DETC2010/MECH-28492

A PORT-BASED AGENT APPROACH TO GUIDING CONCEPT GENERATION FOR CUSTOMIZING MODULAR VARIETIES

Dongxing Cao*

Mechanical Design & Application Lab
Department of Mechanical Engineering
Hebei University of Technology, Tianjin 300130, China

Ming Wang Fu

Department of Mechanical Engineering
The Hong Kong Polytechnic University, H.K. China

Karthik Ramani

Computational Design Innovation Lab
School of Mechanical Engineering
Purdue University, West Lafayette, 47907, USA

Runli Zhang

Mechanical Design & Application Lab
Department of Mechanical Engineering
Hebei University of Technology, Tianjin 300130, China

ABSTRACT

As the description of design requirements at the earlier design stage is inaccurate and vague, it is difficult to figure out functional structure of a product and make sense product configuration. Therefore, it plays an important role to formally represent the process of design for product development in the conceptual design stage. Furthermore, port, as the location of intended interaction, is crucial to capture component concept and realize conceptual design for multi-solution generation. Agent is considered as an effective approach to collaboratively implementing design problem solving and reasoning. Combining both port and agent may be employed to generate new concepts of the product in order to customize product scheme varieties. In this paper, the product module attributes are firstly described. The objective is to implement modeling of design process for obtaining system new concepts to guide multi-solution generation. Secondly, an effective approach to decomposing design process is presented to describe the process of structure generations and product decomposition by formal representation. According to properties of modularity for product development and component connections, we can calculate the number of component connections and density of components. In addition, product module division and coupling degree analysis are conducted, and coupling degrees are calculated by considering the correspondence ratio and the cluster independence. A port-based knowledge building process is described for functional modeling. A port-agent collaborative design framework is given and describes different agent functions to help designers to obtain new design schemes. Finally, a case study is presented to describe the modeling process of conceptual design.

INTRODUCTION

In today's highly changing market, product need is one of the most powerful driving factors of design. This makes the conceptual design to be full of challenging [1]. Product variety mainly focuses on serving the needs of individual customers. Enterprises are forced to seek more efficient and flexible product design and manufacturing strategies [2]. Port, as an effective approach to modeling the components and representing their interface, has been paid extensive attention in the product conceptual design. It is convenient to link multi-components into a system and satisfy user's requirements [3-4]. In engineering design today, product development is a team effort from multi-discipline experts. They work together and closely collaborate with each other in the same design problem will accelerate product development. Agent technologies may provide a promising solution to product design problems. Multi-agents may work together by identifying some sub-problems to solve effectively a design problem through collaboration [5]. However, they may come from different viewpoints for solving and integrating concurrently design results. As agent has the ability to make decision on behalf of human beings, some researchers try to build an intelligent agent model in order to handle various types of fuzzy information during the early design stage. Miao *et al.* [6] presented an intelligent agent model, which has the ability to model, reason and solve problems on behalf of human beings. Tang [7] built an agent-based collaborative system to integrate die-maker's activities into customer product development. The system can assist die design personnel to communicate conveniently, to ensure the part stamped in a cost effective manner during the early stage.

* Professor and author of correspondence, Phone: (086) 022-60204935, Fax: (086)010-950507 to 716480, Email: dongxingcao@gmail.com.

As module design easily realizes one-to-one mapping between function concepts and physical components, it is clearly separated by subsystem boundary or interface in order to reduce the complexity of the product design, facilitate the reuse of subsystems across a product boundary. Product scheme varieties can be generated by combining different modules. Therefore, different module arrangement and combination can customize the product varieties [8].

With the development of technologies, the product functions become more and more complexities. And the existing methodologies are difficult to design and adapt to this changes of complexity [9]. A new approach is appealed to solving complex design problems. This paper will attempt to build a new framework that addresses this issue by including a port-based agent approach. The remainder of the paper is structured as follows. Section 2 presents related work. Module attributes of a product will be described in Section 3. Component connection and its density as well as coupling degree calculation and decoupling design process are given in Section 4. Section 5 gives port-based ontology for product modeling. Section 6 describes port-agent framework for product varieties. A case study and conclusions are given in Section 7 and 8, respectively.

RELATED WORK

In the past decades, we developed a product and usually followed a set of specific principles, design criterion, and phases to build ontology concepts and their relationships. The absence of structured guidelines and methods hinders the development of shared and consensual ontologies [10]. The main reason is that domain experts and customers have no understanding of formal expression in ontology languages. The results of research show that, using the ontology language, experts and customers could gain a full understanding of product concepts and validate their taxonomies [11]. However, they were unable to understand abstract definitions of concepts, relations, functions, and axioms. From the knowledge acquisition viewpoint, they are quite unable to formalize their knowledge [12].

The modularity has been widely used in different contexts, ranging from manufacturing to design of electrical and mechanical product and software. It refers to product, processes, and resources that fulfill various functions through the combination of distinct building blocks [13]. Dahumus et al. [14] presented an approach to architecting a product family that shares interchangeable modules. They developed function structures for common and unique functions. Then rules are applied to determine possible modules. In addition, ontology can not only be expressed in a formal logic form, but also made detailed, accurate, consistent and meaningful explanation among the concepts and relations [15]. Moreover, formal logical form is appropriate for semantic representation in the product development.

Port plays an important role for component concept generation. It constitutes the interface of a component and defines its boundary [3]. Singh & Bettig [16] defined the concept of assembly ports as one or more low-level geometric entities that undergo mating constraints in order to join parts, adopted the port-based composition to describe the hierarchical configurations of complex engineering design,

and realized assembly design through deciding port compatibility and connectability. Breedveld [17] described port as the 'point' of interaction of a system, subsystem or element with its environment in order to realize the port-based modeling of dynamic systems on the basis of bond graphs. Campbell et al. [18] developed a functional representation based on the ports of connectivity with other components to describe how energy and signals are transformed between ports.

The different system architectures can be classified into three categories: centralized approach, federation approach, and autonomy-based approach [19-20]. The centralized architecture probably is the most widely applied approach for developing the network-oriented collaboration systems. Next, the federation architecture is a distributed system approach, where usually there is no central module. Instead, this approach coordinates multi-agent activities via facilitation as a means of reducing overheads, ensuring stability, and providing scalability. Communication and coordination between a local collection of agents and remote agents are therefore made only via middle agents that usually provide the following services: (1) message routing and translation; (2) agent behavior monitoring and notification; and (3) coordination between multi-agent activities. More recently, the autonomy-based approach has been proposed based on the concept of autonomy, and it often consists of a collection of autonomous agents who can define their own goals and actions [21]. They can interact and collaboratively work each other through communication. This approach forms a service exchange network by connecting intractable components that encapsulate specific aspects of an engineering problem. Although powerful in integrating various engineering models transparently, the framework is likely to be only suited to the well-structured and conflict-free problem. Since it mainly focuses on how an integrated model can be modeled by combining a number of distributed modules, the topology of the network is fixed and remains unchanged as the system operates. As all the problems cannot always be well structured or formalized, we should seek to discover new relationships and configure them when the system operates [22]. In addition, some architectures attempt to integrate engineering tools and human specialists by encapsulating them as agents. Since there is, however, no central controller and middle agents, the functional or physical agents are empowered to manage most of the activities related to their own goals and tasks through agent communication. This type of architecture is well suited for developing distributed intelligent systems in an open and dynamic environment. However, it is difficult to produce a globally optimal solution since each local agent has an equal right to compete with one another. No agent has the right to decide on the best strategy. Therefore, there is no theoretical guarantee that the process of agent-based solving will definitively converge [23].

Designers are no longer merely exchanging geometric data, but more general knowledge about design and design process, including specifications, design rules, constraints, etc. In addition to sharing and exchanging information, pressure to reduce product development times has resulted in an increased focus on methods for representing and storing engineering artifact knowledge in a way that facilitates its retrieval and subsequent reuse [24]. As the design is the process of

constantly extracting knowledge, the need for collaborative design to support the representation and use of knowledge has become the most pressing task.

GENERATION OF PORT-BASED MODULARITY

Properties of Modularity

In general, product architecture is divided into two types: modular and integral product architecture [8]. Modular architecture is composed of one-to-one mapping from functional elements in function structure to physical components, and indicates decoupled interfaces between components. On the other hand, integral architecture includes a complex, that is, many to one or one to many, mapping from functional elements to physical components and indicates coupled interfaces between components. In fact, whether functional elements map to more than one component or not depends on the detailed level of the designed components and functional elements. Modular architecture requires relatively more emphasis on system level design than integral architecture [25]. Port description plays a guiding role in the exploration of functional design of system level [26]. The overall function characterizes the general purpose or intention of the designed product. This function may need to be decomposed into a set of sub-functions in a hierarchy. In this phase, we should carefully define component interfaces with modularity and specify the associated standard forms.

Performance targets and acceptance criteria are set for each component, corresponding to the particular functional element implemented by the component. Here, component design is assigned to a designer for drawing system architecture. For integral architecture, this phase focuses on establishing clear targets for the performance of a relatively small number of integrated subsystems. These subsystems are assigned to multi-disciplinary teams that will share responsibility for designing the components that make up the subsystem. Figure.1 shows a modular architecture with different input and output relations, in which each module includes several components [27].

Figure 1. Modularity connections of product

Function-based Modularity

Three main function modularity based on their semantic contents is interdependently described by each other. The most important dependence highlights an integrated function by clustering a set of components. From this point of view, three

typical modules can be identified as follows [8, 28].

- Slot modularity: to allow one primitive device use different components. Each component has the same port and only performs one function shown in Fig.2 (a), for example, the case of LEGO with standard port geometry.
- Bus modularity: to describe a component of the system that is equipped with a standard port that accepts any combination of different functional modules. In most cases, the modules have a standard port that it excludes simultaneous consideration of two design concepts when they demolish, limit or oppose each other functionality shown in Fig.2 (b).
- Web modularity: to show a net connection of modules, each equipped with several ports that specify a set of standard components together through webs of modules rather than a simple chain or bus modularity. Each can individually accomplish different sub-functions, and their recombination on the chain interface, then permits different product function. The modules must be equipped with at least two complementary ports to create a new device as shown in Fig.2 (c).

Figure 2. Three major types of modularity

This research provides a methodology for creating and managing port-based ontology for use in database design. It makes a much richer modeling approach by which more of the semantics and constraints of an application domain are captured [29]. The result is a database that is an accurate representation of the real world created with less designer effort. It will allow port ontology to be used, evolved, and reused. Although a repository for domain port ontology is not necessarily going to make ontology creation less manual, it will provide a more systematic and less time consuming approach. It also will make the management of the port ontology less manual.

COMPONENT CONNECTION AND DENSITY

Component Connection and Density

Ports correspond to the separated interaction points where both components are connected each other. It imposes algebraic constraints on the port variables. This reification description can use to support design refinement and scheme synthesis [3,15]. Therefore, when a component is connected to another one, a connector is generated. Assuming the number

of components is n , its minimum connector number is $(n-1)$, and maximum connector number is represented below.

$$L_n = n(n-1)/2 \quad (1)$$

The density is defined as the level of component arrangement and distribution in a product space occupied [30]. Assuming there is n number of components in a product corresponding to n nodes, there are $(n-1)$ methods to divide n node graphs into two sub-graphs, and there are $(n-2)$ methods to divide two $(n-1)$ node sub-graphs into sub-sub-graphs again. Then, the minimum density of components is calculated as follows.

$$D_{\min} = n(1+2+3+\dots+n) - (1^2+2^2+3^2+\dots+n^2)$$

According to sequence formula sum, then

$$D_{\min} = n(n+1)(n-1)/6 \quad (2)$$

Also, supposing there is n number of components in a product corresponding to n nodes, we will divide n node graphs into two sub-graphs. According to Newton's binomial theory, then

$$(X+a)^n = C_n^0 X^n + C_n^1 X^{n-1} + \dots + C_n^{n-1} a^{n-1} X + C_n^n a^n$$

Where, suppose, $X = a = 1$,

$$1/2 \sum_{i=1}^n C_n^{n-1} = 2^{n-1} - 1$$

Therefore, there are $2^{n-1} - 1$ methods to divide maximum density graphs of n nodes into two graphs. Also there are $2^{n-2} - 1$ methods to divide maximum density graphs of $C_n^1 (2^{n-1} - 1)$ node sub-graphs into two sub-sub-graphs again. Using the same method, we can infer the maximum density of components as follows.

$$D_{\max} = 1/2 (3^n - 2^{n+1} + 1) \quad (3)$$

Table 1 shows minimum and maximum density values of component numbers from 3 to 15. When the functions of a product and its physical structures are independent, assembling product parts is the same procedure as combining product functions. But, this may create problem when the functional representation is not corresponding to the physical representation of design entities.

Table 1. Density values of components

Comp_ No.	Max number of	Min density of comp	Max density of comp
3	3	4	6
5	10	20	90
7	21	56	966
9	36	120	9330
11	55	220	86526
13	78	364	788970
15	105	560	7141686

Coupling Degree Calculation

The dependence between modules can be represented by the physical connections and/or functional interrelations that exist between the modules. Coupling degree between the modules is measured by the correspondence ratio (CR) and the cluster independence (CI) [31]. CR is measured by the ratio of shared components to total components in the modules, meanwhile, CI is the ratio of intra-module connections to the total number of connections in the product shown as follows.

$$CR = (M_i \cap M_j) / (M_i \cup M_j) \quad (4)$$

$$CI = \# \text{ intra-module connections} / \# \text{ total connections} \quad (5)$$

In general, design space is divided into a functional space and a physical space. These spaces are composed of the hierarchies of functional modules and generic physical systems, respectively. Functional modules are representations of behaviors of physical devices and functions are mapped to physical devices that satisfy design specifications. When functions corresponding to physical modules are uncoupling and decoupling as shown in Fig.2, they can implement system functional decomposition in accordance with functional independent axiom [32]. For example, there are three modules corresponding to three functions, in which M1 corresponds to four components, M2 corresponds to four components and M3 corresponds to three components. According to Equ. (4), the correspondence ratio (CR) is calculated as follows.

$$CR_{M12}=1/7 \quad CR_{M23}=1/6 \quad CR_{M13}=1/6$$

Then,

$$CR_{\text{total}} = \Sigma CR / \# \text{ modules} = 0.476$$

Also, according to Equ. (5), the cluster independence (CI) is calculated as follows.

$$CI = \# \text{ number of intra-module connections} / \# \text{ total connections} = 3/9$$

Therefore,

$$\text{Modularity} = 0.5 \times (CR_{\text{total}} + CI) = 0.405$$

A complex product corresponds to a complex functional structure, and it can be decomposed into several independent subsystems that each contains the several modules.

Figure 3. Relationship corresponding to functions and components

The coupling degree as a measure of design complexity has been developed. The coupling degree can be regarded as the modularity, and it embodies the level of connections between design variables. Coupling degree measure requires that design variable is measured by the representation of set-based format. Design process can be represented in set form where the tasks are nodes of a graph and they are connected through variable dependency [32].

PBO FOR PRODUCT MODELLING

Port Compatibility

Assuming X represents the set of components in a product, and a relation R_{port} can be defined in such way that it denotes port compatibility below [33].

$x R_{\text{port}} y$ means that x and y are of compatible port (6)

Where x and y are components in X . R_{port} stands for a compatibility relation, which contains equivalent relation, public relation, inclusion relation and transfer relation. These relations are defined as follows:

Equivalent relation(ER). if x and y have the same port types and port attributes, that is, $x \equiv y$ in mathematics. We will say that they are of compatibility and they can form a port. i.e. $x R_{\text{port}} y$

Public relation(PR). if x and y have the public port types and port attributes, $x \cap y \neq \emptyset$ can be defined from mathematics perspective. It is also compatible and can form a port. i.e. $x R_{\text{port}} y$

Inclusion relation(IR). if the port types and port attributes of x completely belong to y , and unreversed, it can then be represented as $x \subset y$ and $y \not\subset x$. It is also compatible and can form a port. i.e. $x R_{\text{port}} y$

Transfer relation(TR). If x, y, z satisfy $x \subset y$ and $y \subset z$, then $x \subset z$, the ports x, y, z will be of conduction attribute, viz., $x R_{\text{port}} y R_{\text{port}} z$.

These compatibilities are solely based on port names and port attributes. The disadvantage of using only port name is that when a new port class is added to the port ontology, these rules also need to be updated. Even adding a port with the exact same usage but in different name will require revising the rules of compatibilities. A more general approach is to use attributes to describe the constraints of compatibility. A circular-hole-port can be connected to all the ports with similar geometric features. This rule can be expressed by using low-level geometric constraints and dimensions of port features [16].

Port-based Ontology Modeling

Port design ontology has an intentional semantic structure that defines and arranges all related notions which can be either generic notions or specific notions. Port design ontology may also play an important role in forming the art of creating models of artifacts [29]. A model of port-based multi-views is proposed, and it contains three views: behavioral view, functional view and configuration view.

The functional view connoting the functional hierarchies is generated by functional mapping to primitive knowledge base.

It describes the effects of a physical phenomenon corresponding to port connector. Port-based ontological semantics can also describe connection information and structure information between primitive components. Primitive function and sub function are aggregated into the total function in a bottom-up manner. The behavioral view specifies the relation of connection and causality which constitutes a hierarchical semantic net by behavioral semantic mapping, and it makes efforts to bridge the gap between port function and port structure[14].

The configuration view describes which components are involved and whether they are mapped or interact with each other. They specify the relations of component position, topology and kinematics. The position relation describes quantitatively how the artifact is positioned and oriented in a three-dimensional space. The topological relation interprets the condition of their physical connections. Contacts can be direct or indirect. Contacts can also be further specified among the individual points on the surface, domains on the surface and complete the surface. Configuration is identified by primitive form mapping in a hierarchy.

The artifacts are generated by different view projects. The long dot line stands for project surface between views while the short dot line for mapping operations shown in Fig.3. It can realize port ontology project in different view ranges.

Port attributes are crucial to constructing the concept ontology process. Therefore, it is very important to distinctly analyze port attributes before designing concept ontology. Attribute representation of port is shown in Fig. 4. This taxonomy allows the users to quickly find components in an ontology library by using mapping operation, in which it contains component models and an alternative way to access components in the library. For example, two mechanical contacted parts have the same attributes with transferring mechanical energy, and they can form a mechanical port.

Compatibility checking occurs when two component models are connected. The port-based OWL description can be processed by the logic representation, which verifies whether all the attributes of the two ports satisfy the compatibility requirements specified in the port ontology. In a logic description, the port definitions and compatibility rules are stored in the knowledge base which is a collection of axioms for describing the true conditions of the port connection domain. When a port connection is established, the system queries the logic representation to verify if the connected port instances satisfy all the axioms [16]. For example, to connect a circular-hole-port, the port must have either a pin-shape or an axle-shape form attribute. A system can be established as a configuration of components by connecting the components at their ports. At the same time, the connected two ports need to be compatible, or else they don't connect together, such as a square plug does not fit a round hole. In this paper, it illustrates how the port ontology can be used to define general rules for port compatibility. The port ontology that has been defined so far does not include the concept of compatibility yet. We can use the phrase named 'is-compatible-with' to identify the port types that are compatible with each other, that is, it is satisfied with one of the four compatibility. For example, we can define the compatibility for a circular-hole-port whose rule explicitly specifies that only the axle-port and pin-port can be connected to a circular-hole-port. This research defines

the port topological properties (PTP) as a flexible mechanism to link and configure different ontologies into larger ones. For example, there are different types of artifact ontologies. First of all, a technique called include-and-extend, where several rules are included and extended with axioms at the same level of abstraction described in port-based knowledge base, is developed. Secondly, a technique is the include-and-specialize, where several ontological theories are included and subsequently are specialized to an engineering domain by instantiation, term and concept mappings and additional specific axioms called the concept ontology. Finally, a larger ontology, that is system configuration, is synthesized by the compatibility to connect corresponding artifact ontologies. Here, the connection between two artifact ontologies itself assumes the port compatibility. The number of artifact ontological connections is composed of a kind of hierarchical relations through mapping port compatibility.

PORT-AGENT FRAMEWORK FOR VARIETIES

Agent Infrastructure

A multi-agent system is in essence a reactive system that interacts with its environment in order to perform a specific task. A hybrid hierarchical agent architecture that is responsible for creating and improving conceptual design is proposed [5, 34]. The main agents are described in details.

Manager Agent is used for controlling and coordinating the relations of port agent to solve the conflicts and exchange the message with the user. It includes conflict detection agent and user interface agent.

Port Agent is used for design problem solving. It includes mechanical port agent (M_p-agent), electrical port agent (E_p-agent), configuration port agent (C_p-Agent) and I-Agent. I-Agent has the ability to learn from previous cases or past experiences.

Modification Agent is used for transforming port types, if users are not satisfied with the result of conceptual design.

Evaluation Agent is used to evaluate the rationality of collaborative conceptual design in which a reasonable solution is adopted and bad solutions are deserted based on some criteria or rules.

Communication Agent can provide the interactive function among agents by a standard syntax for message and use the knowledge query and manipulation language (KQML) for agent communications [35].

Where agents are defined as port knowledge strategies for solving design problem. They contain port knowledge base, reasoning engine, and reasoning rules. In the current implementation, agents are not autonomous, but are triggered by system or by other agents. They will constitute a federal-level alliance infrastructure [5].

Suppose T_i stands for a design task, in which 'i' is a layer number that represents the decomposed layer location of a design task. For example, an initial design task can be represented as $T_0(0,0,0)$ that locates 0th layer. Its subtasks are $T_0(1,1,0)$, $T_0(1,2,0)$, ..., $T_0(1,n,0)$ that locates in the first layer, and the sub-subtasks of $T_0(1, i, 0)$ are $T_0(1, i, 1)$, $T_0(1, i, 2)$, ..., $T_0(1, i, m)$ separately, i.e. the layer number denotes the decomposed depth while the relation of a design task forms a design task tree. In general, the total task is abstract, and it perhaps needs to iterate several times to decompose into the number of subtasks in detail. These subtasks are independent each other. Then,

$$\left\{ \begin{array}{l} T_0(0,0,0) = \bigcup_{i=1}^n T_0(1,i,0) \\ \forall i_k, 1 \leq i_k \leq n_k, T_0(k, i_k, i_{k-1}) = \bigcup_{i=1}^m T_0(k+1, i_k+1, i_k) \\ n_k = \sum_{i_k=1}^{n(k-1)} n(k+1, i_k) \end{array} \right. \quad (7)$$

Where, $T_0(0, 0, 0)$ stands for the total task that is composed of n subtasks. $n(k+1, i_k)$ stands for the number of sub-tasks of i_k th node in the k th layer of the tree. n_k stands for

Figure 4. A port agent framework for collaborative conceptual design

the total subtasks in the k^{th} layer of the tree.

Assume that A_k denotes k^{th} activity, and D^l denotes a Port agent, in which D means the type of an agent and l represents which agent it belongs to in the agent group, such as D^1 for a port agent with number 1. Furthermore, assume $T_i^{D^l}$ denotes the task i is being done by the port agent l . A set of tasks T_i are described by vector $(T_1, T_2, \dots, T_m)$. The agent l ($l < n$) is performing the task T_i by vector $(T_1^l, T_2^l, \dots, T_m^l)$. An activity A_k can be carried out several subtasks through a port agent operation as follows.

$$A_k \in \bigcup_{i=1}^m \bigcup_{l=1}^n T_i^{D^l} \quad (7)$$

Therefore, there are five kinds of relations between two activities: CON (A_i, A_j), SEQ (A_i, A_j), IND (A_i, A_j), PRI (A_i, A_j, T_{ij}), and COU (A_i, A_j) as follows.

- CON (A_i, A_j) expresses the concurrent relation of the design activity A_i and A_j that can be carried out simultaneously.
- SEQ (A_i, A_j) means sequence relation of the design activity A_i and A_j that can be carried out in series.
- IND (A_i, A_j) expresses an independent relation of the two activities A_i and A_j that cannot be performed simultaneously.
- PRI (A_i, A_j, T_{ij}) indicates the priority of design activity that A_i takes precedence over A_j in design activity, and T_{ij} are the time interval between A_i and A_j .
- COU (A_i, A_j) means the coupling relation of the two activities A_i and A_j that cannot be separated completely.

If $P_i(k)$ is produced by k activities, here, $k=1, 2, \dots, i, \dots, j, \dots, n$ ($i < j$), several agents are applied to solving component P_i . Assume $k=10$, different activities corresponding relations and agent are shown in Tab. 2.

Table 2. Activities of component $P_i(k)$ generation

No	Activities	Relations	Agents
1	Design component specifications	CON (A_1, A_2)	Manager agent, User interface agent
2	Component constraint variables		
3	Goals of design activities	PRI (A_2, A_3, T_{23})	Task assign agent
4	Design variable establishment	SEQ (A_3, A_4)	Method agent
5	Design component matching	SEQ (A_4, A_5)	Port agent, Feature agent
6	Prototype component choice	COU (A_5, A_6)	Task decomposition agent
7	Component tree generation	COU (A_6, A_7)	Method agent
8	Design component evaluation	IND (A_7, A_8)	Evaluation agent
9	Design component modification	PRI (A_8, A_9, T_{89})	Modification agent
10	Change environment	IND (A_9, A_{10})	Manager agent

Mechanism of Port-based Agent Collaboration

The port knowledge representation methods are an important part in developing collaborative conceptual design. Therefore, the representation models of port knowledge are needed to build component base, solving rule base, matching rule base, evaluating rule base, and so on. The proposed research will combine rule-based methods with frame knowledge representations. We build component base by the frame-based methods. In frame-based knowledge representation, the knowledge is represented by the data structures called frames (or objects). Each frame may have a number of attribute descriptions called slots, meanwhile slots may consist of several sides that they are one of two type attributes: abstract or concrete. Here, a frame corresponds to a kind of components, such as mechanical components, electrical components and hydraulic pressure components. A component corresponds to several slots and each slot corresponds to several sides. Meanwhile, rule base may be constituted by a series of conditions and conclusions (IF-THEN). However, these rules are independent on each other, meanwhile they are interrelated with Manager Agent and Communication Agent. They comprise a database that may be visited by all agents [5].

The creation of a complex design is due to collaboration among several different agents. These agents contain knowledge of how the design is based on their individual strategies and preferences. They are constructed to use for the representation of a design state, design problem solving, and finally lead to successful solutions. The strategies used by these agents are based on some algorithms, such as genetic algorithm, neural network, and so on. Here, agents are not autonomous, but are triggered by other agents. The manager agent invokes and harmonizes the activities of agents with the help of a group of function agents, such as port agent, conflict resolution agent, evaluation agent and so on. Their design operations, the conflict solving processes and protocols among the agents are described as follows.

Any design activities may be attributed to collaboration of different agents, in which Port Agent plays a center role in solving design problems. It has two subclasses: configuration-agent (C-Agent) and instantiation-agent (I-Agent), in which C-Agent contains M_p -agent, E_p -agent, and C_p -agent. C-Agents can capture a port type and determine how to obtain a new component. The goal of C-Agent is to decompose the preceding function into several sub-functions in order to make decisions about which component to be added to the design. I-Agent contains past cases and uses existed experience to guide design [5]. As many other knowledge-based systems, the port agents work in reasoning, matching, selecting and triggering cycle. During matching, the facts in the design are matched against the conditions of the rules in the knowledge base. These rules are independent with each other. Their relations can be obtained by database, and the content of database can be visited with all agents. It is easy to add and revise rule base in the synthetic structure. The rules whose conditions are totally satisfied will be triggered with the manager agent. When a rule is fired, its conclusion or action part is added to the design. Through communication and coordination among agents system will finally gain a scheme tree. In order to

evaluate multi-solutions and decide the main performances of each solution, evaluation agent is fired to pass judgment on the solutions. If an alternative is unsatisfactory, modification agent is fired to transform port types again. Of course, communication among agents is realized by KQML for exchanging information and knowledge.

When the users propose a new design problem, the manager agent first fires user-interface agent to carry out the input of design requirements through natural language description. The agents can exchange design data and knowledge via mediators. Then task decomposition agent is fired to decompose the overall task into the number of subtasks. There are several goal tree (AND/OR tree) templates in knowledge base. Task decomposition agent searches the candidate goal tree and recommends it to the Port agent. The port agent makes the decision and passes it to the manager agent. Once the design task decomposition has been fulfilled, the manager agent fires task assignment agent to assign a suitable task to a right agent. After a dynamic assignment process, correlative port agents begin to work solely for the same problem solving.

Each port agent needs manufacturing messages through mediators to transmit. They try to find available solutions based on their own criteria and rules. The manager agent invokes evaluation agent to evaluate all of these proposals. If they satisfy constraint rules, port agent sends the confirmation message and stores these proposals to the KB&DB for case agent in the future. If agreement cannot be reached, the conflict occurs and conflict resolution agent is activated. In order to negotiate these suggestions, conflict resolution agent sends message to manager agent to apply for relaxing some constraints, consistently decreasing their restrictive power to the port agent concerning with each conflict. These steps are repeated until a satisfactory decision is achieved. Here, the knowledge of manager agent includes the names and the addresses of each agent, their skills or competencies, the history records of performed tasks and so on. These can help the manager agent to assign right agent for performing the suitable design tasks. The results of design are stored with tree structure. A better solution can be selected by using evaluation agent. If the designers dissatisfy with all solutions, modification agent will be fired to modulate or revise some parameters for design again (shown in Tab. 2).

A CASE STUDY

The goal of punching-sealing mechanism is to implement plastic box sealing function under heating condition in automatic packaging machine (shown in Fig.5). In general, end-effector runs up and down motion. When it moves on the lowest position, end-effector needs to pause several seconds to keep firmly sealing plastic box. When it moves on the top of position, end-effector needs to go back quickly to raise working efficiency. The users require some sealing parts to realize sealing motion of plastic box.

According to design requirements, a product environment system is constructed. M1 receives input from environment to produce function F1. M2 is a transforming module, and it is either a receiver of function F1, or an executor of function F2. M3 is an output module, and it outputs motion function from system to environment. According to independence axiom, design equation is represented as follows [32].

Figure 5. Punching-sealing mechanism sketch

$$\begin{pmatrix} F1 \\ F2 \\ F3 \end{pmatrix} = \begin{bmatrix} M1 & 0 & 0 \\ 0 & M2 & 0 \\ 0 & 0 & M3 \end{bmatrix} \begin{pmatrix} DP1 \\ DP2 \\ DP3 \end{pmatrix}$$

We can extend and decompose three modules further into corresponding to cam mechanism, lever mechanism and slider mechanism, based on port-agent approach, in which each mechanism corresponds to several components. According to port-agent approach, we can reason several modules. For example, cam mechanism contains cam, wheel and follower, lever mechanism contains lever, fulcrum and support, and slider mechanism contains spring, soleplate and end effector. Tab. 3 gives different module divisions corresponding to CI, and they contain different module numbers. Tab.4 gives different module computing results with different components corresponding to CI and CR, respectively. Some components are changeable, such as cam, fulcrum. By modifying changeable components based on port-agent reasoning, the coupling degree between modules will be changed correspondingly, and we will obtain different modularity. Therefore, when the number of components changes with different manufacturing methods and different kinds of

Table 3. Different module divisions corresponding to CI

Functional clusters	Design activity recycling	Customizing varieties
Module 1: cam, wheel, follower Module 2: lever, fulcrum, support Module 3: soleplate, spring, effector	Module 1: Wheel, fulcrum, spring Module 2: follower, effector, lever Module 3: cam Module 4: support Module 5: soleplate	Module 1: cam Module 2: wheel, spring, fulcrum Module3: follower, end-effector Module 4: lever Module 5: support soleplate
# Modules=3	# Modules=5	# Modules=5
CI=3/9	CI=5/9	CI=5/9

components, they will correspond to different modularity, and produce different product design schemes.

Table 4. Comparison of different module computing results

	Modules	CR	CI	Modularity
Func_clusters	3	0.476	3/9	0.405
Mate_recycling	5	0.656	5/9	0.606
Manu_methods	5	0.532	5/9	0.544

CONCLUDING REMARKS

This paper has reported a port-agent approach to customizing product schemes toward preliminary design. The calculation of module coupling degree has been used to generate scheme variables. Port agent reasoning is crucial to define module concepts in the light of port agent knowledge. Some algorithms should be adopted to customize modules. We are currently investigating port-agent approach to aiming at explicit representations of design knowledge, component knowledge in order to capture product structure and build function modules. Current research is expanding this port-agent approach towards some complex product systems. The port-based ontology modeling for functional knowledge is reported in this paper. It can conveniently capture the intention of designer, determine port types and extend port attributes in a hierarchy. One of the main goals of paper is to clarify the relationships related to functionality, and to provide functional knowledge types for port-based agent reasoning solving. Although engineers may use function decomposition trees for scheme configuration design, they only help designers build on their own port knowledge. In this research, the heuristic approaches can be applied into port agent and specified by users to enable the system to generate various functional hierarchies. In addition, building an integrated product ontology and easy practice for port agent will encourage designers to focus on port knowledge issues. An XML-based representation of port agent is also necessary for an Internet-based collaborative environment in future.

ACKNOWLEDGMENTS

This research is sponsored by the National Nature Science Foundation of China (NSFC) under grant No.50775065 as well as Nature Science Foundation of Hebei Province under grant No. E2008000102.

REFERENCES

- [1] Welch R.V., Dixon J. R. 1994 Guiding Conceptual Design Through Behavioral Reasoning, *Research in Engineering Design* 6 169-188
- [2] Fujita K, 2002. Product variety optimization under modular architecture, *Computer Aided Design*, 34:953-965.
- [3] Singh P, Bettig P, 2003. Port-compatibility and connectability based assembly design, 2003 ASME

International Design Engineering Technical Conferences & Computer and Information in Engineering Conference, DETC/DAC2003, Sept.2-6, Chicago, USA

- [4] Cao D, Jia Q, Yang J, Cui C, 2008. Port-based description of functional modeling for product conceptual design, *Journal of Advanced Manufacturing Systems*, Vol.7 (1): 101-105.
- [5] Cao, D. X., Zhu, N. H., Cui, C. X., *et al.* 2008. An agent-based approach to guiding the conceptual design of mechanical products, *International Journal of Production Research*, Vol. 46 (9): 2381-2396.
- [6] Miao, C. Y., Goh, A. and Miao, Y., *et al.* 2002. Agent that models, reasons and makes decisions, *Knowledge-Based Systems*, 15(3), 203-211.
- [7] Tang D B, 2004. An agent-based collaborative design system to facilitate active die-maker involvement in stamping part design, *Computer in Industry*, 54(3), 253-271.
- [8] Kusiak A, 2002. Integrated product and process design: a modularity perspective, *Journal of Engineering Design*, 13: 223-231.
- [9] Eckert C, Clarksn P and Zanker W, 2003. Change and customization in complex engineering domains, *Research in Engineering Design*, 10.1007/s00163-0031-7.
- [10] Pimmler T, Eppinger S D, 1994. Integration analysis of product decompositions, *Proceedings of ASME Design Engineering Theory and Methodology Conference*, Minneapolis, MN, Sept.
- [11] Kitamura, Y., Mizoguchi, R., 2003. Ontology-based description of functional design knowledge and its use in a functional way server, *Expert System with Application*, Vol. 24, pp. 153-166.
- [12] Schemkel, H., 1998. Functional models and design solutions, *Annals of the CIRP*, Vol. 38(1): 129-132.
- [13] Tsai Y T, Wang K S and Lo, 2003. A study of modularity operation of systems based on maintenance consideration, *Journal of Engineering Design*, 14, pp.41-56.
- [14] Dahmus J B, Gonzalez-Zugasti G and Otto K N, 2001. Modular product architecture, *Design Studies*, 22: 409-424.
- [15] Kitamura, Y., Sano, T., Namba, K., 2002. A functional concept ontology and its application to automatic identification of functional structures, *Advances in Engineering Information*, Vol. 16(2): 145-163.
- [16] Singh, P., Bettig, B., 2004. Port-compatibility and connectability based assembly design, *Journal of Computing and Information Science in Engineering*, Vol. 4(3): 197-205.
- [17] Breedveld P C, 2004. Port-based modeling of mechatronic systems, *Mathematics and Computers in Simulation*, 66, pp. 99-127.

- [18] Campbell, Matthew I., Cagan, Jonathan and Kotovsky, Kenneth, 2003. The A-Design approach to managing automated design synthesis, *Research in Engineering Design*, 14(1), 12-24.
- [19] Gruer P, Hilaime V, Koukam A and Cetnarowicz K, 2002. A formal framework for multi-agent systems analysis and design, *Expert Systems with Application*, 23, 349-355.
- [20] Nahm Y E, Ishikawa H, 2005. A hybrid multi-agent system architecture for enterprise integration using computer networks, *Robotics and Computer-Integrated Manufacturing*, 21, 217-234.
- [21] Campbell, M. I., Cagan, J. and Kotovsky, K, 2000. Agent-Based Synthesis of Electro mechanical Design Configurations, *Journal of Mechanical Design*, 122(1), 61-69.
- [22] Dongxing Cao, Xiaotian Guan, Jian Liu and Chunxiang Cui, 2008. Port-based ontology for concept generation towards modular design, *Proceedings of the 8th International Conference on Frontiers of Design and Manufacturing*, Sept. 23-26, Tianjin, China
- [23] D Cao, Qiang Xue, Y Han, and C Cui, 2008. A port-based information delivery for collaborative conceptual design, *The 12th International Conference on CSCW in Design*, April 16-18, Xi'an, China
- [24] Zha X F, 2002. A knowledge intensive multi-agent framework for cooperative/collaborative design modeling and decision support of assemblies, *Knowledge-Based Systems*, 15, 493-506.
- [25] Kreng, V. B., Lee, T. P., 2004. QFD-based modular product design with linear integer programming-a case study, *Journal of Engineering Design*, 15:261-284.
- [26] Paredis C J J, Diaz-Calderon A, Sinha R and Khosla P K, 2001, Composable models for simulation-based design, *Engineering with Computers*, Vol. 17, pp. 112-128.
- [27] Dongxing Cao, Karthik Ramani, Ming Wang Fu and Runli Zhang, 2009. Port-based ontology semantic similarities for module concept creation, *Proceedings of the SAME International Design Engineering Technical Conference & Computers and Information in Engineering Conference*, Aug.30-Sept.2, San Diego,CA, USA
- [28] Jeffrey, B. J., Gonzales-Zugasti, J. P., and Otto, K. N., 2001. Modular product architecture, *Design Studies*, 22: 409-424.
- [29] D Cao, M W Fu, Y Gu, and H Jia, 2008. Port-based ontology modeling for product conceptual design, *Proceedings of the SAME International Design Engineering Technical Conference & Computers and Information in Engineering Conference*, August 3-6, New York City, New York
- [30] Dongxing Cao and Zhenqi Zhang, 1998. The component disassembly planning in Assembly Plans, *Journal of Hebei University of Technology*, 27 (3): 75-80 (in Chinese)
- [31] Newcomb P J, Bras B and Rosen D W, 1998. Implications of modularity on product design for the life cycle, *Journal of Mechanical Design*, Vol. 120: 483-490.
- [32] Suh N P, 1998. Axiomatic design theory for systems, *Research in Engineering Design*, 10:189-209.
- [33] Liang, V., and Paredis, C. J. J., 2004. A port ontology for conceptual design of systems, *Journal of Computing and Information Science in Engineering*, Vol.4, pp.206-217.
- [34] Zhao, G, Deng, J.T. and Shen, W., 2001. CLOVER: an agent-based approach to systems interoperability in cooperative design systems, *Computers in Industry*, 45, 261-276.
- [35] Koulinich A S, Sheremetov L B, 1998. Coordination and communication issues in multi-agent expert system: concurrent configuration design advisor, *Expert Syetems with Applications*, 15, 295-307.
- [36] Ozawa, M., Cutkosky, M. R., and Howley, B. J., 1998. Model sharing among agents in a concurrent product development team, *Workshop on Knowledge Intensive CAD*, KIC-3, IFIP5.2 Working Group, Tokyo.