Int J Adv Manuf Technol (2006) 30: 589–600
DOI 10.1007/s00170-005-0108-8

ORIGINAL ARTICLE

Chen Liang · Jin Guodong

Product modeling for multidisciplinary collaborative design

Received: 2 February 2005 / Accepted: 20 June 2005 / Published online: 2 December 2005

Abstract The paper analyzes characteristics of multidisciplinary collaborative design (MCD) of product and proposes a new MCD-oriented product information model (MCDPM) that integrates physical structure, design semantic and collaboration management data. Better understanding and coordination among the different disciplines can be achieved through the multi-level relations at the semantic level, structure level, and management level. An extended object-oriented method is introduced to represent the MCDPM. As a component and its attributes are all defined recursively in class form, the model can be continually extended and has good modularity, flexibility, and evolution ability. The class inheritance methods in network environment, the concept of discipline view model (DVM) and the method generating the DVM are given. The coordination and consistency maintenance among multiple DVMs is discussed.

Keywords Collaborative design · Multidisciplinary design · Multi views model · Product modeling

1 Introduction

A complicated mechanical product has generally various functions. It is required that the designers in different disciplines collaboratively work to improve the product quality and shorten the development cycle. The designers in different disciplines observe the same product object from different views. The purpose of effectively supporting the cooperation of multiple disciplines and maintaining their consistency makes it necessary to exchange, share, understand, and coordinate the product information among them. Thus, product modeling for MCD is the foundation for building a design system supporting the MCD.

Product modeling mainly involves model content, model representation and model management, etc., which are interpreted as follows.

The model content is concerned about the information involved in the product design process. Product design is an iterative synthesis-analysis process through requirement-function-behavior (technical principle)-physical entity mapping transformation [1]. Axiom design theory also considers that design is the mapping process from the function domain to the physical domain. As shown in Fig. 1, the X-axis represents the mapping among different design domains, the Y-axis represents the decomposition levels of objects from abstracts to details, and design involves one bi-directional mapping process [2]. The physical structure is the final result of product design, but product design begins at the function requirements, and ends at the satisfactions of the function requirements. Function, behavior, technology principle, and so on reflect the semantic information of product design, and the physical structure is the result of the semantic reasoning; contrarily, the behavior effect can be obtained by the structure analysis, and the function evaluation can be made by behavior effect. Thus, besides the physical structure data, the semantic information is also very important to enhance the design understanding and coordination among the different disciplines [3]. The well-known product models mainly include the geometry model and feature model. The geometry model mainly describes the geometric information of the product (such as geometry, topology, dimension, and so on) and is usually used for product information sharing and visualization in collaborative design [4], but it does not consider the semantic information of product design. The feature model constructs the product model using the application-oriented feature element that contains not only geometry and topology information but also semantic information of

C. Liang · J. Guodong
School of Mechanical Engineering,
Huazhong University of Science and Technology,
430074 Wuhan, People's Republic of China

C. Liang (✉)
College of Mechanical Engineering and Automation,
Fuzhou University, 523 Gongye Road,
350002 Fuzhou, Fujian, People's Republic of China
e-mail: Liangch@fz.gwbn.com.cn
Tel.: +86-591-83852692

geometry entity. However, the feature model mainly focuses on the integration of design and manufacture, and its semantic information mainly expresses the required manufacture information and does not contain the semantic information of product design [5]. In [6], a feature-based collaborative design system is introduced in which the features of a part model can be classified into general feature, collaboration feature, design feature, manufacturing feature, shape feature, process feature, precision feature, and material feature. In [7], a semantic modeling extension (SME) system is put forward. After creating a graphic form using 3D CAD, a designer interactively adds interpretation objects to the graphic form, such as function and behavior, etc, namely adds designer's intents to the graphic form to generate the cognitive model. Because function and behavior are added only after physical structure has been generated, SME isn't in accordance with the design synthesis process.

The model representation is concerned about the description and computerization of the model contents. In [8], a set of generic definitions of functional representation, behavior representation and artifact representation are given, but they don't involve the computerization of the model. The object-oriented method is a common method to represent the product model. In [9], the product model extends traditional OOP and the design objects can be defined recursively without a pre-defined granularity, as a result, the model is implemented as a layered scheme and has good flexibility. In [10], a class feature concept is presented; product modeling libraries are described as class features; a class at a remote location can be used for defining a new class feature at the local site, but the class feature doesn't contain the semantic information of product design. The component technology based on DCOM or CORBA is another common method to represent the product model. In [11], the models or other software applications are encapsulated in CORBA-based components. Components interact with other components by reciprocal service calls. The services constitute the interface of components and describe what each component can provide. However, the relations among these components are predefined, which don't satisfy the requirements of dynamic design changes. Besides, the component technology emphasizes on reuse, and exports the results according to request. Using this method, the client can only obtain output results from the component rather than inherit and expand the component, so that the flexibility of product modeling is weak. The product data exchange standard STEP [12] established by ISO defines the data expressing and exchanging form of product life cycle, but mainly emphasizes longitudinal integration of the different stages of product life cycle and does not support expressing product information about the purpose, function, behavior, and so on [7]. In [13], the product model for MCD is expressed in STEP, but it doesn't better express the function and behavior information. An integrated model to support distributed collaborative design is proposed in [14], which mainly comprises a semantically-rich, object-oriented database that forms the basis for shared design decisions; however, an integrated model isn't suitable for MCD because the different disciplines observe the product model from different views and differ in the concerns of the model data each other.

The model management is concerned about the maintenance and management of the model data during the model evolves. In [15], constraints are used to represent engineering requirements. Solving these constraints results in spaces of feasible values of design variables. Such spaces improve efficiency through avoiding artificial conflicts, enhance change management, and support for collaboration design. However, the research only considers numeric type constraints and doesn't involve other type constraints. In [16], design information is stored as wrapped objects to realize the seamless integration with a CAD system and an object-oriented database (OODB) via an interfacing executable. Through version and configuration management, the alteration in the CAD would automatically update design information to maintain the consistency among the designers. In [17], the coordination among various actors is realized by defining a common repository for knowledge management in a collaborative design situation. In these researches, a common product database is modeled at some location and is shared by many designers located at different places. In MCD, the

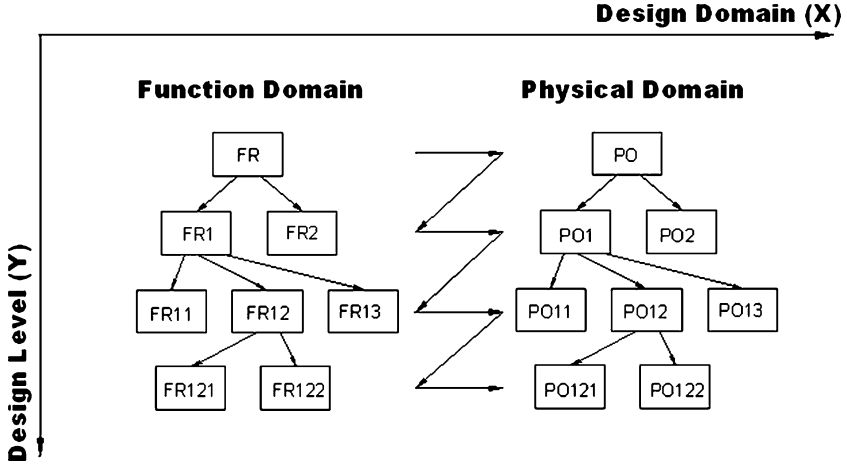

Fig. 1 Axiom design theory

different disciplines build their own databases at different places, and some new methods are needed to maintain the consistency among the distributed databases during design process.

Synoptically, most of the existing models don't effectively merge the design semantics information (such as product function, behavior, technical principle, and designer's knowledge rules, etc.) that reflects designer's design intents; lack flexibility and dynamically modeling ability; and lack collaborative management information (such as designer alliance and component organization, etc). These problems make the existing models unable to effectively support the MCD. To solve these problems, this paper proposes a new product information model for supporting the MCD and the corresponding product modeling methods.

2 Analysis of MCD

In MCD, each discipline concerns some functions of product and makes product design with its own specialized knowledge and criteria, going through a function-behavior-structure mapping inside its own discipline domain. Thus, based on [1, 2], we think that the MCD is a concurrently mapping process from function domain to physical domain in multiple discipline domains, and that the mapping between two discipline domains is also one bi-directional mapping process (as shown in Fig. 2). Because they develop the same product, there must exist the interrelations among the multiple disciplines at semantic level and physical level (as shown in Fig. 3).

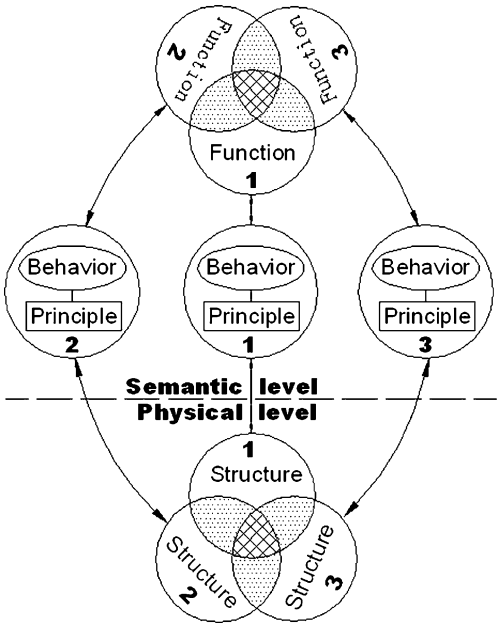

Fig. 3 The interrelations among the multiple disciplines

From Fig. 3, the shade areas are the common concerns of two or more disciplines, which involve functions and structures, and the corresponding physical structure attributes should be jointly determined by all the related disciplines; some blank area is the concern of some

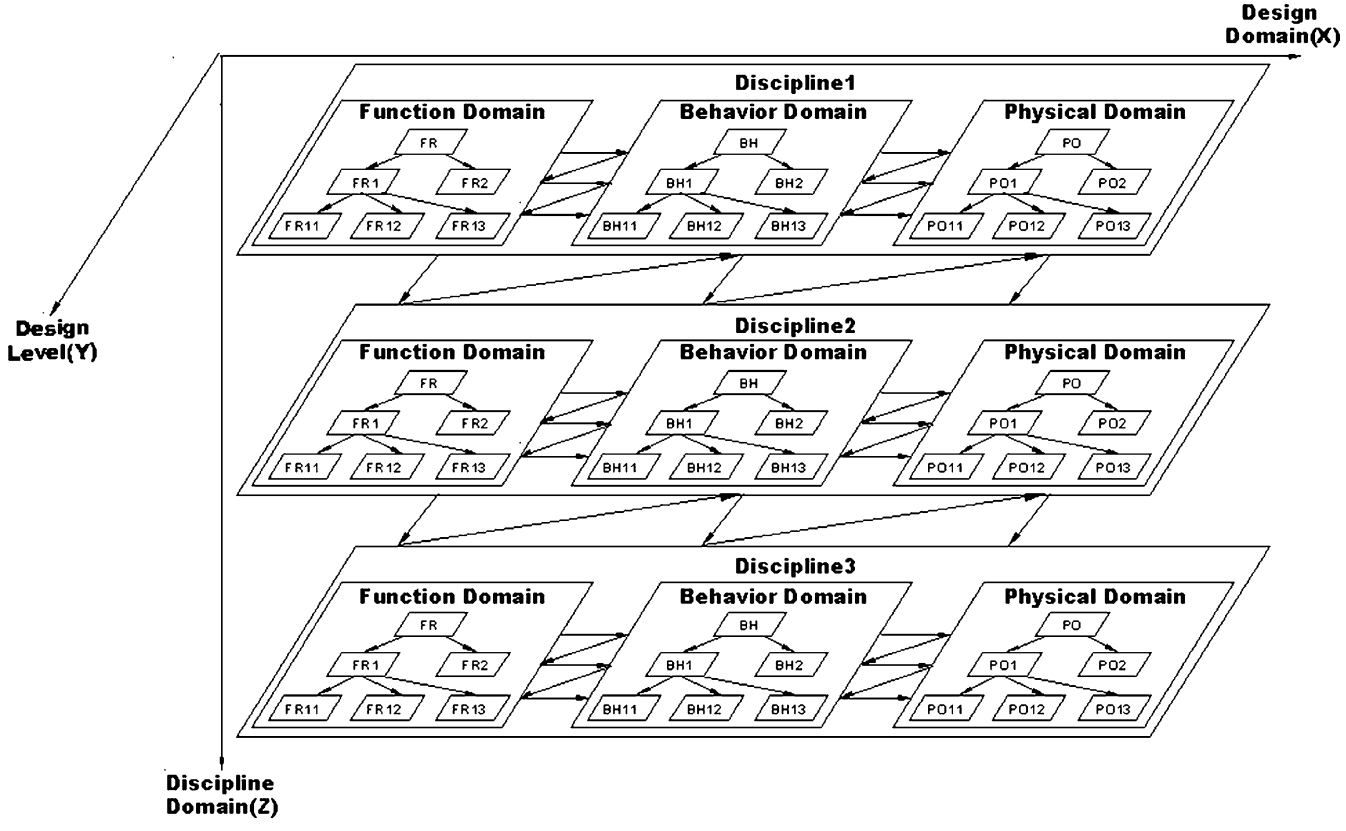

Fig. 2 The three-dimensional framework of the MCD

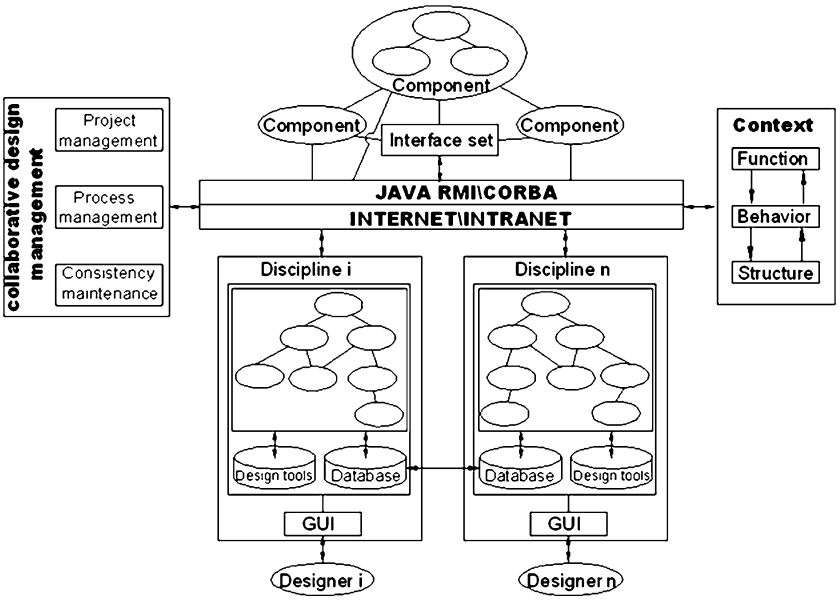

Fig. 4 The system architecture for MCD

discipline, which isn't necessarily shared by the other disciplines, and the corresponding physical structure attributes are independently determined by this discipline. A physical object's behavior is a description of the object's action in a certain circumstance, such as the structural behavior and kinematic behavior. Behavior can be generally illustrated by the working principles of the physical object, such as mechanics, thermotics and tribology principles, etc. The working principle can be used to analyze the performance of the physical object, such as tribology performance and strength performance. Thus, in MCD, behavior reflects discipline's performance requirement on the physical object or product and is strictly a discipline-based concept. Only after all the disciplines' performance requirements on product are satisfied, can the product functions be achieved. For example, the function of gear pair is power transmission, the mechanics discipline has strength performance requirement on gear pair, and the tribology discipline has lubrication performance requirement on gear pair. Only after the strength performance requirement and lubrication performance requirement on gear pair are satisfied, can the power transmission function be achieved. Function, behavior, and technical principle, etc reflect the designer's intents, and are called semantic data and classified into semantic level. The product design result is finally exhibited in form of physical structure that is generated under the drive of the designer's intents, and should be the intersection of multiple disciplines' design results. All the physical structure attributes of product are classified into physical level. The semantic data can enhance the understanding and coordination on product model among multiple disciplines.

Besides, as the MCD is put in practice under network environment (see Fig. 4), the collaborative management is also very important, which mainly involves the project management, process management and consistency maintenance.

The project management: It sets up the design project, establishes the designer alliance, and determines the component organization through product configuration. The main concerns are the information about the designers and components that are related to the design project.

The process management: It controls the proceeding of design process from the initial specification to the final goal,

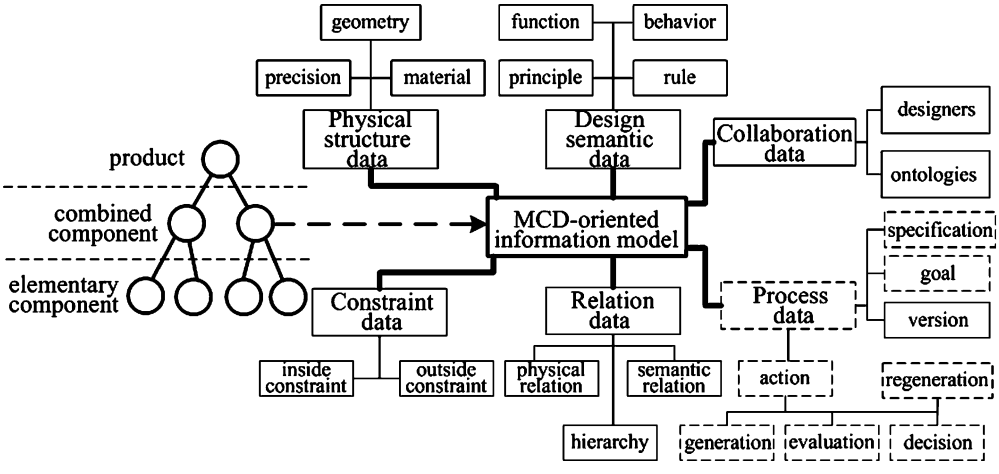

Fig. 5 The MCD-oriented product information model

and involves the information about the associated design activities, such as some process objects (specification, which represents user input; context, current state or version of product model data; and goal, expecting state of product model data in design process.) and some design actions (generation, evaluation (local evaluation for a single discipline and global evaluation for all disciplines), decision and regeneration (update, replace, add and delete), etc.).

The consistency maintenance: It maintains the consistency of product model among multiple disciplines, which is realized by various constraints and relations among the components and multiple disciplines at semantic level, physical level and management level.

Synthetically, to effectively support the MCD, it is necessary to build a new product information model that merges semantic data, physical structure data and collaborative management data.

3 The MCD-oriented product information model

A product consists of multiple components. The component is divided into two types: the elementary component that is at the lowest level in product hierarchy tree and the combined component that is the combination of some components. The combined component at the highest level in product hierarchy tree is the product. Based on the above analysis of MCD, this paper proposes a MCD-oriented information model as shown in Fig. 5.

Physical structure data describe the physical structure information about a product and its components, including the geometry attributes (topology structure and dimension, etc), the precision attributes (roughness and dimension tolerance, etc), and the material attributes (material type, material trademark, rigidity, Young's modulus, and Poisson's ratio, etc).

Design semantic data describe the semantic information of product design, including the product function, the technical principle to realize the function, the behavior to respond to the technical principle, and the designer's knowledge rules, etc. Let's take gear pair as an example, in which, the function is power transmission; the corresponding principles are the calculation principles of contact fatigue strength and bend fatigue strength; the corresponding behaviors are contact bearing capability and bend bearing capability; and the related rules are (IF (bend fatigue strength is insufficient) THEN (increase module)) and (IF (helix angle of tooth is too big) THEN (increase tooth number of pinion gear)), etc.

Relation data depict the relation network among all components in product model and are shown in Table 1.

Constraint data describe the constraint relationships among the attributes inside one component (they are called inside constraints, such as $z \geq 17$ for a gear) or the constraint relationships among the attributes among different components (they are called outside constraints, such as gear1.m=gear2.m for a gear pair). These constraints restrict the feasible value range of the attributes, and maintain consistencies among the different attributes.

Collaboration data describe the information of the ontologies and the designers that are related to the component. Due to the fact that multiple designers may be concerned

Table 1 Relation data

Description		Relation name	Example
Hierarchical relations (see also Fig. 2)	Structure hierarchical relation	Part of Has part	The physical structure "gear" is Part-of the physical structure "gear–pair".
	Function hierarchical relation	Subfunction of Has subfunction	The function "supporting-gear" is Subfunction-of the function "power transmission".
	Behavior hierarchical relation	Subbehavior of Has subbehavior	The behavior "contact fatigue strength" is Subbehavior-of the behavior "fatigue strength".
Semantic relations	The relation between function and structure	Satisfy Satisfied by	The function "supporting-gear" is Satisfied-by the physical structure "axis".
	The relation between function and behavior	Assure Assured by	The function "supporting-gear" is Assured-by the behaviors "bending-resistance" and "twisting-resistance" of the physical structure "axis".
	The relation between behavior and structure	Achieve Achieved by	The behavior "bending-resistance" is Achieved-by the physical structure "axis".
Physical connection relations	The physical connection relation between two physical components	Connect Connected by	The physical component "gear1" Connects the physical component "gear2" by a meshing relation.

on component, the ontology data about the component should be introduced so as to make the designers in the different disciplines have the common understandings on the component. The ontology data are the unified definitions on some basic concepts of the component, such as name, communication address (IP address and port No.), some basic concepts of the component and the corresponding explanations and values. For example, to a gear, the ontology data involves tooth number, pressure angle, module and helix angle that should be same to all designers. The information of the designers related to the component should be recorded in the component model, which can be used to determine the designer alliance related the component, such as the designer's name, address (IP address, port No. and email), authority (modify, suggest, respond and browse) and domain, etc.

Process data describe the design activities operating on the product data. The activities generate the information of product model, and the generated information is also used by process data to generate new information of product model, so repeatedly, until the final results are got. Due to the space limitation, this paper doesn't discuss the other process data (in dot line frames as shown in Fig. 5) except for the version data. The version data capture the evolutionary history of the product model and contain the information such as modifier, modified content, modified time, modified reason and version No., etc.

4 Representation of the MCDPM

Considering a component is shared and operated by multidisciplinary designers in network environment, the representation of the component model should have good modularity, inheritable ability, flexibility and expansibility. Based on [9], a new extended object-oriented method is proposed to represent the MCDPM. According to this method, a component model is presented as a class, namely component class (CC), and each attribute of CC is also defined as a class, namely attribute class (AC). Each AC can be independently defined from the CC, and has own inheriting hierarchy. Thus the MCDPM is modularization architecture, and easy to be organized and modified. A class can be continually refined through inheriting mechanism to generate the different levels of classes. If the class at some level has reached the specified requirements, the instance object at this level can be generated through instancing the class. As required, a new AC can be also added into a CC. In this way, the MCDPM can be recursively extended transversely and lengthwise by inheritance and AC addition, and the evolution and expansion of product model can be realized.

The CC is defined as follows (the item in {} can be repeated 0 or many times; the item in $\{\}^{1}$ can be repeated one or many times; the item in [] is choice (namely, it can be or not be selected)):

$$CC = (Cid,\ PCid,\ \{A\}, \{R\}, \{M\})$$

Where, Cid is the unique identifier of the class. PCid is the unique identifier of the parent class of the class.

{A}is a set of attributes of the CC. As shown in Fig. 5, {A}={geometry, precision, material, function, principle, behavior, rule, designer, ontology data, version}, and each attribute is also defined as a class AC.

{M}is a set of methods of the class. Each method is a three-tuples, M=(type, name, {ptype}), where type is return value type; name is method name; and ptype is argument type.

{R}is a set of relations about the component, and R is also defined as a class, R=(RCid, PRCid, {A}, MO, $\{RO\}^{1}$, {C}), where, the meanings of RCid, PRCid, A and C are the same as the above; MO is the main object in the relation; RO is the object relating to MO. Both MO and RO are a three-tuples (type, name, role), where type is the object type (such as functional object, behavioral object, structural object and so on); name is the object name; role is the role that this object plays in the relation, and the role should be set according to the application domain. A relation at least involves two objects, one MO and at least one RO.

The AC is defined as follows: AC= (ACid, PACid, {A}, {M}, {R}, {C}).

Here, Acid, PACid, M, and R are same to CC.

{A}is a set of attributes of the AC, and each attribute is a three-tuples, A=(type, name, [value]), where name is a unique identifier of the attribute; type is data type of the attribute; value is the default value of the attribute. Furthermore, each attribute here can be also represented as a class, thus a component or product can be defined recursively without a pre-defined granularity.

Table 2 Gear component class (segment)

Cid : CGear				
	ACid	{A}	{M}	{C}

{A}	CGearGeom	int z; // teeth number
		float m,d,d_a,d_f,...;//module, diameter, etc.
			float cal_d (float,int);
			float cal_d_a (float,...);
			float cal_d_f (float,...);
				(c1, z>17)
				(c2, m ϵ {1,1.5,...})

	CGearOnto	string teeth-number="the number of a gear teeth, it should be greater than 17 to avoid the undercut, and ...", helix-angle="...", ...	string add();
			string del();
			string modi();	

Table 3 Function attribute class of gear pair component (segment)

ACid : CGearPairFunc			
{A}	{M}	{R}	{C}
string func1="power transmission";	string set_fun(string); string get_subfun();	nil	nil

{C}is a set of constrains that define the relationships among the attribute parameters in one component or different components, and C is defined as C= (name, content), where name is the unique identifier of the constraint; content is the constraint expression.

An example for gear pair is given as follows (Tables 2, 3, 4, 5, 6).

5 The discipline view model (DVM)

5.1 General description of DVM

The MCDPM gives a general description on the product or its components. Different discipline differs in viewing the product or its components. One discipline builds its own discipline model of the component with its own professional knowledge from its own view, which is called the discipline view model [18]. The DVM is the mapping of the MCDPM in the discipline domain. The mapping has two meanings:

(1) Generate the component discipline class (CDC) by inheriting the attributes of the CC and adding the own attributes of the discipline.
(2) Generate the instance of the CDC according to the discipline's design requirements.

The combination of the CDCs of all the components that are the concern of this discipline and the corresponding instance objects constitutes the discipline view model (DVM) of the product. Each discipline participating in product design builds its own DVM, then all the DVMs form multi-views model of the product (see Fig. 6).

In Fig. 6, the gear drive is taken as an example for analysis. It can be seen in Fig. 6 that all the CCs may distribute at different locations and form a distributed CC library, namely, scattered physically, but concentrated logically. The distributed CC library is independent of the disciplines, and is used as shared resources for all the disciplines to select. The multiple disciplines also distribute at different locations, and communicate, negotiate, and cooperate with each other by Internet, then form a distributed multidisciplinary collaborative design team.

Based on the gear drive design project configured by project management module, each discipline confirms the components it concerned, and then inherits relevant CCs in the distributed CC library and adds its own attributes to generate its own CDCs, and then instantiate the CDCs to generate the instance objects. The components concerned by some discipline are not always the same as the ones concerned by other disciplines. For example, the axis component is the concern of strength discipline and structure discipline, but not the concern of tribology discipline.

As shown in Tables 7 and 8, the CDC of gear pair in strength design discipline CGearPair_CDC_Stre is generated by inheriting the CGearPair (see Table 5) and adding its own attributes (only the added attributes are given, and the inherited attributes are omitted). Similarly, the CDCs of gear pair in other disciplines can be also generated.

5.2 The class inheritance in network environment

As stated above, the CCs distribute at different Internet nodes, and the different disciplines also distribute at different Internet nodes. A CC can be located through specifying its IP address and port at discipline design node or using HTML hyper-link relations among CCs (see Fig. 7).

The CCs and CDCs are all built with Java language. As a CDC is generated by inheriting the corresponding CC in internet environment, the class inheritance method in the network environment needs to be realized. It is implemented through binding Java with XML [19]. The designer at some discipline design node can make some operations on XML files by Web Browser, the conversion between XML document and Java document is done by the programs (Marshalling program is used for converting Java into XML and Unmarshalling program is used for converting XML into Java), and the detailed steps are shown in Fig. 8.

5.3 The coordination and consistency maintenance among multiple DVMs

The coordination and consistency maintenance among multiple DVMs is the key for MCD. As stated above, each

Table 4 Ontology attribute class of gear pair component (segment)

ACid : CGearPairOnto			
{A}	{M}	{R}	{C}
CGearOnto gearonto1; string center_distance= "...", ...	string find(); string add(); ...	nil	nil

Table 5 Geometry attribute class of gear pair component (segment)

ACid: CGearPairGeom							

{A}	CGearGeom gear1, gear2; //gear objects
	float a, i, ...; //center distance, transmission ratio, and so on.						

{M}	float cal_i(int,int); float cal_a(); ...//the functions to calculate a, i, ...						
{C}	(c1, i>3), (c2, a<160), ...//the constraints about a, i, ...						
{R}	RCid	MO			{RO}[1]		
		type	name	role	type	name	role
	Has_part	structure object	gearpair	assembly	structure object	gear1,gear2	component
	Connect	structure object	gear1	active	structure object	gear2	passive

DVM consists of some correlated components, therefore, the coordination among the different DVMs is mainly the coordination among these components in the different DVMs. In the following part, several cases of coordination among different DVMs are discussed.

5.3.1 Coordination among the attributes in one component

The constraints defined in the component are used to maintain the consistency of the attributes. For example, there are some constraints such as d= m*z, $d_{a=}$ d+2*h_a, z≥17 and so on in the gear component. Changing teeth number z will cause the corresponding changes of d and d_a to ensure the consistency among z, d and d_a.

5.3.2 Coordination among the different components in the same DVM

By means of the structure hierarchical relation Has_part, the physical connection relation Connect and the constraint data, the coordination among the different components in the same DVM can be realized. The hierarchy management method is adopted to manage the components, namely the upper component takes charge of managing its lower components. For example, there exist the connect relation, the has_part relation, and the constraint c1, c2 among axis1, gear1 and shafting, which form a relation network, so the changing of axis1 will cause the corresponding changing of gear1 to ensure the consistency between axis1 and gear1 through such relation network (see Fig. 9). The constraint c1 and c2 between axis1 and gear1 are defined in their upper component shafting.

Besides, referring to [20], the coordination among the components can be also realized through semantic reasoning at semantic level, which generally goes through the bottom-up process and the top-down process (see Fig. 10).

(1) The bottom-up process: The changing of some structure attributes of component A (step 1) will cause the changing of its related behavior attributes (step 2), and then will influence its corresponding functions (step 3). For example, in the Fig. 6, the changing of reference circle diameter of gear1 will result in the changing of its loading behavior, and then will result in the changing of its power transmission function.

(2) The top-down process: Based on the function changing of component A, the component B whose function is relevant to the changed function of component A can be determined by semantic reasoning (step 4). In this way, the corresponding function of component B can be determined, and then the corresponding behavior is determined (step 5), and then the corresponding structure attributes are determined (step 6). These structure attributes of component B are updated (step 7) to coordinate with the changed structure attributes of component A (step 8), and finally, the consistency between component A and component B is maintained (step 9). Continuing above example, the component B is determined to be axis1 whose function is power and torque transmission by semantic reasoning, and the corresponding behaviors mainly include the bending _resistance and the twisting_resistance, and then the corresponding structure attributes are axis1.d and axis1.l (see Fig. 9), and finally, these structure attributes are updated to coordinate with the changed reference circle diameter of gear1.

Table 6 Gear pair component class (segment)

Cid : CGearPair							
{A}	CGearPairFunc fun1; CGearPairGeom gearpair1; CGearPairOnto gearpaironto1;						
{M}	float get_a(); float get_i(); string get_fun(); string get_onto(); boolean check_constraint();...						
{R}	RCid	MO			{RO}[1]		
		type	name	role	type	name	role
	Satisfied_by	function object	fun1	goal	structure object	gearpair1	form

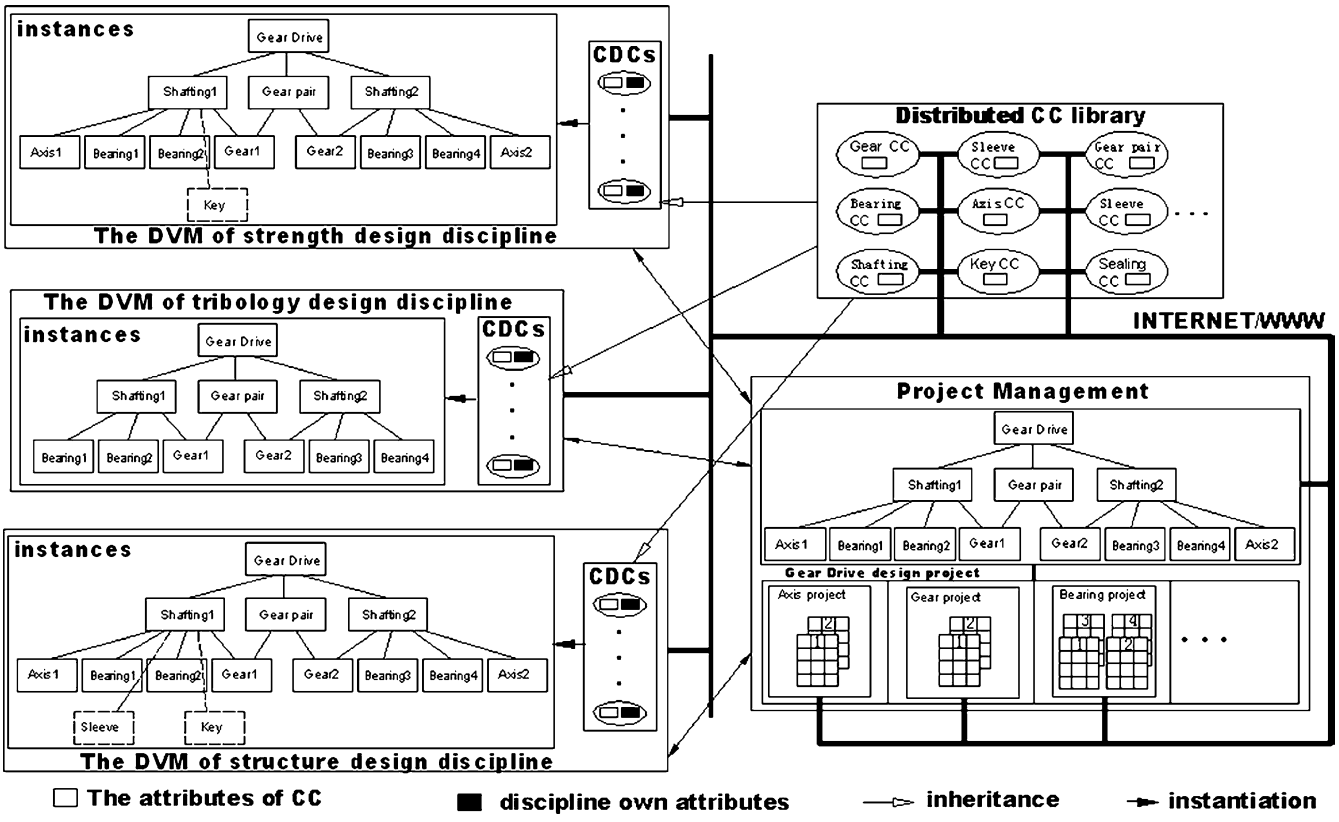

Fig. 6 Product modeling for MCD in network environment

This semantic reasoning method is more flexible, and is similar to the human thinking and reasoning process.

5.3.3 Coordination among the same components in the different DVMs

As shown in Fig. 6, some components such as gear, bearing, axis and so on are the common concerns of multiple disciplines and appear in the corresponding DVMs. As stated in Section 3, the MCDPM contains the information about the designer alliance related one component, such as the designer's name, address (IP address, port number and email), authority and domain, etc. After some discipline inherits one CC and instantiates it, the information about this discipline will be recorded in the data table of this component project in project management module. By the data table, the different DVMs that are related to the same component are correlated each other and the coordination among the same components in the different DVMs can be realized. In Fig. 11, gear1 is the common concern of three disciplines, and the information about three disciplines is recorded in the data table in gear 1 project. For example, when strength discipline makes some operation on gear1, according to the information about the designer alliance in gear project table, the operation information will be automatically transmitted to the structure discipline and tribology discipline by project management module, and then the latter two disciplines respond to the operation to maintain consistency.

Table 7 Gear pair component discipline class in strength discipline (segment)

Cid : CGearPair_CDC_Stre							
PCid: CGearPair							
{A}	CGearPairBeha_Stre beha_stre1;						
{M}	float get_bstress(); //get bend stress value of gear						
	float get_cstress ();//get contact stress value of gear						
{R}	RCid	MO			{RO}[1]		
		type	name	role	type	name	role
	Assured_by	Function object	Fun1	Goal	Behavior object	Beha_stre1	Performance
	Achieved_by	Behavior object	Beha_stre1	Performance	Structure object	Gearpair1	Form

Table 8 Behavior attribute class of gear pair component in strength discipline

ACid: CGearPairBeha_Stre	

{A}	string stre_beha="contact fatigue strength and bend fatigue strength";
	float cs,bs; //contact stress and bend stress
	float pcs,pbs; //permissible values of contact stress and bend stress

{M}	float cal_cs(); float cal_bs();//the functions to calculate cs and bs
{C}	(c3, cs<pcs), (c4, bs<pbs); //the constraints about contact strength and bend strength

5.3.4 Coordination among the different components in the different DVMs

In this situation, the coordination among the different components in the different DVMs can be realized by combining the methods described in Sect. 5.3.2 and in Sect. 5.3.3. In other words, when one component in some DVM is changed, the other relevant components in the same DVM are first determined and updated based on the method in Sect. 5.3.2, and then the relevant components in the other DVMs are determined and updated based on the method in Sect. 5.3.3.

5.3.5 Function semantics-based dynamic coordination among the different DVMs

The above situations belong to the static correlation among the components, that is, the relations among the components have been set up when the product model is built. When one component is changed, by following the existed various relations among the components, other relevant components are activated and updated to maintain consistency.

Besides, there is also a kind of dynamic coordination situation, namely, some discipline dynamically adds some new components into its DVM for considering itself requirements, then how will the other disciplines respond? This kind of dynamic coordination can be realized by the function semantics of the new added components, and the key is whether these new added components in some DVM have the function semantics that are the concerns of other disciplines (as shown in Fig. 3, some components have some common functions concerned by multiple disciplines). For example, as shown in Fig. 6, when the structure discipline adds a key (in a dot line frame) that has two functions (the circumferential fixing and transferring torque) into its DVM, other disciplines examine the function semantics of this component. The strength discipline finds that the transferring torque is its concern, because the torque is directly related to the strength of key, namely the shearing strength and the crushing strength, so adds the key into its DVM; however, the tribology discipline doesn't concern the two functions of the key (here, the key is fixing connection), so doesn't add the key into its DVM. To the sleeve, its main function is to realize the axial fixation of parts on the axis. When the structure discipline adds a sleeve (in a dot line frame) into its DVM, the strength discipline and the tribology discipline don't concern this function of sleeve, so doesn't add the sleeve into their DVMs.

Furthermore, if a new performance analysis needs to be made, the corresponding DVM should be added, such as the DVM of vibration and noise analysis for gear drive is added into the system. Thus, this modeling method supports dynamic product modeling, and such product modeling system is open and flexible and can adapt to dynamic design changes.

6 Conclusions

This paper mainly discusses the problem of product modeling for MCD, including the MCD-oriented product information model (MCDPM), the representation of the MCDPM, the discipline view model (DVM), and the coordination and consistency maintenance among multiple DVMs, etc.

Besides usual physical structure data, the MCDPM also contains design semantic and collaboration management data. The mergence of semantic data, physical structure data and collaboration management data will effectively support the MCD.

The MCDPM is built by using an extended object-oriented method. Each component is defined as a component class (CC), the CC's attribute is also defined as a attribute class (AC), and the AC's attribute can be also defined as a class, thus the MCDPM can be defined recursively without a pre-defined granularity. On the one hand, the attribute classes can be defined independently and be inherited continually; on the other hand, new attribute classes can also be added into the MCDPM according to the needs, thus the MCDPM has good flexibility, expansibility, and evolution ability.

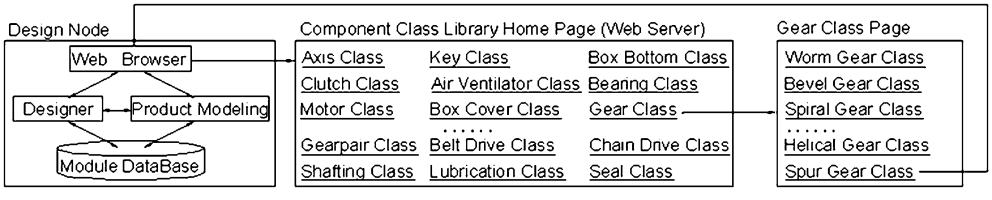

Fig. 7 Hyper-link relations among CCs

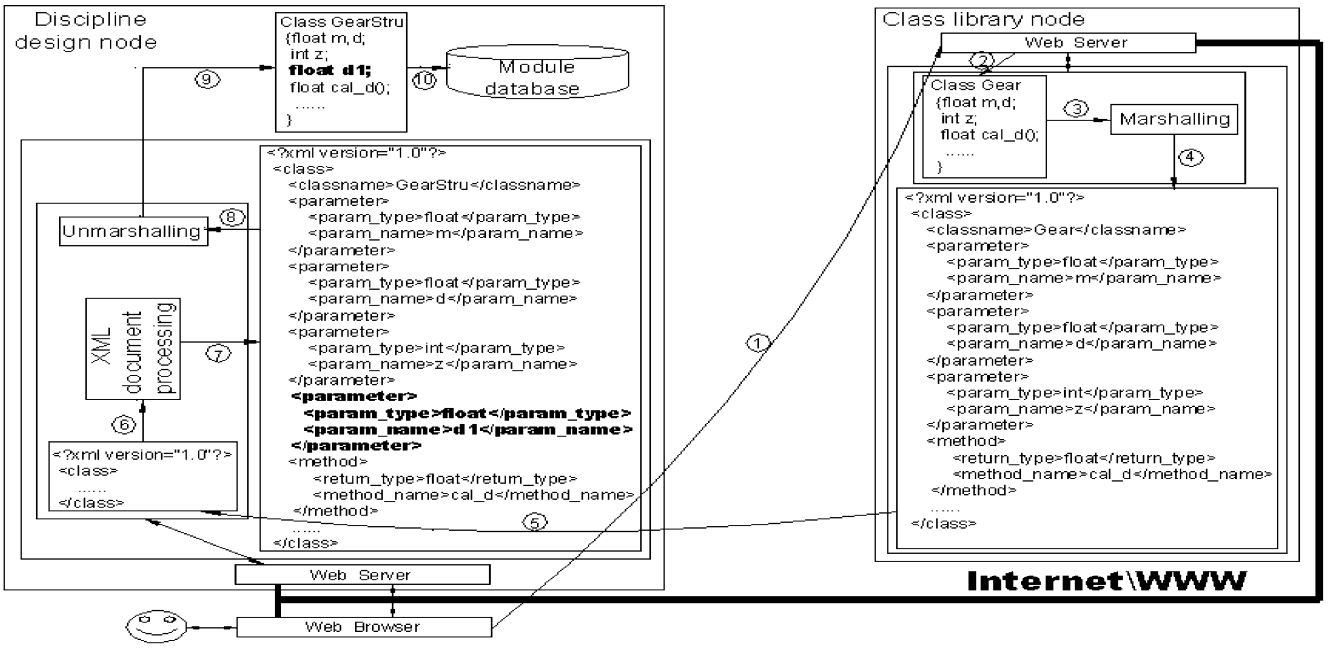

Fig. 8 The class inheritance in distributed network environment

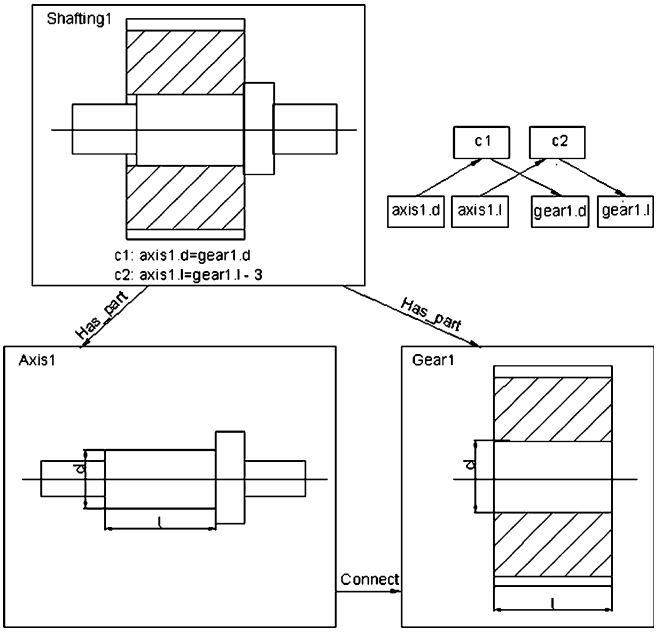

Fig. 9 The consistency maintenance at physical structure level and management level

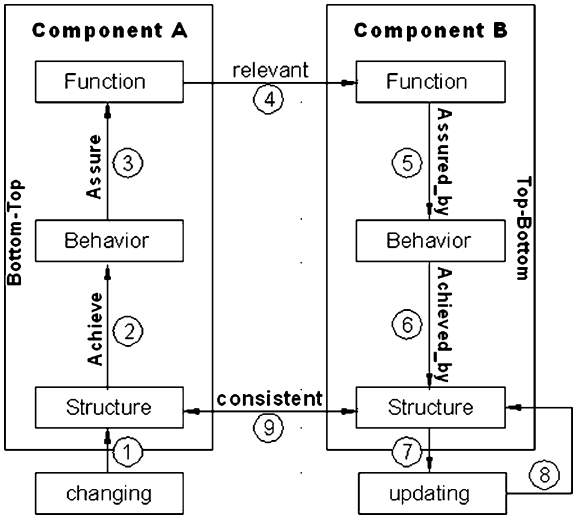

Fig. 10 The consistency maintenance at semantic level and management level

The component discipline class (CDC) is generated through inheriting the attributes of the CC and adding the own attributes of the discipline. The combination of these CDCs and their instances forms the DVM of the product. Consistency maintenance among the DVMs is one key technology of MCD. This paper analyzes and discusses the consistency maintenance problem from five aspects.

We have already developed a prototype system, and take cylindrical gear reducer as an example to check the system's validity. The following issues will be further addressed in the future work:

(1) Design operation propagation: When some discipline makes some operations on the product model, the operation information will propagate to the other relevant disciplines, and then the relevant disciplines make the corresponding operations to maintain consistency among the DVMs. The system currently adopts a simple mechanism to control operation propagation, that is only one discipline can operate the product model at a time, and other disciplines are responders. But in fact, many disciplines maybe operate the product model concurrently, and this will

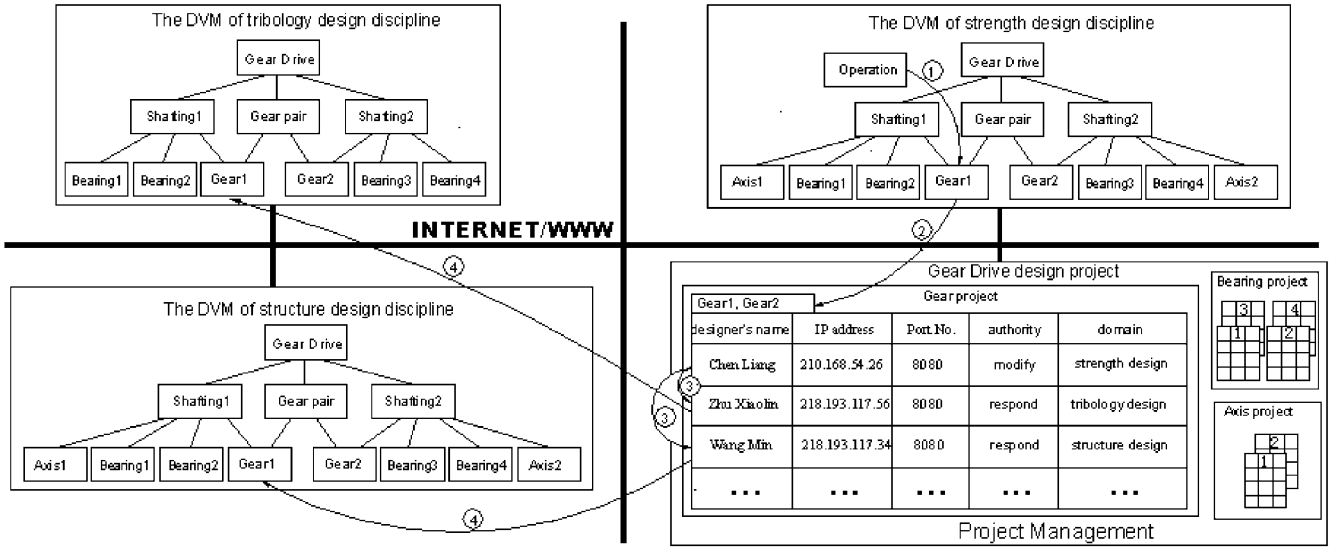

Fig. 11 The consistency maintenance at management level

bring the problems about concurrent operation and dead-locking. The problems are currently being studied and will be introduced in another paper.

(2) Intelligent negotiation mechanism: Because different disciplines differ in goals and knowledge, it is unavoidable that conflicts occur in the design process. Different disciplines need to negotiate each other to resolve the conflicts and reach consistency. We are currently developing the intelligent negotiation system on the basis of fuzzy decision-making and game theory.

Appendix

A notation list of basic terminologies

MCD	multidisciplinary collaborative design.
MCDPM	MCD-oriented product information model.
CC	component class.
AC	attribute class.
CDC	component discipline class.
DVM	discipline view model.

References

1. Paul G, Beitz W (1984) Engineering design. Springer, Berlin Heidelberg New York
2. Suh NP (1990) The principles of design. Oxford University Press, London
3. Yehuda EK (2001) Enhancing multi-disciplinary collaboration through semantically rich representation. Autom Constr (10):741–755
4. Zhang S, Shen WM, Ghenniwa H (2004) A review of Internet-based product information sharing and visualization. Comput Ind 54:1–15
5. Case K, Gao J (1993) Feature technology: an overview. Int J Comput Integr Manuf 6(1):2–12
6. Wang H.F, Zhang YL, Cao J et al (2003) Feature-based collaborative design. J Mater Process Technol 139:613–618
7. Mark JC, John CK, Martin AF (1996) Rapid conceptual design evaluation using a virtual product model. Eng Appl Artif Intell 9(4):439–451
8. Roy U, Pramanik N, Sudarsan R et al (2001) Function-to-form mapping: model, representation and application in design synthesis. Comput Aided Des 33:699–719
9. Gorti SR, Gupta GJ, Sriram RD et al (1998) An object-oriented representation for product and design processes. Comput Aided Des 30(7):489–501
10. Zhang F, Xue D (2002) Distributed database and knowledge base modeling for concurrent design. Comput Aided Des 34:27–40
11. Pahng F, Senin N, Wallace D (1998) Distribution modeling and evaluation of product design problem. Comput Aided Des 30 (6):411–423
12. William C (1995) The STEP integration information architective. Eng With Comput 11:136–144
13. Wang FC (1997) Multidisciplinary collaborative design environments for concurrent product design, Dissertation, University of California, Berkeley
14. Yehuda EK, Lachmi K, Jin WC (1998) An integrated model to support distributed collaborative design of buildings. Autom Constr 7:177–188
15. Lottaz C, Smith IF, Nicoud YR et al (2000) Constraint-based support for negotiation in collaborative design. Artif Intell Eng 14:261–280
16. Miles JC, Gray WA, Carnduff TW (2000) Versioning and configuration management in design using CAD and complex wrapped objects. Artif Intell Eng 14:249–260
17. Yesilbas LG, Lombard M (2004) Towards a knowledge repository for collaborative design process: focus on conflict management. Comput Ind 55:335–350
18. Chen L, Jin GD (2004) Research on product multidisciplinary collaborative design based on multiple views model. J China Mech Eng 15(23):2092–2096
19. McLaugblin B (2003) Java and XML data binding, simplified Chinese edn. O'Relly and China Electric Power Press
20. Rosenman MA, Wang FJ (2001) A component agent based open CAD system for collaborative design. Autom Constr 10:383–397