

DETC2009/DAC-86899

MODULAR PRODUCT DESIGN USING CYBERINFRASTRUCTURE FOR GLOBAL MANUFACTURING

John Jung-Woon Yoo^{*}, Soundar R. T. Kumara, and Timothy W. Simpson
Harold & Inge Marcus Department of Industrial & Manufacturing Engineering
The Pennsylvania State University, University Park, PA 16802, USA

ABSTRACT

Modularization of parts - a fairly recent trend in product development - facilitates part definitions in a standardized, machine-readable form, so that we can define a part based on its input(s), output(s), features, and geometric information. Standardizing part definitions will enable manufacturing companies to more easily identify part suppliers in global, virtual environments. This standard representation of parts also facilitates modular product design during parametric design. We will show that this problem of modular product design can be formulated as an AI Planning problem, and we propose a solution framework to support modular product design. Using part specification information for personal computers, we demonstrate the proposed framework and discuss its implications for global manufacturing.

Keywords: Modularization, Automation, AI Planning, Cyberinfrastructure, Global Manufacturing

1 INTRODUCTION

Global product development is transforming the way many companies do business, and product modularity is a critical component for success [1]. Large multi-national corporations are using modularity to help create innovative product development systems that utilize the best practices among different divisions to speed up development and reduce costs. For example, Ford is leveraging the best practices of its four brands – Ford, Mazda, Volvo, and Aston Martin/Land Rover – to “bring vehicles to the market faster and for less cost” [2]. Boeing’s strategy for developing the 787 Dreamliner is similar, having partnered with 15 companies in ten U.S. States and seven countries to create the major structural systems of the aircraft [3].

Clearly defined interfaces between modules are a key enabler for the success of distributed global product development. They enable geographically distributed teams to work autonomously before integration of modules into the product. Without such modularity, “more intense collaboration across design interfaces is necessary” [1], which invariably

causes delays and missteps in the product development process. The problem is compounded further when working on a global scale, and in today’s global economy, cyberinfrastructure is becoming increasingly critical for seamless integration and maintenance of organizational operations [4]. Having clearly defined interfaces between modules will foster the creation of standardized interface representations because modular components are composed of self-contained functionality and multiple input/output interfaces to other modules. This permits state-of-the-art machine-readable languages developed by computer scientists and information technologists to be adapted to support modular product design.

Our vision is to utilize Service Oriented Architecture (SOA), which is the most advanced Web-based service system architecture [5], to formalize a cyberinfrastructure-based framework to support modular product design for global manufacturing. The proposed framework is rooted in an analogy between SOA’s web services and their composition to modules and modular product design, respectively. To realize this framework, we need to address three challenges that currently hinder the use of SOA in modular product design:

1. Develop an interface-oriented machine-readable representation scheme for modular components;
2. Formalize an accessible cyberinfrastructure-based framework that enables global users to describe, publish, and discover components information in a standardized way; and
3. Adapt Artificial Intelligence (AI) planning algorithms to support modular product design.

In this paper, we focus primarily on the third challenge. The research motivation is discussed in the next section. Section 3 reviews previous literature associated with modular product design and Web service composition, which is analogous to modular product design. Section 4 introduces the proposed framework, focusing on mathematical formulations for automated modular product design using cyberinfrastructure. Section 5 verifies the proposed integer programming formulation through a case study. Finally,

^{*} Please address all correspondences to this author at: jwwoo@psu.edu.

Section 6 concludes the paper and discusses relevant future work.

2 MOTIVATION

The current global manufacturing environment has significantly transformed product development processes. Companies are taking advantage of specialties from diverse companies from all over the world. For example, companies in the U.S. can utilize inexpensive but productive labor from low wage countries like China, and India and also leverage competitive designs from European countries like France or Italy. However, the geographical distances and language barriers hinder the maximal utilization of the resources in the current globalized environment. Salespeople travel all over the world with multi-language versions of hard-copy catalogs that advertise available parts and components. Word of mouth is still one of the common practices for direct marketing.

Benchmarking the current practices of Web service providers gives us indications of the potential direction of product design practices. The number of Web services has exploded recently, and there are more than thirty thousand Web services available [6]. The Web service community has developed a service-oriented framework for Web services on the Internet, called Service Oriented Architecture (SOA). The architecture defines a standard representation scheme for the specification of Web services, called Web Service Description Language (WSDL). The architecture includes a virtual storage of the specifications, called Universal Description Discovery and Integration (UDDI), through which Web services users from all over the world can store, retrieve, discover and use Web services. The standard representation scheme and the virtual storage are prompting the use of Web services from all over the world.

Global product development processes can be facilitated by leveraging both the contemporary modularization trend in product design and the Service Oriented Architecture proposed by the Web service community. Such synergetic efforts will significantly differentiate this new global paradigm from the current best practice, namely, proprietary catalogue-based systems. Figure 1 depicts a hypothetical scenario for global manufacturing based on modular product design using cyberinfrastructure. Suppliers from all over the world publish their product information written in a machine-readable language on a global design repository. Product designers from companies can refer to the components' information stored in the design repository and find proper components which satisfy their design parameters. Table 1 compares the characteristics of the current hardcopy catalogue-based system and the proposed cyberinfrastructure-based system. Note that the proposed cyberinfrastructure-based system enables unlimited access to global information through automated search engines leveraging machine-readability and reusability. In this paper, we focus on detailed, parametric product design within the overall product development process, particularly modular product design. We identify three major elements for the modular product design for global manufacturing as: (1) a standard representation scheme for the specification of

modularized components, (2) a digital design repository to capture these specifications, and (3) a product design “engine” that reduces the feasible set of design alternatives and possibly finds the best solution among them. The next section reviews literature related to the proposed work.

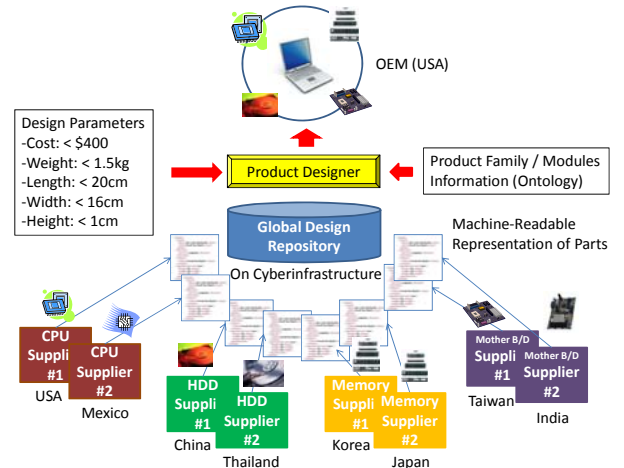


Figure 1. Modular Product Design Framework for Global Manufacturing using Cyberinfrastructure

Table 1. Characteristics of Hardcopy Catalogues and Global Design Repository

	Catalogues (Hardcopy)	Global Design Repository (using Cyberinfrastructure)
Representation	Non-standard	Standard (machine-readable)
Search	Brute-force search Word of mouth	Search engine Case-based reasoning
Information	Local	Global
Accessibility	Limited	Unlimited
Reusability	Not reusable	Reusable

3 BACKGROUND AND RELATED WORK

Machine-readable representation of modules is the first step for the success of modular product design for global manufacturing in the proposed cyberinfrastructure-based framework. The National Institute for Standards and Technology (NIST) has proposed the use of eXtensible Markup Language (XML) for the description of functions and associated flows in computer-based design [7-10]. Devanathan, et al. [11] also represent the concept of a component in XML. Bohm, et al. [12] introduced an extensive data schema to capture fundamental elements of design information. These representation schemes for products or parts are designed mostly from the viewpoint of the materials or energy flow, which is sequential or *flow-oriented*; however, the recent trend of modularization demands a totally new viewpoint for representation. The new viewpoint is *interface-oriented*, in which physical interfaces help connect modularized components. An interface can consist of a set of flows, where

the flows are described more in detail than those in Functional Basis for detailed, parametric product design.

The standardization of language for the representation of components is a crucial element for the common understanding on components. Standardization efforts yielded the Functional Basis [13-14], which abstracts terms of function and flow, limits the number of terms, and recommends using those terms in product design. The Functional Basis plays an important role in the systematic description of components as well as the transparent communications among designers or design software. However, abstraction gives rise to incompatibility issues due to the ambiguity of the abstracted terms.

Bryant, et al. [15-16] developed a computational tool for automated concept generation. After creating a function chain, they select relevant components utilizing Function-Component Matrix (FCM), and check the compatibility among components using Design Structure Matrix (DSM). However, their concept generator currently considers neither physical nor non-functional properties of components, such as weight or price, respectively. Their tool also does not support branching scheme during the assembly of modules and limits its application to components with single input/output.

Regarding other computational product design research, Campbell, et al. [17-18] implemented an agent-based approach to automated design synthesis for electromechanical products. Their system, called A-design, not only generates but also iteratively improves design configurations based on pre-set multiple objectives. A-design is based on the agent-based and adaptive nature of the process [17-18]. Mittal, et al. [19] implemented an expert system, called PRIDE, for the design of paper handling systems. It acquires knowledge from expert designers and performs knowledge-guided search for possible designs that satisfy requirements. Navinchandra, et al. [20] present a case-based approach to exploit the knowledge embodied in prior designs. They captured and saved prior designs regardless of success or failure and utilized them to take advantage of the prior success and not to repeat the previous failure. They applied this case-based approach to the conceptual design of hydro-mechanical systems. Finally, Titus and Ramani [21] explore design spaces using the constraint satisfaction approach.

In parallel, as briefly mentioned above, computer scientists and information technology (IT) researchers have been developing a formal way to describe, publish, store, and discover software components, called Service Oriented Architecture (SOA). Figure 2 depicts SOA-based Web services. The Universal Description Discovery and Integration (UDDI) and Web services technology, which are two major components of SOA, form a cyberinfrastructure for services on the Web. UDDI is a registry or storage for published Web services from various Web service providers. Conceptually, in our proposed framework, a UDDI corresponds to the digital Design Repository like the one implemented by the Missouri University of Science & Technology and NIST [10, 22] while a Web service corresponds to a physical component whose

specification is stored in the repository. The digital Design Repository stores specification data of selected products and their components and relationship data among components, so that users can review the data through Web browsers. In addition, the repository provides users with design tools, such as FCM and DSM, as well as a product concept generator.

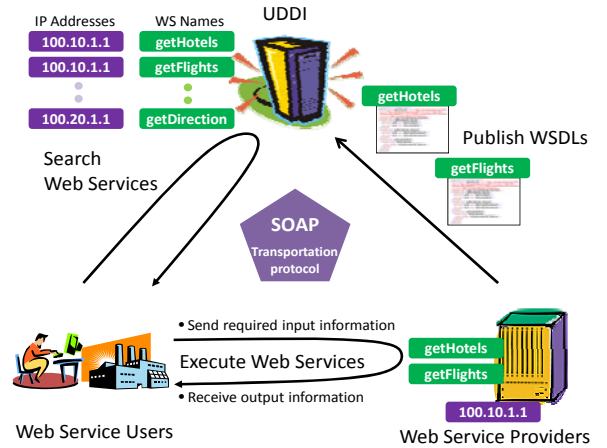


Figure 2. SOA using Web Services

A similar approach to modular product design has been identified in the Web community. The research field is commonly known as Web service composition [23], where a software agent selects and arranges appropriate Web services while matching input and output information to obtain pre-set goal information [24-25]. Thus, the composed Web services can provide a solution that a single Web service cannot. Figure 3 illustrates an example of Web service composition for the search of a “Thai” restaurant around a hotel. Note that the input parameters of a Web service should be satisfied in order to invoke it, and that the output parameters from one Web service are used for the input of other Web services.

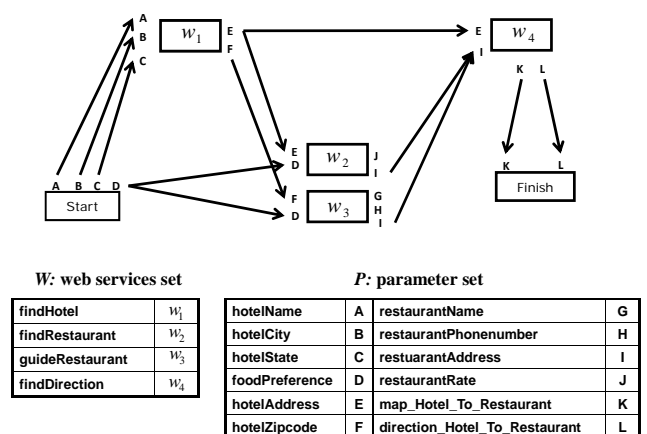


Figure 3. Web Service Composition Example [24]

The composition of Web services to respond to users' queries can be analogous to modular product design except

that Web services do not have physical properties that modules in a product have, such as size or weight. Since Web services do not have physical properties, most of Web service composition algorithms leverage logic-based approaches, such as propositional logic or satisfiability techniques, which consider only functional requirements. Recently, Yoo, et al. [26] have proposed a mathematical programming framework for Web service composition, based on Optiplan proposed by Van Den Briel and Kambhampati [27]. The framework can incorporate not only functional requirements but also non-functional attributes, such as cost or quality of service.

The digital Design Repository and machine-readable component information form a cyberinfrastructure for modular product design that can be utilized to support global product development. Standardized component information published by suppliers can be shared with manufacturers from all over the world through the Internet. Additionally, machine-readable components information facilitates modular product design processes regardless of the location of companies.

Our review of related work highlights the limitations of current research. Consequently, we propose an algorithm to facilitate the synthesis of modular product design concepts, which leverages the machine-readable XML representation of modularized components. The XML schema describes modules in terms of features, interfaces, and geometrical information. The proposed modular product design framework can incorporate not only functional requirements of a product but also non-functional ones, such as weight, cost, etc. into its mathematical model, and support branching and merging and multiple input/output, which prior research has overlooked.

4. METHODOLOGY

This section discusses the proposed cyberinfrastructure for global manufacturing. The major elements of the cyberinfrastructure are composed of the formal representation of components, a component repository, and a modular product design framework. The following sections discuss each element in detail, and our major focus in this paper is on the AI Planning-based modular product design framework.

4.1 Formal representation of components

Previous research performed by the Missouri Institute of Science and Technology (MS&T, former the University of Missouri – Rolla, UMR) and the National Institute of Standards and Technology (NIST) identified functionality, input/output flows, and physical parameters (e.g., dimensions) as key elements for component representation [14]. However, recent trends in modularization draw our attention to interfaces among components. A modularized component consists of a functional body and multiple interfaces to other components. For instance, a hard disk drive has a functional body that is used for reading/writing digital data, and two interfaces to the power supply and the motherboard as shown in Figure 4. The power supply interface can be either AC or DC with a certain voltage, while the motherboard interface can be either SCSI or IDE.

We introduce an interface-oriented, machine-readable representation scheme for modularized components. In particular, we extend the current representation scheme developed by NIST and MS&T to include precisely defined interfaces attributed to the modularization trend, as shown in Figure 5. The proposed interface-oriented representation can contribute not only to conceptual design, but also detailed, parametric design that the current representation could not support. Figure 6 topologically illustrates the concept of modularization and interface-oriented modular product design. Before modularization, there exist non-standard connections among parts, and designers should consider all the connections during design; however, modularization helps designers focus on external, standardized interfaces. Such modularization simplifies the representation of components as well as helps reduce factors to consider during design since complex interactions can be abstracted by interfaces between modules.

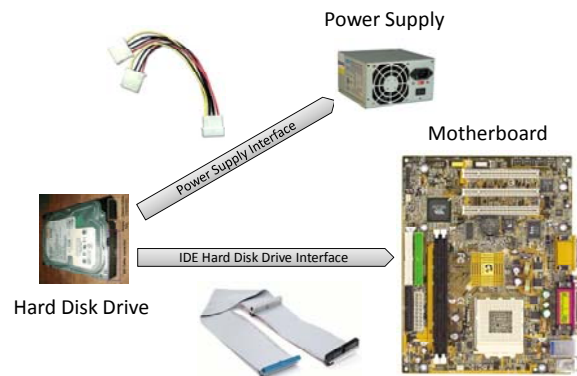


Figure 4. Standardized Interfaces: Examples from a Hard Disk Drive

```
<?xml version="1.0" encoding="UTF-8" ?>
<tns:Component xmlns:tns="http://product.repository"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://product.repository Component.xsd">
<generalInfo>
<companyName>Intel</companyName>
<componentName>MotherBoard</componentName>
<modelName>BOXD845PEBT2</modelName>
<price unit="USD">300.00</price>
<weight unit="lb">2.5</weight>
<warranty unit="year">3</warranty>
<geometricSpecification unit="in">
<width>3.0</width>
<height>4.0</height>
<depth>1.0</depth>
</geometricSpecification>
</generalInfo>
<inputInfo>
<voltage unit="V">16</voltage>
</inputInfo>
<outputInfo>
<cpu>Socket372</cpu>
<memory>DIMM</memory>
</outputInfo>
<interfaceInfo>
<hdd>SCSI</hdd>
</interfaceInfo>
<functionInfo>Control</functionInfo>
</tns:Component>
```

Figure 5. Machine-Readable XML Representation

The proposed interface-oriented representation scheme is machine-readable, so that component representations can be

globally shared. Machine-readability can be achieved by leveraging Extensible Markup Language. XML, a W3C recommendation, is designed to describe data and the structure of the data by using self-defined tags, different from HTML (Hypertext Markup Language), which is designed to display data by using pre-defined tags, e.g., <h1>, <p>, [26]. XML's self-descriptiveness, which comes from the Document Type Definition (DTD) or XML Schema, provides maximal degrees of freedom for defining new data types.

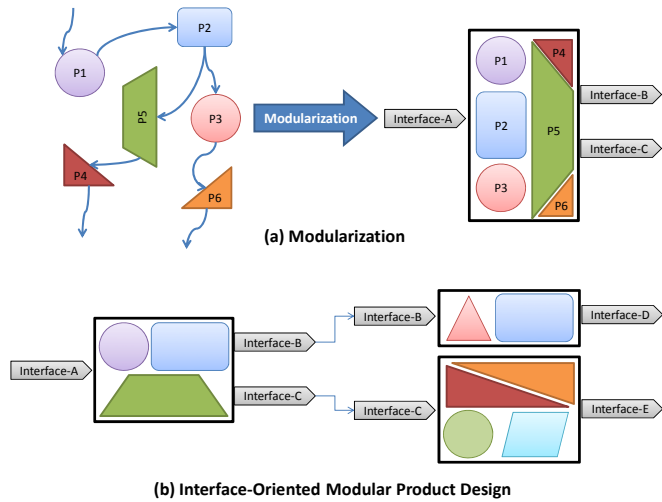


Figure 6. Concept of Modularization and Interface-Oriented Modular Product Design

4.2 Modular product design as an AI Planning problem

The XML-based machine-readable representation scheme facilitates automated modular product design. The generated design for the modular product should satisfy functional and non-functional requirements. Functional requirements require component compatibility, while non-functional requirements constrain the size, cost, quality, or weight of products. Functional requirements are the most essential because the designed product should work functionally; however, the current trend of mass customization emphasizes the significance of non-functional requirements. The following describes how the modular product design problem can be transformed to an AI Planning problem.

AI planning problem attempts to solve navigating through a state space, stating from an initial one to reach a goal state(s). An AI Planning problem, P , can be represented as $P = (\Sigma, s_0, g)$ and, $\Sigma = (S, A, \gamma)$ where S is the set of states, A is the set of actions; $\gamma: S \times A \rightarrow 2^S$ is a state transition function; s_0 is the initial state; and g is the goal state. A solution to P is a sequence of actions that are taken to reach from the initial state to the goal state [25, 28].

The proposed algorithm is rooted in an analogy between Web services to their composition and modules to modular product design. Our earlier work [25] has formulated Web service composition problems as AI Planning problems. The main difference between Web service composition and

modular product design resides in that modules in a product have physical shape, while Web services do not. Table 2 summarizes the similarity between modular product design and Web service composition and how the elements of the two correspond to the key terms of AI Planning. In short, the initial state corresponds to the fundamental component or resources in product design, such as a power supply; the goal state corresponds to the final product; states correspond to currently available interfaces and features; and actions correspond to the assembly of components. However, there exist significant differences among the two problems, caused by the physical characteristics of modules. The differences are discussed in terms of functional and non-functional requirements.

Table 2. AI Planning Problems: Modular Product Design and Web Service Composition

AI Planning	Modular Product Design	Web Service Composition
Action	Assemble	Compose
State	Available interfaces and features	Known parameters
Initial State	Fundamental components or resources (e.g., power)	Given (initial) information
Goal State	final product (e.g., iPod)	Goal information

(1) Functional requirements

Functional requirements require adjacent components to be interface compatible, and each interface to be used non-redundantly. Compatibility checks are essential for both modular product design and Web service composition; however, non-redundant use of interfaces is unique to modular product design due to its physical characteristics. Known parameters during Web service composition are data, which can be shared (reused) by other Web services. On the other hand, once an available interface of a module is connected to that of another module, then the interface cannot be shared by other modules because the interface is already consumed and does not exist. The AI Planning based modular product design algorithm supports such capability as compatibility check and non-redundant use of interfaces. Bryant, et al. [15-16], in one of the most recent works, used a DSM to verify compatibility between two adjacent components. However, DSM supports neither branching & merging of components in a product nor multiple input/output of components, which commonly exist in product design. The proposed approach can also incorporate the consideration of branching and merging into the AI Planning model. The machine-readable representation for modularized components plays a key role in satisfying functional requirements.

(2) Non-functional requirements

Few existing computational product design methods consider non-functional requirements, such as weight, or price, even though such requirements are crucial for the current market trend of mass customization. Furthermore, most AI Planning approaches are logic based, which cannot incorporate numerical constraints into their models. Therefore,

we use an Integer Programming (IP) formulation that can incorporate the non-functional requirements into its mathematical model, in which the optimal product design is generated based on users' objectives. For example, the proposed model can have maximal or minimal weight or price as its objective, depending on the targeting market segment. The detailed IP formulations are described next.

4.3 Integer Programming (IP) Formulation

The following sections define the IP formulation for the proposed framework.

4.3.1. Domain Definition

P is the set of parts published in the cyberinfrastructure.

I is the set of all interfaces of parts in P .

F is the set of all features of parts in P .

$I_{In} \subseteq I$ is the set of interfaces that are used as input to any part.

$I_{Out} \subseteq I$ is the set of interfaces that are used as output in any part.

$F_{Out} \subseteq F$ is the set of features that are obtained from any part.

$I_{Initial} \subseteq I$ is the set of interfaces that are initially given.

$I_{Goal} \subseteq I$ is the set of interfaces that are goals.

$F_{Goal} \subseteq F$ is the set of goal features.

$P_i^{input} \subseteq P, \forall i \in I$ is the set of parts that have interface i as input.

$P_i^{output} \subseteq P, \forall i \in I$ is the set of parts that have interface i as output.

$P_f^{output} \subseteq P, \forall f \in F$ is the set of parts that have feature f as output.

SF_s is the set of features obtained until Stage s .

Stage (s): $1 \leq s \leq S$, where S is the maximal number of stages for modular product design.

P_s is the set of parts simultaneously used in product design at Stage s .

4.3.2. Variable Definition

For all $p \in P, s \in 1, \dots, S, y_{p,s}$'s are *part usage variables*.

$$y_{p,s} = \begin{cases} 1 & \text{if part } p \text{ is used in stage } s, \\ 0 & \text{otherwise.} \end{cases}$$

The following variables are *interface usage variables*.

$$x_{i,s}^{available-unused} = \begin{cases} 1 & \text{if interface } i \text{ is available but not used in stage } s, \\ 0 & \text{otherwise.} \end{cases}$$

$$x_{i,s}^{input} = \begin{cases} 1 & \text{if part } p \text{ is used in stage } s \text{ such that } p \in P_i^{input}, \\ 0 & \text{otherwise.} \end{cases}$$

$$x_{i,s}^{output} = \begin{cases} 1 & \text{if part } p \text{ is used in stage } s \text{ such that } p \notin P_i^{input} \wedge p \in P_i^{output}, \\ 0 & \text{otherwise.} \end{cases}$$

$$x_{f,s}^{feature} = \begin{cases} 1 & \text{if part } p \text{ is used in stage } s \text{ such that } f \in F, \\ 0 & \text{otherwise.} \end{cases}$$

4.3.3. Formulation

(1) Objective Function

We can define any numerical expression for the objective function. Based on the characteristics of targeting market segments, weight, or price can be the objective. Furthermore, multiple objectives can be defined. In such case, goal programming or other multi-criteria optimization methodologies can be utilized. The following objective function is an example of a single objective of cost.

$$\text{Minimize } \sum_{p \in P} \sum_{s \in S} c_p \cdot y_{p,s}, \text{ where } c_p \text{ is the cost for the use of part } p.$$

(2) Initial constraints

The initial input interfaces are expressed by setting 1 to output interface usage variables at Stage 0.

$$x_{i,0}^{output} = 1, x_{i,0}^{input} = x_{i,0}^{available-unused} = 0 \quad \forall i \in I_{Initial} \quad : \quad \text{Given}$$

interfaces at initial stage

$$x_{i,0}^{input}, x_{i,0}^{output}, x_{i,0}^{available-unused} = 0, \forall i \notin I_{Initial} \quad : \quad \text{Other interfaces at initial stage}$$

$$x_{f,0}^{output} = 0, \forall f \in F \quad : \quad \text{All features at initial stage}$$

(3) Goal constraints

The goal of modular product design is represented in goal constraints. If the goal features are used as output variables at the last stage, the goal is achieved.

$$x_{f,S}^{input} + x_{f,S}^{output} + x_{f,S}^{available-unused} \geq 1 \quad \forall f \in F_{Goal} \quad : \quad \text{Goal features at the final stage}$$

$$x_{i,S}^{input} + x_{i,S}^{output} + x_{i,S}^{available-unused} \geq 1 \quad \forall i \in I_{Goal} \quad : \quad \text{Goal interfaces at the final stage.}$$

(4) Compatibility constraints

The compatibility constraints play a role of making the functional requirements satisfied.

$$A. \sum_{p \in P_i^{output} \setminus P_i^{input}} y_{p,s} \geq x_{i,s}^{output} \quad \forall i \in I, s \in 1, \dots, S$$

$$B. y_{p,s} \leq x_{i,s}^{output} \quad \forall p \in P_i^{output} \setminus P_i^{input}, \forall i \in I, s \in 1, \dots, S$$

$$C. \sum_{p \in P_f^{output}} y_{p,s} \geq x_{f,s}^{output} \quad \forall f \in F, s \in 1, \dots, S$$

$$D. y_{p,s} \leq x_{f,s}^{output} \quad \forall p \in P_f^{output}, \forall f \in F, s \in 1, \dots, S$$

$$E. \sum_{p \in P_i^{input}} y_{p,s} \geq x_{i,s}^{input} \quad \forall i \in I, s \in 1, \dots, S$$

$$F. y_{p,s} \leq x_{i,s}^{input} \quad \forall p \in P_i^{input}, \forall i \in I, s \in 1, \dots, S$$

Each pair of Constraints A and B, Constraints C and D, and Constraints E and F enforces, respectively, available inputs, outputs, and features of an interface to be used only one time at a stage, which is a key to functional requirements.

(5) Non-concurrency constraints

Once interface i is decided to be used as an input or output variable, which means $x_{i,s}^{input}$ or $x_{i,s}^{output}$ is set to 1,

$x_{i,s}^{available-used}$ should not be used. In other words, it should not be set to 1. $x_{i,s}^{output} + x_{i,s}^{available-used} \leq 1 \quad \forall i \in I, s \in 1, \dots, S$

$$x_{i,s}^{input} + x_{i,s}^{available-used} \leq 1 \quad \forall i \in I, s \in 1, \dots, S$$

(6) Sequence constraints

Only when interface i is used as an output in previous stages, it can be used as input or available-unused variables. Such sequential requirements are represented in the sequence constraints.

$$x_{i,s}^{input} + x_{i,s}^{available-used} \leq x_{i,s-1}^{input} + x_{i,s-1}^{output} + x_{i,s-1}^{available-used} \quad \forall i \in I, s \in 1, \dots, S$$

(7) Binary variables

Here are all the variables defined in the formulation.

$$x_{i,s}^{input}, x_{i,s}^{output}, x_{i,s}^{available-used} \in \{0,1\} \quad \forall i \in I, s \in 1, \dots, S$$

$$x_{f,s}^{output} \in \{0,1\} \quad \forall f \in F, s \in 1, \dots, S$$

$$y_{p,s} \in \{0,1\} \quad \forall p \in P, s \in 1, \dots, S$$

4.4 SOA-based cyberinfrastructure to support global manufacturing

A central repository for the component information available from all over the world is essential for global manufacturing. More importantly, the standardization of the representation of components is the key to it. The central repository can be built in a global level or a corporate level. This paper describes major features of the proposed SOA-based cyberinfrastructure for global manufacturing, but the implementation of it is beyond the scope of this paper.

The repository should be able to store and share the formal representation of components published by suppliers. In order to standardize and automate such sharing processes, the cyberinfrastructure should provide SOA-based subscribe/publish/ query mechanisms. Figure 7 depicts the proposed SOA-based cyberinfrastructure for modular product design. The proposed cyberinfrastructure should provide three types of Web services for suppliers and OEMs, managerial services, atomic query services and composite query services. Managerial services consist of Web services for subscribe/ unsubscribe/ publish/ update/ delete operations. Atomic query services are single task Web services, such as searching a type of components (e.g., getCPU). Finally, composite query services provide value-added services that an atomic query service cannot perform, such as designing a product or checking compatibility among components. Suppliers can subscribe to the cyberinfrastructure, and publish/update their components information by invoking Web services.

Table 3 summarizes the differences between the proposed cyberinfrastructure for modular product design and the existing digital Design Repository developed by NIST and MS&T. The differences are categorized into two aspects, namely, contents and architecture. In terms of contents, the proposed cyberinfrastructure contains information regarding modularized components, while the digital design repository does non-modularized components. In addition, the module

representation of the proposed approach is interface-oriented, while that of the digital design repository is flow-oriented. The provided services by the two are also different. The proposed cyberinfrastructure can provide users with three types of services, managerial, simple and composite ones, such as services for generating multiple modular product designs or checking compatibility, while the digital design repository provides them with only simple query services. With respect to architecture, since the proposed cyberinfrastructure is based on SOA, it is remotely accessible by invoking Web services, while the digital design repository is not. In addition, the services of the proposed cyberinfrastructure are discoverable through the standard discovery mechanism for Web services, while the design repository is not. The case study described next elucidates these similarities and differences further.

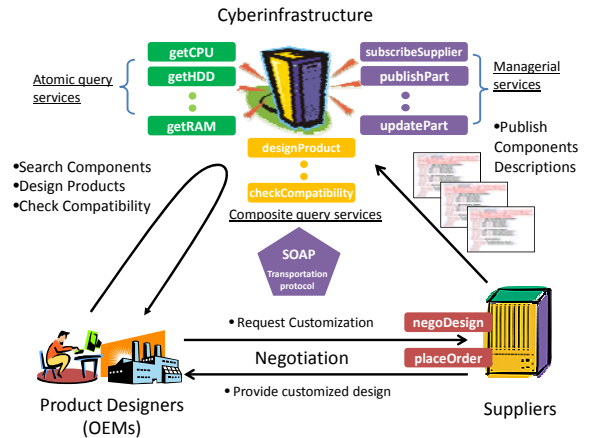


Figure 7. Overview of the proposed SOA-based Cyberinfrastructure for Modular Product Design

Table 3. Comparison of Proposed Design Cyberinfrastructure and Existing Design Repository

	Proposed Design Cyberinfrastructure	Existing Design Repository
Component type	Modularized components	Non-modularized components
Module representation	Interface-oriented	Flow-oriented
Services	Composite query and simple query	Simple query
Remote accessibility	SOA based remote invocation	Web based access
Discoverability	Discoverable thanks to standard description and search mechanism of SOA	N/A

5 CASE STUDY

This section presents a case study to demonstrate how the proposed IP-based AI Planning formulation works. To facilitate understanding of the case study, we use a familiar product, namely, a desktop PC that consists of a motherboard, a hard disk drive (HDD), a random access memory (RAM), a central processing unit (CPU), and a power supply. Table 4

summarizes the components used in the case study. There are five different components, and each component has two alternatives. We assume that there exists a power source with the voltage of 110V, which corresponds to the initial constraint. We set the goal constraints as obtaining all five features. The objective is to minimize the cost of the final product assuming that functional requirements (compatibility among components) are satisfied.

Table 4. Parts Used in the Case Study

Components	Features	Alternatives	Price(\$)
CPU	Processing	Socket 370(*)	250
		Socket 462	300
HDD	Storage	IDE(*)	80
		SCSI	120
RAM	Memory	DIMM(*)	100
		SIMM	80
Power	Energy	110V/8V(*)	50
		220V/16V	40
Mother B/D	Control	A:8V/IDE/Socket370/DIMM(*)	110
		B:8V/IDE/SCSI/Socket370/DIMM	100
		C:16V/SCSI/Socket462/SIMM	90

We formulated this case study using the proposed IP formulation introduced in Section 4.3 as follows. Due to the page constraints, we highlight only the meaningful aspects of the formulation for this particular problem.

(1) Initial constraints

Since only 110V power supply is assumed to be given, the following constraints are the whole initial constraints:

$$x_{110V,0}^{output} = 1, x_{110V,0}^{input} = x_{110V,0}^{available-unused} = 0$$

(2) Goal constraints

Goal constraints associated with the features of Control and Memory are presented:

$$x_{Control,s}^{input} + x_{Control,s}^{output} + x_{Control,s}^{available-unused} \geq 1$$

$$x_{Memory,s}^{input} + x_{Memory,s}^{output} + x_{Memory,s}^{available-unused} \geq 1$$

(3) Compatibility constraints

Compatibility constraints associated with motherboards are presented:

$$y_{MotherBD-A,s} + y_{MotherBD-B,s} + y_{MotherBD-C,s} \geq x_{CONTROL,s}^{output}$$

$$y_{MotherBD-A,s} \leq x_{CONTROL,s}^{output}$$

$$y_{MotherBD-A,s} \geq x_{8V,s}^{input} \quad y_{MotherBD-A,s} \geq x_{Socket370,s}^{input}$$

$$y_{MotherBD-A,s} \geq x_{IDE,s}^{input} \quad y_{MotherBD-A,s} \geq x_{DIMM,s}^{input}$$

$$y_{MotherBD-A,s} \leq x_{8V,s}^{input} \quad y_{MotherBD-A,s} \leq x_{Socket370,s}^{input}$$

$$y_{MotherBD-A,s} \leq x_{IDE,s}^{input} \quad y_{MotherBD-A,s} \leq x_{DIMM,s}^{input}$$

(4) Non-concurrency constraints

Non-concurrency constraints associated with DIMM-type RAMs are presented:

$$x_{DIMM,s}^{output} + x_{DIMM,s}^{available-unused} \leq 1$$

$$x_{DIMM,s}^{input} + x_{DIMM,s}^{available-unused} \leq 1$$

(5) Sequence constraints

Sequence constraints associated with IDE-type HDDs are presented:

$$x_{IDE,s}^{input} + x_{IDE,s}^{available-unused} \leq x_{IDE,s-1}^{input} + x_{IDE,s-1}^{output} + x_{IDE,s-1}^{available-unused}$$

(6) The optimal solution

$$y_{CPU,1,1} = 1, y_{HDD,1,1} = 1, y_{RAM,1,1} = 1, y_{Power,1,1} = 1, \text{ and } y_{MotherBD-A,2} = 1.$$

Components marked with a * in Table 4 correspond to the optimal design that minimizes cost. This solution was obtained using ILOG CPLEX IP solver. Figure 8 illustrates the optimal design, which shows how all of the functional requirements are satisfied and that all of the required features are provided. For comparison, we enumerated all 48 (=2x2x2x2x3) possible combinations and checked them against the constraints. This yielded the 4 feasible solutions listed in Table 5. Solution 1 is the best choice since it provides the feasible design with the lowest cost. This is the same solution that the IP formulation identified without enumerating all 48 combinations.

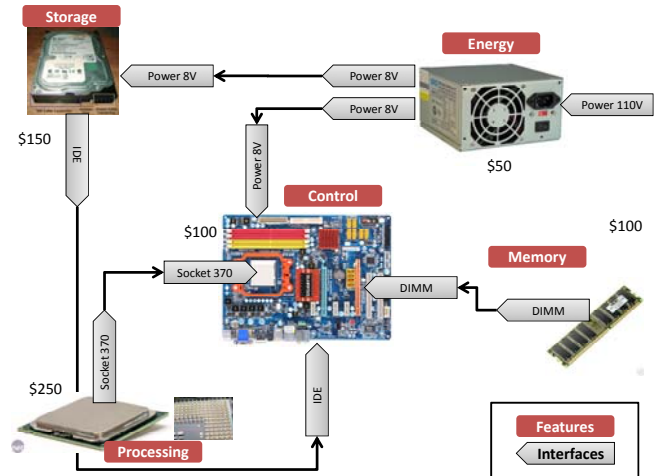


Figure 8. Optimal Solution from IP-based Formulation

Table 5. Feasible Design Alternatives

Type	Solution 1	Solution 2	Solution 3	Solution 4
CPU	Socket 370	Socket 370	Socket 370	Socket 462
HDD	IDE	IDE	SCSI	SCSI
RAM	DIMM	DIMM	DIMM	SCSI
Power	110V/8V	110V/8V	110V/8V	220V/16V
Mother B/D	Type A	Type B	Type B	Type C
Price	\$540	\$580	\$620	\$630

This case study shows the AI Planning based approach can deal not only with branching and merging of components, but also with multiple inputs and outputs. Additionally, the IP-based formulation provides a way to incorporate non-

functional attributes into the formulation. Therefore, the IP-based AI Planning formulation considers non-functional requirements as well as functional ones.

6 CONCLUDING REMARKS

In this paper, we introduce a computational modular product design framework using cyberinfrastructure and show the viability of the proposed framework through the case study. The machine-readable component and module specifications stored in cyberinfrastructure will facilitate global manufacturing because product designers from all over the world can more easily identify suitable components produced by suppliers from around the world by examining the specifications stored in the cyberinfrastructure. Additionally, the paper shows how to transform the modular product design problem into an AI Planning problem and proposes an IP formulation to solve the modular product design problem. Compared to the existing computational product concept generation tool proposed by Bryant, et al. [15], the AI Planning based approach can handle branching and merging of components as well as multiple inputs and outputs. Furthermore, the capability of incorporating non-functional constraints, such as weight or price, into the IP formulation facilitates to deal with the current mass customization trend.

In future research, 3-D geometric constraints can be considered in the IP formulation, which will generate a product that also meets size constraints. Additionally, the current IP formulation generates an optimal solution according to the pre-set objective function. However, generating alternative solutions will broaden the range of options from which decision makers can choose. Therefore, methods for generating multiple good solutions, such as K-best solutions, should also be investigated.

7 REFERENCES

- [1] Eppinger, S. D. and Chitkara, A. R., 2006, "The New Practice of Global Product Development," *MIT Sloan Management Review*, 47(4), 22-30.
- [2] Hoffman, B. G., 2006, "Ford Retools Auto Design," *The Detroit News*, Dearborn, MI, June 21, 2006.
- [3] Kotha, S., Olesen, D. G., Nolan, R. and Condit, P. M., 2005, "Boeing 787: Dreamliner," *Harvard Business School Case Study*, 9-305-101, Boston, MA.
- [4] Atkins, D. E., Droegemeier, K. K., Feldman, S. I., Garcia-Molina, H., Klein, M. L., Messerschmitt, D. G., Messina, P., Ostriker, J. P. and Wright, M. H., 2003, *Revolutionizing Science and Engineering Through Cyberinfrastructure*, National Science Foundation, Arlington, VA.
- [5] Erl, T., 2004, "Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services", Prentice Hall, Upper Saddle River, NJ.
- [6] Bachlechner, D. K. Siorpaes, D. Fensel, and I. Toma, 2006, "Web Service Discovery – A Reality Check", Technical Report DERI-TR-2006-01-17, Digital Enterprise Research Institute, available online at: <http://www.deri.ie/fileadmin/documents/DERI-TR-2006-01-17.pdf>
- [7] Szykman, S., Racz, J., and Sriram, R., 1999, "The Representation of Function in Computer-Based Design", *ASME Design Engineering Technical Conferences – Design Theory & Methodology Conference*, Las Vegas, NV, ASME, Paper No. DETC99/DTM-8742.
- [8] Szykman, S., Senfaute, J., and Sriram, R., 1999, "The Use of XML for Describing Functions and Taxonomies in Computer-Based Design", *ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Las Vegas, NV, ASME, Paper No. DETC99/CIE-9025.
- [9] Szykman, S., 2002, "Architecture and Implementation of a Design Repository System", *ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Montreal, Canada, Paper No. DETC2002/CIE-34463
- [10] Szykman, S. and Sriram, R., 2006, "Design and Implementation of the Web-enabled NIST Design Repository", *ACM Transactions on Internet Technology*, 6(1), pp. 85-116.
- [11] Devanathan, S., and Ramani, K., 2007, "Combining Constraint Satisfaction and Non-Linear Optimization to Enable Configuration Driven Design", *International Conference on Engineering Design*, Paris, France (ICED'07).
- [12] Bohm, M. R., Stone, R. B., Simpson, T. W., and Streva, E. D., 2008, "Introduction of a Data Schema to Support a Design Repository", *Computer-Aided Design*, 40(7), pp. 801-811.
- [13] Stone, R. and Wood, K., 2000, "Development of a Functional Basis for Design", *ASME Journal of Mechanical Design*, 122(4), pp. 359-370.
- [14] Hirtz, J., Stone, R., McAdams, D., Szykman, S., and Wood, K., 2002, "A Functional Basis for Engineering Design: Reconciling and Evolving Previous Efforts", *Research in Engineering Design*, 13(2), pp. 65-82.
- [15] Bryant, C.R., McAdams, D. A., Stone, R. B., Kurtoglu, T., and Campbell, M. I., 2005, "A Computational Technique for Concept Generation", *ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Long Beach, CA, ASME, DETC2005-85323*.
- [16] Bryant, C.R., McAdams, D. A., Stone, R. B., Kurtoglu, T., and Campbell, M. I., 2006, "A Validation Study of an Automated Concept Generator Design Tool", *ASME Design Engineering Technical Conferences, Philadelphia, PA, ASME, Paper No. DETC2006-99489*.
- [17] Campbell, M. I., Cagan, J., and Kotovsky, K., 2003, "The A-Design Approach to Managing Automated Design Synthesis", *Research in Engineering Design*, 14, pp. 12-24.
- [18] Campbell, M. I., Cagan, J., and Kotovsky, K., 2000, "Agent-Based Synthesis of Electromechanical Design Configurations", *ASME Journal of Mechanical Design*, 122, pp. 61-69.
- [19] Mittal, S., Dym, C., and Morjara, M., 1985, "PRIDE: An Expert System for the Design of Paper Handling Systems", *IEEE Computer*, 19(7), pp. 102-114.

- [20] Navinchandra, D., Sycara, K. P., and Narasimhan, S., 1991, "A transformational Approach to Case-Based Synthesis", *AIEDAM*, 5, pp. 31-45.
- [21] Titus, N., and Ramani, K., 2005, "Design Space Exploration Using Constraint Satisfaction," *International Joint Conferences on Artificial Intelligence (IJCAI)*, Configuration Workshop, pp. 31-37.
- [22] Bohm, M. and Stone, R., "Product Design Support: Exploring a Design Repository System", 2004, *Proceedings of IMECE*, IMECE2004-61746, Anaheim, CA.
- [23] Blake, M. B., Cheung, W., and Wombacher, A., 2007, "Web Services Discovery and Composition Systems", *International Journal of Web Services Research*, 4(1), pp. 3-8.
- [24] Oh, S.-C., 2006, "Effective Web-Service Composition in Diverse and Large-Scale Service Networks", Ph.D. Dissertation, Industrial & Manufacturing Engineering, The Pennsylvania State University, University Park, PA.
- [25] Oh, S.-C., Lee, D., and Kumara, S., 2008, "Effective Web Service Composition in Diverse and Large-Scale Service Networks", *IEEE Transactions on Services Computing*, 1(1), pp. 15-32.
- [26] Yoo, J., Oh, S.-C., and Kumara, S., 2008, "Application of Semantic Web Technology to Scheduling Interoperability", *Proceedings of Industrial Engineering Research Conference*, Vancouver, Canada.
- [27] Van den Briel, M. and Kambhampati, S., "Optiplan: Unifying IP-based and Graph-based Planning", *J. of Artificial Intelligence Research*, 24, pp. 919-931, 2005.
- [28] Nilsson, N. J., 1998, "Artificial Intelligence: A New Synthesis", Morgan Kaufmann Publishers, San Francisco, CA.