Submitted to Proceedings of DETC'2005
2005 ASME Design Engineering Technical Conference
September 24-28, 2005, Long Beach, California, USA

DRAFT

DETC2005-85608

DISTRIBUTED MODELING AND FRAMEWORK FOR COLLABORATIVE EMBEDDED SYSTEM DESIGN

Xuan F Zha*, Ram D Sriram

Manufacturing System Integration Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

ABSTRACT

In this paper, we develop distributed design models and framework for collaborative embedded system design. Based on the component-based approach, a unified design-with-modules scheme is proposed to model the embedded system design process and a web-based distributed module modeling and evaluation (WebDMME) framework is developed as a collaborative cyber-infrastructure to support distributed network-centric embedded system design. The framework intends to link distributed, heterogeneous hardware/software (HW/SW) design models and tools and assist designers in evaluating design alternatives, visualizing trade-offs, finding optimal solutions, and making decisions on the web. It also enables designers to build integrated design models using both the local and distributed resources (e.g., local and distributed HW/SW modules) and to cooperate by exchanging services. The client (browser) / knowledge server architecture allows embedded system design models to be published and connected over the web to form an integrated intelligent models/modules network. Finally, as an illustration, a model for modular micro-robotic systems design is developed.

Keywords: Embedded system, hardware/software modules, hardware/software co-design, component-based approach, distributed module modeling and evaluation (DMME), collaborative design, cyber-infrastructure

1. INTRODUCTION

As we progress closer to a knowledge economy, the need for an infrastructure based upon distributed computing, information and communication technology (i.e., a Cyber-Infrastructure), becomes increasingly paramount. A recent report to NSF indicates that such a Cyber-Infrastructure (CI) will play a pivotal role in supporting and shaping future predictive product realization systems and processes (http://www.nsf.gov/).

Design process is a knowledge-intensive and collaborative task. The knowledge intensive design support in a cyber-infrastructure becomes critical and is recognized as a key solution toward future competitive advantages in product development. Integrated design requires skills of many designers/ users and experts that each participant creates models and tools to provide information or simulation services to other participants given appropriate input information. It is the goal that the network of participants exchanging services in a cyber-infrastructure forms a concurrent integrated model for design.

An embedded system is a hybrid of hardware and software, which combines software's flexibility and hardware real-time performance. Many industries are witnessing a rapid evolution toward solutions that integrate hardware and software or incorporate complete systems on a single chip (SoC). Modern embedded and hybrid systems have characteristics (ever-increasing complexity and diversity for more functionality, packed into smaller spaces consuming less power) that demand new approaches to their specification, design and implementation. Thus, embedded system design problems embody significant levels of complexity, which make it unlikely that a single designer can work alone on a complex design problem. Knowledge-intensive collaborative design has emerged as a promising discipline for dealing with the modeling and decision-making processes in distributed embedded system design. There exist many informal or semi-formal models and methodologies for separate hardware/software design. However, there is as yet no unified formal representation, simulation, and synthesis framework. At NIST, in the project of Representation for Embedded Systems (Zha and Sriram 2004), we are developing a standards-based framework for modeling information and knowledge in embedded systems design, including hardware/software co-design methodologies; an

*

integrated framework for design, modeling and testing; standard representations and protocols for exchanging and reusing system-level information and knowledge so as to enable semantic interoperability between design software systems in virtual, distributed and collaborative environments through the entire lifecycle.

This work aims to build up an information cyber-infrastructure to facilitate the rapid construction of integrated distributed design models for embedded systems. A unified design-with-object scheme is proposed for modeling and representing the distributed network-centric design process. A distributed framework is developed for collaborative design of embedded systems. A component-based topology and a feature-based model structure are defined for integrated representation/ configuration of HW/SW components that constitute an embedded system.

The organization of this paper is as follows. Section 2 is the current research status review. Section 3 gives an overview of the proposed approach to collaborative embedded system design. Section 4 discusses distributed module modeling and evaluation for embedded systems design process. Section 5 proposes an integrated knowledge representation scheme. Section 6 provides a description of the WebDMME framework. The knowledge server architecture for supporting different types of collaborative design activities in a distributed design environment is described. In Section 7, a model for modular micro-robotic systems design is built upon the WebDMME framework. Section 8 summarizes the paper.

2. CURRENT RESEARCH STATUS

2.1. Collaborative Design Frameworks and Systems

Computer Supported Cooperative Design (CSCD) is the use of computer technology assisting people working in the engineering design field. King (1989) considers that the fundamental issue of CSCD is focused on the computerization to establish a concept-sharing and seamless coordination among engineering design participants for concept formation. While many individuals and organizations may provide services so that an integrated product model can be constructed, it is not likely that each participant will disclose the full details or structure of their proprietary models and data. Providing a means for encapsulating expert knowledge or know-how is essential. An object-oriented approach provides a framework for such knowledge encapsulation. Furthermore, an object-oriented architecture is also highly suited to a distributed computing environment (Toye et al 1993). Distributed object technology, such as CORBA (Common Request Broker Architecture) (Siege 1996) and DCOM / ActiveX (Chappel 1996) can be used to address the issue of distributed computing environment. A computer platform and language-independent interface definition allows software applications to communicate with each other provided a neutral interface has been agreed upon. For example, the WWW has gained its popularity and momentum through a platform-independent protocol (i.e., HTTP) and a language-independent scheme (i.e, HTML) for presenting information. The software component technology is adopted to build a collaborative CAD system. Rosenman and his collaborators (1999) proposed an approach to collaborative design based on the software component mechanism. However, their work is only focused on the data, not knowledge.

Distributed design systems might have two distinct forms: distributed designers with access to centralized resources, or distributed designers with distributed resources (e.g., engineering models, databases, software applications, etc.) "Decentralized" means that the coordination between design participants and models is not centrally modeled or controlled (analogous to the WWW). This is important because centrally controlling the interactions of all distributed resources may restrict system growth and flexibility. If there was a centralized control over the WWW for linking the hyper documents, it could not evolve so rapidly. Pahng *et al* (1997, 1998a-b) developed a web-based framework for collaborative design modeling and decision support, based on the distributed object modeling and evaluation (DOME). In the DOME architecture, the resultant service-exchange network forms an integrated concurrent system model if module services are connected. There are other architectures or frameworks for network-centric collaborative design. These include the centralized multi-user system architecture, e.g., the blackboard-based DICE (Sriram and Logcher, 1993, Sriram 2002), DIS (Bliznakov et al 1995), data and model exchange system, e.g., SHARE (Toye et al, 1993), EDN (Lewis and Singh 1995), MADEFast (1999), NIIIP (1999), RaDEO (1998), and multi-agent distributed system architectures (Sun et al. 2001).

There are many information and distributed computing systems that have been built to support the collaborative design modeling and decision-making process. According to their different purposes and/or focuses, these systems can be generally grouped into the following categories (Li et al. 2004):

(1) Collaborative product data/information management systems for engineers to timely obtain the necessary product data and knowledge (Cheng and Liang 2000, Hardwick et al. 1996, Kim et al. 2001, Chao and Wang 2001);
(2) Network based collaborative design systems which can be further divided into web-independent (Pahng et al. 1998a-b, Shyamsundar and Gadh 2001, Tan et al. 1996, Sun et al. 2001, Whitfield et al. 2002) and web-dependent systems (Kim et al. 2001);
(3) Process-centered collaborative design and workflow management systems (Lu et al. 2001, Jin et al. 1999);
(4) Conflict detection, management and resolution systems for collaborative design (Jin et al. 1999, Wong 1997, Klein 1991);

(5) Flexibility and security focused collaborative design system (Camarinha et al. 2001);
(6) Interoperability approaches in heterogeneous collaborative design systems (Abrahamson et al. 2000, Zhao et al. 2001, Han et al. 1999)

Since most of them are developed for the needs of collaborative design, the current systems can assist designers one way or another in collaboration of embedded system design.

2.2. Electronic and Embedded System CAD Frameworks

Embedded and hybrid systems or integrated electronic systems are among the most complex artifacts. Decades ago, when the first integrated circuits were developed, small groups of engineers could handle the design without sophisticated computer aid. However, most of the current achievements rely on the work of many designers and design automation tools. From the need to support the numerous tools needed in a design cycle of an embedded and hybrid system (even an integrated circuit), the concept of electronic CAD frameworks/systems was crafted. This concept has evolved over the years, incorporating new engineering techniques to better serve its purpose of:

(1) support tool developers by providing building blocks - to accelerate implementation - and interfaces - to grant interoperability with other tools and data repositories;
(2) support tool administrators by providing a platform where tools and data repositories can be integrated and managed together;
(3) support designers by providing an integrated environment for the complete design flow.
(4) support designers by providing an integrated & distributed collaborative environment for the complete design flow.

It is clear that the next generation electronic CAD framework/systems are to support designers by providing an integrated & distributed collaborative design and development environment for the complete design flow. Currently, there exist some informal or semi-formal models and methodologies for separate hardware/software design and several frameworks have been proposed (Eggermont 2002). However, there is as yet no unified formal representation, simulation, and synthesis framework for supporting hardware and software co-design in a distributed collaborative environment despite the progress made in several research projects such as Ptolemy (Lee 2003) and Metropolis (Balarin et al. 2003).

From the literature review, we can summarize the current research status of collaborative design modeling and support for embedded systems design as follows. There are many research efforts working on enabling technologies or cyber-infrastructure to assist designers in the computer-aided network-centric design environment. Some of them intend to help designers to collaborate or coordinate by sharing product information and manufacturing services through formal or informal interactions. Others propose frameworks and develop systems that manage conflicts between design constraints and assist designers in making decisions. The overview of the current collaborative design frameworks and systems shows that they can assist designers in one way or another in collaboration and their functionality can eliminate a large amount of work during the design process such as data/information sharing and exchange. However, they do not provide a structured and formalized framework for modeling the characteristics of multidisciplinary and multi-objective design problems, in particularly hardware and software co-design in a complex embedded system. Thus, they are unable to effectively and efficiently support and coordinate highly distributed/decentralized collaborative design and modeling activities for embedded systems. As a result, an intelligent collaborative tool that can provide efficient coordination and intelligent decision-making mechanisms for designers is still needed.

3. COLLABORATIVE EMBEDDED SYSTEM DESIGN: AN OVERVIEW OF THE PROPOSED APPROACH

As reviewed above, distributed design systems might have two distinct forms: distributed designers with access to centralized resources, or distributed designers with distributed resources (e.g., engineering models, databases, software applications, etc.). This work focuses on the latter architecture. Thus, the coordination between design participants and models is not centrally modeled or controlled (analogous to the WWW). The motivation and vision of this research share some similar themes with DOME (Pahng *et al* 1997, 1998) but emphasize knowledge-level or knowledge intensive design modeling, decision-making support, and search/optimization and navigation. Even more, the proposed approach decomposes design tasks into several HW/SW "design components" or modules that can be developed separately by collaborative developers. We investigate the application of the modular design and development approach both for HW and SW artifacts. The distributed module-based approach follows the component technology and distributed object technology, which is called distributed module modeling and evaluation (DMME) framework. In the component technology, a component is a reusable application whose data and methods is exposed and can be accessed and operated by other applications. Thus, the DMME framework asserts that multidisciplinary problems are decomposed into modular sub-problems. Modularity divides overall complexity and distributes knowledge and responsibility amongst designers. It also facilitates the reuse of modeling elements. Therefore, distributed modules allow designers to define mathematical models or HW/SW modules and integrate or interconnect them to form large system models. In DMME, a multiple attribute decision method is used to capture preferences and evaluate design alternatives from different viewpoints. The resultant service-exchange network forms an integrated concurrent system model if module services are connected. The comparisons between the DMME architecture

and other architectures or frameworks including DOME are described as follows:

(1) DMME share some characteristics with DOME, but it differs from the DOME architecture (Fig.1b) in that it supports both hardware and software system modeling at feature and knowledge level.

(2) For the DMME/DOME architecture, the distinct characteristic is that when module services are connected, the resultant service exchange network forms an integrated concurrent system model.

(3) For the centralized multi-user system (Fig.1a), multiple users have access to the centralized main system, which stores and manages information such as product design models, design information and design history. Although powerful, a central system is less suited for loose and flexible collaborations as it is not an open environment and does not allow for true knowledge encapsulation. However, such architecture could be supported within a module of a larger DMME/DOME network.

(4) The data and model exchange system architecture (Fig.1c) tends to provide an "over-the-wall" sequential interaction between designers and models. When a designer receives a model or data from another designer, he/she works on the design and sends the result of design modification to others. Therefore, this architecture is not intended to provide concurrent system modeling functionality.

(5) Multi-agent based architectures are more appropriate for loosely coupled environments where mutual interactions between objects are not well defined. In the DMME/DOME architecture, the interactions between sub-problems are explicitly defined through design negotiation so that a communicating object paradigm is appropriate. However, within the DMME/DOME, agents are useful when designers are not certain about what modules can provide the service they require. Agents could locate appropriate modules.

More details about the proposed approach are discussed below.

4. DISTRIBUTED MODULE MODELING AND EVALUATION FOR EMBEDDED SYSTEM DESIGN PROCESS

In this section, based on the component-based approach, a design with modules scheme is proposed for embedded system design and a distributed module modeling and evaluation (DMME) framework is developed for modeling the network-centric embedded system design process.

4.1 Design with Modules Scheme for Embedded Systems

Over half of industrial products have subsystems or modules or components as a part of their basic designs. The modularity design concept has been widely used in design for flexibility, rapid responsiveness, ease of maintenance, and rapid deployment. As a matter of fact, during the design process, information processing is inherently model-based because the design object is structural in type. Therefore, an object orientation scheme is employed so that both calculating and reasoning work in design can be carried out. The integrated design object model is, in fact, an attempt to set up a knowledge intensive framework in such a way that it becomes possible to process various types of knowledge in a top-down design process (Gero 1990). Object-oriented programming technique allows designers to look at a design problem as a collection of objects/components/modules or sub-problems linked together by rules. Thus it provides the designers with an expressive power to represent complex problems or information in an effective manner. If a designer can break a design problem in the form of well defined, clearly operable chunks with their own self-containing information which are interrelated through a series of rules and constraints, then these problems lead themselves well to object-oriented programming application and conveniently to be solved (Pahng et al 1998).

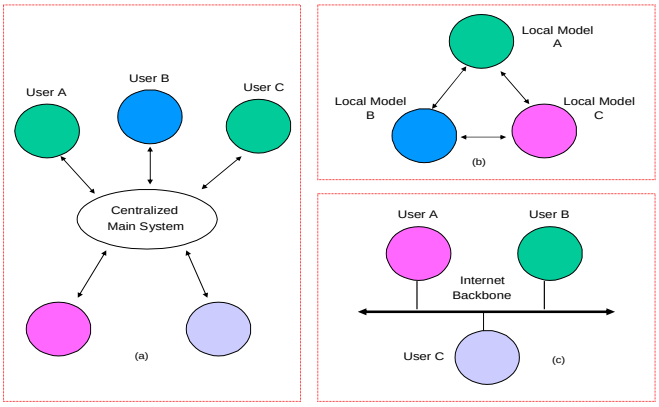


Fig.1: Interactions between modules exchanging services

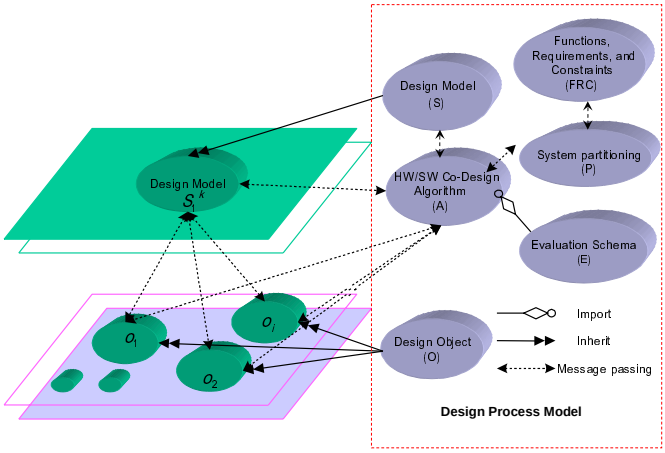


Fig.2: Overall architecture of the design with modules scheme for embedded systems

The central embedded system design process inherent in the design-with-modules scheme proposed in this work could be represented as the architecture as shown in Fig.2 with six main types of objects involved, namely, design models (*S*), design objects (O), co-design algorithms (A), functions (requirements and constraints) (FRC), system partitioning (P) and the evaluation schema (E). Object operators can express the relationship between these objects: inheritance, import, and message passing. The architecture in Fig.2 shows how the particular instance of a design model, S_1^k, is obtained from the HW/SW co-design algorithm, system partition, evaluation schema, requirements, constraints and the design model object. For the pure formulation design or creative design, a new design model object, *S*, is defined that describes the form of the model. A specific instance, S_1^k, of this design model can then be created. For pure parametric design then the design model object S has already been defined and the design process therefore only involves the determination of a specific instance, S_1^k, of the design model. Note that additional objects can be defined within the overall architecture.

4.2 Module-based Design and Modularization/ Partitioning Process

A design model is created using object attribute variables and relations between them. Modules are variable containers. Variables can be grouped together into modules according to logical, functional or physical component-based decompositions (Senin et al. 1997a). This grouping process is normally a bottom up decomposition as elementary entities are aggregated to build more complex entities. A designer may also prefer to use a top-down approach by defining high-level modules first and then detailing the low-level variables and relations. Using the bottom-up and top-down techniques, integrated design models are built by interconnecting modules corresponding to different sub-problems. Modules interact by service calls. A module might request information from other modules to perform internal computations and then provide the results as services to other modules. In fact, a HW/SW module can be seen as a collection of HW/SW components that covers one or more sub-functions. For example, an electro-mechanical product (e.g. modular robot) module can be defined by a combination of bill-of-materials (BOM) and technical drawings. Thus, physically, a modular product system is a collection of interchangeable modules (e.g., link and joint modules in modular robots) that can be assembled into many different types and configurations of products. The modular design of products provides the ability to achieve product variety for customer requirements through the combination/configuration and standardization of modules.

A module is a building block capable of performing calculations and performing information through service calls invoked by its user. Defined as shown in Fig.3, a module represents knowledge related to different aspects of the design in the form of variables and relations. The variables contained in the module are represented as interconnected circles. The directed arcs imply dependency. The outputs and inputs to the module constitute the interface of the module. Modules can be interconnected through interfaces. The calculations internal to a module constitute its embedded model. Customer-created computer programs and third-party applications (software modules), such as domain specific analysis tools or CAD systems, can also be embedded into a module. Modules interact with each other by exchanging information and services, reacting to each other's changes for an integrated system model. Modules can distribute over the network and collectively form a distributed model for a collaborative, multidisciplinary and concurrent design evaluation.

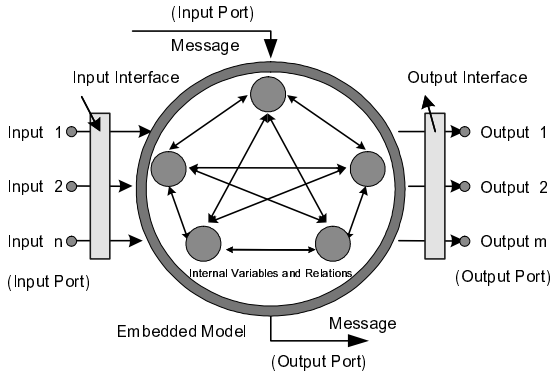


Fig.3: Module definition and embedded model

Decomposing the problem into modules and defining how modules are related to one another creates the model of a design problem. This is actually a modularization/partitioning process. The modularization/partitioning process is fulfilled through the following steps:

(1) Functional specification analysis is carried out from the customer requirement viewpoint using design functions deployment technique and Hatley/Pirbhai technique (Sivaloganathan et al 2001; Rushton & Zakarian, 2000). A function-function interaction matrix is generated.
(2) The combination of heuristic and quantitative clustering algorithms is used to modularize/partition the product/system architecture, and a modularity matrix is constructed.
(3) All modules are identified through the modularity matrix, and the types of all these modules can be further identified according to the module classifications (HW/SW modules).
(4) Functional modules are mapped to structural modules using the function-structure interaction matrix. Module attribute parameters or features can represent its structure (Zha and Sriram 2004).
(5) Hierarchical building blocks (modules) are used to represent the product/system architecture from both the functional and the structural perspectives.
(6) Optimization algorithms are used to optimize the product/system architecture to achieve one main objective. Other design objectives are transformed into constraints for modules and their attributes as well as their assemblies or

configurations. In addition, the cost or profit models can also be built as system constraints.

4.3 Distributed Co-Design Modeling and Evaluation

In a distributed design environment, each group of designers can define their own HW/SW modules, loading them into their local work area and eventually connecting them to the other parts of the design problem through appropriate networked interfaces. Fig.4a shows a distributed co-design model involving two designers and three modules. Designer 1 defines HW modules A and B while Designer 2 defines SW module C. Two domains communicate through an Internet connection. Once the whole problem is loaded and interconnected, each group of designers typically has write access to local parts of the model, i.e., they can exert decisions within their local range of influence, and read access to relevant aspects of remote parts of the model. This allows designers to see the remote effects of their local decisions. Fig.4b shows that both Designers 1 and 2 might see the complete problem, but with different access privileges. Designer 1 can see modules A and B as local and module C as remote. Conversely, module C is local to Designer 2. The remote part seen by Designer 2 could show modules A and B or just a single distributed object (AB) if Designer 1 restricts their visibility/access.

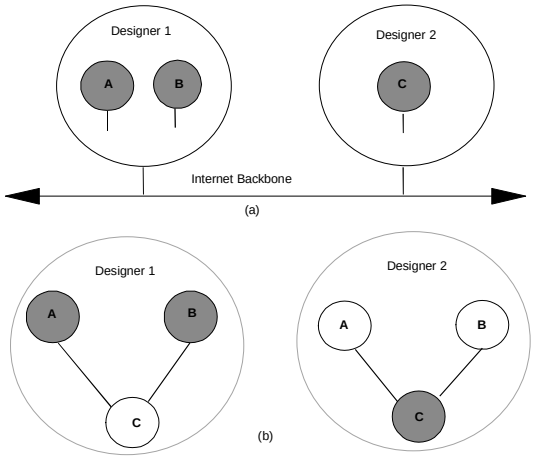


Fig.4: Distributed modules

The modeling and implementation layers are illustrated in Fig.5. The user-visible layer or modeling layer is the designer's viewpoint. At this level the designer defines the problem in terms of modules and interactions among modules (Fig.5a). In this example, an implementation is provided only for the local HW modules A and B, while SW module C is remote. The designer provides location constraints for compatible modules that may be used as C. The unseen implementation layer (Fig.5b) is created to provide the functionalities described in the modeling layer. This layer locates remote modules. The remote modules must be distributed objects capable of communicating via a standard communication protocol. A distributed interface is wrapped around the group of standard modules (A and B) to allow the local and distributed modules to communicate with each other. This distributed module's external interface offers service calls to and from the remote module. A design problem model sees the distributed module as a separate application that is capable of providing services upon request. Fig.6 shows an implementation for module network.

The interactions between distributed modules can be achieved by publishing and subscribing services. The term publish refers to making the services of one's local model visible together designers. The term subscribe refers to making use of published services. Such design problem models are mixed variables, where independent parameters within modules are set and catalog selections might be used to substitute entire modules. Design solutions can be assessed and compared with each other using the decision-making tool embedded in the framework.

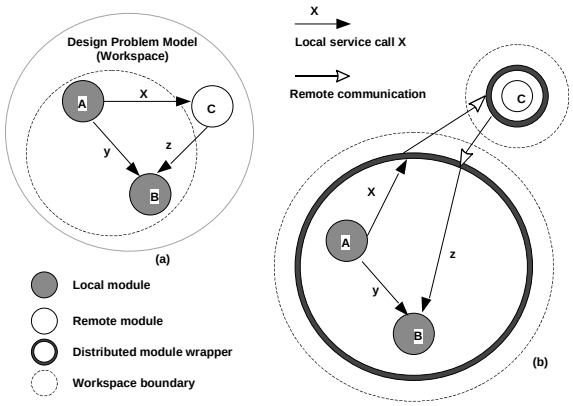


Fig.5: Module network modeling

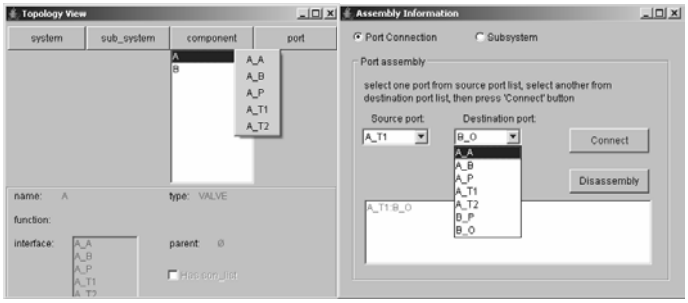


Fig.6: Module network implementation

5. INTEGRATED KNOWLEDGE REPRESENTATION SCHEME

During the design process, both product and design process design knowledge representations are needed to be dealt with (Gorti et al 1998). Since the design knowledge is very extensive, we focus only on the product/system and some selected activities in the design process. A systematic methodology is developed for knowledge modeling and

management in the design process, as shown in Fig.6. In our previous work, we have developed an object-oriented model (OESM) for representing embedded systems (Zha and Sriram 2004, Zha and Sriram 2005). OESM is extended from the NIST core product model (Fenves 2001). Specifically, a complete information model is defined, which consists of customer requirements, design specifications, HW/SW modules (artifacts, functions-behaviors, geometry and material for HW form, architecture and code for SW form), module interfaces, etc., as follows:

Embedded System Model {
    Requirements;
    Specifications;
    HW/SW Artifacts;
    HW/SW Features;
    HW/SW Functions-behaviors-forms;
    HW/SW Performance objectives and constraints;
    Relationships;
    Design rationale;
}

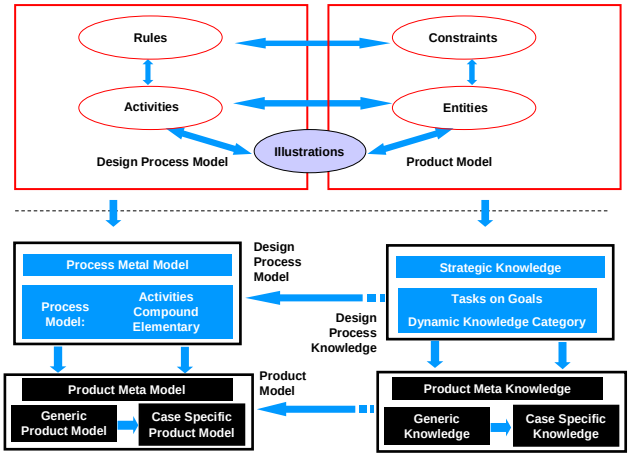


Fig.7: Product, design process, and knowledge

Based on the OESM, an integrated knowledge representation approach is proposed for embedded system design in this work. In what follows, we concentrate on introducing the integrated knowledge representational scheme related to the design process. It deals mainly with declarative representation, production rules and object-oriented concepts. Procedural representation using conventional languages such as C was not emphasized. Integrating knowledge in its multiple forms, multiple levels and multiple functions can fulfill design processes and activities, especially the more complex type. The integration is very challenging, as the overall effect may be greater than the sum of its parts. The integrated knowledge can solve problems, which cannot be attained by the individual knowledge alone. Based on a combination of elements of semantic relationships with the object-oriented data model, a multi-level hierarchical representation schema (enterprise-level, system level, component-level, feature level) is adopted to represent the embedded system design process knowledge. To effectively manage and utilize the design process knowledge, a generalized design knowledge matrix is proposed for organizing design knowledge. All tasks in the design process are listed in column while all information and design knowledge are categorized in rows. The contents of design knowledge for each task are recorded in the corresponding cells of the design knowledge matrix with appropriate representations.

More specifically, the object-oriented knowledge representation is based on a mixed representation method and object-oriented programming (OOP) techniques (Sriram 2002), and allows designers to look at the design problem as a collection of objects or sub-problems linked together by rules. Thus it provides the designers with an expressive power to represent complex problems or information in an effective manner. If a designer can break the design problem into the form of well-defined, clearly manipulative chunks with their own self-containing information, which is interrelated through a series of rules and constraints, then the problem can be easily solved. The basic structure of this representation is described as a module. The class of an object and its instances are described by the module structure. An object-oriented module is composed of four types of slots, which are the attribute slot, relation slot, method slot and rule slot as follows (Sriram 2002): 1) The attribute slots are used for describing the static attributes (variables) of design object. 2) The relation slot is used for describing the static relations among objects. With the help of the relation slot and according to the relation of classification, the design object can be described as a hierarchical structure. Its classes and subclasses can share the knowledge in super class. The messages that control the design process can be sent among all instances of objects. In addition, if needed, other kind of relation slots can be defined, such as the resolution, position and assembly, etc. These slots create the foundation for describing a graph in design. The hierarchical structure of object oriented knowledge representation is formed. 3) The method slot is used for storing the methods of design, sending messages and performing procedural control and numerical calculation. 4) The rule slot is used for storing sets of production rules. The production rules can be classified according to the differences among objects being treated and stored respectively in rule slots in the form of slot value. Thus, the integrated knowledge representation scheme realizes the advantages of both object-oriented representation and rule-based representation.

6. KNOWLEDGE INTENSIVE FRAMEWORK FOR COLLABORATIVE EMBEDDED SYSTEM DESIGN

In this section, a web knowledge server based distributed module modeling and evaluation (WebDMME) framework is proposed for collaborative embedded system design.

6.1 WebDMME Framework Architecture

The widespread use of the Internet and WWW provides an opportunity for making expert systems widely available. By implementing knowledge-based expert systems as knowledge servers that perform their tasks remotely, developers can publish expertise on the web. Technologies and cyber-infrastructures that make this approach feasible are emerging. Simultaneously, the interest in AI support for network navigation services is growing. The internet and WWW now allow developers to provide intelligent knowledge servers (Eriksson 1996). Knowledge-based expert systems running on servers can support a large-scale group of users who communicate with the system over the network. In this approach, user interfaces based on web protocols provide access to the knowledge servers, and users do not need special hardware or software to consult these services with appropriate web browsers. To make knowledge servers available, developers must distribute the software front ends that allow users to communicate with the servers. The remaining parts are concerning how expert systems technology can be used to assist designer in navigating design knowledge present on the WWW, drawing on an alternative view of KBS, and making decisions in the design process.

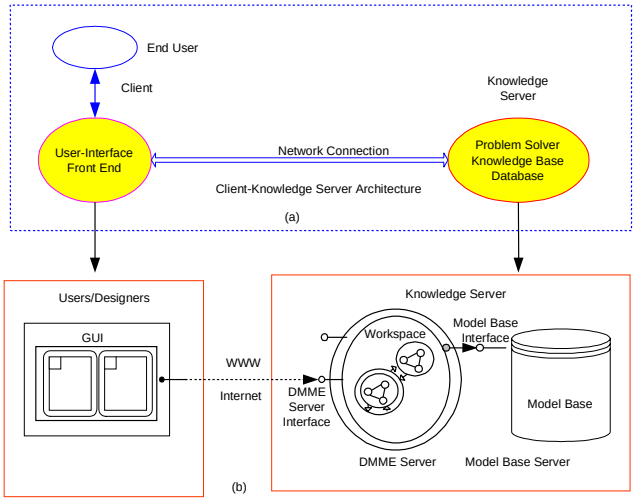


Fig.8: (a) Client-knowledge server architecture (b) and main components for WebDMME

The proposed web based design framework (WebDMME) adopts the design with modules, modules network, and knowledge server paradigms in which the knowledge server utilizes the connectivity provided by the internet to increase the size of the user base whilst minimizing distribution and maintenance overheads. Therefore, HW/SW modules under WebDMME framework are connected together so that they can exchange services to form a larger integrated model. The module structure of WebDMME leads itself to a client (browser) / knowledge server oriented architecture using the distributed object technology. Fig.8 shows the main system components of the proposed client (browser) and knowledge server architecture. Each of these components interacts with one another using a communication protocol (e.g. CORBA) so that it is not required to maintain the elements on a single machine. As a gateway to provide services, the interface of a system component invokes the necessary actions to provide requested services. To request a service, a system component must have an interface pointer to the desired interface (Pahng et al 1998a,b). In the WebDMME architecture, the resultant service-exchange network forms an integrated distributed concurrent system model for embedded systems when module services are connected over the web. The distinct characteristics of the WebDMME architecture are described as follows:

(1) It is well suited for loose and flexible collaborations as it is an open environment and allows for true knowledge encapsulation. The centralized multi-user system architecture could be supported within a module in a larger WebDMME network so that multiple users have the ability to access to the centralized module.

(2) WebDMME architecture is intended to provide concurrent system modeling functionality, i.e., concurrent interactions between designers and models. When a designer receives a model or data from another designer, he/she works on the design and sends the result of design modification to others.

(3) In the WebDMME architecture, the interactions between sub-problems are explicitly defined through design negotiation so that a communicating object paradigm is appropriate. Within the WebDMME, agents are useful when designers are not certain about what modules can provide the service they require. Agents could locate appropriate modules (Li et al. 2004).

6.2 Module Interactions for Exchanging Services

The WebDMME architecture is designed to allow designers or experts to publish and subscribe to design modeling and decision support services on the web. These services operate when information is received from other clients or knowledge servers. When module services are connected, the resultant service exchange network forms a concurrent integrated system model. Any service request in the module network can invoke a chain of service requests if needed to provide correct information. When a design alternative is evaluated, the local model asks for the services of subscribed models. If the subscribed models themselves need services from other models in order to provide the request services, they again request those services from their own network to remote models. Thus, the service requests are propagated through the connected modules. However, the complete system may not be visible to any given model. Since modules can only interact through services, it is possible for a module or local model to encapsulate its internal modules and hide intellectual property (IP) if desired. Before a designer publishes his/her model(s), they can assign access privileges for their services. Three levels of models access are required: owner, builder, and user. The owner is the original creator of the model and has access to all the services defined in

the model and control over their publication. The builder can see the internal details of a model the owner chooses to make public and can add new modules. However, they cannot destroy modules created by the owner or other builders. The users can subscribe only to the published services.

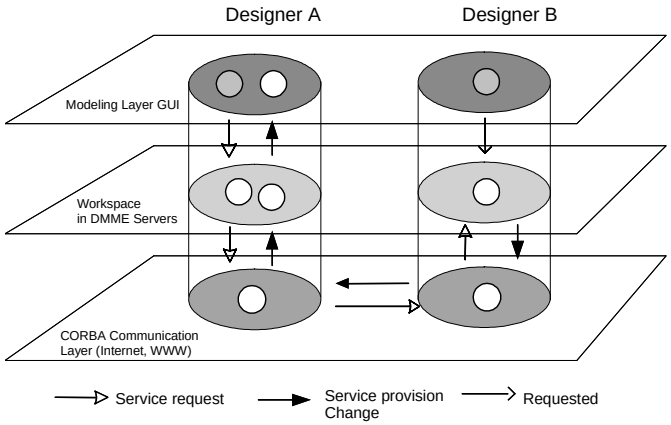


Fig.9: Service exchanges between distributed modules.

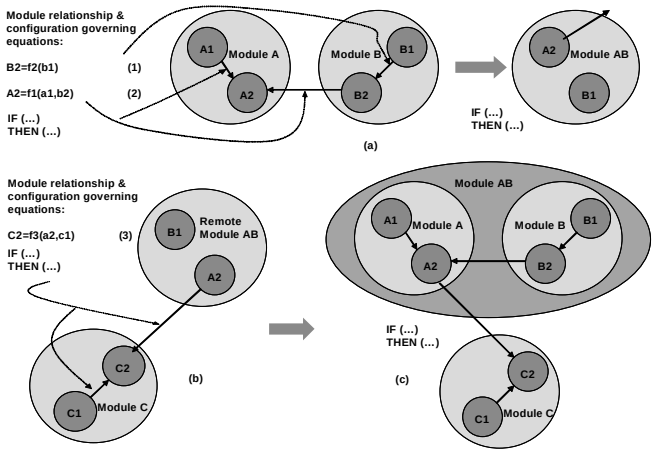


Fig.10: Simple distributed design model with two modules and a remote module: (a) Modules A and B, (b)-(c) remote module AB

The WebDMME framework provides methods and interfaces needed for the interaction with other modules in the networked environment. These interactions are graphically depicted in Fig.9. When Designer B makes a change, the service corresponding to the request from Designer A will reflect this change. The enumerated request shows the sequence for obtaining the service needed by Designer A. The light gray module seen by Designer A is a remote module published by Designer B.

6.3 Modules Network Formulation

As discussed above, the modularization/partitioning process decomposes a design problem into HW/SW modules and defines how HW/SW modules are related to one another. The relationships amongst modules specify how outputs of a module are connected to the inputs of other modules. The embedded model of a module produces outputs using its internal design resources as well as inputs from other modules. Fig.10 illustrates a simple distributed module network model used for a design process. The variables of the model are governed by a set of equations and/or rules. The interface connections between variables in different modules (e.g. modules A, B) can be established interactively or defined explicitly using the Model Definition Language (MDL) (Siegel 1996, Phang et al. 1998). The embedded models defined with the variable declaration can also be created separately and linked to the model definition using keywords. Modules A and B are local to the problem. Using the remote module AB, a new design model (ABC) can be created. As such, the problem model is made available for use as a distributed module with the outward appearance in Fig.10a. These distributed modules allow users to utilize variables and their dependencies such as Module A (A_1, A_2, $\rightarrow a_1$, $\rightarrow a_2$), Module B (B_1, B_2, $\rightarrow b_1$, $\rightarrow b_2$), Module C (C_1, C_2, $\rightarrow c_1$, $\rightarrow c_2$), and Module AB (A_1, A_2, B_1, B_2, $\rightarrow a_1$, $\rightarrow a_2$, $\rightarrow b_1$, $\rightarrow b_2$). Fig.10b illustrates the model from the viewpoint of the ABC designer. Module C is local to the designer. Fig.10c illustrates the true integrated model created when the remote module AB and the local module C are connected. The problem model ABC is thus created, which requires additional information such as the distributed module's name and IP address. The description of the distributed model can be illustrated in Table 1.

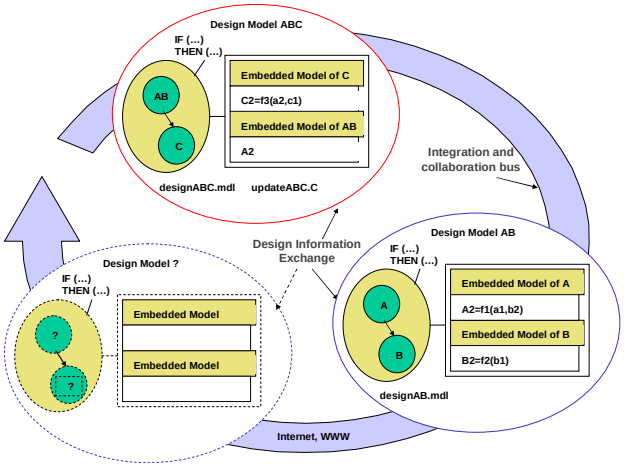


Fig.11: Module network configuration under the WebDMME framework

It is shown that the relations between modules do not need to be changed even if the embedded mathematical model of a remote module (i.e., module AB) is changed. This flexibility enables a designer to define a model independently from the actual location (i.e., local or remote) of embedded models. When the designer utilizes the remote module AB in conjunction with the local module C, the resulting integrated model forms a distributed computing system comprised of two autonomous

computing elements. Fig.11 illustrates the configuration process for distributed modules using the system components/modules, including the internet and web resources. The embedded model of the module AB in design problem model ABC contains an object connector that manages the design information exchange with the distributed design object AB.

Table 1: The description of the distributed model

Local Module Integration	Remote Module Integration

module: "A" (
Variable "A1"()
Variable "A2"()
Dependency "a1"
Dependency "b2"
EmbeddedModel "calculateA2"(A2=f1(a1,b2))
RuleSet "A" (Rule A1:
IF (a1=)
THEN (…)
)
)
module: "B" (
Variable "B1"()
Variable "B2"()
Dependency "b1"
EmbeddedModel "calculateB2" (B2=f2(b1))
RuleSet "B" (Rule B1:
IF (…)
THEN (…)
)
)
Design: Two-module design (
Module: A
Module: B
}
	module: "Remote_AB" (
	URL: 159.69.1.19 // IP address
	Receive "a2"
	RuleSet "AB" (Rule AB1:
	IF (…)
	THEN (…)
	)
	)
	module: "C" (
	Variable "C1"()
	Variable "C2"()
	Dependency "c1"
	Dependency "a2"
	EmbeddedModel "calculateC2" (C2=f3(a2,c1))
	RuleSet"C" (Rule C1:
	IF (…)
	THEN (…)
	)
	)
	Design: Design with a remote module (
	Module "AB"()
	Module " C"()
	}

7. CASE STUDY

To verify and validate the proposed modeling approach and framework, we carried out a few case studies, including the hydraulic measurement and control system for car ABS (Antilock Brake System), intelligent e-maintenance and service system, robotic system (controller), micro/medical device, etc. In this section, the design of micro robotic system is used as an example to illustrate the distributed modeling for embedded system design. The case is chosen because of its embedded but relatively simple nature. The research results from this particular case can be generalized to cover other designs that require collaboration and integration of multiple domains. The focus of the illustration is on how designers from different teams may participate to create an integrated design model: modules, modules network and HW/SW module configurations.

The design session creates modules in the design workspace. Designers can use any commercial web browser to access and work on these modules. As the robot system design and operation are tightly coupled, it would make sense for designers in these groups to share a common model. Thus, while designers from different groups are in remote locations, they can access into the same workspace, which is referred as a shared workspace. Fig.12 shows the design workspace as viewed by the designers from the robot system design team and the robot operation team. The robot system design team is connected to the robot and gripper manufacturing teams so that their robot system design integrated with the gripper and robot models can be tested. In this implementation and demonstration, the robotic system is assembled through the use of predefined fixed types of HW modules (joint and link modules) and SW modules (control system) in distributed HW/SW module inventories (repositories). These modules are published and can be accessed.

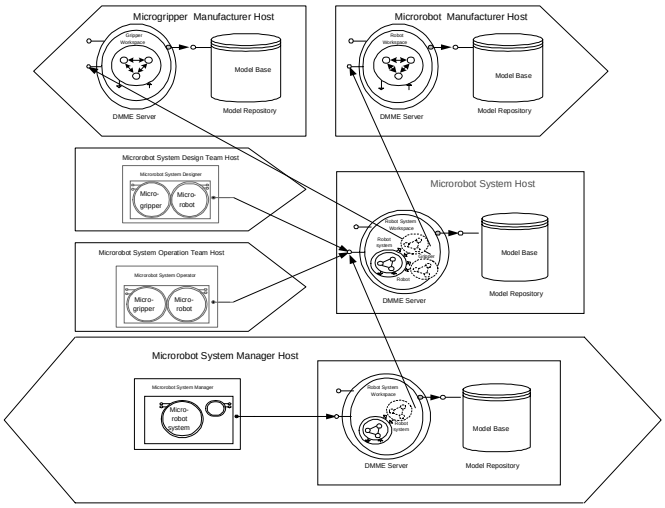
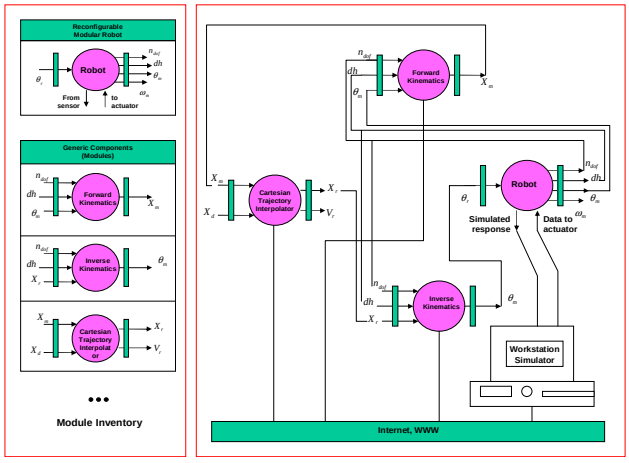


Fig.12: Shared design workspace



Note: n_{dof} - the number of degree-of-freedom; dh-D-H parameters; X_m -the measured configuration; X_r -the real configuration; X_d -the dynamic configuration; θ_m -the measured joint variable (vector); θ_r -the real joint variable; ω_m -the measured joint velocity; ω_r -the real joint velocity; V_r - the real linear velocity

Fig.13: Robot design analysis modules and modules network

The users or operation team can share their workspaces with the design team. The design team creates modules while the robot system operation team makes the rest design. In this case the design team owns the session and the operation team have joined as a builder. Although builders cannot modify the modules created by other builders or owners, they can add new modules and utilize all services. For example, the operation team can use a service from a design module to obtain the robot accuracy and the open distance of the gripper and can build new modules in the workspace that utilize this information. Similarly, the design team can also use services from the models published by the robot and gripper manufacturing team. Utilizing models provided by other designers is referred as subscribing to a model. It is the responsibility of the design team to provide these data or to locate other models that can provide these data as services. The robot system managers want to evaluate the design from in term of costs and they may link their models to the design module to obtain the information services needed by their models. The design team has only published cost related aspects of their models. This means that the robot system managers can only observe elements of the design models that were published, as the designers wanted to protect their proprietary models.

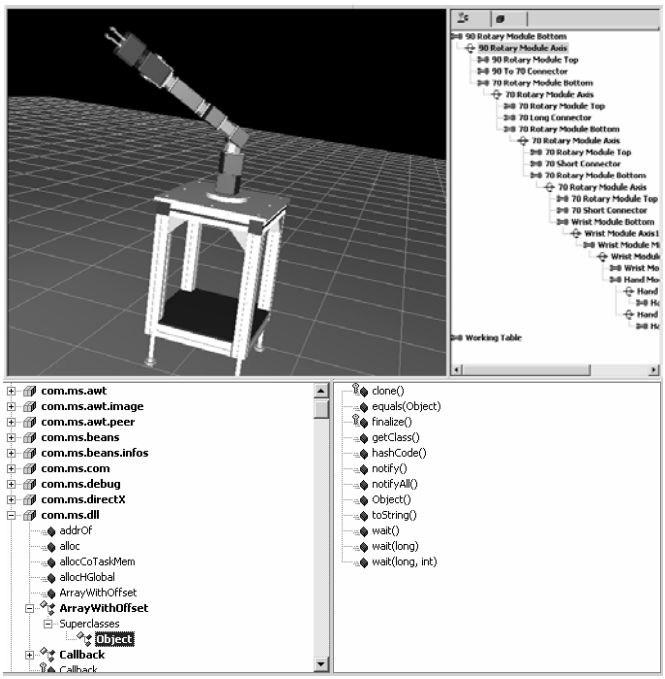


Fig.14: HW/SW modules configurations in robotic system design

The problem of robotic system simulation and design are also tightly coupled so that the design and simulation teams should share a common model and access into the same workspace, although these teams may be in remote locations. The robot system can be operated by means of a virtual robot manipulation system constructed in the web scheme, in which 3D models of the components are manipulated virtually in a computer graphics. The robotic system simulator developed by the simulation team provides a new design tool for designers in the design team to carry out the flexible assembly and the intuitive operations and simulations. This can help the designers to verify the design. When a simulation sequence is running, users or designers can control positions and orientations of the robot and the components, and open-close states of the gripper by clicking on them. The user interface graphically displays robot configurations, gripper states, and the component states. The simulation results can also help the designers in the design team to modify and redesign the design if necessary. Fig.13 shows the robotic system design and analysis modules and modules network. Fig.14 gives HW/SW module configurations in the robotic system design.

8. SUMMARY AND CONCLUSIONS

In this paper, we presented a design with modules scheme and a knowledge intensive cyber-infrastructure framework (WebDMME) for collaborative embedded system design modeling and support. Because of the heterogeneous structure, the design and simulation processes require different grades of abstraction and need the cooperation and collaboration of different disciplines and resources. The developed framework can provide distributed designers with a tool for collaboratively building integrated models. The advantage of the demonstrated modular concept consists in the flexibility of the program structure and the reduction of costly software support by integrating design tools and simulators. Large problems are decomposed into sub-problems with modules. Models or other software applications are encapsulated in modules. A module can provide information services through its interface, and the network of modules exchanging services form a concurrent design model. Therefore, the behavior of complex systems and the interactions of components can be analyzed and optimized during the design process, resulting in shorter manufacturing cycles.

As the knowledge-server based framework was built to provide the module network architecture for integrating modeling services available on the network, it can accommodate top-down and bottom-up approaches in the context of both the traditional HW/SW separate design process and the HW/SW co-design process. In the module network, design resources, models, data, and activities are not centralized nor concentrated in one location. They are distributed among many companies, designers, or design participants working together over the Internet/Intranet. Thus, the module network architecture is extended to a computer network environment, focusing on the web-based knowledge intensive and collaborative design modeling and support. Fully implementing the locally- defined modules and subscribing to the services of the remote modules create design modules and modules network. In the module network architecture, when modules services are connected, the resultant service exchange network

creates an integrated concurrent system model or module network that invoke a chain of service requests if needed to provide correct information.

Compared to existing research efforts, the framework presented in this paper differs in its focus to create an intelligent design modeling scheme that handles the different variable types and knowledge needed in embedded system design, integrate multiple objective evaluation and optimization with design models, and provide an object oriented design methodology to facilitate the intelligent integration of design models and their utilization in an open and distributed intelligent design environment.

Disclaimer

No approval or endorsement of any commercial product, service or company by the National Institute of Standards and Technology is intended or implied.

REFERENCES

Abrahamson, S., Wallace, D., Senin, N., Sfereo, P., Integrated design in a service marketplace, Computer-Aided Design 32 (2) (2000) 97–107

Balarin, F., Watanabe, Y., Hsieh, H., Lavagno, L., Passerone, C., Vincentelli, A.S., Metropolis: An integrated electronic system design environment, Computer, IEEE Computer Society, pp. 45-52, 2003

Bidarra, R., van den Berg, E., Bronsvoort, W.F., Collaborative modeling with features, Proceedings of 2001 ASME Design Engineering Technical Conferences, Pittsburgh, Paper No. DETC2001/CIE-21286

Bliznakov, P. I., Shah, J. J., Jeon, D. K. and Urban, S. D., Design information system infrastructure to collaborative design in a large organization, Proceedings of ASME DETC 95, Boston, MA, 1: pp. 1-8, 1995.

Camarinha-Matos, L.M., Afsarmanesh, H., Osorio, A.L., Flexibility and safety in a web-based infrastructure for virtual enterprises, International Journal of Computer Integrated Manufacturing 14 (1) (2001) 66–82.

Chappel, D., Understanding ActiveX and OLE, Redmond, WA: Microsoft Press, 1996

Chao, P.-Y., Wang, Y.-C., A data exchange framework for networked CAD/CAM, Computers in Industry 44 (2) (2001) 131–140.

Cheng, K., Pan, P.Y., Harrison, D.K., Web-based design and manufacturing support systems: implementation perspectives, International Journal of Computer Integrated Manufacturing 14 (1) (2001) 14–27.

Chen, Y.M. Liang, M.W., Design and implementation of a collaborative engineering information system for allied concurrent engineering, International Journal of Computer Integrated Manufacturing 13 (1) (2000) 11–30.

Eggermont, L. D.J. (ed.) (2002), Embedded Systems Roadmap 2002, Vision on Technology for the Future of PROGRESS, 30 March

Eriksson, H., Expert systems as knowledge servers. IEEE Expert, 14(3): 14-19, 1996

Fenves, S. J. (2001), A core product model for representing design information, NISTIR 6736, NIST, Gaithersburg, MD

Gero, J.S., Design prototypes: a knowledge representation schema for design. AI Magazine, 11(4): 6-36, 1990

Gorti, S. R., Gupta, A., Kim, G. J., Sriram, R. D., and Wong, A., 1998, An object-oriented representation for product and design process, Computer-Aided Design, 30(7): 7, pp. 489–501

Hardwick, M. Spooner, D.L. Rando, T. Morris, K.C., Sharing manufacturing information in virtual enterprise, Communications of the ACM 39 (2) (1996) 46–54

Han, C.S., Kunz, J.C., Law, K.H., An internet-based distributed service architecture, Proceedings of the 1999 ASME Design Engineering Technical Conferences, Las Vegas, NV, 12–15 September 1999 (CD).

Hassani, Mehrdad, A Component-based Methodology for Real-time Decision-making Embedded Systems, PhD Dissertation, University of Maryland, 2000

Jin, Y., Zhao, L., Raghunath, A., ActiveProcess: a process-driven and agent-based approach to supporting collaborative engineering, Proceedings 1999 ASME Design Engineering Technical Conferences, Las Vegas, NV, 12–16 September 1999 (CD).

Klein, M., Supporting conflict resolution in cooperative design systems, IEEE transactions on systems, Man and Cybernetics 21 (6) (1991) 1379–1390.

Kim, Y., Kang, S.-H., Lee, S.-H., Yoo, S.B., A distributed, open intelligent product data management system, International Journal of Computer Integrated manufacturing 14 (2) (2001) 224–235

Kim, C.-Y., Kang, S.-H., Kim, N., O'Grady, P., Internet-based concurrent engineering: an interactive 3D system with markup, Proceedings of ASME 18th Computers in Engineering Conference, Atlanta, GA, 13–16 September 1998 (CD)

Kim, H., Lee, J.Y., Han, S.-B., Process centric distributed collaborative design based on the web, Proceedings of ASME 19th Computers in Engineering Conference, Las Vegas, NV, 12–15 September 1999 (CD)

King, S., Co-design: a process of design participation, New York: Van Nostrand Reinhold, 1989

Konduri, G. and Chandrakasan, A., Framework for collaborative and distributed web-based design, Proceedings of Design Automation Conference, Proceedings of the 1999 36th Annual Design Automation Conference (DAC), p.898-903, Jun 21-Jun 25 1999, New Orleans, LA, USA

Kusiak, A. and Wang, J., Dependency analysis in constraint negotiation, IEEE transactions on systems, Man and Cybernetics, 25 (9) (1995) 1301–1313

Lander, S. E., Issues in multi-agent design systems, IEEE Expert: Intelligent System & Their Application, 12(2), 1997

Lee, J.Y., Kim,H., and Han, S.B., Network-centric feature-based modeling, Proceedings of Pacific Graphics'99, pp. 280-289, Seoul, Korea, 1999

Lee,Edward A., Overview of the Ptolemy Project, Technical Memorandum No. UCB/ERL M03/25, University of California, Berkeley, CA, 94720, USA, July 2, 2003.

Lewis J. W. and Singh, K. J., Electronic design notebooks (EDN): Technical issues, Proceedings of Concurrent Engineering: A Global Perspective, McLean, VA, pp. 431-436, 1995

Li, Y.L., Shao, X.Y., Li, P.G., and Liu, Q., Design and implementation of a process-oriented intelligent collaborative product design system, Computer in Industry, 53, pp.205-229, 2004

Lu, S.C.Y., Cai, J., Burkett, W., Udwadia, F., A methodology for collaborative design process and conflict analysis, Annals of CIRP 49 (1) (2000) 69–73

MADEFast, http://madefast.stanford.edu/, 1999

NIIIP, 1999, http://www.niiip.org/

O'Grady,P., Liang, W.Y., An object oriented approach to design with modules, Iowa Internet Laboratory Technical Report TR98-04, 1998

Pahng, F., Senin, N. and Wallace, D. R. Modeling an evaluation of product design problems in a distributed design environment, CD ROM Proceedings of ASME DETC, Sacramento, CA, 1997

Pahng, .F., Senin, N., and Wallace, D.R., Distribution modeling and evaluation of product design problems, Computer-Aided Design, 30(6): pp.411-423, 1998a

Pahng, F., Bae, S.H., and Wallace, D.R., Web-based collaborative design modeling and decision support, Proceedings of DETC'98, Atlanta, Georgia, USA, 1998b

Petrie, C. Cutkosky, M. and Park, H., Design space navigation as a collaborative aid, Proceedings of Third International Conference on Artificial Intelligence in Design, Lausanne, Switzerland, 1994

Pena-Mora, F., Sriram, R. D. and Logcher, R., Conflict mitigation system for collaborative engineering, AI EDAM -Special Issue of Concurrent Engineering, 9(2): pp. 101-123, 1995

Pena-Mora, F. Sriram, R.D. and Logcher, R., SHARED DRIMS: SHARED design recommendation and intent management system, Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE Press, pp. 213-221, 1993

RaDEO, 1998, http://elib.cme.nist.gov/radeo/

Rosenman, M. and Wang, F.J., CADOM: A component agent-based design-oriented model for collaborative design, Research in Engineering Design, 11: pp.193-205, 1999

Rushton, G. J., and Zakarian, A., Development of Modular Vehicle Systems, Department of Industrial and Manufacturing Systems Engineering, University of Michigan, Dearborn, 2000

Sriram, R. D. and Logcher, R., The MITDICE Project, IEEE Computer, pp. 64-65, 1993

Sriram, R.D., Distributed and Integrated Collaborative Engineering Design, Sarven Publishers, Glenwood, MD 21738, Dec. 2002

Shyamsundar, N., Gadh, R., Internet-based collaborative product design with assembly features and virtual design spaces, Computer-Aided Design 33 (9) (2001) 637–651.

Singh, A. K., CONSENS - An IT solution for concurrent engineering, Proceedings of Concurrent Engineering: A Global Perspective, McLean, VA, pp. 635-644,1995

Senin., N., Wallace, D. R., Borland, N., and Jakiela, M.J., A framework for mixed parametric and catalog-based design problem modeling and optimization, MIT CAD Lab - Technical Report: 97.02, 1997a

Siegel, J., CORBA: Fundamentals and Programming: OMG, 1996

Stokes, M., Managing Engineering Knowledge: MOKA Methodology for Knowledge Based Engineering Applications, MOKA Consortium, London

Sun, J., Zhang, Y.F., Nee, A.Y.C., A distributed multi-agent environment for product design and manufacturing planning, International Journal of Production Research, 39 (4) (2001) 625–645

Sivaloganathan, S., Andrews, P.T.J., and Shahin, T.M.M., Design function deployment: A tutorial introduction, Journal of Engineering Design, vol.12, No.1, pp.59-74, 2001

Tan, G.W., Hayes, C.C., Shaw, M., An intelligent-agent framework for concurrent product design and planning, IEEE Transactions on Engineering Management, 43 (3) (1996) 297–306

Toye, G., Cutkosky, M. R., Tenenbaum, J. M. and Glicksman, J., SHARE: A methodology and environment for collaborative product development, Proceedings of Second Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Morgantown, West Virginia, pp. 33-47, 1993

Wang, Kai-Lu and Jin, Yan, Managing dependencies for collaborative design, Proceedings of 2000 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Baltimore, MA, 10–13 September 2000 (CD).

Wood III, W.H. and Agogino, A.M., Case based conceptual design information server for concurrent engineering, Computer-Aided Design, 8(5): 361-369, 1996

Whitfield, R.I., Duffy, A.H.B., Coates, G., Hills, W., Distributed design coordination, Research in Engineering Design 13 (2002) 243–252

Wong, S.T.C., Coping with conflict in cooperative knowledge based systems, IEEE Transactions on Systems, Man and Cybernetics—Part A, Systems and Humans, 27 (1) 57–72, 1997

Zha, X.F., and Sriram, R.D., 2004, Collaborative product development and customization: a platform-based strategy and implementation, Proceedings of ASME DETC 2004, Paper No.: DETC2004-57709

Zha, X. F. and Sriram, R.D., 2004 Feature-based component model for design of embedded system, in Intelligent Systems in Design and Manufacturing V, edited by B. Gopalakrishnan, Proceedings of SPIE Vol.5605 (SPIE, Bellingham, WA, 2004), pp. 226-237

Zha, X.F., Fenves, S. and Sriram, R.D. (2005), A feature-based approach to embedded system hardware/software co-design, submitted to 2005 ASME DETC, 2005

Zha, X.F., Fenves, S. and Sriram, R.D. (2005), Object-oriented representation for embedded systems using UML, Working Paper, National Institute of Standards and Technology, 2005

Zhao, G., Deng, J., and Shen, W., CLOVER: an agent-based approach to systems interoperability in cooperative design systems, Computers in Industry, 45 (3) (2001) 261–276.