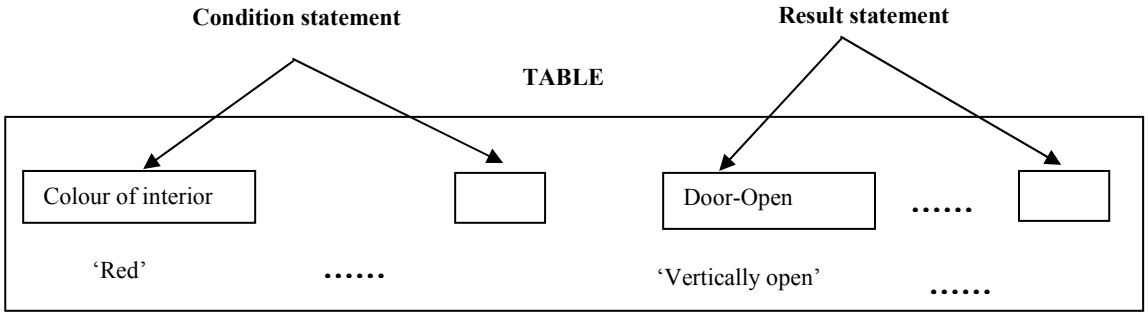

Fig. 10. The tabular form of representing knowledge for configuration design

Fig. 11 presents a data model for knowledge for configuration design. It should be noted that the domain of the class 'product_feature' is included by the domain of the class "PDO" (see a previous discussion). Several remarks can be made regarding this model.

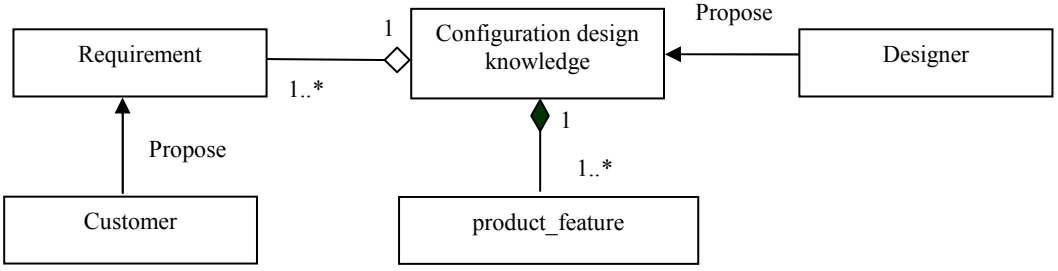

Fig. 11. A data model for the design knowledge for configuration design

Remark 1: When the multiplicity indicted at the end of the class 'product_feature' is '1', this means that there is only one 'product_feature' associated with the class 'Configuration design knowledge'. In this case, the design knowledge which makes sense is the domain of the respective 'product_feature'.

Remark 2: When the multiplicity indicated at the end of the class 'product_feature' is '2', this means a binary (or n-nary) constraint. When the multiplicity is '3', this means a ternary constraint.

Remark 3: In this representation, the design requirement is associated with the configuration design knowledge. The result of a configuration design process is the representation of (1) a particular instance of design requirement, and (2) a particular instance of configuration design knowledge base; see Fig. 12. In this case, the data models for the requirement and for the configuration design knowledge become a kind of template [Sinha, 2004].

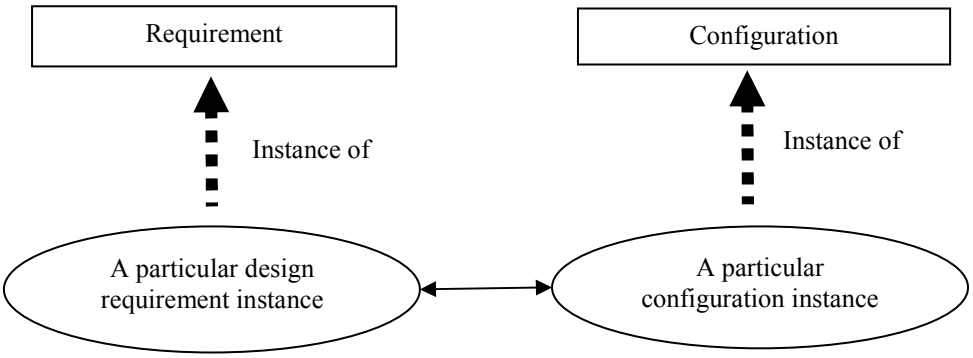

Fig. 12. The representation of design requirement and design knowledge

Fig. 13 shows a new knowledge base which contains all the pairs of requirement instances and design instances. Such a knowledge base is called design case base. The design case base can improve the efficiency of configuration design in the following sense. For any new configuration design task, the first step may be to search the design case base by attempting to match the requirement instance given the requirement instance of the new design task. If there is a match, then simply to retrieve the corresponding design result of the matched requirement instance completes the design task.

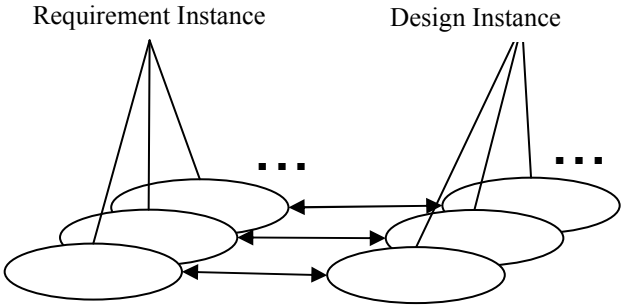

Fig. 13. A new knowledge base

4.4 Integration of Structural and Semi-structural Data

The structural data is one that conform to a particular data model framework (e.g., relational, etc.), while the semi-structural data does not. Typically, the semi-structural data appears like note or document. The cost of modeling the semi-structural data using the conventional concept of data model will be very high. The tag-based data modeling language (e.g., HTML) emerged with the development of the Internet technology; more recently XML, and it is inherently geared with the relatively free format of document data. The structural data and semi-structural data should

be integrated because they both represent the product and its development process. The key idea to integrate them is to make them share the same ontology (see Fig. 14). This means that both the tag in the XML and the attribute in the PLM use a same set of vocabulary.

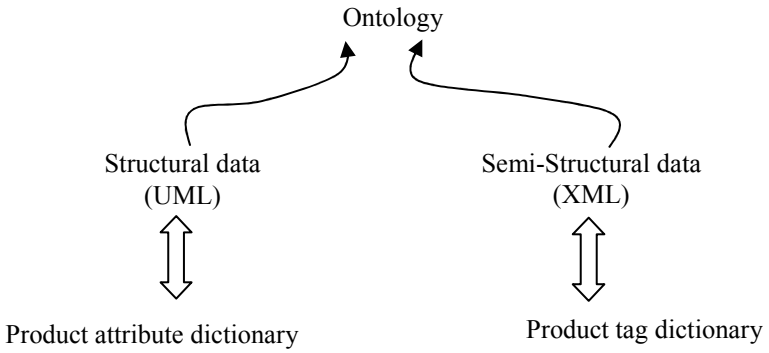

Fig. 14. Integration of structural and semi-structural data

5. RELATED WORK

Configurator was originally designed as an interface on the top of the ERP (Enterprise Resource Planning) system. This is especially introduced to facilitate the acquisition of products from customers with a focus on the price and delivery time. The commercial configurators, such as Windchill™ Product and Sales Performer™, are a successful implementation of the configurator concept; in addition, they usually provide the customer with access to their systems at any time and any place through the Internet technology. This kind of configurator may be called the ERP-based configurator.

One of the essential assumptions underlying the ERP-based configuraor is that products are pre-designed. Specifically, components are ready to go, and they just need to be assembled [Tiihonen and Soininen, 1997]. The ERP-based configurator cannot work for the situation where the product needs some manufacturing work. For example, the customer may not prefer a color of the interior of the elevator which is not available in the current component repository. In this case, the elevator manufacturer may reject the customer's request for that special color, but the customer oriented manufacturing practice would try to tailor whether the component with that color could be painted and produced, which may need to contact the manufacturer's supplier or sub-contractor who does the painting job for the interior. Another example is such that the customer may want to install a video camera in a place where the existing elevator is designed not to hold the camera in that place. To accommodate that requirement from the customer will require a little bit of engineering work to assembly a product.

It might be quite true that the configurator without the capability of accommodating engineering and manufacturing works (more or less) will considerably compromise the manufacturer's philosophy: customer-oriented product development. Mesihovic and Malmqvist [2000] suggested a PLM integrated configurator. This idea is promising; yet they have not given details of their system. The study reported in the present paper differs from their work in that (1) we provide more detailed information about how PLM and configuration design system merge, and (2) we specifically tie to CSP.

The data modeling for general product development has been extensively studied. However, most of the studies focused on the data modeling for product structures, and they did not model the function and behavior and nor the relationship among the function, behavior and structure except our previous studies [Zhang, 1994; Zhang et al., 1997; Zhang et al., 2000; Shrikhande, 2000]. Specifically, in [Zhang et al., 1997] we demonstrated an integration of the structural, behavioral and functional information in the case of mechanisms design. In the present paper we extended the discussion there to the problem of configuration design. In [Shrikhande, 2000] we elaborated on the need to represent the semi-structural data which are mostly about the background for product structures. In this paper, we

proposed a scheme to integrate the structural data and the semi-structural data by making both share the same vocabulary and notion. In the study reported by Wang and Nnaji [2004], a different idea was proposed with which XML was employed to write the structural data as well as the semi-structural data.

6. CONCLUSION

In this paper we proposed a framework that integrates PLM and CSP due to their strengths in two respective domains which should however be integrated, namely the semantic data representation or model with the PLM and the computational model with CSP. This integrated framework is called PLM-CSP and is applied to configuration design which follows the assembly to order product development pattern. An integrated data model for design requirement (function and constraint), product structure, and design knowledge is presented in this paper.

ACKNOWLEDGEMENT: This research is supported by an NSERC CRD grant to the corresponding author.

7. REFERENCES

[1] Bowen, J. and Bahler, D., 1991. Conditional existence of variables in generalized constraint networks. *Proc. AAAI,* pp. 215-220.

[2] Li, J.X., 2004. *A Novel Approach and System to Assembly-to-Order Configuration Design*, M. Sc. Thesis, University of Saskatchewan, Saskatoon, Canada.

[3] Mesihovic, S. and Malmqvist, J., 2000. Product Data Management (PDM) system support for the engineering configuration process. *14th European conference on artificial intelligence ECAI 2000 configuration workshop*, Aug. 20-25, 2000, Berlin, Germany

[4] Shrikhande, S.V., 2000. *On effective information modeling for computer-based configuration management of complex products in manufacturing environments*, M. Sc. Thesis, University of Saskatchewan, Saskatoon, Canada.

[5] Sinha, N., 2004. *Modeling for effective computer support to MEMS product development*, M. Sc. Thesis, University of Saskatchewan, Saskatoon, Canada.

[6] Tiihonen, J. and Soininen, T., 1997. Product configurators – information systems support for configurable products. TAI research center and laboratory of information processing science, Product Data Management Group, Helsinki University of Technology, Finland.

[7] Wang, Y. and B.O. Nnaji, 2004. UL-PML: constraint-enabled distributed product data model, Int. J. Prod. Res., 42 (17), 3743-3763.

[8] Yost, G.R., 1996. Configuring elevator systems, *Int. J. Human-Computer Studies,* 44, pp 521-568.

[9] Zhang, WJ, Li, JX, Xie, Helen, and Shi, Z.Z., 2003. A general approach to e-learning software*, Proceedings of CIE'03: ASME international 23rd Computers and Information in Engineering (CIE) Conference*, September 2-6, 2003, Chicago, Illinois.

[10] Zhang W.J., 1994. *An Integrated Environment for CADCAM of Mechanical Systems*, Ph.D. thesis, printed by Delft University of Technology, The Netherlands, ISBN 90-370-0113-0, pp. 1-263.

[11] Zhang W.J. and Werff van der K. 1993a, Guidelines for product data model formulation using database technology, *Proc. of Int. Conf. on Engineering Design (ICED'93),* Vol.3, The Hague, 1618-1626.

[12] Zhang W.J., K. van der Werff, and D. Zhang, 1997. *Toward an integrated data representation of function, behavior and structure for computer aided conceptual mechanical system design*, in Integrated Product and Process Development: Methods, Tools and Technologies, John M. Usher et al. (Eds.), John Wiley & Sons, 1997, 85-124.

[13] Zhang W.J., S.N. Liu, and Q. Li, 2000. Data/Knowledge representation of modular robot and its working environment, *Int. J. of Robotics and CIM*, Vol. 16, No.2 (June), 143-159.