The Research of Platform-Based Product Configuration Model

Huiqiang Yan[1], Qunsheng Guan[2], Qinghai Li[1], Fei Lu[1], and Xiujuan Wang[1]

[1] Institute of Design for Innovation, Hebei University of Technology, 300130, TianJin, China
[2] Military Traffic and Transportation Research Institute, 300161, TianJin, P.R. China
{Huiqiang.Yan,Qunsheng.Guan,Qinghai.Li,Fei.Lu,Xiujuan.Wang,
transparentyan}@gmail.com

Abstract. Product configuration system, being a knowledge intensive system, plays an important role in order to realize Mass Customization. The Platform-based generic product configuration tool(PB-GPCT) being developed by Institute of Design for Innovation, Hebei University of Technology. PB-GPCT is a structure-based and domain independent configuration tool. For the realization of PB-GPCT, UML is chosen to construct the configuration model and OCL(Object Constraint Language) to express constraints. In order to manage the constraints of product easily, the approach of checking consistency of configuration model is presented. The theory of constraints hierarchies is introduced into the system of PB-GPCT in order to express customer requirements of different levels.

Keywords: Object Constraint Language, Constraint hierarchies, Configuration model, Configuration model consistency.

1 Introduction

Nowadays, companies must shorten product cycles and can meet individual customer's requirements at a lower price in order to survive. Product configuration is a common method which is employed to realize mass customization. Mittal & Frayman defines configuration as a form of design, which selects assembly of components from a set of pre-defined components to meet customer's requirements [1]. "The result of each configuration will be a model of the configured product, configured product model [2]." So that configuration is to instantiate configuration model using pre-defined components. Configuration model is the product model which is scoped within the conceptualization of configuration domain [3].

Distinguishing with other definitions, the platform-based configuration model is defined as follows:

CM= ($\{C_1,\ldots, Cn\}\{P_1,\ldots,P_m\}\{R_1,\ldots,R_s\}$). C_i is the element of the configuration model which is called component type; P_i is the platform which is composed of component instances; R_i is the constraints among platforms and component types.

This paper is organized into 4 sections. In section 1, the definition of platform-based configuration model is introduced. In section 2, the approach of representing

R. Tan, G. Cao, and N. León (Eds.): CAI 2009, IFIP AICT 304, pp. 89–96, 2009.

configuration model is presented. In section 3, the algorithm of checking configuration model consistency is proposed. Lastly, the approach of constructing configuration model is discussed.

2 Representation of Configuration Model

In this paper UML is employed to express the configuration models for the following reasons [4]:

1. UML (Unified Modeling Language) is the leading industrial object-oriented modeling language for software engineering.
2. UML is extensible for domain-specific purposes, for the semantics of the basic modeling concepts can be further refined in order to be able to provide domain-specific modeling concepts.
3. The Object Constraint Language (OCL) being a built-in constraint language can perfectly describe the constraints about the objects in the model.

2.1 The Elements of Configuration Model

According to the definition of configuration model stated above, the elements include Component type, Component instance, Relations, Rules and Platforms. Part of configuration model of configurable pc is showed in Fig. 1.

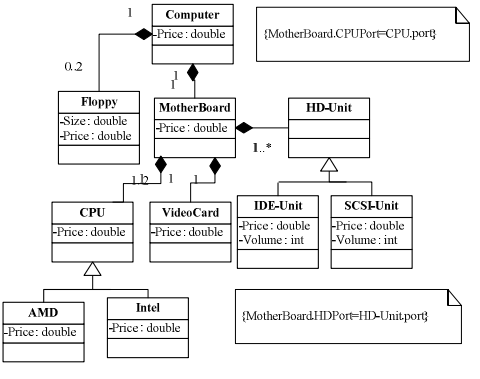

Fig. 1. Part of configuration model of configurable pc

Component Type. There are two kinds of component types. One is *virtual component* which does not have a physical correspondence, e.g. HD-Unit and CPU. The other is *physical component* which typically has a bill-of-material associated with it, e.g. VideoCard and MotherBoard.

Component Instance. When all the properties of component are given values, we call it Component instance. Virtual component and physical components all have component instance associated with it. However, the properties of physical component are given by product experts before configuration, the properties of virtual component are assigned by sales engineers at the configuration phase.

Relations. The taxonomies of relations are Part-of and Is-a. ComponentA part-of componentB means that componentB is part of componentA, e.g. CPU, VideoCard and HD-Unit are parts of MotherBoard. ComponentB Is-a componentA means that componentB is a kind of componentA, e.g. IDE-Unit and SCSI-Unit are two kinds of HD-Unit.

Rules. Rules not only include the constraints among component types, but also include the constraints among component instances. The taxonomies of constraints are unary, binary, global and incrementally constraints [5].

Platform. Platform consists of common elements from which a range of products can be derived. For a configurable PC, the platform elements are MotherBoard and CPU. Sales engineer first selects platform (determine the MotherBoard and CPU) according to customers requirements, then based on the platform other components are configured.

2.2 The Object Constraint Language

We select OCL to express constraints among components for the following reasons [6]:

(1)OCL is a formal language which is easy to read and write for users.

(2)OCL is a pure specification language; therefore, an OCL expression is guaranteed to be without side effects. When an OCL expression is evaluated, it simply returns a value. It cannot change anything in the model.

OCL is a typed language so that each OCL expression has a type. Examples of the operations on the predefined types are showed in table1. Collection types are also supported besides predefined types, the operations frequently used are select(), reject(), size() etc.

Table 1. The predefined types of OCL

Type	Operations
Integer	*, +, -, /, abs()
Real	*, +, -, /, floor()
Boolean	and, or, xor, not, implies, if-then-else
String	Concat(), size(), substring()

2.3 Constraint Hierarchies

Constraint hierarchies theory is presented by Alan Borning [7]. A labeled constraint is defined as: a constraint labeled with strength, written sc, where s is strength and c is a constraint. A constraint hierarchy is defined as a multiset of labeled constraints. Given a constraint hierarchy H, H_0 denotes the required constraints in H. In the same way, the sets H_1, H_2, Hn for levels 1, 2,…, n. For k>n ,$H_k=\phi$. Constraints H are also called hard constraints, and constraints H_1, H_2,…,H_n are called soft constraints. The constraints hierarchies are divided into required, strong, medium, weak and weakest [8].

We introduced the constraint hierarchies into configuration system in order to express the customer's requirements of different levels. The constraints are divided into five levels, required, strong, medium, weak and weakest, and each constraint has a weight associated with it. At the same time, the OCL is expanded to express the constraint hierarchies and the following is an example.

Context MotherBoard **inv**:
Self.CPUPortType = CPU.PortType **required 1**

The keyword **Context** introduces the constraints belong to which components. In this example, the constraint is belong to the component MotherBoard. The keyword **inv** denotes stereotypes of the constraints. For **inv**, the constraint must be true when the component is instantiated. This constraint means the selected MotherBoard's port type must be in line with the port type of CPU which is chosen. The keyword **required** indicates the level of the constraint. The value "1" is the weight of this rule, and the scope of weight is between 0 and 1.

Writing constraints in OCL is a time consuming and an error prone task. So that we developed an edit window which is showed in Fig. 2. The main function of this window is lexical analysis, syntactic analysis and semantic of analysis [9]. Utilizing this window, users can easily write correct constraints for the configuration model.

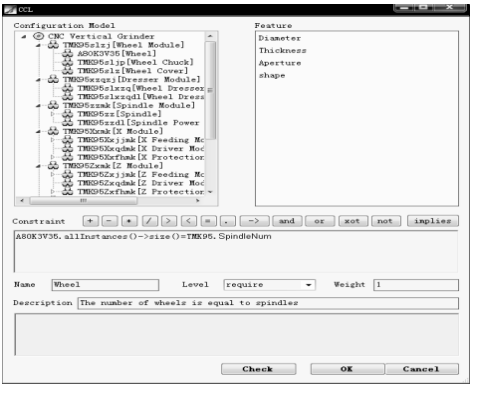

Fig. 2. Constraint edit window

3 Configuration Model Consistency

3.1 Constraint Graph of Configuration Model

For the component C_i, R_j is one of the constraints of C_i. R_j =f (C_i, C_m,…, C_n) means R_j has relationship with components C_m,…, C_n. According to this constraint R_j, drawing directed edges start from C_i to components C_m,…, C_n. If one edge has been existed, the action is ignored. For the configuration model of Fig 3, the following rules exist.

R_{A1}=f (A, B); R_{B1}=f (B, C); R_{B2}=f (B, D); R_{E1}=f (E, F);

According this conversion method, the corresponding constraint graph is showed in Fig 4.

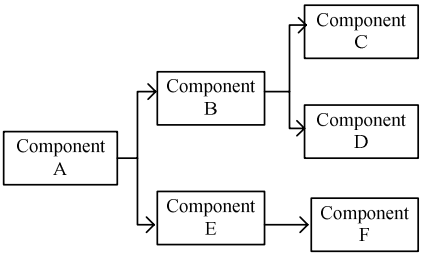

Fig. 3. Configuration model

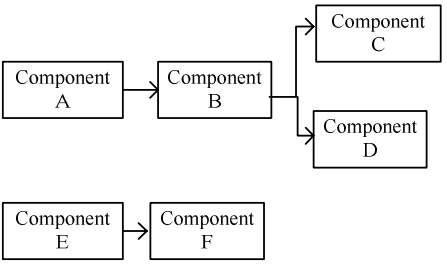

Fig. 4. Constraint graph

3.2 Component Consistency

We define function $Sat(\theta, H)$ as follows:

$$Sat(\theta_k(C_i), H(C_i)) = \begin{Bmatrix} True; \theta_k(C_i) \text{ satisfies } H(C_i) \\ False; \theta_k(C_i) \text{ does not satisfies } H(C_i) \end{Bmatrix}$$

H is the hard constraints of the component C_i; $\theta_k(C_i)=\{CI_k,CI_m,\ldots,CI_n\}$, CI_k is one of instance of component C_i, $C_m,\ldots,C_n$ are the children or brothers of component C_i in constraint graph, $CI_m,\ldots,CI_n$ are the instances of components $C_m,\ldots,C_n$. For example, θ_k (B) = (BI_k,CI_m,DI_n). The component consistency is defined as: for any instance CI_k of component C_i, there is valuation θ_k (C_i) which satisfies the hard constraints of component C_i, or formally,

$$\forall CI_k, \exists \theta_k Sat(\theta_k(C_i), H(C_i)) = true$$
$$\theta_k = (CI_k, CI_m, \ldots CI_n)$$

3.3 Path Consistency

In the constraint graph, there may be more than one path from root component to another component. For instance, there are two paths from component A to component D in the figure 5, namely, P_1=A→B→C→D and P_2=A→B→D. The component B is cross component of these two paths. The path like this is called cross path. We define path consistency as: there are at least two valuations along the cross

paths from the cross component to the target component which satisfy all the constraints of the two paths, and the valuations have the same cross component instance and target component instance, or formally,

$$\exists \theta_m Sat(\theta_m, H(P_i)) = true \wedge \exists \theta_n Sat(\theta_n, H(P_j)) = true$$
$$\theta_m = (CI_s, ..., CI_t); \theta_n = (CI_s, ..., CI_t)$$

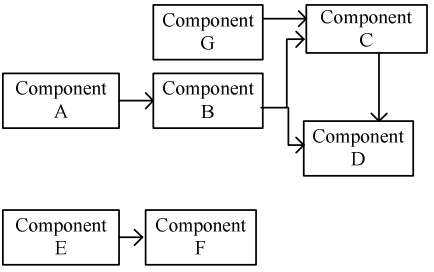

Fig. 5. Cross path and parallel path

The indegree of component C is two too and there are two paths, namely, P_1=A→B→C and P_2=G→C. However, there is no cross component between these two paths, we call it parallel path. It is unnecessary to check consistency of parallel path.

3.4 Configuration Model Consistency

M.Wahler [10] defines that a UML/OCL model M is strongly-consistent if and only if there exists a state in which all classes of M are instantiated. The state of a given UML/OCL model can potentially contain an infinite number of objects, however the state of a given configuration model contains a finite number of component instances. Consequently, we only discuss the problem of finite states. Configuration model CM is consistency if and only if all the components and paths is consistency.

4 Constructing the Configuration Model

For customers the process of configuration is to instantiate the elements of configuration model. Configuration model is constructed by product engineer. The proposed development process for constructing configuration model is shown in Fig 6 and it is divided into six steps. (1) Firstly, the platform of the product is designed by platform designer. (2)The product engineers build the product tree. The product tree is composed of components type. (3)In terms of component type of the product tree, the component instances are created. (4)According to the platform designed at first step, input the platform elements into the product tree and the configuration model is constructed. (5)The consistency of the configuration model has to be checked. The consistency checking has to be done repeatedly to ensure the configuration model is consistent. (6)All the elements have to be stored into Database which is called knowledge base.

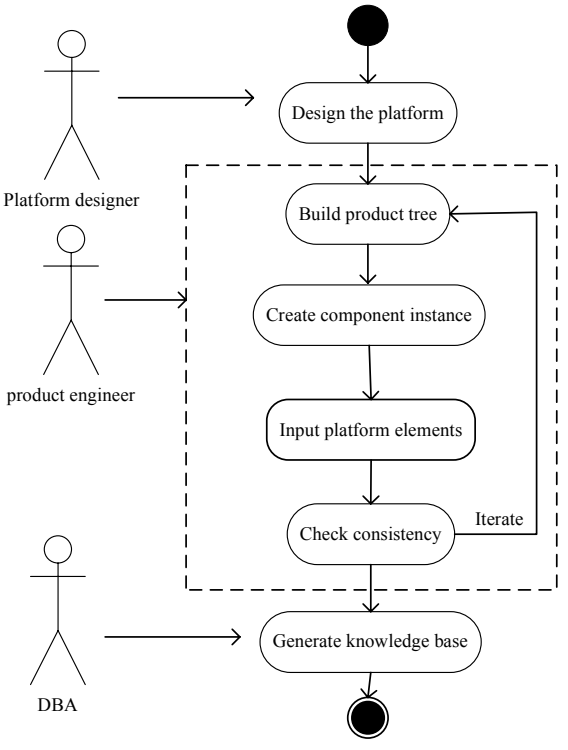

Fig. 6. Construct the configuration model

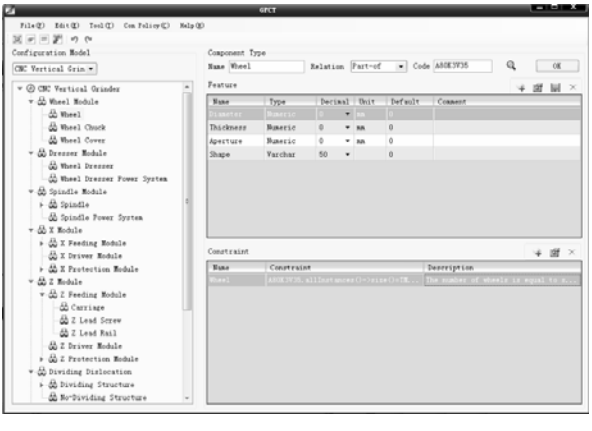

Fig. 7. The system of PB-GPCT

5 Conclusion and Future Works

In this paper, the configuration model of platform-base configuration system is presented. PB-GPCT has been developed utilizing the theory stated above and has been

applied to TIANJIN NO. 2 MACHINE TOOL CO., LTD. The main GUI of PB-GPCT is showed in Fig. 7. But a lot of effort have to be done in the future, for example, the evaluation of configuration, automatic configuration etc.

Acknowledgements

This research is supported in part by he Key Project of the Ministry of Science and Technology of the People's Republic of China under Grant Numbers 2008IM030100, and the science and technology key project of Hebei Province under Grant Numbers 09212102D, the Natural Science Foundation of Hebei under Grant Numbers E2008000101.

References

1. Mittal, S., Frayman, F.: Towards a generic model of configuration tasks. In: The 11th IJCAI, pp. 1395–1401. Morgan Kaufman, San Mateo (1989)
2. Kaj, A., Jørgensen: Manufacturing Information Systems. In: Proceedings of the Fourth SMESME International Conference (2001)
3. Tiihonen, J., Lehtonen, T., Soininen, T., et al.: Modeling Configurable Product Families. In: 4th WDK Workshop on Product Structuring, Delft University of Technology, October 22-23 (1998)
4. Felfernig, A., Salbrechter, A.: Applying function point analysis to effort estimation in configurator development. In: International Conference on Economic, Technical and organisational aspects of Product Configuration Systems, Kopenhagen, Denmark, pp. 109–119 (2004)
5. Altuna, A., Cabrerizo, A.: Co-operative and Distributed Configuration. In: NOD 2004, Erfurt, Germany, September 2004, pp. 27–30 (2004)
6. OMG.: Object Constraint Language Specification (2006)
7. Borning, A., Freeman-Benson, B., Wilson, M.: Constraint Hierarchies. Lisp and Symbolic Computation 5(3), 233–270 (1992)
8. Sannella, M.: The SkyBlue Constraint Solver, R 92-07-02, Department of Computer Science and Engineering, University of Washington (February 1993)
9. Yan, H.Q., Xiao, G.X.: The research and realization of configuration tool based on OCL. Computer Engineering and Applications 45(6), 73–77 (2009)
10. Wahler, M.: Using Patterns to Develop Consistent Design Constraints. PhD thesis, ETH Zurich (2008)