The Pennsylvania State University

The Graduate School

The Department of Industrial and Manufacturing Engineering

A MATHEMATICAL FRAMEWORK FOR SEMANTIC WEB SERVICE COMPOSITION WITH APPLICATION TO MODULAR PRODUCT DESIGN

A Dissertation in

Industrial Engineering

by

Jung-Woon Yoo

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2010

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

The dissertation of Jung-Woon Yoo was reviewed and approved* by the following:

Soundar R.T. Kumara
Allen E. Pearce/Allen M. Pearce Chaired Professor of Industrial Engineering
Dissertation Advisor
Chair of Committee

Dongwon Lee
Associate Professor of Information Sciences and Technology

Jose A. Ventura
Professor of Industrial Engineering

Timothy W. Simpson
Professor of Industrial and Mechanical Engineering

Paul Griffin
Department Head and Professor of Industrial Engineering

*Signatures are on file in the Graduate School

ABSTRACT

Web services applications prevail in mobile devices such as the iPhone and Blackberry as well as among numerous networked computers. Web services are a state-of-the-art, leading technology based on Service Oriented Architecture in the World Wide Web environment. Each Web service is described in a standard language, such as WSDL (Web Service Definition Language), and is published in a global registry called UDDI (Universal Description, Discovery, and Integration). Individual users or software agents invoke Web services based on the Web service descriptions registered with UDDI. A simple information or service request can be fulfilled by a single Web service, but complicated requests cannot be satisfied by a single Web service. Consequently, a composition of multiple Web services in an appropriate sequence is required.

As the number of Web services increases in dynamic business environments, the automation of Web service composition becomes an essential feature for commercial Web services. Automatic Web service composition software should be able to consider not only functional requirements, but also quality of service (QoS) aspects. Functional requirements force the composition software to generate feasible solutions, while QoS aspects make the composition software satisfy user objectives, such as cost, time, or reliability.

Web service composition problems can be classified in two ways: (1) syntactically and (2) semantically. Semantic issues have recently presented challenges to Web service composition. Due to insufficient understanding of semantics, Web service composition solutions can be inferior, or even impossible to generate.

In this research, a mathematical solution framework that guarantees the optimal solution to semantic Web service composition is introduced. The Integer Programming (IP) based mathematical framework considers not only functional requirements but also QoS aspects of Web

service composition. Furthermore, the framework can incorporate semantics-processing mechanisms into its IP formulation. The proposed approach guarantees the optimality of the solutions from both syntactic and semantic perspectives.

Finally, a k-best solution method for Web service composition is presented. Using k-best solutions provides a holistic view of the Web service composition solution space rather than a myopic view that is focused only on the optimal solution. Knowing k-best solutions and the summary statistics among them (such as the range of objective values) provides a broader view when composing Web services.

Key words: Web service composition, Semantic Web, Integer Programming, quality of service, k-best solution approach.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

The objective of this dissertation is to develop a mathematical framework for semantic Web service composition. This chapter introduces the problem domain, motivation and contributions of this research, followed by the outline of this dissertation.

1.1 Introduction to Web Services

Web services is a state-of-the-art, leading method for providing a variety of real-time technologies in the World Wide Web environment [1]. Recently, Web-based service providers have created a central registry, or "yellow pages directory" of their technologies using UDDI (Universal Description, Discovery and Integration). Individual users or intelligent software agents who subscribe to the service providers can use the registered Web services for their own purposes. Each Web service is described using standard languages, such as WSDL (Web Service Definition Language), which describe the input and output of the Web service, and other information such as transport protocol and message format. SOAP (Simple Object Access Protocol) is the communication protocol between Web service providers and users over the Internet.

A Web service is a software component that is not dependent upon a platform or implementation methodology and can be [2]:

1) *described* using a service description language;
2) *published* to a registry (UDDI) of services;
3) *discovered* through a standard mechanism at runtime or design time;

4) *invoked* through a declared API, usually over a network; and

5) *composed* with other services.

Web services are based on Service Oriented Architecture (SOA) [1], the state-of-the-art information system architecture shown in Figure 1-1. The following describes the major components of Service Oriented Architecture, which are essential for the implementation of Web services.

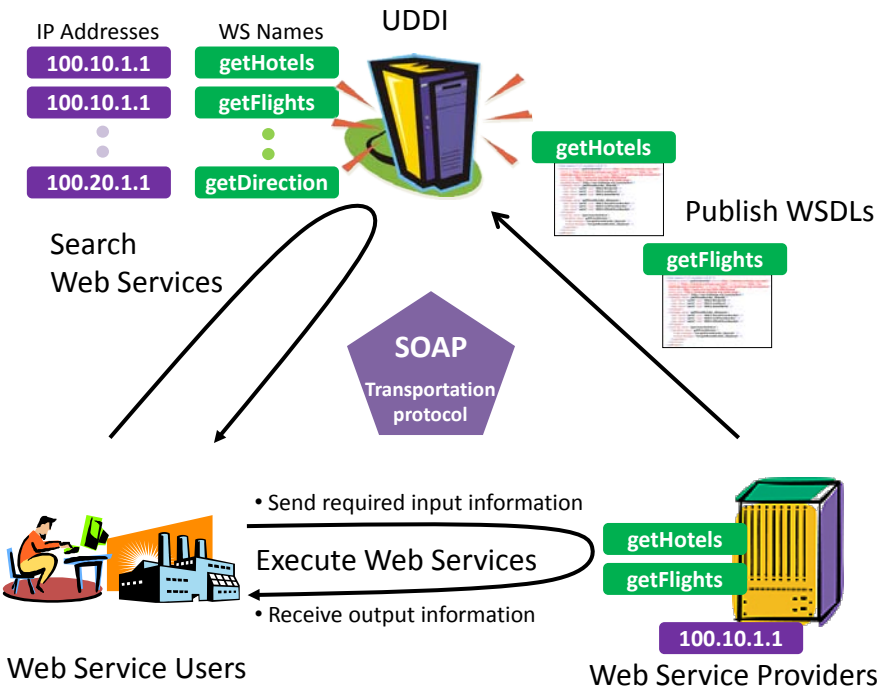


Figure 1-1. Service Oriented Architecture

1.1.1 SOAP (Simple Object Access Protocol)

SOAP [3] was originally intended to provide networked computers with remote-procedure call (RPC) services [4] written in XML (eXtensible Markup Language) [1]. As its name implies, SOAP is a lightweight communication protocol that is widely used for e-business transactions. A SOAP message consists of a header and a body. The header includes control

information for the SOAP message, such as authentication and encoding, while the body includes the actual content to be transferred from a sender to a receiver. One well-known SOAP message pattern is SOAP RPC. It contains: (1) the destination address, (2) the name of the method to be invoked, and (3) the input and output value.

1.1.2 WSDL (Web Service Description Language)

Web services use a de facto standard description language: Web Service Description Language (WSDL), shown in Figure 1-2. WSDL is an XML [5] language for describing a programmatic interface to a Web service [6]. The basic contents of a WSDL file for a Web service are composed of input and output message formats, data types, network addresses, port type, and binding information. Port type describes the operations provided by the service, and binding describes communication protocols required to use the Web service. WSDL specifies the name of the Web service (such as GetDirection or GetPhoneNumber), the data types of input/output parameters (such as string or integer), the operations provided by the Web service (such as GetRoute or GetTime), the communication protocol (such as SOAP), and the network address in the form of a URL (Uniform Resource Locator).

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions name="getContactInfo" xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:w="http://schemas.xmlsoap.org/wsdl/" xmlns:WSCC="http://ws-
  challenge.org/composition" xmlns:tns="http://ws-challenge.org/composition"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  targetNamespace="http://ws-challenge.org/composition">
 - <message name="getPhoneNumber_Request">
    <part name="part0" type="WSCC:StudentID" />
    <part name="part1" type="WSCC:LastName" />
    <part name="part2" type="WSCC:DateOfBirth" />
  </message>                                                    } Input Parameters
 - <message name="getPhoneNumber_Response">
    <part name="part0" type="WSCC:HomePhoneNumber" />
    <part name="part1" type="WSCC:CellPhoneNumber" />
    <part name="part2" type="WSCC:OfficePhoneNumber" />
  </message>                                                    } Output Parameters
 - <portType name="getContactInfoPort">
  - <operation name="getPhoneNumber">
     <input message="tns:getPhoneNumber_Request" />
     <output message="tns:getPhoneNumber_Response" />
    </operation>                                                } Operations
  </portType>
</definitions>
```

Figure 1-2. An example of Web Service Description Language

A Web service can be invoked through a stub code[1] that can be generated from the corresponding WSDL file of each Web service. Such code generation is possible because of the standardized format provided by WSDL. Furthermore, the standard mechanism enables software agents to compose multiple Web services to answer complex queries that cannot be answered by a single Web service.

1.1.3 UDDI (Universal Description, Discovery, and Integration)

The Universal Description, Discovery, and Integration (UDDI) specification [7] describes a mechanism for registering and locating Web services [1]. UDDI is a platform-independent, cross-industry framework designed to create a registry standard for Web service description and

[1] A stub code, or skeleton code, is a piece of code generated from a Web Service Description Language (WSDL) file or an Interface Definition Language (IDL) file. Most programming languages have SOAP utility programs that are designed to generate stub code from a WSDL or IDL file.

discovery, together with a registry facility that supports the publishing and discovery processes [8]. Web services described in WSDL are published in a UDDI registry so that software agents or individual users can discover them.

UDDI has three main components: white pages, yellow pages, and green pages [7]. UDDI white pages include business names, descriptions, contact information, and identifiers for Web service providers; UDDI yellow pages include industries, products and services, and geographical location information for Web service providers; UDDI green pages describe how other businesses can conduct electronic commerce with the registered providers [1].

A Web services model is illustrated in Figure 1-1. First, providers describe their Web services using WSDL and publish them on UDDI. Then users are able to find appropriate Web services using UDDI. Since UDDI is used to store all WSDL files published by Web service providers, users can find and download them. Once Web service users download the WSDL files, they can generate stub codes and invoke the corresponding Web services. There is direct communication between Web service providers and users.

1.2 Research Motivation

Simple requests, such as finding directions from City A to B, can be answered by a single, atomic Web service, whereas complicated queries cannot be answered by a Web service. For example, suppose that there is a car accident and a person is severely injured. A 911 control center needs to find a way to deliver a requested amount of a specific blood type from a blood bank to the closest transfusion-capable hospital. To answer such a complicated query, multiple Web services would need to be invoked in an appropriate sequence. The sequence would need to include branches and merges in the Web service invocation, which means that some Web services would be invoked in parallel and others would be invoked in serial.

The task of arranging relevant Web services to answer a complex query is called Web service composition. Web service composition can be done manually [9]. However, as the number of open Web services increases, and as a variety of emerging, new service requests appear in the current competitive, complex, and changing business environment, automatic Web service composition becomes an essential feature of commercial Web services.

To automate the composition of Web services, functional and non-functional requirements must be satisfied. First, a functional requirement of the Web service composition problem is that input parameters of a Web service must be satisfied in order to invoke it. Most approaches from the previous literature have focused mostly on the functional aspects of automation [10, 11]. In particular, logic-based approaches (such as description logics, theorem proving, propositional satisfiability techniques, situational calculus, etc.) focus mostly on functional requirements. Non-functional requirements of the Web service composition problem such as invocation cost, response time, and provider reputation are often overlooked.

As Web services become more popular and better-utilized by individual users and intelligent software agents, they will be inevitably commercialized. This will lead to significant composition changes. Traditionally, matching parameters with the minimum number of Web services required in order to meet functional requirements has been the primary way of addressing the composition problem. However, considering both functional and non-functional attributes together when solving Web services composition problems would likely produce superior outputs.

Figure 1-3 illustrates the same blood delivery Web service composition problem discussed earlier, with additional consideration for non-functional attributes. Suppose that in a virtual medical industry UDDI, there would be five relevant Web services, each with a nominal fee: WS-A: getHospital ($10), WS-B: getBlood ($10), WS-C: askDelivery ($10), WS-D: getDirection ($10), and WS-E: getHospitalwithBloodBank ($50). If the main objective for the

composition is to minimize the number of Web services invoked, then the best solution would be to invoke only one Web service, WS-E at the cost of $50, which satisfies the parameter matching requirement. The invocation of a single Web service, WS-E, would be sufficient since it is provided by a hospital with a blood bank. On the other hand, if the main objective for the composition is to minimize cost, the best solution would be to invoke four Web services, WS-A and WS-B concurrently and then WS-D and WS-C sequentially, at the cost of $40. After securing an available hospital and a blood bank with requested blood type available, a transportation company would need to be hired to deliver the blood from the blood bank to the hospital.

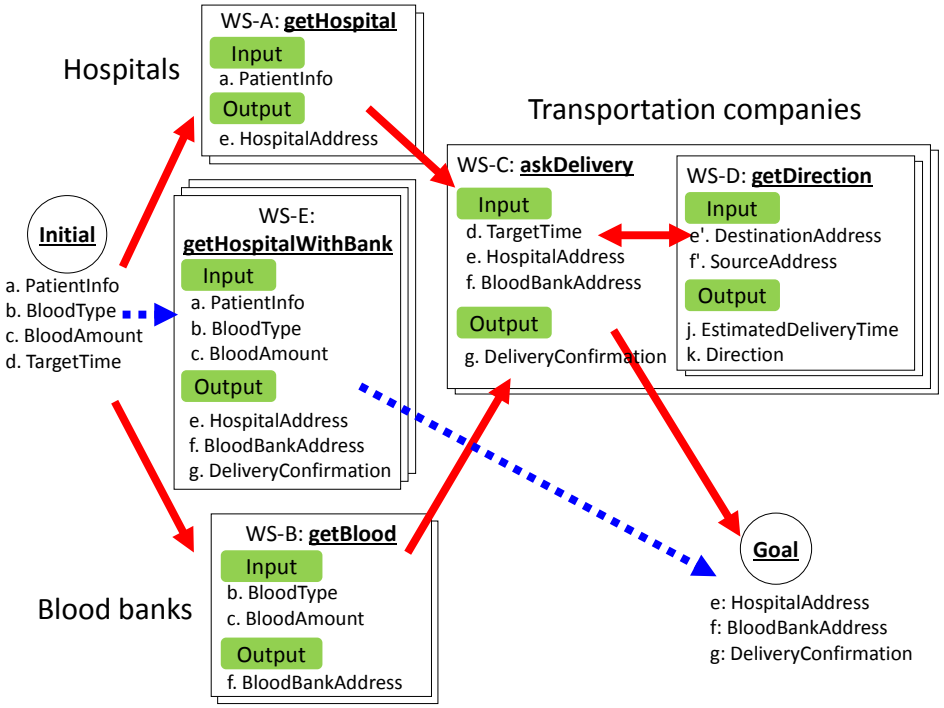


Figure 1-3. A motivating example: Urgent blood delivery

There could be other objectives for this example, such as minimal processing time or maximum reputation if the situation is very urgent or requires high credibility, respectively. This example clearly explains our claim that the best solution depends on the user's objectives, which

are non-functional attributes. Therefore, in this research, we propose a framework to address how to incorporate such non-functional attributes into a software agent for Web service composition.

1.3 Problem Statement

This Web service composition problem is formulated as an Artificial Intelligence (AI) planning problem. The AI planning formulation is denoted as $P = (\Sigma, t_0, g)$, where t_0 is an initial state, g is a goal state and Σ is a state transition system. The state transition system consists of three component; a set of states (T), a set of actions (A), and a state transition function ($\gamma : T \times A \rightarrow T$). The solution to the planning problem is a sequence of actions, $w^1, w^2, ..., w^k$, where k is the index of the sequence. The corresponding sequence of state transitions can be denoted as $t^1 = \gamma(t^0, w^1), t^2 = \gamma(t^1, w^2), ..., t^k = \gamma(t^{k-1}, w^k) = g$. Therefore, the Web service composition problem is stated as given $w_1, w_2, ..., w_n$ Web services, where n is the total number of Web services, and given the initial knowledge of t_0 and the goal knowledge of g, the goal of this problem is to generate an execution sequence of Web services that optimizes a given objective function.

1.4 Research Objectives and Contributions

The main objectives of this research are summarized as follows:

1) Develop a mathematical solution framework for obtaining the optimal solution to Web service composition;
2) Develop semantics processing mechanisms for Web service composition; and
3) Develop k-best solution methods for Web service composition.

These objectives, as well as the potential contributions of this research are discussed next.

1.4.1 Optimal solution framework

Much research has focused on heuristic approaches to Web service composition. This is partially due to long solution times stemming from the highly complex nature of the problem. As the query requirement increases, the solution generation becomes intractable. However, most Web service composition applications, like AroundMe iPhone Apps, which use the current location information of users, are being implemented during the design phase, rather than the run-time phase. We strongly believe that our optimal framework can contribute to the identification of potential composition solutions during the design process. Furthermore, our optimal approach is expected to provide an evaluation guideline for heuristic approaches.

1.4.2 Semantics processing

Semantic issues are challenging problems that have recently surfaced in Web service composition. Due to insufficient understanding of semantics, Web service composition solutions can be inferior, or even worse, the algorithms may not be able to find any solutions at all. The uniqueness of our approach to semantic issues is that all semantic relationships are incorporated into the Integer Linear Programming (ILP) formulation, a significant departure from other approaches in which relationships are pre-processed. Our approach guarantees the optimality of the composition solutions both syntactically and semantically.

1.4.3 k-best solution methods

Identifying k-best design alternatives provides a holistic view of the Web service composition solution space rather than a myopic view, which focuses only on the optimal solution. Suppose that the three best Web service (WS) composition solutions in terms of cost are: WS-A, WS-B and WS-C ($100); WS-A, WS-B and WS-D ($101); and WS-A, WS-B, and WS-E ($120). If a decision maker considers only the optimal solution, the first composition solution will be selected. However, if the provider of WS-D has a better reputation than WS-C, then the second solution can still be attractive, since the cost difference between the top two solutions is small. Although the reputation of Web service providers could be included as another decision making criterion and a different optimization problem could be formulated, it may be simpler to evaluate k-best solutions. Hence, knowing k-best composition solutions and the summary statistics among them (such as the range of objective values), can broaden a decision maker's view when composing Web services.

1.5 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 classifies Web service composition problems into two categories, syntactic and semantic, and defines each problem in detail. Background technologies and approaches (especially those related to Semantic Web services composition) are introduced to explain how the proposed methodology works.

Chapter 3 reviews background literature related to Web service composition and identifies the research gaps that this thesis addresses. Previous works are classified into logic-based approaches and mathematical programming approaches in Section 3.1. Looking at the literature in a different way, Section 3.2 classifies related papers on four dimensions: optimal

versus heuristic solutions, semantic versus syntactic composition, functional requirements versus quality of service concerns, and parameter- versus operation-level composition.

Chapter 4 proposes a mathematical framework for semantic and syntactic Web service composition. Semantic and syntactic Web service composition problems are formulated using Integer Programming. The mathematical formulation makes it possible to obtain the optimal solution for Web service composition. Semantic relationships among input and output parameters of Web services are also formulated in the proposed framework. Experimental results show the performance of our approach, and the system architecture for Web service composition is presented.

Chapter 5 introduces an application of Web service composition to modular product design, creating an analogy between them. Differences between them are also analyzed and reflected in the formulation.

Finally, Chapter 6 summarizes the contributions of this research and offers ideas for future research topics.

Chapter 2

Problem Definition

The Web service composition problem can be transformed into an AI planning formulation [12, 13, 14]. An AI planning problem, P, can be represented as $P = (\Sigma, t_0, g)$ and $\Sigma = (T, A, \gamma)$, where T is the set of states, A is the set of actions, $\gamma : T \times A \to T$ is a state transition function, t_0 is the initial state, and g is the goal state. A solution to P is a sequence of actions. In Web service composition, known parameters correspond to the initial state, goal parameters correspond to the goal state, the set of Web services corresponds to the set of actions, the set of known parameters corresponds to the set of states, an invokable Web service with the known parameters corresponds to input, and the parameters resulting from the Web service invocation (output) correspond to the state transition function. Depending on the consideration of semantic aspects, we can classify the problem into syntactic Web service composition and semantic Web service composition. In this chapter, we define these two types of problems in detail.

2.1 Syntactic Web Service Composition

The Web service composition problem and the Web service discovery problem (a special type of Web service composition problem) can be defined as follows.

Definition 1. Web Service Composition Problem: Given $P_{Initial}$ and P_{Goal}, where $P_{Initial}$ is the set of parameters that are known at the outset, and P_{Goal} is the set of parameters that are to be obtained, the purpose of Web service composition is to find a sequence of the set of Web services ($W_1, W_2, ..., W_S$) that can be simultaneously invoked at each stage, where S denotes the index of maximal stage.

Definition 2. Web Service Discovery Problem: The Web service discovery problem is a special case of the Web service composition problem where there is a solution at the first stage, and W_1 (the set of Web services in the first stage) consists of a single Web service. In other words, a single Web service with $P_{Initial}$ satisfies P_{Goal}.

In Figure 1-3, the solid line solution that consists of WS-A, WS-B, WS-C, and WS-D is an example of the Web service composition problem, while the dotted line solution that consists of WS-E is an example of the Web service discovery problem.

Figure 2-1 illustrates invokable Web services. Given the initial information of A and B, Web service 1 can be invoked at the first stage since it requires the information of A and B, and the information of C, D, and E is obtained. At the second stage, only Web service 2 is invokable since it requires the information of A, C, and E that are already known. However, since Web service 3 requires the information of H, but H is not known at this stage, the Web service is not invokable. Note that the size of the knowledge base is non-decreasing. This characteristic is used in the problem formulation.

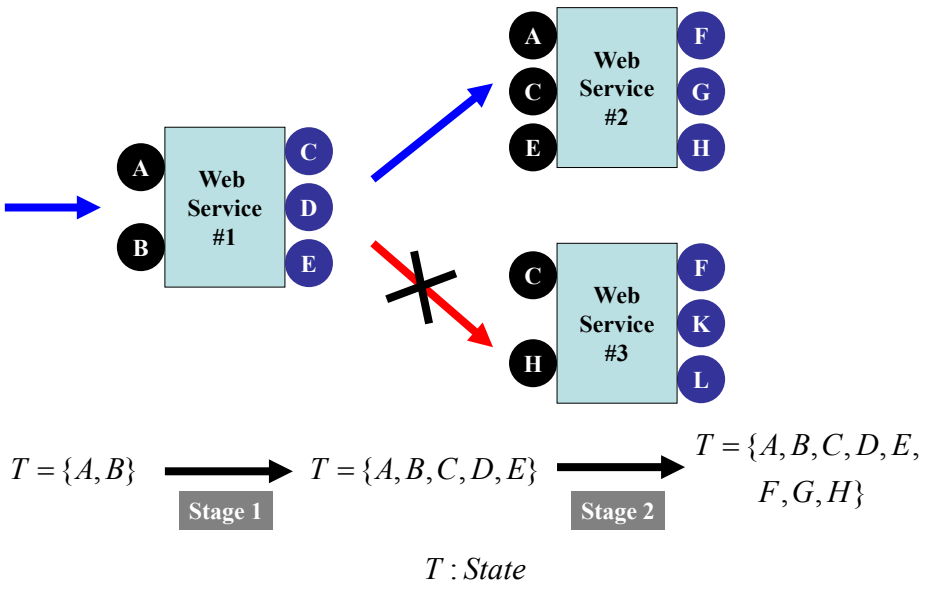


Figure 2-1. An example of invokable Web services

2.2 Semantic Web Service Composition

Thus far, we have discussed Web service composition from a syntactic perspective. In this section, we discuss Web service composition from a semantic perspective.

Figure 2-2 illustrates a semantic issue in Web service composition. "U.S. Address" and "BloodAmount" are initial knowledge and "LocalBloodBankAddress" is goal knowledge, and there are two Web services that perform the same function but are offered by different providers at different costs. Intuitively, we see that "getBlood #1" can be invoked with the initial knowledge. However, we assume that the "U.S. Address" data is inherited from "Address" data based on the principles of object-oriented programming [15]. If Web service composition software understands the inheritance relationship between "U.S. Address" and "Address" data, and "PartAmount" is equivalent to "PartVolume," then the composition software will select the

cheaper solution of "getBlood #2" instead of "getBlood #1." Such semantic relationships can be described using a web ontology language, such as OWL [27]. This example clearly illustrates a case in which considering semantics facilitates the discovery of a better solution, proving the necessity of semantic consideration in Web service composition. Figure 2-3 illustrates the semantic relationships of hierarchy and equivalence.

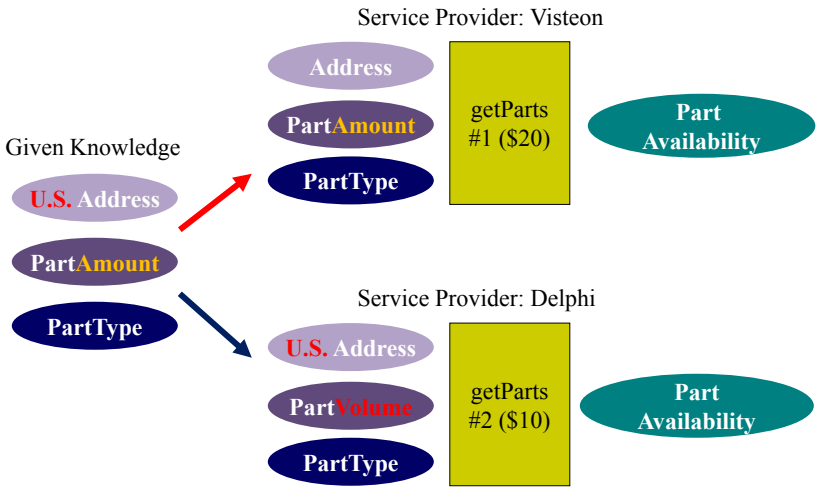


Figure 2-2. An Example of Semantic Web Service Composition

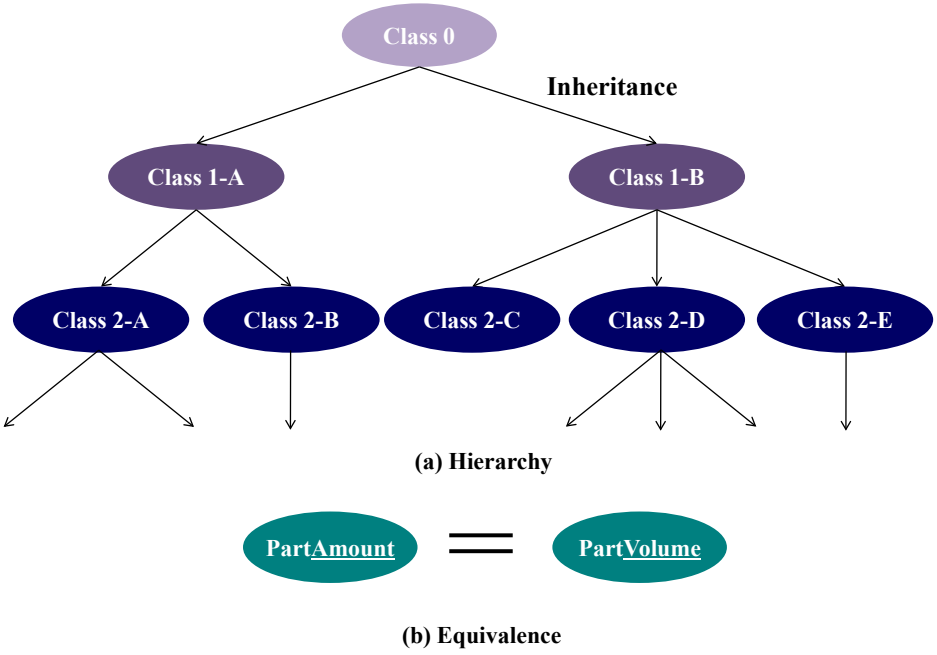


Figure 2-3: Semantic Relationships in Web Service Parameters

To resolve semantic issues occurring during the Web service composition process, we utilize the Semantic Web. The Semantic Web was proposed in 2001 by Berners-Lee et al. [16], the first author credited with inventing the World Wide Web (WWW). The Semantic Web represents the actual Web contents represented by the character strings in the current World Wide Web (which are often meaningless to humans). The main idea behind the Semantic Web is to create another dimension of the current Web that enables machines to comprehend the semantics of Web content. Internet users can browse not only the World Wide Web but also the Semantic Web, as illustrated in Figure 2-4.

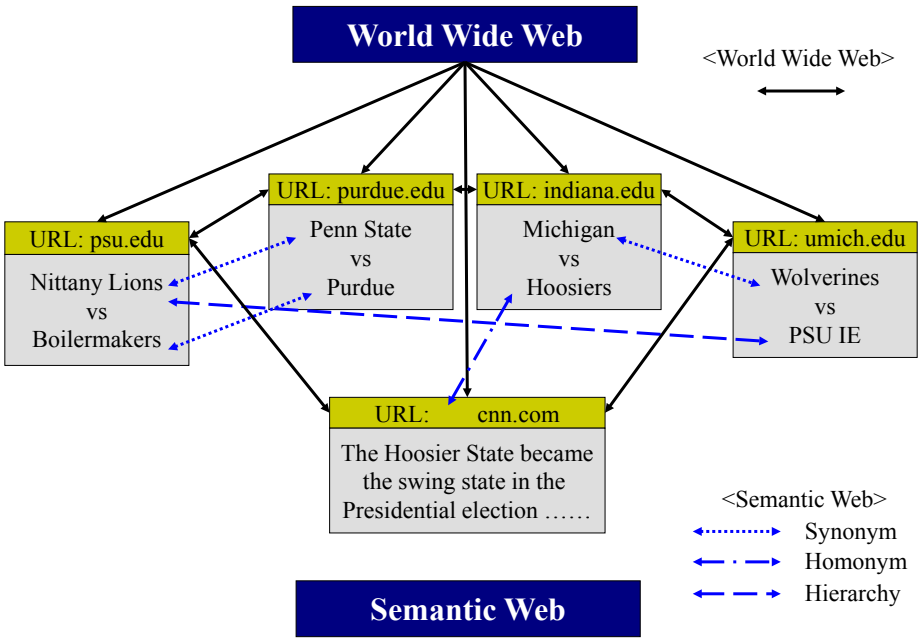


Figure 2-4. The World Wide Web and the Semantic Web

The realm of the Semantic Web is not limited to cyberspace. Berners-Lee et al. [16] predict that the Semantic Web will break out of the virtual realm and extend into the physical world. There is a group of enabling components called Semantic Web technology which may potentially be utilized to solve various problems in the industrial domain.

The core of Semantic Web technology consists of three components: XML, RDF/RDFS and ontologies. In the context of ontologies, the Web Ontology Language (OWL) is introduced, as it is the recommended standard for ontology representation. After a brief description of the components, representative applications of Semantic Web technology is presented.

2.2.1 XML (Extensible Markup Language)

XML (Extensible Markup Language) [17], a W3C recommendation, is designed to describe data and data structures by using self-defined tags. It differs from HTML (Hypertext Markup Language) [18], which is designed to display data by using pre-defined tags (e.g., <h1>, <p>, <b>). XML is self-descriptive because a Document Type Definition (DTD) [19] or an XML Schema [5] describes data and the structure of the data in detail, which provides maximal freedom to define new data types. XML has been successfully utilized in the WWW environment for describing and transferring data. However, in the Semantic Web, where the meaning of data or the meaning of the structure of the data is essential, additional information is required.

2.2.2 RDFS (Resource Description Framework Schema)

RDF (Resource Description Framework) [20] is a language originally designed to represent information pertaining to "Web resources." However, by generalizing the scope of what constitutes a Web resource, any entity can be accessed through the Web, including physical objects such as devices or products. RDF can also be used to represent information about any object [20] and is utilized as a way of expressing the underlying meanings of objects in Semantic Web technology. It encodes the meaning of an object by using a triple, which consists of a subject, a predicate, and an object. URIs (Uniform Resource Identifiers) [21] are used to identify differences among object meanings. The URIs ensure that object meanings are identified by unique RDF descriptions on the Web. An illustrative example is provided in Figure 2-3, which is an RDF representation of the triple set for an RFID tag [22]. The example represents data in an RFID tag where the name is "L233", the RFID type is "active", and the Electronic Product Code

(EPC) [23, 24] is "01000111." Two name spaces, lisqont and epccode, are utilized to describe the tag, which refer to XML schemas defined elsewhere.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:lisqont="http://www.ie.psu.edu/lisq/ont/"
         xmlns:epccode="http://www.epc.code/EPC/">
   <rdf:Description rdf:about="http://www.ie.psu.edu/lisq/ont/RFID/L233">
      <lisqont:RFID-type>Active</ lisqont:RFID-type>
      <epccode:RFID-maker>Alien Technology</ epccode:RFID-maker>
          <lisqont:RFID-code rdf :resource= "http://www.epc.code/EPC/Code/01000111"/>
</rdf:Description>
</rdf:RDF>
```

Figure 2-2. An RDF representation of triples

URIs play an effective role in differentiating homonyms and synchronizing synonyms in RDF. Different URIs are utilized to identify different meanings among homonymous terms. For example, assuming that two different companies use the same term "PC" to describe a generic personal computer and a Windows machine, respectively, having a unique URI for each meaning makes it possible to differentiate the usages of the term between the two companies. Resource Description Framework Schema (RDFS) is a semantic extension of RDF, which provides mechanisms for describing groups of resources and properties along with application-specific relationships among them [25]. RDFS is not used to instantiate resources and properties.

2.2.3 Ontologies: OWL (Web Ontology Language)

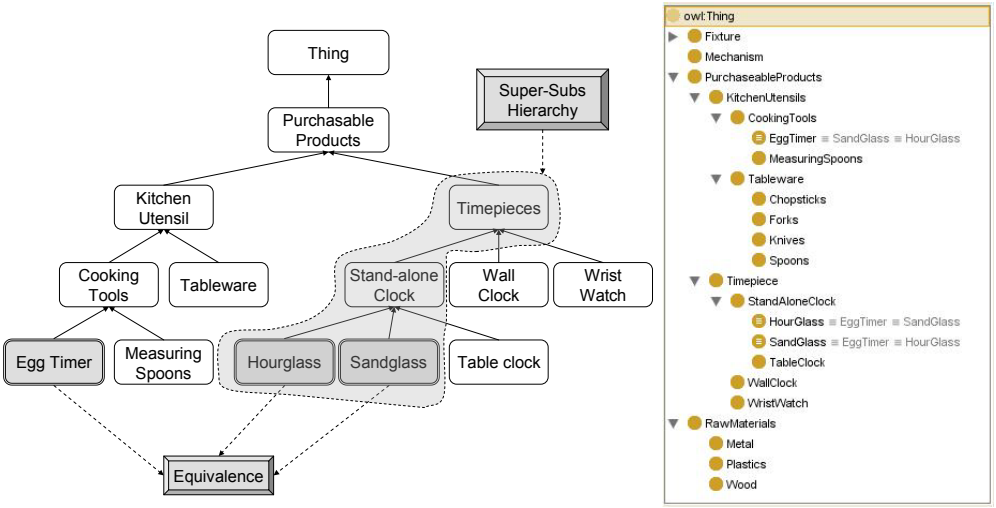
Ontology is a philosophical term indicating a branch of metaphysics that deals with the nature of existence. However, the Artificial Intelligence and Web community have given the term new meaning, referring to "ontologies" as "defined objects and relationships among them." Tim Berners-Lee defined an ontology as a document or file that formally defines the relationships

among terms [16]. In order for machines to understand a Web document, the document should define terms and relationships among terms.

There are several ontology languages in use; however, in this research, we focus on Web Ontology Language (OWL), as it is the World Wide Web Consortium's (W3C) [26] recommendation for Web ontology [27]. OWL builds on the foundation of RDF/RDFS by providing additional descriptors (constructs) and making it possible for machines to interpret the meaning of Web content. OWL has three types of sublanguages, OWL Lite, OWL DL, and OWL Full [27], which are differentiated in terms of their expressiveness.

OWL provides a way to define semantic relationships either among objects or within an object. It can define object classes and properties as well as their characteristics, such as hierarchy, equivalence, and cardinality restrictions, including header, versioning and annotation information.

Figure 2-4 is a graph representation of OWL in action as it describes purchasable goods (hourglasses and egg timers). Each rounded rectangle in Figure 2-4(a) represents an OWL class. Each OWL class can be connected to other classes and each connection describes the relationship between them, which can be regarded as data semantics. The overall description is made in a hierarchical way using OWL and RDF/RDFS keywords, such as owl:Class, rdf:Property, rdfs:subClassOf and rdfs:subPropertyOf, which are illustrated in the upper right bevel of Figure 2-4 (a). However, OWL key words, such as owl:equivalentClass and owl:equivalentProperty, make the description a graph, which is illustrated in the lower left bevel of Figure 2-4(a). These keywords play an important role in synchronizing synonyms in an ontology. For example, assuming that "hourglass" is a synonym of "sandglass" in the inventory management system of a retail store, both products can be retrieved by either ontology query because of the reference capability of keywords in RDFS. Figure 2-4(b) illustrates the hierarchy and equivalence relationships of the purchasable products ontology.



(a) (b)

Figure 2-3. An example of OWL applied to a product catalog

An ontology plays an important role in specifying the scope of objects, properties, and their relationships. An intelligent agent can search, query, and infer based on the specified ontology. In the event of additional or deprecated objects and properties, they can be easily added, updated, or deleted from the ontology instance. Related ontology instances can be integrated with an existing one via "reconciliation" in the design phase, which is the process of defining relationships in the design details of each ontology. Defining additional hierarchies, properties, and relationships is a part of reconciliation.

Chapter 3

Literature Review

In this chapter, we present a review of previous research related to Web service composition. Before any problem can be solved, its complexity must be assessed. Oh et al. [9] extended the proof that the complexity of the STRIPS planning problem [28] is NP-complete, which is presented by Bylander [29], and showed that the Web service composition problem is also NP-complete. Various methodologies have been used to solve Web service composition problems, each taking different factors into account. We have grouped these methodologies into logic-based approaches, mathematical programming approaches, and other approaches. Approaches in the literature can also be classified based on other factors, which we compare and contrast: (1) optimal versus heuristic solution, (2) syntactic versus semantic, (3) quality of service (QoS) versus functional, and (4) parameter-level versus operation-level. As we frame the literature in this way, we are able to identify the gaps that our research intends to fill.

3.1 Classification in Terms of Methodology

The AI-Planning problem has been addressed by a variety of methodologies, such as planning-graph techniques, propositional satisfiability techniques, constraints satisfaction techniques, situational calculus, rule-based planning, theorem-proving, and integer programming [13, 30, 31, 32]. We classify these methodologies into logic-based approaches, mathematical programming approaches and other approaches, and discuss them in detail in this section.

3.1.1 Logic-based methods

Logic-based methods have been used to address the functional aspects of Web service composition without concern for quality of service (QoS). McIlraith et al. [33, 34, 35] utilized ConGolog [36, 37] (a high-level logic programming language based on situational calculus) to create an automatic Web service composition. Possible Web service compositions are represented as a tree of situations. In this approach, a Web service corresponds to an action, while a situation is a sequence of Web services from the initial state. States correspond to parameters of Web services.

Rao et al. [38] applied the Linear Logic theorem to solve the Web service composition problem. This approach considered not only functional attributes but also non-functional ones in composing Web services. We have the same objective in our Integer Linear Programming (ILP)-based methodology. However, the Linear Logic approach partially incorporates a decision maker into the solution procedure, using the attributes of core services already selected by the user. Our approach differs on this point. Oh et al. [11] represented the Web service composition problem using Description Logic and solved the problem by using the flexible parameter matching framework.

3.1.2 Mathematical programming methods

Mathematical programming approaches have contributed to incorporating QoS factors into the solution procedure for Web service composition problems. In 1999, Vossen et al. [39, 40] and Kautz and Walser [41, 42] were the first to use an ILP-based approach to solve AI-Planning problems. In 2005, Van Den Briel and Kambhampati [43, 44] reported that the ILP approach showed relatively good or even better performance than the most efficient SAT-based methods. Yoo and Kumara [14] then extended its application, formulating the Web service composition problem based on Van Den Briel and Kambhampati's Optiplan [43]. They identified unique problem characteristics of the Web service composition problem and reflected those characteristics into their ILP formulation. In their formulation, the number of variables and constraints does not increase exponentially as the number of Web services increases.

The most significant benefit of the ILP approach is the ability to incorporate not only functional attributes (e.g., parameter matching between anterior and posterior Web services), but also non-functional ones, (e.g., cost or time spent in invoking Web services). Note that most logic-based approaches only address functional requirements. Furthermore, multiple objectives can be incorporated into the objective function, such as both cost and time, so alternative Web service composition solutions that are on the efficient frontier[2] of the multiple objectives can be found. Such multiple-objective ILP problems can be solved using various algorithms [45, 46].

As shown in Figure 3-1, there have been a few research studies related to ILP-based Web service composition, which consider non-functional quality of service (QoS) factors [47, 48, 49]. However, Web service composition in those studies was not performed on a parameter level, but on an operation level [47, 48, 49, 50, 51]. Multiple alternative Web services were selected for

[2] An efficient frontier is a solution set where efficient solutions exist. An efficient solution can improve an objective only at the expense of at least one other objective.

each operation using non-ILP methods, and an ILP-based method was leveraged in an attempt to find the composition with the best QoS among the services.

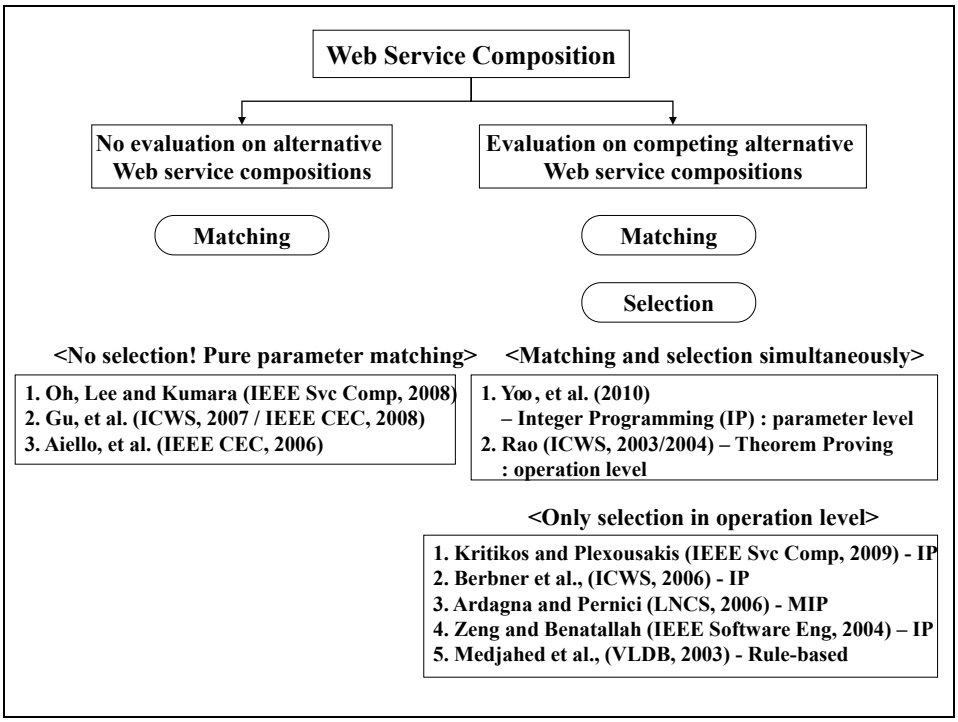


Figure 3-1. Web service composition with functional and/or non-functional attributes

3.1.3 Other methods

There are a few of other methodologies that do not fit into the two categories, but are worth noting. SWORD [52] leverages a rule-based expert system that calculates possible service outputs based on given inputs, and creates an appropriate Web services composition. Sirin et al. [53] also proposed a prototype version of a semi-automatic method involving users for Web service composition. Their method provides possible Web services to users at each composition step by matching Web services based on functional properties and filtering them out based on non-functional attributes.

3.2 Other Classifications

Web service composition approaches can also be classified in the following four ways: (1) optimal versus heuristic solution, (2) semantic versus syntactic, (3) functional requirements versus quality of service (QoS) concerns, and (4) parameter-level versus operation-level. The approaches are explained and compared next.

3.2.1 Optimal versus heuristic solution approaches

Optimal solution approaches aim to solve Web service composition problems in the best possible way regardless of time, while heuristic approaches seek to find workable solutions quickly. Generally, if a Web service composition problem requires real-time response, then a heuristic approach is appropriate. However, if there is sufficient design time for an application, then an optimal approach provides the best possible solution. Many previous research studies related to Web service composition have proposed heuristic approaches [10, 11, 12, 54, 55, 56, 57], while just a few papers [48, 49, 58] have proposed optimal solution approaches to Web service composition problems. Optimal solution approaches can be further classified into two categories: operation-level and parameter-level (see Figure 1-2). All previous papers proposing optimal solution approaches solved Web service composition problems at the operation level. However, we propose an optimal solution approach at the parameter level, which is one of the contributions of this research.

3.2.2 Semantic versus syntactic approaches

As discussed in the previous chapter, some of the most challenging problems of Web service composition are semantic issues. As the name implies, Semantic Web service composition considers the meaning of content. Conversely, Syntactic Web service composition does not consider semantics.

In recent times, semantic processing capability has become more important. This is evidenced by the evolution of the Web Service Challenge, a collaborative competition sponsored by the IEEE International Conference on Commerce and Enterprise Computing, where researchers work on Web service composition problems [59]. Since 2008, the competition has required participants to be equipped with semantic processing capability.

Several representative research papers [14, 32, 55, 57] address Semantic Web service composition problems. Our semantic approach differs in that it extracts two types of relationships among Web service input/output parameters from the ontology written in OWL: (1) equivalence and (2) hierarchy. We formulate the relationships using Integer Programming and integrate them into Syntactic Web service formulations.

3.2.3 Functional requirements versus quality of service concerns

Functional requirements are the most basic conditions that must be met for a Web service composition solution to be valid. A universal functional requirement for a Web service composition is that all input parameters of a Web service must be satisfied in order to invoke it. As discussed in the problem definition in Chapter 2, a solution to a Web service composition consists of a set of Web services and their invocation sequence, either serially or in parallel. Therefore, the sequence of Web services must satisfy functional requirements.

Many logic-based approaches [9, 33, 36, 38] consider functional requirements to be the only constraints of Web service composition, and they attempt to find solutions that invoke the minimum number of Web services. However, as discussed in the research motivation part of Chapter 1, a solution with a minimum number of Web services may not be the best solution if Web service users have their own specific objectives, such as cost or time. To address such a case, many scholars have tried to incorporate quality-of-service (QoS) into Web service composition [47, 48, 49, 57, 58, 60]. These representative research papers considered not only functional requirements but also QoS factors. Likewise, our approach considers both during the Web service composition process.

3.2.4 Parameter-level versus operation-level composition

Details in Web service composition can be classified as being either parameter-level or operation-level. As shown in Figure 1-2, parameters are the components of an operation which must be considered in order to complete a composition solution. Several research studies [33, 48, 49] have considered operations that are functionally categorized in advance, and other scholars [10, 14, 32, 54, 56] have addressed parameter-level Web service composition. Our research considers parameters during the composition process.

3.3 Summary

Table 3-1 compares several representative works with our research in terms of the framing we have used for literature review. Our work proposes an optimal solution approach to parameter-level Semantic Web service composition, considering both functional requirements and quality of service. To the best of our knowledge, our work is the first to consider all four framing aspects compared in the table.

Table 3-1. Comparison of Representative Research to This Study

	Optimal or Heuristic	Semantic or Syntactic	QoS or Functional	Parameter or Operation
Yoo, et al. (2010)	Optimal	Both	Both	Parameter
Kritikos and Plexousakis (2009)	Optimal	Both	Both	Operation
Oh, et al. (2008)	Heuristic	Both	Functional	Parameter
Gu, et al. (2007)	Heuristic	Syntactic	Functional	Parameter
Zeng, et al. (2004)	Optimal	Syntactic	Both	Operation

Chapter 4

Solution Methodology

Three main objectives are addressed in this research. The first objective is to find a Web services composition solution that both satisfies functional requirements (i.e., parameter matching) and optimizes non-functional attributes (e.g., cost, time, reputation). The second objective is to find not only the best composition solution, but also near-optimal composition solutions (k-best solutions). The third objective is to find composition solutions that consider semantic relationships among parameters (e.g., equivalence and hierarchy).

In order to address the first issue, we propose to formulate the Web service composition problem mathematically while incorporating both functional and non-functional requirements. Integer Linear Programming (ILP) [61] is used when all variables are integers. In cases where some, but not all, variables are integers, a Mixed-Integer Linear Program (MILP) [62] is appropriate.

Various approaches have been taken in order to find k-best solutions for combinatorial optimization problems [63, 64, 65, 66, 67, 68, 69], one example being a binary search algorithm. To find k-best solutions to the ILP-based Web service composition problem, we propose leveraging cutting planes that eliminate the optimal solution while retaining other feasible solutions in binary integer programs. By sequentially applying the cutting planes to the current best solution, k-best solutions are identified.

These two issues can be regarded as syntactic problems; however, in the Web service composition problem, semantic problems also exist. In cases where certain parameters have hierarchical or synonymous relationships, the solution space is enlarged, and superior solutions can be found when Web service composition software agents understand the semantic

relationships. To account for semantics in the Web service composition problem, we propose the utilization of Semantic Web Technology as discussed in Chapter 2.

In summary, Integer Linear Programming is used for the mathematical formulation, a cutting plane method is utilized to generate k-best solutions, and semantic issues addressed by leveraging the semantic Web technology. The solution methodologies for these three issues are discussed in detail in this section along with proposed system architecture.

4.1 Integer Programming Formulation for Syntactic Web Service Composition

We propose formulating Web service composition problems using Integer Linear Programming (ILP). This section presents a general formulation for Web service composition problems along with analysis of variables and constraints. To demonstrate how to formulate a Web service composition problem using the proposed formulation, an example Web service composition problem is used. Refer to Appendix A for a detailed formulation of the example.

4.1.1 Domain definition

W is the set of Web services in a UDDI system.

P is the set of all Web service parameters in W.

$P_{In} \subseteq P$, is the set of parameters that are used as input for any Web service.

$P_{Out} \subseteq P$, is the set of parameters that are used as output for any Web service.

$P_{Initial} \subseteq P$, is the set of initially given parameters.

$P_{Goal} \subseteq P$, is the set of goal parameters.

$W_p^{input} \subseteq W, \forall p \in P$, is the set of Web services that have parameter p as input.

$W_p^{output} \subseteq W, \forall p \in P$, is the set of Web services that have parameter p as output.

Stage (s): $1 \le s \le S$, where S is the maximum number of stages for Web service composition.

W_s is the set of Web services simultaneously invoked at Stage s.

4.1.2 Problem classification

If $S = 1$ and $|W_1| = 1$, then the problem is a *Web Services Discovery Problem*; otherwise, if ($S > 1$ or $|W_1| > 1$), then it is a *Web Services Composition Problem*.

4.1.3 Variable definition

For all $w \in W$, $s \in 1,...,S$, $y_{w,s}$ are *invocation variables*, and they are defined as follows.

$$y_{w,s} = \begin{cases} 1 & \text{if Web services } w \text{ is invoked in stage } s, \\ 0 & \text{otherwise.} \end{cases}$$

The following variables are *parameter usage variables* and are defined as follows.

$$x_{p,s}^{input} = \begin{cases} 1 \text{ if Web services } w \text{ is invoked in stage } s \text{ such that } w \in W_p^{input}, \\ 0 \text{ otherwise.} \end{cases}$$

$$x_{p,s}^{output} = \begin{cases} 1 \text{ if Web services } w \text{ is invoked in stage } s \text{ such that } w \notin W_p^{input} \wedge w \in W_p^{output}, \\ 0 \text{ otherwise.} \end{cases}$$

$$x_{p,s}^{known-unused} = \begin{cases} 1 \text{ if parameter } p \text{ is known but not used in stage } s, \\ 0 \text{ otherwise.} \end{cases}$$

Parameter usage variables consist of three types of parameters, (1) *input* parameters, (2) *output* parameters, and (3) *known-unused* parameters. Parameters can be used as *input* parameters or *output* parameters in a stage, or they are carried to the next stage, in which case parameters would be *known-unused* parameters.

4.1.4 Formulation

(1) Objective Function

One of the major contributions of this research is the incorporation of quality-of-service (Qos) factors into the formulation. Any numerically describable QoS factors can form the objective function.

$$Minimize \sum_{w \in W} \sum_{s \in S} f(y_{w,s}), \text{ where } f \text{ is the function of Web service } w \text{ and Stage } s. \quad (1)$$

(2) Constraints and Variables

a. Initial knowledge constraints

Initially-known parameters are represented by the initial constraints. To define the initial constraints, Stage 0 variables are introduced. The initially-known parameters are expressed by setting *output* variables of Stage 0 to "1," which can be interpreted that the initial knowledge is the output knowledge of Stage 0. All other *input* and *known-unused* variables at Stage 0 are set to "0."

$$x_{p,0}^{output} = 1,\ x_{p,0}^{input} = x_{p,0}^{known\text{-}unused} = 0 \quad \forall p \in P_{Initial} : \text{Given parameters at initial stage} \quad (2)$$

$x_{p,0}^{input}, x_{p,0}^{output}, x_{p,0}^{known\text{-}unused} = 0, \forall p \notin P_{Initial}$: Unknown parameters at initial stage (3)

* Number of constraints = $3 \cdot |P|$

b. Goal knowledge constraints

Goal parameters are the parameters to be discovered during Web service composition. Goal parameters are represented by goal constraints. Initially-unknown parameters become known through the invocation of Web services, which means the unknown parameters will be the *output* parameters of invoked Web services. Once a parameter is known, it is used as an *input* parameter for other Web services. In cases where some known parameters are not used in a stage (*known-unused*), the parameters are carried to the next stage until they are carried to the last stage. Therefore, the goal parameters can be defined using the variables for the last stage. In the last stage, the goal parameters should be used as *input* parameters, *output* parameters, or *known-unused* parameters of Stage S.

$x_{p,S}^{output} + x_{p,S}^{known\text{-}unused} + x_{p,S}^{input} \geq 1 \quad \forall p \in P_{Goal}$: Goal parameters at the final stage (4)

* Number of constraints = $|P_{Goal}|$

c. Web services invocation constraints

Web services invocation constraints play an important role in meeting the functional requirements of Web service composition solutions. Functional requirements mean that all the required input parameters of a Web service should be ready before invoking the Web service, and that once the Web service is invoked, output parameters of the Web service become known.

$$\sum_{w \in W_p^{input}} y_{w,s} \geq x_{p,s}^{input} \quad \forall p \in P, s \in 1,...,S \tag{5}$$

$$y_{w,s} \leq x_{p,s}^{input} \quad \forall w \in W_p^{input}, \forall p \in P, s \in 1,...,S \tag{6}$$

These two sets of constraints guarantee that all input parameters of a Web service should be ready before invoking the Web service, while the below two sets of constraints guarantee that all output parameters of a Web service should become known after invoking the Web service.

$$\sum_{w \in W_p^{output} \setminus W_p^{input}} y_{w,s} \geq x_{p,s}^{output} \quad \forall p \in P, s \in 1,...,S \tag{7}$$

$$y_{w,s} \leq x_{p,s}^{output} \quad \forall w \in W_p^{output} \setminus W_p^{input}, \forall p \in P, s \in 1,...,S \tag{8}$$

* Number of constraints = $|S| \cdot \left(|P_{In}| + \sum_p |W_p^{input}| + |P_{Out}| + \sum_p |W_p^{output}| \right)$

d. Non-concurrency constraints

Three types of parameters are discussed in Section 4.1.3. Among the three types of parameters, non-concurrency constraints should be satisfied. Non-concurrency means that if a parameter is used as a *known-unused* parameter, then the parameter should be neither an *input* parameter nor an *output* parameter. Such non-concurrency constraints are represented as follows.

$$x_{p,s}^{output} + x_{p,s}^{known\text{-}unused} \leq 1 \quad \forall p \in P, s \in 1,...,S \tag{9}$$

$$x_{p,s}^{input} + x_{p,s}^{known\text{-}unused} \leq 1 \quad \forall p \in P, s \in 1,...,S \tag{10}$$

* Number of constraints = $2 \cdot |S| \cdot |P|$

e. Sequence constraints

In Web service composition, only known parameters can be used to invoke Web services as input parameters. If a parameter is unknown in a stage, then the parameter cannot be used in the next stage, which means that there is a sequence of knowledge usage. In other words, once a parameter is known in Stage s-1 (which means the parameter is used as an input, output or known-unused parameter in a stage), the parameter can be used as an input or known-unused parameter in Stage s. Such constraints are defined as follows.

$$x_{p,s}^{input} + x_{p,s}^{known\text{-}unused} \le x_{p,s-1}^{input} + x_{p,s-1}^{output} + x_{p,s-1}^{known\text{-}unused} \quad \forall p \in P, s \in 1,...,S \tag{11}$$

* Number of constraints = $|S| \cdot |P|$

f. Knowledge increment constraints

The parameters known in each stage during Web service composition processes keeps increasing. Once a parameter is known in a stage, the known parameter is unlimitedly reusable after the stage. The following knowledge increment constraints represent this characteristic.

$$\sum_{p \in P} (x_{p,s-1}^{input} + x_{p,s-1}^{output} + x_{p,s-1}^{known\text{-}unused}) \le \sum_{p \in P} (x_{p,s}^{input} + x_{p,s}^{output} + x_{p,s}^{known\text{-}unused}) \quad \forall s \in \{1,...,S\} \tag{12}$$

* Number of constraints = $|S|$

g. Redundant invocation prevention constraints

As discussed previously, known parameters can be infinitely reusable. Therefore, there is no use in invoking a Web service multiple times. The following redundant invocation prevention constraints prohibit a Web service from being called multiple times.

$$\sum_{s\in\{1,2,...,S\}} y_{w,s} \le 1, \quad \forall w \in W \tag{13}$$

* Number of constraints = $|W|$

h. Binary variables

$$x_{p,s}^{input}, x_{p,s}^{output}, x_{p,s}^{known\text{-}unused} \in \{0,1\} \quad \forall p \in P, s \in 1,...,S \tag{14}$$

$$y_{w,s} \in \{0,1\} \quad \forall w \in W, s \in 1,...,S \tag{15}$$

* Number of variables = $3 \cdot |S| \cdot |P| + |S| \cdot |W|$

The formulation from (1) to (15) defines variables and constraints for syntactic Web service composition. The following summarizes the number of variables and constraints required in this formulation.

* Total number of constraints:

$$3 \cdot |P| + |P_{Goal}| + |S| \cdot \left(|P_{In}| + \sum_p |W_p^{input}| + |P_{Out}| + \sum_p |W_p^{output}| \right) + 3 \cdot |S| \cdot |P| + |W| + |S|$$

where $|W| \le \sum_p |W_p^{input}| < |W| \cdot |P|$ and $|W| \le \sum_p |W_p^{output}| < |W| \cdot |P|$.

* Total number of variables: $3\cdot(|S|)\cdot|P|+(|S|)\cdot|W|+3\cdot|P|$

4.2 Cutting Plane Methods for k-Best Solutions to Web Service Composition

This section illustrates a basic concept for generating cutting planes that eliminate the optimal solution while retaining other feasible solutions in binary integer programs. In addition, a naïve cutting plane method and two improved cutting plane methods for k-best solutions are proposed.

4.2.1 A naïve cutting plane method

Figure 4-1 illustrates cutting planes that eliminate the optimal solution while retaining other feasible solutions in binary integer programs, where Circle 1 is the optimal solution and Circle 2 is the second-best one. In two-dimensional space, the optimal solution is found in the following four cases: (a) is (1,1); (b) is (0,1); (c) is (0,0); and (d) is (1,0). The gray area represents feasible spaces for each case. The (red) dotted lines are the corresponding cutting planes that eliminate the optimal solution, but not other feasible solutions for each case.

$X = P \cap Z^2$

$P = \{x \in R^2 \mid g_j(x) \le 0, j = 1,2,...,m\}$, where $g_j(x)$'s represent constraints.

$Z^2 = (b_1,\ b_2),\ b_i \sim binary, i = 1,2.$

$x^* = (b_1^*, b_2^*)$ $\sim$ the optimal solution

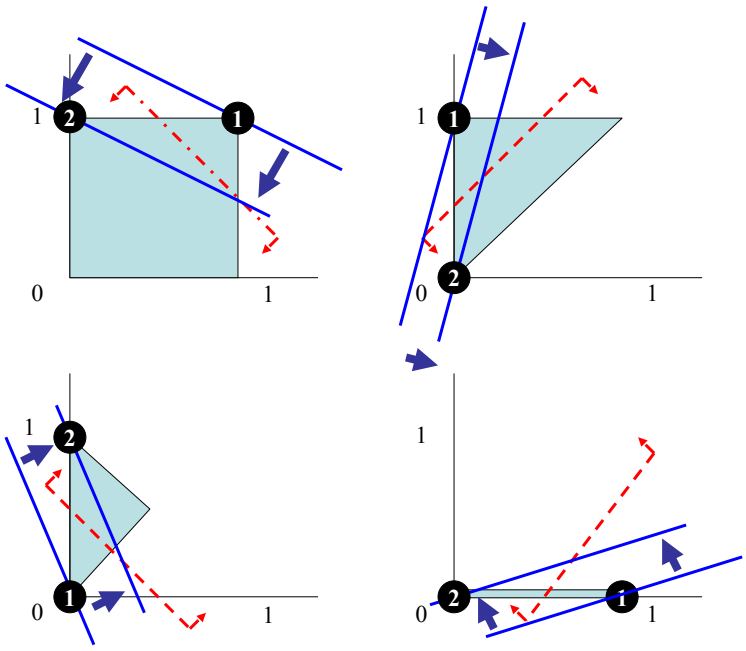


Figure 4-1. Cutting planes for k-best solutions

In the case of (a) in Figure 4-1, where the optimal solution is $x^* = (1,1)$, the cutting plane for the second best solution is:

$$\alpha x_1 + \beta x_2 = 1, \text{ where } \alpha \text{ and } \beta$$

$$\ni \alpha \left|1 - \frac{1}{2}\right| + \beta = 1,$$

$$\alpha + \beta \left|1 - \frac{1}{2}\right| = 1,$$

$$\therefore \alpha = \frac{2}{3} \text{ and } \beta = \frac{2}{3}$$

$$\Rightarrow \frac{2}{3}x + \frac{2}{3}y = 1 .$$

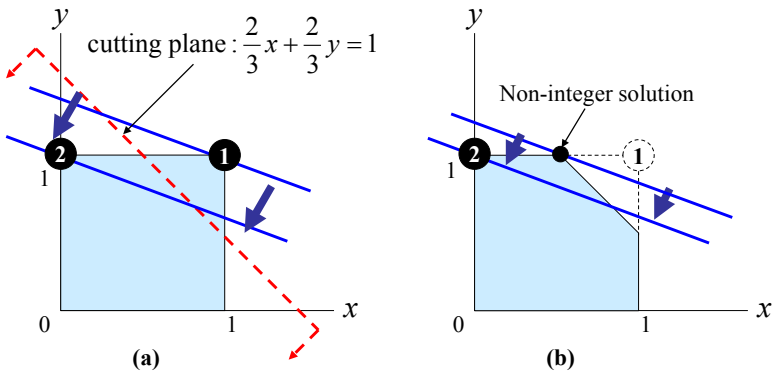


Figure 4-2. How the cut works to obtain the next best solution

4.2.2 General cutting planes for k-best solutions

$$X = P \cap Z^n$$

$P = \{x \in R^n \mid g_j(x) \le 0, j = 1,2,...,m\}$, where $g_j(x)$'s represent constraints.

$Z^2 = (b_1, b_2, ..., b_i, ..., b_n)$, $b_i \sim binary, i = 1,2,...,n.$

$x^{(1)*} = (b_1^{(1)*}, b_2^{(1)*}, ..., b_i^{(1)*}, ..., b_n^{(1)*})$ $\sim$ the (first) optimal solution

When the (first) optimal solution is $x^{(1)*} = (b_1^{(1)*}, b_2^{(1)*}, ..., b_i^{(1)*}, ..., b_n^{(1)*})$, the cutting plane for the second-best solution is:

$$\alpha_1^{(2)}x_1+\alpha_2^{(2)}x_2+\ldots+\alpha_i^{(2)}x_i+\ldots+\alpha_n^{(2)}x_n=1,\ \text{where}\ \alpha_i^{(2)}\ (i=1,2,\ldots,i,\ldots n)$$

$$\text{э}\ \alpha_1^{(2)}\left|b_1^{(1)*}-\frac{1}{2}\right|+\alpha_2^{(2)}b_2^{(1)*}+\ldots+\alpha_i^{(2)}b_i^{(1)*}+\ldots+\alpha_n^{(2)}b_n^{(1)*}=1,$$

$$\alpha_1^{(2)}b_1^{(1)*}+\alpha_2^{(2)}\left|b_2^{(1)*}-\frac{1}{2}\right|+\ldots+\alpha_i^{(2)}b_i^{(1)*}+\ldots+\alpha_n^{(2)}b_n^{(1)*}=1,$$

.....

$$\alpha_1^{(2)}b_1^{(1)*}+\alpha_2^{(2)}b_2^{(1)*}+\ldots+\alpha_i^{(2)}\left|b_i^{(1)*}-\frac{1}{2}\right|+\ldots+\alpha_n^{(2)}b_n^{(1)*}=1,$$

.....

$$\alpha_1^{(2)}b_1^{(1)*}+\alpha_2^{(2)}b_2^{(1)*}+\ldots+\alpha_i^{(2)}b_i^{(1)*}+\ldots+\alpha_n^{(2)}\left|b_n^{(1)*}-\frac{1}{2}\right|=1.$$

The number in parenthesis is the order of the k[th] optimal solution. At this stage, since we find the cutting plane for the *second* optimal solution from the *first* optimal solution, the superscript of x is (1), and that of α is (2).

When the (k-1)[th] optimal solution is $x^{(k-1)*}=(b_1^{(k-1)*},b_2^{(k-1)*},\ldots,b_i^{(k-1)*},\ldots,b_n^{(k-1)*})$, the cutting plane for the k[th] best solution is:

$$\alpha_1^{(k)}x_1+\alpha_2^{(k)}x_2+\ldots+\alpha_i^{(k)}x_i+\ldots+\alpha_n^{(k)}x_n=1,$$

where $\alpha_i^{(k)}\ (i=1,2,\ldots,i,\ldots n)$ is the solution to the following LP problem.

Min (*an arbitrary objective of* $\alpha_i^{(k)}$, *i.e.* $\alpha_1^{(k)}$, $\alpha_n^{(k)}$, *or* $\sum_{i=1}^{n}\alpha_i^{(k)}$)

subject to:

$$\left|b_1^{(k-1)*}-\frac{1}{2}\right|\alpha_1^{(k)}+b_2^{(k-1)*}\alpha_2^{(k)}+\ldots+b_i^{(k-1)*}\alpha_i^{(k)}+\ldots+b_n^{(k-1)*}\alpha_n^{(k)}=1,$$

$$b_1^{(k-1)*}\alpha_1^{(k)}+\left|b_2^{(k-1)*}-\frac{1}{2}\right|\alpha_2^{(k)}+\ldots+b_i^{(k-1)*}\alpha_i^{(k)}+\ldots+b_n^{(k-1)*}\alpha_n^{(k)}=1,$$

.....

$$b_1^{(k-1)*}\alpha_1^{(k)}+b_2^{(k-1)*}\alpha_2^{(k)}+\ldots+\left|b_i^{(k-1)*}-\frac{1}{2}\right|\alpha_i^{(k)}+\ldots+b_n^{(k-1)*}\alpha_n^{(k)}=1,$$

.....

$$b_1^{(k-1)*}\alpha_1^{(k)}+b_2^{(k-1)*}\alpha_2^{(k)}+\ldots+b_i^{(k-1)*}\alpha_i^{(k)}+\ldots+\left|b_n^{(k-1)*}-\frac{1}{2}\right|\alpha_n^{(k)}=1,$$

where $\alpha_i^{(k)}\sim$ *unrestricted* $(i=1,2....n)$.

Key findings from the cutting plane generation for k-best solutions are as follows.

(1) The cutting plane for the k-best solutions is independent of the objective function as well as constraints.

(2) The fact that all variables in the formulation of the proposed Web service composition are binary enables us to generate cutting planes.

4.2.3 An improved cutting plane approach

We also propose an improved cutting plane approach (see Figure 4-3), which eliminates a larger non-integer space than the naïve cutting plane method. In addition, the cuts eliminate areas up to adjacent binary points.

$$\sum_{i \in I} x_i + \sum_{i \in O} (1 - x_i) \leq n - 1,$$

where:

I is the index set of variables that have the value of 1,

O is the index set of variables that have the value of 0, and

$n = |I| + |O|$

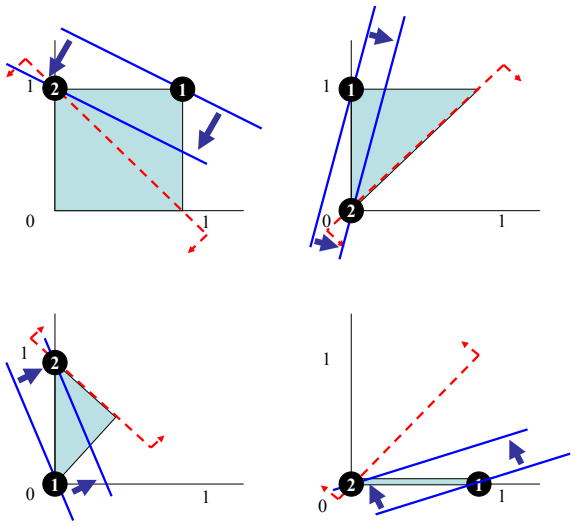


Figure 4-3. Improved cutting plane approach for k-best solutions

As a pilot, we applied the improved cuts to obtain the two best solutions in the previous small Web service composition example illustrated in Figure 1-3. It turns out that the optimal solution was WS-E (getHospitalWithBloodBank). However, in order to reach the second-best solution (1st Stage: WS-A and WS-B; 2nd Stage: WS-D; and 3rd Stage: WS-C), 131 additional cuts were required. We analyzed the solutions after applying each cut to identify the reason why such a large number of cuts were needed to obtain the next best optimal solution. It turns out that the intermediate solutions included the first optimal solution and some other Web services, such as 1st Stage WS-E and WS-A. WS-A did not reach the goal state, however, since the solution included the first optimal solution, it satisfied the objective (the goal state) with a slightly higher cost than the first optimal solution. Such a solution did not provide any added value. In order to find other ways of reaching the goal state, we created the following cutting plane approach.

4.2.4 A cutting plane approach that negates previous solutions

This cutting plane approach negates all previous solutions, so that a next best solution does not include the exact same solutions that were found previously. However, this approach does allow the next best solution to include parts of previous solutions. To implement such cuts, we introduced additional binary variables to express binary-choice-type constraints as follows.

$$\sum_{i\in I} x_i^{(k)} \neq |I| \Leftrightarrow \begin{cases} \sum_{i\in I} x_i^{(k)} \leq |I| + 1 - M \times (1 - b^{(k)}) \\ \sum_{i\in I} x_i^{(k)} \leq |I| - 1 + M \times b^{(k)} \end{cases}$$

where $\{x_i^{(k)}\}$ is the variable set
that includes Web service variables except parameter variables,
$b^{(k)}$ is a binary variable introduced at the k-th solution,
and M is a positive large number greater or equal to $|I|+1$.

This cutting plane works more effectively. We were able to obtain the next best solution as soon as we added this cutting plane to the original formulation.

4.2.5 Elimination of only the current best solution

The proposed k-best solution methods do not eliminate other feasible solutions and only exclude the k-1 best solutions. The methods take advantage of the characteristics of the proposed binary Integer Programming formulation for Web service composition. As illustrated in Figure 4-3, since the generated cutting planes, which are dotted lines, eliminate only one solution that is the current best solution, there is no possibility that the k-best method discard any feasible solutions.

4.2.6 Analysis of k-best solutions

Figure 4-4 shows 5-best solutions, in terms of cost, to the blood delivery example discussed in Section 1.2, generated using the proposed k-best solution approach. The best solution costs $40 with four Web service invocations, the second best one $50 with three, the third best one $60 with three, the fourth best one $70 with four, and the fifth best one $100 with only one invocation. The first, second and fifth best solutions form an efficient frontier of this problem. Using this k-best solutions, decision makers can broaden their understanding on the solution space and can choose the most preferable solution from their own point of view. This k-best solution problem is similar to multiple criteria decision making problems [70, 71, 72], terms that both problems can have multiple solutions that form efficient frontier. However, while the k-best solution approach focuses on only one objective and generates other good solutions in terms of

the objective, multiple criteria optimization approaches use an integrated objective that is a function of multiple objectives.

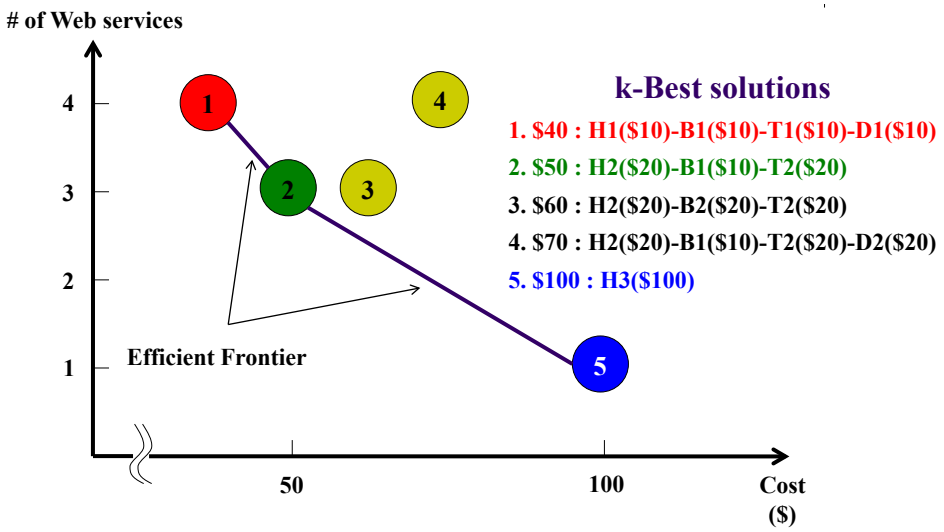


Figure 4-4. Examples of k-best solutions

4.3 Integer Programming Formulation for Semantic Web Service Composition

This section presents a general formulation for semantic Web service composition, which considers semantic relationships among parameters, in this case hierarchy and equivalence. Both relationships are formally defined in the Web Ontology Language (OWL) using rdfs:subClassOf and equivalentClass key words [27]. There are common constraints between syntactic and semantic Web service composition formulations. Therefore, this section focuses only on the additional constraints introduced to address semantic issues.

4.3.1 Variable definition

In addition to the variables defined in Section 4.1.1, for semantic propagation, *semantic variables* are defined as follows.

$$x_{p,s}^{semantics} = \begin{Bmatrix} 1 \text{ if parameter } p \text{ is known through "semantic propagation" in stage } s, \\ 0 \text{ otherwise.} \end{Bmatrix}$$

4.3.2 Formulation

(1) Initial constraints

The initial constraints of semantic Web service composition are almost the same as those of syntactic Web service composition. The only difference is that semantic variables for the initial stage, $x_{p,0}^{semantics}$, are added.

$x_{p,0}^{output} = 1, x_{p,0}^{input} = x_{p,0}^{known\text{-}unused} = x_{p,0}^{semantics} = 0, \ \forall p \in P_{Initial}$: Given parameters at initial stage (16)

$x_{p,0}^{input}, x_{p,0}^{output}, x_{p,0}^{known\text{-}unused}, x_{p,0}^{semantics} = 0, \forall p \notin P_{Initial}$: Unknown parameters at initial stage (17)

(2) Goal constraints

The goal constraints of semantic Web service composition are almost the same as those of syntactic Web service composition. The only difference is that semantic variables for the last stage, $x_{p,S}^{semantics}$, are added.

$x_{p,S}^{output} + x_{p,S}^{known\text{-}unused} + x_{p,S}^{input} + x_{p,S}^{semantics} \geq 1 \ \ \forall p \in P_{Go\ a}$: Goal parameters at the final stage (18)

(3) Semantics propagation constraints

Semantic relationships among Web service parameters are represented in the semantics propagation constraints. *Semantics propagation* occurs when one parameter has a semantic relationship with another parameter, and the knowledge of one parameter affects the knowledge of the other parameter. For example, when used as a parameter of a Web service, it is assumed that the "U.S. Address" class inherits from the "Address" class, meaning the former has additional member variables or methods than the latter according to the definition of inheritance in objected-oriented programming. Hence, once the content of U.S Address is known, then that of Address is known as well. This semantic chaining of knowledge is the essence of semantics propagation. This research limits its scope to two instances of semantics propagation: hierarchy and equivalence. However, the proposed ILP (Integer Linear Programming) framework for Semantic Web service composition has capacity to handle other complex semantic relationships described in Web Ontology Language. We define the semantics propagation constraints next.

First-order semantics propagation constraints are needed to describe the fact that once parameter p becomes known, then the semantic variable for parameter p becomes also known, and are as follows:

$$x_{p,s}^{semantics} \geq x_{p,s}^{output} \ \forall p, s \in \{1, ..., S\}. \tag{19}$$

Second-order semantics propagation constraints define semantics propagation between a child parameter and its direct parent parameter and are as follows:

$$x_{p,s}^{semantics} \geq x_{q,s}^{semantics} \ \forall (p,q) \text{ where } p \text{ is the direct parent of } q. \tag{20}$$

Third-order semantics propagation constraints define semantics propagation in the whole parameter hierarchy, and are as follows:

$$x_{p,s}^{semantics} - x_{q,s}^{output} \le x_{p,s}^{h} \ \forall s, \forall (p,q), \text{ where } p \text{ is a parent of } q \text{ in parameter type hierarchy.} \quad (21)$$

$$\sum_{h} x_{p,s}^{h} \le f(p) - 1 \ \forall p, s \in \{1,...,S\}, \text{ where } f(p) \text{ is the number of child parameters of } p. \quad (22)$$

The equivalence relationships among parameters are defined as follows.

$$x_{p,s}^{semantics} = x_{q,s}^{semantics} \ \forall (p,q) \text{ where } p \text{ is equivalent to } q. \quad (23)$$

(4) Sequence constraints

Only when parameter p is known from the previous stages can it be used as an input or a known-unused variable. Such sequential requirements are represented in the sequence constraints.

$$x_{p,s}^{input} + x_{p,s}^{known\text{-}unused} \le x_{p,s-1}^{input} + x_{p,s-1}^{output} + x_{p,s-1}^{known\text{-}unused} + x_{p,s-1}^{semantics} \quad \forall p \in P, s \in \{1,...,S\}. \quad (24)$$

(5) Binary variables

The binary variables of semantic Web service composition are almost the same as those of syntactic Web service composition. The only difference is that semantic variables are added.

$$x_{p,s}^{input}, x_{p,s}^{output}, x_{p,s}^{known\text{-}unused}, x_{p,s}^{semantics} \in \{0,1\} \quad \forall p \in P, s \in \{1,...,S\} \quad (25)$$

$$y_{w,s} \in \{0,1\} \quad \forall w \in W, s \in 1,...,S \quad (26)$$

4.4 Experimental Results

We used the test sets from the 2008 Web Service Challenge (WSC 2008), which is a competition for semantic Web service composition held along with the IEEE Joint Conference on

E-Commerce Technology (CEC) and Enterprise Computing, E-Commerce and E-Services (EEE) [73]. Each test set contained a WSDL file, an OWL file, and an XML file. The WSDL file included descriptions for all Web services, such as name, input/output parameters, etc. The OWL file defined semantic relationships among the input/output parameters of the given Web services. The XML file specified initially-known parameters and goal parameters. The composition results were generated in WSBPEL (Web Services Business Process Execution Language) [74].

Table 4-1 summarizes the test set specifications and results. The test sets were formulated using Integer Linear Programming and solved using ILOG CPLEX Optimizer [75], a commercial Integer Programming solver, through our composition software agent. The composition software agent found the optimal solutions to Problem sets from 1 to 4. However, it could not solve Problem set 5 due to lack of memory. Our agent successfully found out that Problem set 6 does not have any solution.

Table 4-1. Problem sizes of the test sets provided by WSC 2008

Test Sets	# of Parameters	# of Services	Solution time (seconds)	# of Variables	# of Constraints	Results (#Stage, #WS)
1	5,000	100	23.025	193,750	293,452	Optimal (3,10)
2	5,000	500	35.731	393,780	654,491	Optimal (3, 5)
3	10,000	1,000	38.917	838,791	1,354,145	Optimal (5, 10)
4	10,000	1,000	370.583	777,254	1,289,791	Optimal (8, 21)
5	40,000	2,000	N/A	3,511,879	5,280,279	Out of memory
6	10,000	500	22.216	764,886	1,177,609	Infeasible

For this experiment, we used high performance computing (HPC) resources provided by the HPC Group at Penn State, which consisted of Quad 2.5 GHz AMD Processors and 32 GB of RAM. CPLEX 11.0 was used to obtain the optimal solution for each test set.

4.4.1 Consideration of quality-of-service attributes

Table 4-2 summarizes the results of two composition runs. The two objectives of the Integer Linear Programming formulations are minimizing cost and the number of Web services, respectively. The cost of each Web service was randomly generated for this experiment because the test sets from the 2008 Web Service Challenge did not include cost information. Depending on the objectives, different solutions were found. Table 4-2 clearly shows that cost and number of Web services are conflicting criteria, which generate Pareto optimal solutions. That is, a composition solution with the minimum number of Web services does not mean that the solution is the most inexpensive than other solutions, which can be analogous to airline ticket price. Usually, flights with multiple stops are cheaper than non-stop flights. This result demonstrates the significance of QoS aspects in Web service composition.

Table 4-2. Composition results in terms of cost and number of Web services

Test sets	Main Objective: Cost ($)		Main Objective: Services (#)	
	Cost	Services	Cost	Services
1	37	10	59	3
2	20	5	23	3
3	46	5	61	5
4	72	10	139	8
5	No solutions		No solutions	
6	Out of memory		Out of memory	

4.4.2 Consideration of semantics

Semantic considerations enabled us to find solutions for the WSC 2008 test set that could not be found with only syntactic considerations. Table 4-3 summarizes the results of two composition runs. One run considered syntax only (pure string matching), and the other run

considered both syntax and semantics. Consideration of semantics made it possible to find the optimal solution for Test sets 1 through 4. By considering only syntax, no feasible solution could be found in the test sets. These experimental results clearly show the significance of semantic consideration. Test set 5 was designed not to have any solution by the competition host. Our software successfully detected the infeasibility of the test set. Unfortunately, our software could not find out the optimal solution to Test set 6, due to insufficient computer memory.

Table 4-3. Solution existence when semantics were considered

Test Sets	Solution Existence	
	Syntax Only	Semantics & Syntax
1	No	Yes
2	No	Yes
3	No	Yes
4	No	Yes
5	No	No
6	No	No

4.5 Solution Optimality

The optimality of the solution found by our composition software is verified through the comparison with the provided optimal solution from the 2008 Web Service Challenge (WSC 2008). Our composition software successfully found the optimal solution to Problem set 1 to 4 and found out that there are no solutions in Problem set 6.

4.6 System Architecture

Figure 4-5 illustrates the architecture of the Web service composition software in which the proposed framework is implemented. The software consists of three steps: bootstrapping, query processing, and execution. In this section, the three steps are discussed in detail.

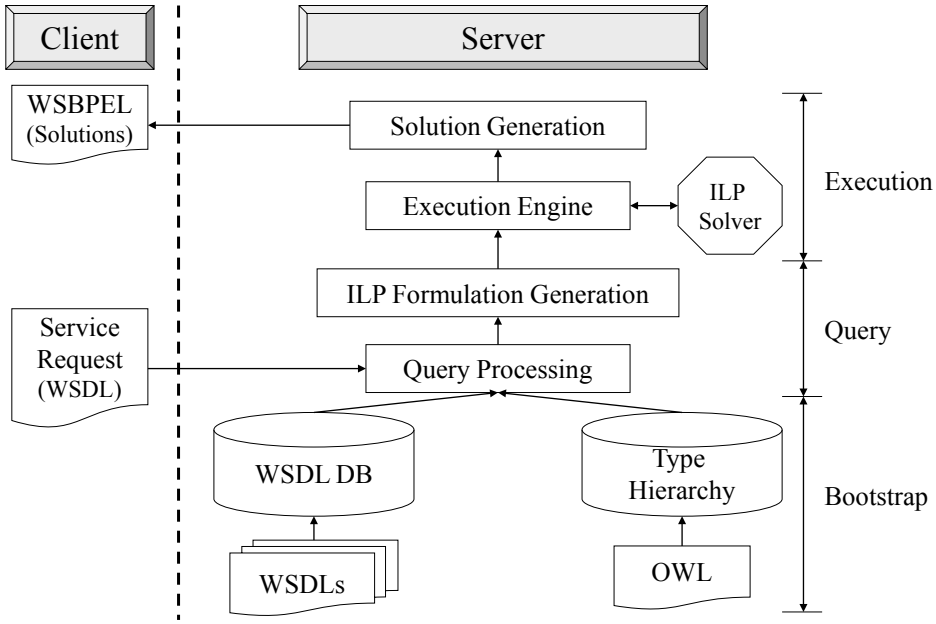


Figure 4-5. System architecture

4.6.1 Bootstrapping

In the bootstrapping step, the Web service composition software reads all input WSDL (Web Service Description Language) files [6] and the OWL (Web Ontology Language) file [27] that includes semantic relationship information. The WSDL files include all Web services under consideration for the Web service composition process. From a Service-Oriented Architecture (SOA) [1] point of view, those Web services are the ones stored in UDDI (Universal, Description, Discovery, and Integration) [7]. Once it is finished reading the WSDL and OWL files, the

composition software is ready to run the composition process. Samples of the actual WSDL files and OWL files are presented in Appendix B.

4.6.2 Query processing

The query processing step starts when query files are received. The query file is written in XML and includes initially-known parameters and goal parameters. This step begins to generate the Integer Linear Programming formulation (ILP) based on the WSDL and OWL files that were read in the previous step.

4.6.3 Execution

Finally, the execution step performs ILP problem-solving and generates a WSBPEL (Web Service Business Process Execution Language) file [74], in which a Web service composition solution is specified in detail. Figure 4-6 shows an example solution in WSBPEL.

```
− <bpel:process name="WSC08" targetNamespace="http://www.ws-challenge.org
  /WSC08CompositionSolution/">
  − <bpel:sequence name="main">
      <bpel:receive name="receiveQuery" portType="solutionProcess" variable="query"/>
    − <bpel:switch name="SolutionAlternatives">
      − <bpel:case name="Alternative-Solution0">
        − <bpel:sequence>
          − <bpel:flow>
              <bpel:invoke name="svcGetHospital" portType="pt15" operation="op15"/>
              <bpel:invoke name="svcGetBloodBank" portType="pt68" operation="op68"/>
            </bpel:flow>
            <bpel:invoke name="svcAskDelivery" portType="pt41" operation="op41"/>
            <bpel:invoke name="svcGetDirection" portType="pt18" operation="op18"/>
          </bpel:sequence>
        </bpel:case>
      </bpel:switch>
    </bpel:sequence>
  </bpel:process>
```

Figure 4-6. A composition result in WSBPEL

4.7 Summary

The proposed methodologies for semantic Web service composition have presented in this chapter. The proposed IP formulation for semantic Web service composition guarantees to generate the optimal solution. Leveraging the proposed k-best solution methods, other good solutions are generated, which play a significant role of broadening decision maker's view on the solution space. The system architecture for the Web service composition systems has been also discussed in detail.

Chapter 5

An Application of Web Service Composition: Modular Product Design

Global product development is transforming the way many companies conduct business, and product modularity is a critical component for success [76]. Large multinational corporations are using product modularity to help create innovative product development systems utilizing best practices among different divisions to speed up development and reduce costs. For example, Ford is leveraging the best practices of four brands in its Global Product Development System (GPDS) – Ford, Mazda, Volvo, and Aston Martin/Land Rover – to "bring vehicles to the market faster and for less cost" [77]. 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 [78].

Clearly defined interfaces between modules enable the success of distributed global product development. They allow geographically-distributed teams to work autonomously before modules are integrated into a product. Without such modularity, "more intense collaboration across design interfaces is necessary" [76], which invariably causes delays and missteps in the product development process. The global scale of today's economy compounds the problem further, as cyberinfrastructure is becoming increasingly critical for seamless integration and maintenance of organizational operations [79]. Clearly defined interfaces between modules will foster the design of machine-readable interface representations, because modular components are composed of multiple input/output interfaces to other modules. As a result, state-of-the-art, machine-readable languages that have been developed by the Web service community will then be able 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 [2], to formalize a cyberinfrastructure-based framework

for modular product design in a global manufacturing environment. The cyberinfrastructure consists of a machine-readable representation scheme for components, a design repository to store component descriptions, and a software agent that aids in product design. The proposed framework is rooted in an analogy between SOA's web services and their composition and modules and modular product design. To realize this framework, we need to address three challenges that currently hinder the use of SOA in modular product design:

1. Developing an interface-oriented machine-readable representation scheme for modularized components;

2. Formalizing a cyberinfrastructure-based framework that enables global users to describe, publish, and discover component information in a standardized way; and

3. Adapting Artificial Intelligence (AI) planning algorithms to support modular product design.

In this chapter, we propose such a cyberinfrastructure-based modular product design framework. The motivation behind the application is discussed in Section 5.1. Section 5.2 reviews previous literature associated with modular product design and Web service composition. Section 5.3 discusses the elements of the proposed framework in detail. Section 5.4 demonstrates the framework through a case study. Finally, Section 5.5 discusses the implications of this application and relevant future work.

5.1 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, U.S.-based organizations 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, geographical distance and language barriers hinder the maximum utilization of resources in the current globalized environment. Salespeople travel all over the world with multi-language versions of hard-copy catalogs advertising available parts and components. Word of mouth is still one of the common practices for direct marketing.

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 [1]. Such synergetic efforts will significantly differentiate this new global paradigm from the current best practice, namely, proprietary catalog-based systems. Figure 5-1 shows a hypothetical scenario for global manufacturing based on modular product design using cyberinfrastructure. Suppliers from all over the world publish their product information in a machine-readable language on a global design repository. Note that even though repositories and machine-readable language are global standards, they can also be built and standardized locally, at the industry- or company-level. Product designers from different organizations can refer to component information stored in the design repository and find proper components satisfying their design parameters.

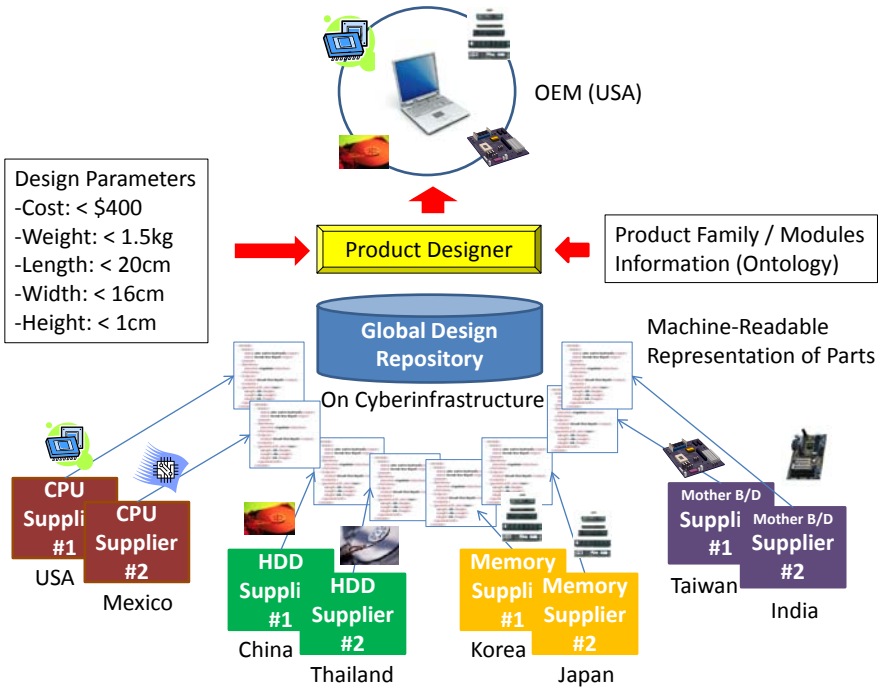


Figure 5-1. Modular Product Design Framework for Global Manufacturing

5.2 Background and Related Work

Machine-readable module representation is the first step to ensuring 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) [17] to describe functions and associated flows in computer-based design [80, 81, 82, 83]. Devanathan et al. [84] presented the concept of components in XML, and Bohm et al. [85] introduced an extensive data schema to capture fundamental elements of design information. These representation schemes for products or parts were designed primarily from a materials or energy flow point of view, which is sequential or flow-oriented; however, the recent trend toward modularization makes the old schemes obsolete. We propose interface-oriented representation schemes as a way of solving this problem.

An interface consists of a set of flows and specifies detailed parameters for the constituent flows. However, an interface can also encapsulate detailed flow information. For example, an electrical flow, a connector type, and a communication protocol together form the USB (Universal Serial Bus) interface of a computer peripheral, but the interface name (USB) and the version number encapsulate the specifications of its three elements.

To store design knowledge, a digital design repository has been developed by the Missouri University of Science & Technology (MS&T, former the University of Missouri – Rolla) and the National Institute of Standards and Technology (NIST) [83, 86, 87]. The digital design repository is a web-based repository that stores specifications of selected components and relationship data among them, which users can review through Web browsers. Currently, it contains detailed design knowledge pertaining to approximately 120 consumer products [88]. In addition, the repository provides users with design tools, such as the Function-Component Matrix (FCM) and Design Structure Matrix (DSM). However, the digital design repository does not hold sufficient component information with enough detail to actually design a functional, working product. In addition, the repository provides users with one-way service, which means that it stores pre-selected product design data that can only be used for reference purposes; it does not allow users to participate in building the data.

Standardization efforts yielded the Functional Basis [89, 90], which abstracts terms of function and flow, limits the number of terms, and recommends those terms as component descriptors. The Functional Basis plays an important role in the systematic description of components, and helps create transparent communications among designers or design software applications. Abstraction, however, gives rise to incompatibility issues due to the ambiguous nature of the abstracted terms.

Bryant et al. [88, 91] developed a computational tool for automated concept generation. After creating a function chain, relevant components are selected utilizing the Function-

Component Matrix, and component compatibility is checked using the Design Structure Matrix. However, the tool does not support a branching and merging scheme during module assembly. Also, the tool cannot directly handle function chains with multiple inputs and outputs. Instead, it decomposes a function chain with multiple inputs and outputs into multiple function chains which each contain only a single input and output.

In other computational product design research projects, Campbell et al. [92, 93] 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 a set of pre-determined objectives. A-design is based on the agent-based and adaptive nature of the process [92, 93]. Mittal et al. [94] implemented an expert system called PRIDE to design paper handling systems. It acquires knowledge from expert designers and performs a knowledge-guided search for possible designs that satisfy requirements. Navinchandra et al. [95] presented a case-based approach to exploit the knowledge embodied in prior designs. They captured and saved prior designs regardless of success or failure in order to build on prior successes and learn from previous failures. They applied this case-based approach to the conceptual design of hydro-mechanical systems. Finally, Titus and Ramani [96] formulated concept design problems as constraint satisfaction problems.

Likewise, as briefly mentioned above, the Web service community has been developing a formal way to describe, publish, store, and discover software components, called Service Oriented Architecture (SOA). Figure 1-1 illustrates SOA-based Web services. Universal Description Discovery and Integration (UDDI) and Web services technology, which are two major components of SOA, form part of the cyberinfrastructure for services on the Web. UDDI serves as a registry or storage for published Web services from various Web service providers. In our proposed framework, a UDDI corresponds conceptually to the digital design repository (like the one implemented by MS&T and NIST [83, 86]) while a Web service corresponds to a

physical component whose specification is stored in the repository. The digital design repository stores specification data for selected products and their components, as well as relationship data among components, so that users can review the data through Web browsers. Table 5-1 summarizes the analogy between Web service composition and modular product design.

The specific composition of Web services used to respond to user queries is analogous to a modular product design that is used to satisfy customer preferences; a Web service composition solution consists of a set of Web services and their invocation sequence, while a modular design solution consists of a set of components and their assembly sequence. One major difference is that Web services do not have physical properties like modules in a product, such as size or weight. Since Web services do not have physical properties, most Web service composition algorithms leverage logic-based approaches, such as propositional logic or satisfiability techniques, which consider only functional requirements. The functional requirements demand that preceding Web services provide required inputs for those that follow in a composition solution.

We argue that the modular product design problem is similar to the Web service composition problem. Just as each module description can be published to a digital design repository, each Web service description can be registered with a UDDI; hence, the design problem translates into a Web service composition problem. This problem, as discussed in [12], is a planning problem, which means that AI planning algorithms can be used to generate a design. In the next section, we discuss our approach, which can support branching and merging and multiple inputs/outputs, scenarios previous research has overlooked.

5.3 Methodology

This section discusses the proposed cyberinfrastructure-based framework for modular product design in detail. The major elements of the proposed framework are a formal representation of components, a component repository, and modular product design software. The following sections discuss features and roles of each element in detail. In addition, the AI planning formulation for modular product design using Integer Programming is described.

5.3.1 Formal representation of components

Previous research conducted by MS&T and NIST identified functionality, input/output flows, and physical parameters (e.g., dimensions) as key elements for component representation [89];, 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 consists of circular disks with a head for reading/writing digital data, and two interfaces to the power supply and motherboard, as shown in Figure 5-2. The power supply interface can be either AC or DC with a certain voltage, while the motherboard interface can be either SCSI or IDE.

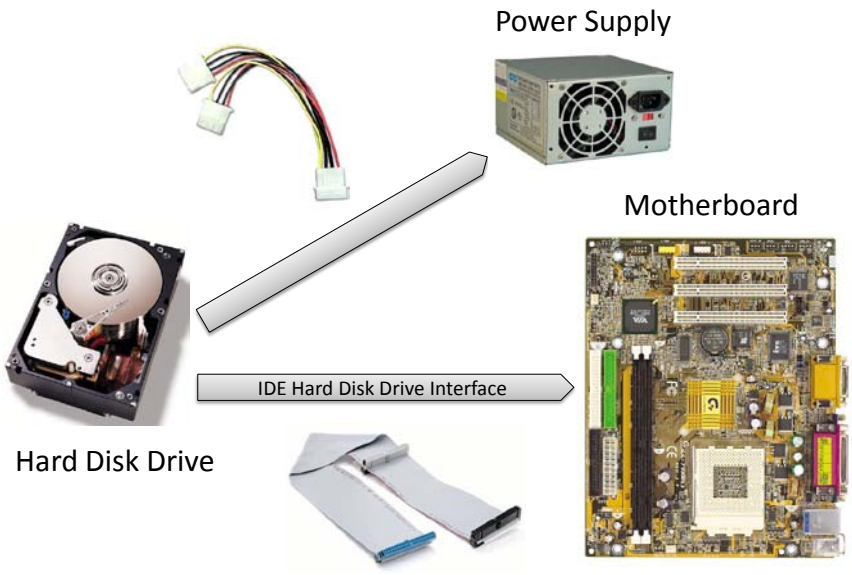


Figure 5-2. Standardized Interfaces: Examples from a Hard Disk Drive

Here, we introduce an interface-oriented, machine-readable representation scheme for modularized components. In particular, we extend the current representation scheme developed by MS&T to include interfaces attributed to the modularization trend, as shown in Figure 5-3. The representation of a component consists of input/output interfaces, features, functions specified in the functional model discussed in Section 5.3.2, and general information, including the component name, manufacturer, and geometric specifications. While current representations only support conceptual design, the proposed interface-oriented representation will contribute not only to conceptual design but also to detailed, parametric design. Figure 5-4 illustrates the concept of modularization and interface-oriented modular product design. Before modularization, non-standard connections existed among parts, and designers had to check the compatibility of all connections during design; however, modularization has helped designers to focus on external, standardized interfaces. Such modularization has simplified the representation of components

while helping to reduce the number of factors under consideration during the design process, since complex interactions between modules can be abstracted by interfaces.

```
<?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>MicroMoon</companyName>
  <componentName>MotherBD</componentName>
  <price>110</price>
  <weight>10</weight>
  <geometricSpecification width="6.0" height="0.5" depth="4.0" />
</generalInfo>
<inputs>
   <input name="Power" type="8"/>
   <input name="BUS" type="IDE"/>
 </inputs>
 <outputs>
   <output name="Socket" type="370"/>
   <output name="Pin" type="DIMM"/>
 </outputs>
 <functions>
   <function name="Position"/>
 </functions>
<features>
```

Figure 5-3. Machine-Readable XML Representation

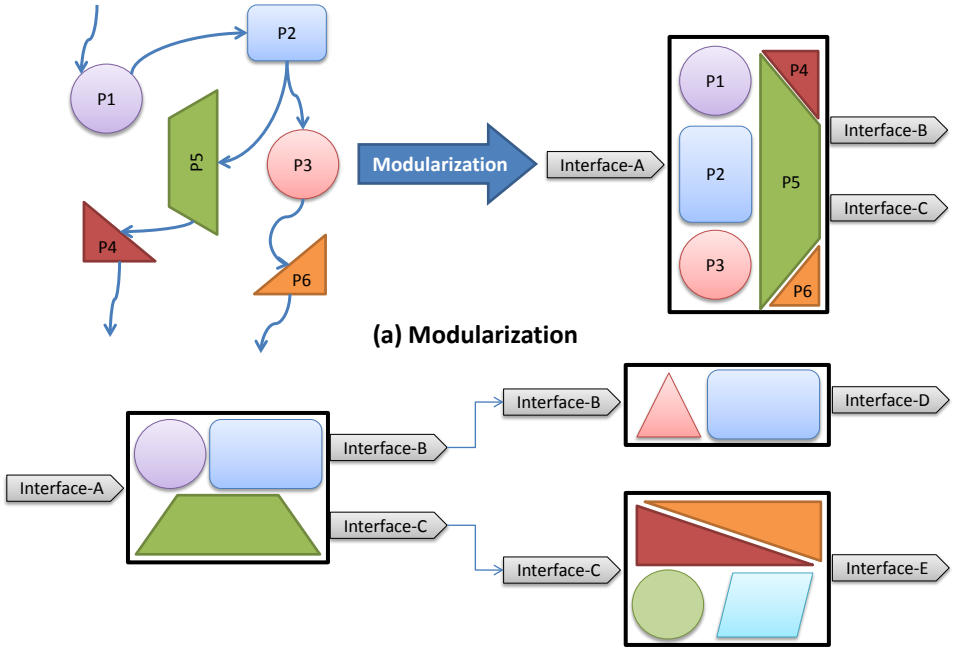


Figure 5-4. Concept of Modularization and Interface-Oriented Modular Product Design

5.3.2 Modular product design as an AI planning problem

As stated earlier, a modular product design problem can be transformed into an AI planning problem [12]. An AI planning problem attempts to navigate through a state space, starting from an initial state with the intent of reaching a goal state(s). An AI planning problem P can be represented as $P = (\Sigma, t_0, g)$ and, $\Sigma = (T, A, \gamma)$ where T is the set of states, A is the set of actions, $\gamma : T \times A \rightarrow T$ is a state transition function, t_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 [12, 28].

Table 5-1 compares modular product design with Web service composition showing how the elements of the two problems correspond to the key terms of AI planning. In the case of modular product design, the initial state corresponds to given interfaces, such as a 110V power source. The goal state corresponds to the desired function set of the goal product, such as random access memory (RAM) of a certain size (gigabytes), and a central processing unit of a certain speed (gigahertz). The states correspond to interfaces available at the time and acquired functions during the assembly process, and the actions correspond to the assembly of components. Finally, the solution is a set of components and their assembly sequence.

Table 5-1. AI Planning Problems: Modular Product Design and Web Service Composition

AI Planning	Modular Product Design	Web Service Composition
Action	Assemble	Compose
State	Available interfaces / functions	Known information
Initial State	Given interfaces	Given (initial) information
Goal State	Desired function set of a goal product	Goal information
Solution	A set of components and their assembly sequence including branch & merge scheme	A set of Web services and their invocation sequence including branch & merge scheme

The non-redundant use of interfaces is a distinctive characteristic of modular product design based on Web service composition. In Web service composition, known information can be infinitely reusable by other Web services. On the other hand, once an available interface of a component is connected to the corresponding interface of another component, then the interface cannot be used by other components. In addition to the non-redundant use of interfaces, the proposed AI planning-based modular product design formulation has compatibility check capabilities, which were described in Section 4.4.3.

The AI planning approach to the modular product design problem offers a branching and merging scheme for product architecture and multiple input/output interfaces for components. This approach was not supported by Bryant et al. [88, 91] in a recent work on automated product concept generation, where DSM was used to verify compatibility between two adjacent components. However, DSM supports neither a branching and merging scheme, nor multiple input/output interfaces, both of which are common in product architecture. Figure 5-5, the functional model [97] of a computer system, illustrates such a case. The power supply branches into the HDD and motherboard, and the power supply and HDD subsequently merge into the motherboard. The power supply has multiple output interfaces, while the motherboard has multiple input and output interfaces.

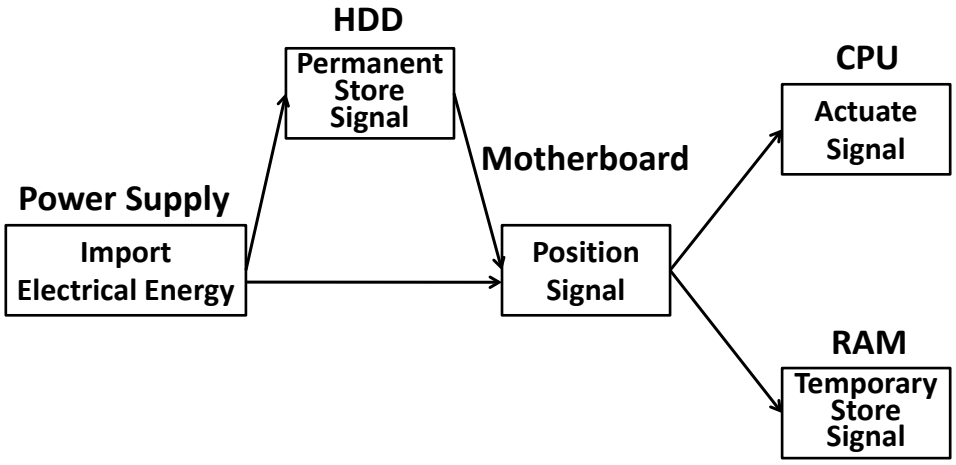


Figure 5-5. A simple functional model of a desktop PC

Functional modeling describes the functional flow of components in a product [89], and its incorporation into the AI planning formulation can help prevent the product design solution from malfunctioning. The proposed AI planning formulation, specifically Section 5.3.3, incorporates functional models into its formulation to make sure that the product design solution follows the pre-defined functional flows of a product.

The proposed AI planning approach guarantees not only that the functional requirements of a product are met, but also that the non-functional attributes, such as the size, cost, quality, or weight of a product are satisfied. Functional requirements are the most essential because products should functionally work; however, the current trend of mass customization also emphasizes the significance of non-functional attributes of products. For example, some customers may want lightweight laptops at a low price (e.g., less than four pounds for less than $500), while some other customers may want high-speed laptops, regardless of weight or price.

Few existing computational product design methods consider non-functional attributes, such as size, weight, or price. Furthermore, most AI planning approaches are logic-based, which cannot incorporate numerical constraints for non-functional attributes into their models [98]. Therefore, we propose an AI planning-based Integer Programming (IP) formulation for modular product design problems that can incorporate non-functional attributes into its mathematical model, where the optimal product design is generated based on a single objective or a pre-defined set of multiple objectives. Detailed formulations are described in the next section.

5.3.3 Integer Programming (IP) formulation

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

5.3.3.1 Domain Definition

P is the set of parts published in the proposed design repository.

I is the set of all part interfaces in P.

F is the set of all part functions in P.

M_s is the set of parts having the s^{th} function of the goal product functional model.

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

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

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

$E_{Out} \subseteq E$ 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 functions.

$E_{Goal} \subseteq E$ is the set of goal features.

$P_i^{input-consumed} \subseteq P, \forall i \in I$ is the set of parts that have interface *i* as input, consuming (using) the interface.

$P_i^{input-unconsumed} \subseteq P, \forall i \in I$ is the set of parts that have interface *i* as input where the interface remains as an output.

$P_i^{output} \subseteq P, \forall i \in I$ 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 function *f* as output.

$P_e^{output} \subseteq P, \forall e \in E$ is the set of parts that have feature *e* as output.

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

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

5.3.3.2 Variable Definition

All $p \in P$, $s \in 1,...,S$, $y_{p,s}$ 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, function, and feature usage variables*:

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

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

$$x_{i,s}^{input-unconsumed} = \left\{ \begin{array}{l} 1 \text{ if part } p \in P_i^{input-unconsumed} \text{ is used at stage } s, \\ 0 \text{ otherwise.} \end{array} \right\}$$

$$x_{i,s}^{output} = \left\{ \begin{array}{l} 1 \text{ if part } p \notin P_i^{input} \wedge p \in P_i^{output} \text{ is used at stage } s, \\ 0 \text{ otherwise.} \end{array} \right\}$$

$$x_{f,s}^{function} = \left\{ \begin{array}{l} 1 \text{ if part } p \in P_f^{output} \text{ is used at stage } s, \\ 0 \text{ otherwise.} \end{array} \right\}$$

$$x_{e,s}^{feature} = \left\{ \begin{array}{l} 1 \text{ if part } p \in P_e^{output} \text{ is used at stage } s, \\ 0 \text{ otherwise.} \end{array} \right\}$$

5.3.3.3 Formulation

(1) Objective Function

We can define any numerical expression as an objective function (including one with multiple objectives) based on target market segment characteristics (e.g., price, weight). In the case of multiple objectives, various multi-criteria optimization methodologies, such as goal programming, can be utilized. The following is a general objective function for multiple criteria:

$$Minimize \quad w_1 \sum_{p \in P} \sum_{s \in S} c_p^1 \cdot y_{p,s} + w_2 \sum_{p \in P} \sum_{s \in S} c_p^2 \cdot y_{p,s} + \ \ldots \ + w_n \sum_{p \in P} \sum_{s \in S} c_p^n \cdot y_{p,s},$$

where c_p^i is the cost of part p in terms of the i^{th} criteria and $\sum_{i=1}^{n} w_i = 1$.

(2) Initial constraints

The initial input interfaces are expressed by setting output interface usage variables to "1" at Stage 0, as shown in constraint (1). Since all the other interfaces, functions and features are not given at the initial stage, all other variables are set to "0", as in constraint (2), (3), and (4).

$x_{i,0}^{output} = 1, x_{i,0}^{input-unconsumed} = x_{i,0}^{available-unused} = x_{i,0}^{input-consumed} = 0 \ \ \forall i \in I_{Initial}$: Given interfaces at the initial stage (1)

$x_{i,0}^{input-unconsumed}, x_{i,0}^{input-consumed}, x_{i,0}^{output}, x_{i,0}^{available-unused} = 0, \forall i \notin I_{Initial}$: Other interfaces at the initial stage (2)

$x_{f,0}^{output} = x_{f,0}^{available-unused} = 0, \forall f \in F$: All functions at the initial stage (3)

$x_{e,0}^{output} = x_{e,0}^{available-unused} = 0, \forall e \in E$: All features at the initial stage (4)

(3) Goal constraints

The goal of a modular product design is represented by goal constraints. If all the goal interfaces, functions, and features are acquired at the last stage, the goal is achieved. These goal constraints for interfaces, functions, and features are shown in constraints (5), (6), and (7), respectively.

$x_{i,S}^{input-unconsumed} + x_{i,S}^{input-consumed} + x_{i,S}^{output} + x_{i,S}^{available-unused} \geq 1 \ \ \forall i \in I_{Go\ a}$: Goal interfaces at the final stage (5)

$x_{f,S}^{output} + x_{f,S}^{available-unused} \geq 1 \ \ \forall f \in F_{Goal}$: Goal functions at the final stage (6)

$x_{e,S}^{output} + x_{e,S}^{available-unused} \geq 1 \ \ \forall e \in E_{Goal}$: Goal features at the final stage (7)

The compatibility checks between adjacent components are accomplished through input/output constraints, non-concurrency constraints, and sequence constraints. These three sets of constraints ensure that adjacent components have compatible interfaces. Note that functions and features are only used as output, while interfaces are used as input as well as output.

(4) Input/output constraints

Constraints (8) through (12) verify whether the required input interfaces of a component are provided and, in turn, whether output interfaces of a component are generated. Constraint (12) specifically guarantees the non-redundant use of interfaces. The constraint makes sure that once an interface of a component is connected to a component, it cannot be connected to another component at the same stage. Constraints (13) and (14) ensure proper generation of output functions, while constraints (15) and (16) ensure that of output features.

$$\sum_{p \in P_i^{output}} y_{p,s} \geq x_{i,s}^{output} \quad \forall i \in I, s \in 1,...,S \tag{8}$$

$$y_{p,s} \leq x_{i,s}^{output} \quad \forall p \in P_i^{output}, \forall i \in I, s \in 1,...,S \tag{9}$$

$$\sum_{p \in P_i^{input-unconsumed}} y_{p,s} \geq x_{i,s}^{input-unconsumed} \quad \forall i \in I, s \in 1,...,S \tag{10}$$

$$y_{p,s} \leq x_{i,s}^{input-unconsumed} \quad \forall p \in P_i^{input-unconsumed}, \forall i \in I, s \in 1,...,S \tag{11}$$

$$\sum_{p \in P_i^{input-consumed}} y_{p,s} = x_{i,s}^{input-consumed} \quad \forall i \in I, s \in 1,...,S \tag{12}$$

$$\sum_{p \in P_f^{output}} y_{p,s} \geq x_{f,s}^{output} \quad \forall f \in F, s \in 1,...,S \tag{13}$$

$$y_{p,s} \leq x_{f,s}^{output} \quad \forall p \in P_f^{output}, \forall f \in F, s \in 1,...,S \tag{14}$$

$$\sum_{p \in P_e^{output}} y_{p,s} \geq x_{e,s}^{output} \quad \forall e \in E, s \in 1,...,S \tag{15}$$

$$y_{p,s} \leq x_{e,s}^{output} \quad \forall p \in P_e^{output}, \forall e \in E, s \in 1,...,S \tag{16}$$

(5) Non-concurrency constraints

Once it has been decided whether interface *i* is to be used as an input-unconsumed, input-consumed or output variable, (i.e., $x_{i,s}^{input-unconsumed}$, $x_{i,s}^{input-consumed}$ or $x_{i,s}^{output}$ is set to 1, respectively), $x_{i,s}^{available-unused}$ will not be set to 1. If $x_{i,s}^{available-unused}$ is equal to 1, it means that interface *i* has not been used at stage *s,* and will be saved for a future stage. On the other hand, if $x_{i,s}^{input-unconsumed}$, $x_{i,s}^{input-consumed}$ or $x_{i,s}^{output}$ is equal to 1, then it means that interface *i* has been used at stage *s*. These two cases cannot happen at the same time; so, these non-concurrency constraints are required:

$$x_{i,s}^{output} + x_{i,s}^{available\text{-}unused} + x_{i,s}^{input\text{-}consumed} \le 1 \quad \forall i \in I, s \in 1,...,S \tag{17}$$

$$x_{i,s}^{input-unconsumed} + x_{i,s}^{available\text{-}unused} + x_{i,s}^{input\text{-}consumed} \le 1 \quad \forall i \in I, s \in 1,...,S \tag{18}$$

(6) Sequence constraints

Only when interface *i* is an output of a component in previous stages can it be used as input or an available-unused variable in a later stage. Such sequential requirements are represented in the following sequence constraints:

$$\begin{aligned} & x_{i,s}^{input-unconsumed} + x_{i,s}^{available\text{-}unused} + x_{i,s}^{input\text{-}consumed} \\ & \le x_{i,s-1}^{input-unconsumed} + x_{i,s-1}^{available\text{-}unused} + x_{i,s-1}^{output} \quad \forall i \in I, s \in 1,...,S \end{aligned} \tag{19}$$

(7) Functional model constraints

The functional model of the goal product specifies required functions at each stage. The functional model constraints check whether solutions meet the pre-defined functional model of

the goal product. If the functional model is not given, then the functional model constraints should be omitted.

$$\sum_{p \in M_s} y_{p,s} \geq 1, \ s \in 1,...,S \tag{20}$$

(8) Binary variables

Here are all the variables defined in the formulation:

$$x_{i,s}^{input-unconsumed}, x_{i,s}^{input-consumed}, x_{i,s}^{output}, x_{i,s}^{available\text{-}unused} \in \{0,1\} \ \ \forall i \in I, s \in 1,...,S$$

$$x_{f,s}^{output}, x_{f,s}^{available\text{-}unused} \in \{0,1\} \ \ \forall f \in F, s \in 1,...,S$$

$$x_{e,s}^{output}, x_{e,s}^{available\text{-}unused} \in \{0,1\} \ \ \forall e \in E, s \in 1,...,S$$

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

5.3.4 SOA-based cyberinfrastructure to support global manufacturing

We propose a central repository to store information on available components from all over the world much like UDDI registers all Web services. The central repository can be uniquely built globally or locally, such as at the corporate or industry 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 work.

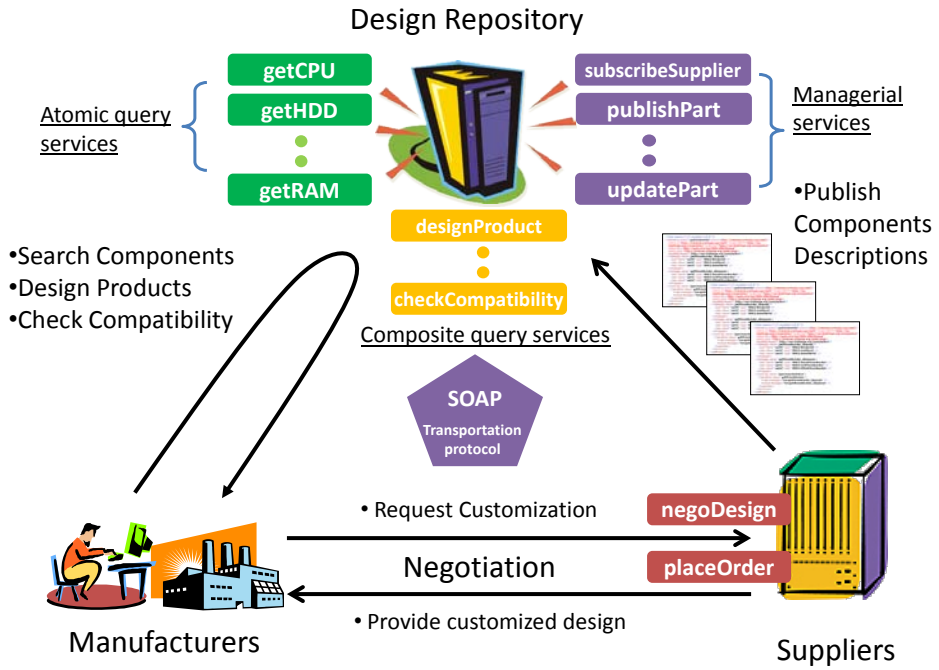


Figure 5-6. Overview of the Proposed SOA-based Cyberinfrastructure for Modular Product Design

The repository should be able to store and share the formal representations of components published by suppliers with manufacturers. In order to standardize and automate such sharing processes, the cyberinfrastructure should provide SOA-based subscribe/publish/query mechanisms. Figure 5-6 depicts the proposed SOA-based cyberinfrastructure for modular product design. The proposed cyberinfrastructure provides three types of Web services for both 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 for a type of component (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 will be able to

subscribe to the cyberinfrastructure and publish/update component information by using Web services.

Table 5-2 summarizes the differences between the proposed cyberinfrastructure for modular product design and the existing digital design repository developed by MS&T and NIST. The differences are categorized as pertaining to either content or architecture. In terms of content, the proposed cyberinfrastructure contains information on modularized components, while the current digital design repository does not consider whether or not the stored components are modularized. In addition, module representation in the proposed approach is interface-oriented, while that of the digital design repository is flow-oriented. The services provided by each are also different. The proposed cyberinfrastructure provides users with managerial, simple and composite services, while the digital design repository only provides simple query services. With respect to architecture, since the proposed cyberinfrastructure is based on SOA, it will be remotely accessible by invoking Web services, while the existing design repository is accessible only through Internet browsers. In addition, the services in the proposed cyberinfrastructure are discoverable through the standard discovery mechanism for Web services, while the services in the design repository are not discoverable.

Table 5-2. 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	Browser-based access
Discoverability	Discoverable via standard description and SOA search mechanism	N/A

5.4 Case Study

This section presents a case study demonstrating how the proposed cyberinfrastructure-based framework facilitates modular product design in a global manufacturing environment. The scenario of this case study is based on a Dell-like business model, with an Internet-based ordering process and assemble-to-order products with a variety of customization options. Additionally, we assume that any product design processes occurring after customer customization will be conducted through the proposed cyberinfrastructure-based framework. To facilitate understanding of this case study, we use a familiar product, namely, a desktop personal computer (PC) that consists of a motherboard, a hard disk drive (HDD), random access memory (RAM), a central processing unit (CPU), and a power supply.

Table 5-3 summarizes the specifications for 17 components published by international suppliers and stored in the design repository of an Internet-based assemble-to-order PC maker. In this case study, the design repository is defined to be corporate and not globally unique. To make the case study tractable, we consider five major components of a PC, each with three or four alternatives. The specifications are written in the proposed machine-readable representation scheme exemplified in Figure 5-3, and we assume these component specifications are stored in the proposed design repository on the modular product design cyberinfrastructure through managerial services, as illustrated in Figure 5-6. Authorized suppliers and manufacturers can use the proposed simple and composite query services. Security issues are implicated in this cyberinfrastructure, but are beyond the scope of this work.

The scenario begins with a customer submitting an online order form on a PC manufacturer's website, requiring the company to optimally design the product based on customer specifications. Optimal PC design is the selection of proper components and assembly sequences so as to satisfy the customer while maximizing efficiency and profit for the company.

The proposed framework provides the PC maker with a solution to this problem through a composite service illustrated in Figure 5-6. The composite service utilizes the AI planning approach to design a modularized product.

Table 5-3. Parts Used in the Case Study

Component	Function	Feature	Alternatives [**Supplier**-input:output]	Price ($)
CPU	Actuate	2.0GHz	**A**-Socket 370(*)	250
	Actuate	2.0GHz	**B**-Socket 462	300
	Actuate	3.0GHz	**C**-Socket 370	320
HDD	Store (Permanent)	160GB	**A**-16V:IDE(*)	80
	Store (Permanent)	160GB	**B**-8V:SCSI	120
	Store (Permanent)	320GB	**C**-4V:IDE	130
RAM	Store (Temporary)	2GB	**A**-DIMM	110
	Store (Temporary)	2GB	**B**-SIMM	90
	Store (Temporary)	2GB	**C**-DIMM(*)	100
Power	Import	N/A	**A**-110V:8V/16V(*)	50
	Import	N/A	**B**-110V:4V/8V	60
	Import	N/A	**C**-220V:8V/16V	70
Mother B/D	Position	DualBIOS	**A**-4V/SCSI:Socket370/DIMM	110
	Position	DualBIOS	**B**-8V/IDE:Socket370/DIMM(*)	100
	Position	DualBIOS	**C**-16V/SCSI:Socket462/SIMM	90

Suppose that a U.S. bsed customer, who uses 110V power source, wants to buy a PC at the lowest price, which corresponds to the initial constraints. The customer desires four specific PC features: a 2GHz CPU, 2GB RAM, a 160GB HDD and a dual BIOS motherboard. The online PC maker may have a functional model for the architecture of the PC, as shown in Figure 5-5, to

ensure proper functioning of the assembled PC. The features requested by the customer and the specifications in the functional model designed by the PC maker form the goal constraints.

As mentioned in Section 5.3.3, multiple objectives can be defined; however, in this case study, we use only a single objective of minimizing price. We formulated this case study using the proposed AI planning-based IP formulation introduced in Section 5.3.3. We highlight only key parts of the formulation for this particular problem.

(1) Initial constraints

Since only a 110V power supply is given as the initial condition, the following constraints are the whole initial constraints. The other variables for stage 0 are set to "0".

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

(2) Goal constraints

Goal constraints associated with the five goal functions listed in Figure 5-5 and four goal features specified above are as follows.

- Five goal functions

$$x_{Actuate,5}^{input-unconsumed} + x_{Actuate,5}^{input-consumed} + x_{Actuate,5}^{output} + x_{Actuate,5}^{available-unused} \geq 1$$

$$x_{Temporary-Store,5}^{input-unconsumed} + x_{Temporary-Store,5}^{input-consumed} + x_{Temporary-Store,5}^{output} + x_{Temporary-Store,5}^{available-unused} \geq 1$$

$$x_{Permanent-Store,5}^{input-unconsumed} + x_{Permanent-Store,5}^{input-consumed} + x_{Permanent-Store,5}^{output} + x_{Permanent-Store,5}^{available-unused} \geq 1$$

$$x_{Import,5}^{input-unconsumed} + x_{Import,5}^{input-consumed} + x_{Import,5}^{output} + x_{Import,5}^{available-unused} \geq 1$$

$$x_{Position,5}^{input-unconsumed} + x_{Position,5}^{input-consumed} + x_{Position,5}^{output} + x_{Position,5}^{available-unused} \geq 1$$

- Four goal features

$$x_{2GB-HDD,5}^{input-unconsumed} + x_{2GB-HDD,5}^{input-consumed} + x_{2GB-HDD,5}^{output} + x_{2GB-HDD,5}^{available-unused} \geq 1$$

$$x_{2GHz-CPU,5}^{input-unconsumed} + x_{2GHz-CPU,5}^{input-consumed} + x_{2GHz-CPU,5}^{output} + x_{2GHz-CPU,5}^{available-unused} \geq 1$$

$$x_{160GB-HDD,5}^{input-unconsumed} + x_{160GB-HDD,5}^{input-consumed} + x_{160GB-HDD,5}^{output} + x_{160GB-HDD,5}^{available-unused} \geq 1$$

$$x_{DualBIOS,5}^{input-unconsumed} + x_{DualBIOS,5}^{input-consumed} + x_{DualBIOS,5}^{output} + x_{DualBIOS,5}^{available-unused} \geq 1$$

(3) Input-output constraints

Input-output constraints associated with motherboard Type-B are as follows:

- Interfaces

$$y_{MotherBD-B,s} \geq x_{8V,s}^{input\_unconsumed}, y_{MotherBD-B,s} \geq x_{Socket370,s}^{input\_unconsumed},$$
$$y_{MotherBD-B,s} \geq x_{IDE,s}^{input\_unconsumed}, y_{MotherBD-B,s} \geq x_{DIMM,s}^{input\_unconsumed}$$

$$y_{MotherBD-B,s} \leq x_{8V,s}^{input\_unconsumed}, y_{MotherBD-B,s} \leq x_{Socket370,s}^{input\_unconsumed},$$
$$y_{MotherBD-B,s} \leq x_{IDE,s}^{input\_unconsumed}, y_{MotherBD-B,s} \leq x_{DIMM,s}^{input\_unconsumed}$$

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

$$y_{MotherBD-B,s} = x_{8V,s}^{input\_consumed}, y_{MotherBD-B,s} = x_{Socket370,s}^{input\_unconsumed}, y_{MotherBD-B,s} = x_{Socket370,s}^{input\_unconsumed},$$

$$y_{MotherBD-B,s} = x_{DIMM,s}^{input\_unconsumed}, \text{ for } s \in 1,...,5.$$

- Functions

$$y_{MotherBD-B,s} \geq x_{Position,s}^{output}, \; y_{MotherBD-B,s} \leq x_{Position,s}^{output}, \text{ for } s \in 1,...,5.$$

- Features

$$y_{MotherBD-B,s} \geq x_{DualBIOS,s}^{output}, \; y_{MotherBD-B,s} \leq x_{DualBIOS,s}^{output}, \text{ for } s \in 1,...,5.$$

(4) Non-concurrency constraints

For all interfaces, non-concurrency constraints should be defined. Following is the non-concurrency constraints of the interface of DIMM for RAMs:

$$x_{DIMM,s}^{output} + x_{DIMM,s}^{available\text{-}unused} + + x_{DIMM,s}^{input\text{-}consumed} \le 1, \text{ for } s \in 1,...,5$$

$$x_{DIMM,s}^{input-unconsumed} + x_{DIMM,s}^{available\text{-}unused} + x_{DIMM,s}^{input-consumed} \le 1, \text{ for } s \in 1,...,5$$

(5) Functional model constraints

Figure 5-5 illustrates the functional model of the target PC. There are five functions, and the corresponding functional model constraints are as follows:

$y_{Power-A,1} + y_{Power-B,1} + y_{Power-C,1} + y_{Power-D,1} \ge 1$: for the import function

$y_{HDD-A,2} + y_{HDD-B,2} + y_{HDD-C,2} \ge 1$: for the permanant store function

$y_{MotherBD-A,3} + y_{MotherBD-B,3} + y_{MotherBD-C,3} \ge 1$: for the position function

$y_{CPU-A,4} + y_{CPU-B,4} + y_{CPU-C,4} \ge 1$: for the actuate function

$y_{RAM-A,5} + y_{RAM-B,5} + y_{RAM-C,5} \ge 1$: for the temporary store function

(6) Sequence constraints

For all interfaces, sequence constraints are defined. Following is the sequence constraints of the IDE interface for HDDs:

$$x_{IDE,s}^{input-unconsumed} + x_{IDE,s}^{available\text{-}unused} + x_{IDE,s}^{input\text{-}consumed} \le x_{IDE,s-1}^{input-unconsumed} + x_{IDE,s-1}^{available\text{-}unused} + x_{IDE,s-1}^{output}, \text{ for } s \in 1,...,5$$

(7) The optimal solution

The optimal solution is as follows, which satisfies the entire set of specified constraints:

$y_{Power-A,1} = 1,\ y_{HDD-A,2} = 1,\ y_{MotherBD-B,3} = 1, y_{CPU-A,4} = 1$ and $y_{RAM-A,5} = 1$

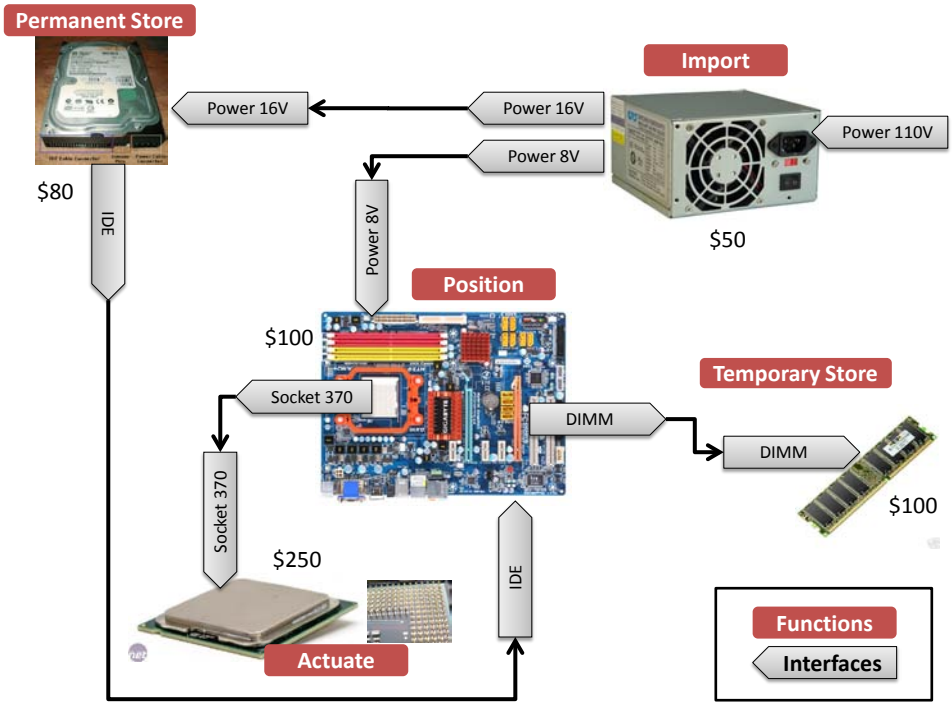


Figure 5-7. Optimal Solution from an IP-based Formulation

The components marked with a * in Table 5-3 are the ones used in the optimal design that minimize cost. Figure 5-7 shows the optimal design, which confirms that all of the compatibility requirements between adjacent components are satisfied and that all of the required functions and features are provided. For comparison, we enumerated all 243 possible combinations using brute-force search. The best four feasible solutions are listed in Table 5-4. The best solution is the same solution that the IP formulation identified without enumerating all combinations, shown in Section 5.4. Once generating the optimal solution, using the k-best solution approach identifies top four best solutions.

From the identified k-best design alternatives, we can draw three suggestions for the U.S.-based PC manufacturer and supplier-A. First, because the price difference between the first and the second best design is relatively small yet the difference between the second and the third best design is relatively large, the manufacturer may want to take only the first or second best design into consideration. Second, the best design needs three different suppliers, while the second best design only uses two. Therefore, if the decision makers prefer to manage fewer suppliers, which may reduce the indirect cost of the supplier, then they may want to choose the second best design rather than the best design. Third, the only difference between the best and the second design comes from RAM. If the price of supplier-A drops ten dollars or more, then the RAM from supplier-A becomes less expensive than those from supplier-C, and then it becomes part of the best solution. Therefore, supplier-A may want to reduce the price by more than ten dollars to gain additional businesses.

Table 5-4. Feasible Design Alternatives

Component	The Best Design	2nd Best Design	3rd Best Design	4th Best Design
CPU	**A:** Socket 370	**A:** Socket 370	**A:** Socket 370	**B:** Socket 462
HDD	**A:** IDE	**A:** IDE	**B:** SCSI	**B:** SCSI
RAM	**C:** DIMM	**A:** DIMM	**A:** DIMM	**B:** SIMM
Power	**A:** 110V	**A:** 110V	**B:** 110V	**A:** 110V
Mother B/D	**B:** DualBIOS	**B:** DualBIOS	**A:** DualBIOS	**C:** DualBIOS
Price	$580	$590	$650	$650

This case study shows how the proposed cyberinfrastructure-based framework for modular product design can be useful for online assemble-to-order PC makers. The Internet allows suppliers from all over the world to publish their components to manufacturer design repositories, allowing for maximal use of global resources. The machine-readable representation of

components facilitates the identification of proper components as well as the automation of product design. In the next chapter, we summarize and conclude this research with discussions about the contributions of this research and future work.

Chapter 6

Conclusions and Future Research Plan

The Internet has changed not only our daily lives but also business paradigms. Web services play a central role in the World Wide Web that runs on the Internet. This research has developed a solution framework for semantic Web service composition, which is one of the key features of commercial Web services. The main objectives of this research are as follows:

1) Development of a mathematical solution framework to obtain the optimal solution for Web service composition;
2) Development of semantics-processing mechanisms for Web service composition;
3) Development of k-best solution methods for Web service composition.

In this chapter, we present a summary of the research, scholarly contributions based on these objectives, and a discussion of future research topics.

6.1 Research Summary

In this research, a mathematical solution framework that guarantees the optimal solution for Web service composition was introduced. The mathematical framework considers not only functional requirements but also QoS aspects of Web service composition. Furthermore, the framework can incorporate semantics-processing mechanisms into its mathematical formulation. The proposed approach guarantees both the syntactic and semantic optimality of the composition solutions .

Finally, a k-best solution method for Web service composition was presented. The use of k-best solutions provides a holistic view of the Web service composition solution space rather than a myopic view which focuses only on the optimal solution. Knowing k-best solutions and the

summary statistics among them (such as the range of objective values) provides a broader view when composing Web services.

6.2 Contributions

6.2.1 Development of an optimal solution framework

The proposed mathematical framework for Web service composition can incorporate not only functional requirements but also QoS (quality of service) aspects into its formulation, so that the framework is guaranteed to generate a functionally-working solution that is optimal in terms of the QoS goals of Web service users, such as lowest cost or shortest time. As Web services become commercialized, the preferences of individual users will be the main objectives of Web service composition, similar to the trend of mass customization in product design. Therefore, we expect that our framework will play a significant role in providing customized Web service composition solutions to individual users in the near future.

The ability to take parameters into consideration will contribute to finding more precise composition solutions, as compared to other methods which focus at the operation level. In addition, the proposed optimal framework can significantly contribute to the identification of the best composition solution during the Web service design stage, when most composition takes place. Furthermore, the proposed optimal approach is expected to provide a guideline for evaluating the performance of heuristic approaches.

6.2.2 Consideration of semantic relationships

Semantic issues have recently become challenging aspects of Web service composition because they are ingrained in the problem. The proposed framework is capable of incorporating semantic relationships among Web service parameters into its mathematical formulation, which leads to generation of the optimal solution. The optimal solution considering semantics is superior to syntactic optimal solutions because of the broader solution space.

The major contribution of the proposed framework in terms of semantic issues is the development of a general formulation for hierarchical relationships among Web service parameters. The parameters of Web services are defined in XML and XML Schema, which follow the state-of-the-art object-oriented paradigm in which inheritance relationships are described in a hierarchical way. Therefore, the proposed framework can formulate any data structure that follows the object-oriented paradigm.

6.2.3 Generation of k-best solutions

Identifying k-best design alternatives provides a holistic view of the Web service composition solution space, rather than a myopic view that is focused on only the optimal solution. Optimal solution approaches have been criticized because they disregard other good alternatives. For example, when we search for a hotel on expedia.com, it returns a list of choices rather than offering only the best hotel. Diverse options provide not only more freedom, but also more information for users. Therefore, the proposed k-best solution methods will help Web service users make better decisions by providing more detailed information on the solution space.

6.3 Future Research

Many aspects need to be considered in order to improve Web service composition. Regarding future research topics, we suggest two interesting issues that have not been considered and two potential applications of the proposed framework in this section.

6.3.1 Interactive Web service composition

The consideration of QoS (quality of service) in this research has provided an excellent way of incorporating inputs from users into Web service composition processes. Using the proposed framework, users can define their own objectives and obtain the optimal composition solution in terms of those objectives. While the proposed framework can consider user inputs in the objective functions and constraints of the mathematical formulations, an interactive way of obtaining user inputs will help to further reflect the preferences of users by taking their immediate feedback into consideration. As Web services become commercialized, interactive Web service composition is expected to be a key feature that Web service providers should provide for users.

6.3.2 Modular product design

An application of the proposed framework to modular product design is discussed in Chapter 5. In the application, Web services and the resulting compositions are viewed as being analogous to modules and product assembly. Modules are part of a product as Web services are part of a composition solution. A key consideration for modules and Web services is whether or not they are comprised of physical objects. Chapter 5 proposes a mathematical approach to modular product design as well as a mechanism for describing modules in a machine-readable way. The approach successfully generates a functionally-working solution that is optimal in terms

of design objectives, such as total cost. However, the approach does not consider the physical aspects of modules (i.e., geometric constraints). Since both aesthetic design and size are important in product design, the consideration of geometric constraints is expected to contribute to product design practices.

6.3.3 Agent-based Web service composition

UDDI (Universal Description, Discovery, and Integration) [7] is a central registry of Web services which could be vertically categorized. For example, getHotel or getRentalCar Web services could be provided by travel service providers; getHospital or getBloodbank Web services could be provided by medical service providers. Each industry has its own business processes with specific terminology and jargon that other industries may not always understand. Multi-agent technology [99, 100] could contribute to such cases by leveraging distributed computing capabilities based on communications and intelligence. Each agent could be specialized by a company or in other ways. The specialized agents could accumulate knowledge in an assigned area and make intelligent decisions. Through communications, they could share their knowledge with other agents or relay decisions, or local optimums, to other agents. In addition, while the proposed framework can consider semantic issues, if a specialized agent for each category of Web services existed, they would be able to understand the semantics of terms frequently used in the assigned category.

6.3.4 Collaborative medical services

The healthcare industry has been one of the industries in which collaborative activities among service providers are hardly found. Healthcare Information Technology (HIT) has been

recognized as an enabler for healthcare collaboration [101]. Although HIT starts from local integration, such as the electronic medical record, computerized physician order entry, and decision support systems that integrate and improve access to health- and patient-related data [102], HIT is evolving toward global integration, called HIT Networks [103] to achieve the exchange of information among providers, insurers, and patients.

The proposed framework of this research is expected to contribute to facilitating collaborative medical services in the healthcare industry. The key connection between collaborative medical services and Web services is that medical services can be represented as Web services in machine-readable or even machine-understandable form. In this way, as in Web services that have been shared by Internet users, medical services can be shared efficiently by healthcare service users. The proposed framework, based on Service Oriented Architecture, is expected to contribute to such collaborative healthcare services.

Appendix A:

An Example of IP Formulation for Web Service Composition

The following formulation is based on the example shown in Figure 1-3.

$W = \{A, B, C, D\}$

$P = \{a, b, c, d, e, f, g, h, i, j, k, l\}$

$P_{In} = \{a, b, c, d, e, f, g, h, i, j\}$

$P_{Out} = \{f, g, h, i, j, k, l\}$

$P_{Initial} = \{a, b, c, d, e\}$

$P_{Goal} = \{f, j, l\}$

$W_a^{input} = \{A\}, W_b^{input} = \{A\}, W_c^{input} = \{B\}, W_d^{input} = \{B\}, W_e^{input} = \{B\}, W_f^{input} = \{C, D\},$

$W_g^{input} = \{C, D\}, W_h^{input} = \{C\}, W_i^{input} = \{C\}, W_j^{input} = \varnothing, W_k^{input} = \varnothing, W_l^{input} = \varnothing$

$W_p^{output} = \varnothing, W_b^{output} = \varnothing, W_c^{output} = \varnothing, W_d^{output} = \varnothing, W_e^{output} = \varnothing, W_f^{output} = \{A\}, W_g^{output} = \{B\},$

$W_h^{output} = \{B\}, W_i^{output} = \{B\}, W_j^{output} = \{D\}, W_k^{output} = \{D\}, W_l^{output} = \{C\}$

Let $S = 3$.

Total number of constraints:

$$3 \cdot |P| + |P_{Goal}| + |S| \cdot \left(|P_{In}| + \sum_p |W_p^{input}| + |P_{Out}| + \sum_p |W_p^{output}| \right) + 3 \cdot |S| \cdot |P|$$

$$= 3 \cdot 12 + 3 + 3 \cdot (10 + 12 + 7 + 7) + 3 \cdot 3 \cdot 12 = 255$$

Total number of variables: $3 \cdot |S| \cdot |P| + |S| \cdot |W| = 3 \cdot 3 \cdot 12 + 3 \cdot 4 = 120$

A.1 Objective Function

$$Minimize \sum_{w \in W} \sum_{s \in S} y_{w,s} = y_{A,1} + y_{A,2} + y_{A,3} + y_{B,1} + y_{B,2} + y_{B,3} + y_{C,1} + y_{C,2} + y_{C,3} + y_{D,1} + y_{D,2} + y_{D,3}$$

A.2 Constraints

(1) Initial conditions:

Number of constraints = $3 \cdot |P| = 3 \cdot 12 = 36$

$x_{a,0}^{output} = 1, x_{a,0}^{input} = x_{a,0}^{known\text{-}unused} = 0$, $x_{b,0}^{output} = 1, x_{b,0}^{input} = x_{b,0}^{known\text{-}unused} = 0$,

$x_{c,0}^{output} = 1, x_{c,0}^{input} = x_{c,0}^{known\text{-}unused} = 0$, $x_{d,0}^{output} = 1, x_{d,0}^{input} = x_{d,0}^{known\text{-}unused} = 0$,

$x_{e,0}^{output} = 1, x_{e,0}^{input} = x_{e,0}^{known\text{-}unused} = 0$.

$x_{f,0}^{input} = x_{f,0}^{output} = x_{f,0}^{known\text{-}unused} = 0$, $x_{g,0}^{input} = x_{g,0}^{output} = x_{g,0}^{known\text{-}unused} = 0$,

$x_{h,0}^{input} = x_{h,0}^{output} = x_{h,0}^{known\text{-}unused} = 0$

$x_{i,0}^{input} = x_{i,0}^{output} = x_{i,0}^{known\text{-}unused} = 0$, $x_{j,0}^{input} = x_{j,0}^{output} = x_{j,0}^{known\text{-}unused} = 0$,

$x_{k,0}^{input} = x_{k,0}^{output} = x_{k,0}^{known\text{-}unused} = 0$

$x_{l,0}^{input} = x_{l,0}^{output} = x_{l,0}^{known\text{-}unused} = 0$

(2) Goal conditions:

Number of constraints = $|P_{Goal}| = 3$

$$x_{f,3}^{output} + x_{f,3}^{known\text{-}unused} + x_{f,3}^{input} \geq 1,\ x_{j,3}^{output} + x_{j,3}^{known\text{-}unused} + x_{j,3}^{input} \geq 1,\ x_{l,3}^{output} + x_{l,3}^{known\text{-}unused} + x_{l,3}^{input} \geq 1$$

(3) Web services invocation constraints:

$$\text{Number of constraints} = |S|\left(|P_{In}| + \sum_{p} |W_p^{input}| + |P_{Out}| + \sum_{p} |W_p^{output}| \right) = 3(10+12+7+7) = 104$$

Stage 1:

Output : All Web services where parameter *p* is used.

$$* \ y_{A,1} \geq x_{f,1}^{output},\ y_{B,1} \geq x_{g,1}^{output},\ y_{B,1} \geq x_{h,1}^{output},\ y_{B,1} \geq x_{i,1}^{output},\ y_{C,1} \geq x_{l,1}^{output},\ y_{D,1} \geq x_{j,1}^{output},\ y_{D,1} \geq x_{k,1}^{output}$$

Output : Each Web service where parameter *p* is used.

$$* \ y_{A,1} \leq x_{f,1}^{output},\ y_{B,1} \leq x_{g,1}^{output},\ y_{B,1} \leq x_{h,1}^{output},\ y_{B,1} \leq x_{i,1}^{output},\ y_{C,1} \leq x_{l,1}^{output},\ y_{D,1} \leq x_{j,1}^{output},\ y_{D,1} \leq x_{k,1}^{output}$$

Input : All Web services where parameter *p* is used.

$$* \ y_{A,1} \geq x_{a,1}^{input},\ y_{A,1} \geq x_{b,1}^{input},\ y_{B,1} \geq x_{c,1}^{input},\ y_{B,1} \geq x_{d,1}^{input},\ y_{B,1} \geq x_{e,1}^{input},\ y_{C,1} + y_{D,1} \geq x_{f,1}^{input},$$

$$y_{C,1} + y_{D,1} \geq x_{g,1}^{input},\ y_{C,1} \geq x_{h,1}^{input},\ y_{C,1} \geq x_{i,1}^{input},\ y_{C,1} \geq x_{j,1}^{input}$$

Input : Each Web service where parameter *p* is used.

$$* \ y_{A,1} \leq x_{a,1}^{input},\ y_{A,1} \leq x_{b,1}^{input},\ y_{B,1} \leq x_{c,1}^{input},\ y_{B,1} \leq x_{d,1}^{input},\ y_{B,1} \leq x_{e,1}^{input},\ y_{C,1} \leq x_{f,1}^{input},\ y_{D,1} \leq x_{f,1}^{input}$$

$$y_{C,1} \leq x_{g,1}^{input},\ y_{D,1} \leq x_{g,1}^{input}\ \ y_{C,1} \leq x_{h,1}^{input},\ y_{C,1} \leq x_{i,1}^{input},\ y_{C,1} \leq x_{j,1}^{input}$$

Stage 2:

Output : All Web services where parameter *p* is used.

$* y_{A,2} \geq x_{f,2}^{output}, y_{B,2} \geq x_{g,2}^{output}, y_{B,2} \geq x_{h,2}^{output}, y_{B,2} \geq x_{i,2}^{output}, y_{C,2} \geq x_{l,2}^{output}, y_{D,2} \geq x_{j,2}^{output},$

$y_{D,2} \geq x_{k,2}^{output}$

Output : Each Web service where parameter *p* is used.

$* y_{A,2} \leq x_{f,2}^{output}, y_{B,2} \leq x_{g,2}^{output}, y_{B,2} \leq x_{h,2}^{output}, y_{B,2} \leq x_{i,2}^{output}, y_{C,2} \leq x_{l,2}^{output}, y_{D,2} \leq x_{j,2}^{output},$

$y_{D,2} \leq x_{k,2}^{output}$

Input : All Web services where parameter *p* is used.

$* y_{A,2} \geq x_{a,2}^{input}, y_{A,2} \geq x_{b,2}^{input}, y_{B,2} \geq x_{c,2}^{input}, y_{B,2} \geq x_{d,2}^{input}, y_{B,2} \geq x_{e,2}^{input}, y_{C,2} + y_{D,2} \geq x_{f,2}^{input},$

$y_{C,2} + y_{D,2} \geq x_{g,2}^{input}, y_{C,2} \geq x_{h,2}^{input}, y_{C,2} \geq x_{i,2}^{input}, y_{C,2} \geq x_{j,2}^{input}$

Input : each Web service that parameter p is used.

$* y_{A,2} \leq x_{a,2}^{input}, y_{A,2} \leq x_{b,2}^{input}, y_{B,2} \leq x_{c,2}^{input}, y_{B,2} \leq x_{d,2}^{input}, y_{B,2} \leq x_{e,2}^{input}, y_{C,2} \leq x_{f,2}^{input}, y_{D,2} \leq x_{f,2}^{input}$

$y_{C,2} \leq x_{g,2}^{input}, y_{D,2} \leq x_{g,2}^{input}\ y_{C,2} \leq x_{h,2}^{input}, y_{C,2} \leq x_{i,2}^{input}, y_{C,2} \leq x_{j,2}^{input}$

Stage 3 :

Output : All Web services where parameter *p* is used.

$* y_{A,3} \geq x_{f,3}^{output}, y_{B,3} \geq x_{g,3}^{output}, y_{B,3} \geq x_{h,3}^{output}, y_{B,3} \geq x_{i,3}^{output}, y_{C,3} \geq x_{l,3}^{output}, y_{D,3} \geq x_{j,3}^{output},$

$y_{D,3} \geq x_{k,3}^{output}$

Output : Each Web service where parameter *p* is used.

* $y_{A,3} \le x_{f,3}^{output}$, $y_{B,3} \le x_{g,3}^{output}$, $y_{B,3} \le x_{h,3}^{output}$, $y_{B,3} \le x_{i,3}^{output}$, $y_{C,3} \le x_{l,3}^{output}$, $y_{D,3} \le x_{j,3}^{output}$,

$y_{D,3} \le x_{k,3}^{output}$

Input : All Web services where parameter *p* is used.

* $y_{A,3} \ge x_{a,3}^{input}$, $y_{A,3} \ge x_{b,3}^{input}$, $y_{B,3} \ge x_{c,3}^{input}$, $y_{B,3} \ge x_{d,3}^{input}$, $y_{B,3} \ge x_{e,3}^{input}$, $y_{C,3} + y_{D,3} \ge x_{f,3}^{input}$,

$y_{C,3} + y_{D,3} \ge x_{g,3}^{input}$, $y_{C,3} \ge x_{h,3}^{input}$, $y_{C,3} \ge x_{i,3}^{input}$, $y_{C,3} \ge x_{j,3}^{input}$

Input : Each Web service where parameter *p* is used.

* $y_{A,3} \le x_{a,3}^{input}$, $y_{A,3} \le x_{b,3}^{input}$, $y_{B,3} \le x_{c,3}^{input}$, $y_{B,3} \le x_{d,3}^{input}$, $y_{B,3} \le x_{e,3}^{input}$, $y_{C,3} \le x_{f,3}^{input}$, $y_{D,3} \le x_{f,3}^{input}$

$y_{C,3} \le x_{g,3}^{input}$, $y_{D,3} \le x_{g,3}^{input}$ $y_{C,3} \le x_{h,3}^{input}$, $y_{C,3} \le x_{i,3}^{input}$, $y_{C,3} \le x_{j,3}^{input}$

(4) Non-concurrency constraints

Number of constraints = $2 \cdot |S| \cdot |P| = 2 \cdot 3 \cdot 12 = 72$

Stage 1:

$x_{a,1}^{output} + x_{a,1}^{known\text{-}unused} \le 1$, $x_{b,1}^{output} + x_{b,1}^{known\text{-}unused} \le 1$, $x_{c,1}^{output} + x_{c,1}^{known\text{-}unused} \le 1$,

$x_{d,1}^{output} + x_{d,1}^{known\text{-}unused} \le 1$

$x_{e,1}^{output} + x_{e,1}^{known\text{-}unused} \le 1$, $x_{f,1}^{output} + x_{f,1}^{known\text{-}unused} \le 1$, $x_{g,1}^{output} + x_{g,1}^{known\text{-}unused} \le 1$,

$x_{h,1}^{output} + x_{h,1}^{known\text{-}unused} \le 1$

$x_{i,1}^{output} + x_{i,1}^{known\text{-}unused} \le 1$, $x_{j,1}^{output} + x_{j,1}^{known\text{-}unused} \le 1$, $x_{k,1}^{output} + x_{k,1}^{known\text{-}unused} \le 1$,

$x_{l,1}^{output} + x_{l,1}^{known\text{-}unused} \le 1$

$$x_{a,1}^{input} + x_{a,1}^{known\text{-}unused} \le 1, x_{b,1}^{input} + x_{b,1}^{known\text{-}unused} \le 1, x_{c,1}^{input} + x_{c,1}^{known\text{-}unused} \le 1, x_{d,1}^{input} + x_{d,1}^{known\text{-}unused} \le 1$$

$$x_{e,1}^{input} + x_{e,1}^{known\text{-}unused} \le 1, x_{f,1}^{input} + x_{f,1}^{known\text{-}unused} \le 1, x_{g,1}^{input} + x_{g,1}^{known\text{-}unused} \le 1, x_{h,1}^{input} + x_{h,1}^{known\text{-}unused} \le 1$$

$$x_{i,1}^{input} + x_{i,1}^{known\text{-}unused} \le 1, x_{j,1}^{input} + x_{j,1}^{known\text{-}unused} \le 1, x_{k,1}^{input} + x_{k,1}^{known\text{-}unused} \le 1, x_{l,1}^{input} + x_{l,1}^{known\text{-}unused} \le 1$$

Stage 2:

$$x_{a,2}^{output} + x_{a,2}^{known\text{-}unused} \le 1, x_{b,2}^{output} + x_{b,2}^{known\text{-}unused} \le 1, x_{c,2}^{output} + x_{c,2}^{known\text{-}unused} \le 1,$$

$$x_{d,2}^{output} + x_{d,2}^{known\text{-}unused} \le 1$$

$$x_{e,2}^{output} + x_{e,2}^{known\text{-}unused} \le 1, x_{f,2}^{output} + x_{f,2}^{known\text{-}unused} \le 1, x_{g,2}^{output} + x_{g,2}^{known\text{-}unused} \le 1,$$

$$x_{h,2}^{output} + x_{h,2}^{known\text{-}unused} \le 1$$

$$x_{i,2}^{output} + x_{i,2}^{known\text{-}unused} \le 1, x_{j,2}^{output} + x_{j,2}^{known\text{-}unused} \le 1, x_{k,2}^{output} + x_{k,2}^{known\text{-}unused} \le 1,$$

$$x_{l,2}^{output} + x_{l,2}^{known\text{-}unused} \le 1$$

$$x_{a,2}^{input} + x_{a,2}^{known\text{-}unused} \le 1, x_{b,2}^{input} + x_{b,2}^{known\text{-}unused} \le 1, x_{c,2}^{input} + x_{c,2}^{known\text{-}unused} \le 1, x_{d,2}^{input} + x_{d,2}^{known\text{-}unused} \le 1$$

$$x_{e,2}^{input} + x_{e,2}^{known\text{-}unused} \le 1, x_{f,2}^{input} + x_{f,2}^{known\text{-}unused} \le 1, x_{g,2}^{input} + x_{g,2}^{known\text{-}unused} \le 1, x_{h,2}^{input} + x_{h,2}^{known\text{-}unused} \le 1$$

$$x_{i,2}^{input} + x_{i,2}^{known\text{-}unused} \le 1, x_{j,2}^{input} + x_{j,2}^{known\text{-}unused} \le 1, x_{k,2}^{input} + x_{k,2}^{known\text{-}unused} \le 1, x_{l,2}^{input} + x_{l,2}^{known\text{-}unused} \le 1$$

Stage 3:

$$x_{a,3}^{output} + x_{a,3}^{known\text{-}unused} \le 1, x_{b,3}^{output} + x_{b,3}^{known\text{-}unused} \le 1, x_{c,3}^{output} + x_{c,3}^{known\text{-}unused} \le 1,$$

$$x_{d,3}^{output} + x_{d,3}^{known\text{-}unused} \le 1$$

$$x_{e,3}^{output} + x_{e,3}^{known\text{-}unused} \le 1, x_{f,3}^{output} + x_{f,3}^{known\text{-}unused} \le 1, x_{g,3}^{output} + x_{g,3}^{known\text{-}unused} \le 1,$$

$$x_{h,3}^{output} + x_{h,3}^{known\text{-}unused} \le 1$$

$$x_{i,3}^{output} + x_{i,3}^{known\text{-}unused} \le 1, x_{j,3}^{output} + x_{j,3}^{known\text{-}unused} \le 1, x_{k,3}^{output} + x_{k,3}^{known\text{-}unused} \le 1,$$

$$x_{l,3}^{output} + x_{l,3}^{known\text{-}unused} \le 1$$

$$x_{a,3}^{input} + x_{a,3}^{known\text{-}unused} \le 1, x_{b,3}^{input} + x_{b,3}^{known\text{-}unused} \le 1, x_{c,3}^{input} + x_{c,3}^{known\text{-}unused} \le 1, x_{d,3}^{input} + x_{d,3}^{known\text{-}unused} \le 1$$

$$x_{e,3}^{input} + x_{e,3}^{known\text{-}unused} \le 1, x_{f,3}^{input} + x_{f,3}^{known\text{-}unused} \le 1, x_{g,3}^{input} + x_{g,3}^{known\text{-}unused} \le 1, x_{h,3}^{input} + x_{h,3}^{known\text{-}unused} \le 1$$

$$x_{i,3}^{input} + x_{i,3}^{known\text{-}unused} \le 1, x_{j,3}^{input} + x_{j,3}^{known\text{-}unused} \le 1, x_{k,3}^{input} + x_{k,3}^{known\text{-}unused} \le 1, x_{l,3}^{input} + x_{l,3}^{known\text{-}unused} \le 1$$

(5) Sequence constraints

Number of constraints = $|S| \cdot |P| = 3 \cdot 12 = 36$

Stage 1:

$$x_{a,1}^{input} + x_{a,1}^{known\text{-}unused} \le x_{a,0}^{input} + x_{a,0}^{output} + x_{a,0}^{known\text{-}unused},$$

$$x_{b,1}^{input} + x_{b,1}^{known\text{-}unused} \le x_{b,0}^{input} + x_{b,0}^{output} + x_{b,0}^{known\text{-}unused}$$

$$x_{c,1}^{input} + x_{c,1}^{known\text{-}unused} \le x_{c,0}^{input} + x_{c,0}^{output} + x_{c,0}^{known\text{-}unused},$$

$$x_{d,1}^{input} + x_{d,1}^{known\text{-}unused} \le x_{d,0}^{input} + x_{d,0}^{output} + x_{d,0}^{known\text{-}unused}$$

$$x_{e,1}^{input} + x_{e,1}^{known\text{-}unused} \le x_{e,0}^{input} + x_{e,0}^{output} + x_{e,0}^{known\text{-}unused},$$

$$x_{f,1}^{input} + x_{f,1}^{known\text{-}unused} \le x_{f,0}^{input} + x_{f,0}^{output} + x_{f,0}^{known\text{-}unused}$$

$$x_{g,1}^{input} + x_{g,1}^{known\text{-}unused} \le x_{g,0}^{input} + x_{g,0}^{output} + x_{g,0}^{known\text{-}unused},$$

$$x_{h,1}^{input} + x_{h,1}^{known\text{-}unused} \le x_{h,0}^{input} + x_{h,0}^{output} + x_{h,0}^{known\text{-}unused}$$

$$x_{i,1}^{input} + x_{i,1}^{known\text{-}unused} \le x_{i,0}^{input} + x_{i,0}^{output} + x_{i,0}^{known\text{-}unused},$$

$$x_{j,1}^{input} + x_{j,1}^{known\text{-}unused} \le x_{j,0}^{input} + x_{j,0}^{output} + x_{j,0}^{known\text{-}unused}$$

$$x_{k,1}^{input} + x_{k,1}^{known\text{-}unused} \le x_{k,0}^{input} + x_{k,0}^{output} + x_{k,0}^{known\text{-}unused},$$

$$x_{l,1}^{input} + x_{l,1}^{known\text{-}unused} \le x_{l,0}^{input} + x_{l,0}^{output} + x_{l,0}^{known\text{-}unused}$$

Stage 2:

$$x_{a,2}^{input} + x_{a,2}^{known\text{-}unused} \le x_{a,1}^{input} + x_{a,1}^{output} + x_{a,1}^{known\text{-}unused},$$

$$x_{b,2}^{input} + x_{b,2}^{known\text{-}unused} \le x_{b,1}^{input} + x_{b,1}^{output} + x_{b,1}^{known\text{-}unused}$$

$$x_{c,2}^{input} + x_{c,2}^{known\text{-}unused} \le x_{c,1}^{input} + x_{c,1}^{output} + x_{c,1}^{known\text{-}unused},$$

$$x_{d,2}^{input} + x_{d,2}^{known\text{-}unused} \le x_{d,1}^{input} + x_{d,1}^{output} + x_{d,1}^{known\text{-}unused}$$

$$x_{e,2}^{input} + x_{e,2}^{known\text{-}unused} \le x_{e,1}^{input} + x_{e,1}^{output} + x_{e,1}^{known\text{-}unused},$$

$$x_{f,2}^{input} + x_{f,2}^{known\text{-}unused} \le x_{f,1}^{input} + x_{f,1}^{output} + x_{f,1}^{known\text{-}unused}$$

$$x_{g,2}^{input} + x_{g,2}^{known\text{-}unused} \le x_{g,1}^{input} + x_{g,1}^{output} + x_{g,1}^{known\text{-}unused},$$

$$x_{h,2}^{input} + x_{h,2}^{known\text{-}unused} \le x_{h,1}^{input} + x_{h,1}^{output} + x_{h,1}^{known\text{-}unused}$$

$$x_{i,2}^{input} + x_{i,2}^{known\text{-}unused} \le x_{i,1}^{input} + x_{i,1}^{output} + x_{i,1}^{known\text{-}unused},$$

$$x_{j,2}^{input} + x_{j,2}^{known\text{-}unused} \le x_{j,1}^{input} + x_{j,1}^{output} + x_{j,1}^{known\text{-}unused}$$

$$x_{k,2}^{input} + x_{k,2}^{known\text{-}unused} \le x_{k,1}^{input} + x_{k,1}^{output} + x_{k,1}^{known\text{-}unused},$$

$$x_{l,2}^{input} + x_{l,2}^{known\text{-}unused} \le x_{l,1}^{input} + x_{l,1}^{output} + x_{l,1}^{known\text{-}unused}$$

Stage 3:

$$x_{a,3}^{input} + x_{a,3}^{known\text{-}unused} \le x_{a,2}^{input} + x_{a,2}^{output} + x_{a,2}^{known\text{-}unused},$$

$$x_{b,3}^{input} + x_{b,3}^{known\text{-}unused} \le x_{b,2}^{input} + x_{b,2}^{output} + x_{b,2}^{known\text{-}unused}$$

$$x_{c,3}^{input} + x_{c,3}^{known\text{-}unused} \le x_{c,2}^{input} + x_{c,2}^{output} + x_{c,2}^{known\text{-}unused},$$

$$x_{d,3}^{input} + x_{d,3}^{known\text{-}unused} \le x_{d,2}^{input} + x_{d,2}^{output} + x_{d,2}^{known\text{-}unused}$$

$$x_{e,3}^{input} + x_{e,3}^{known\text{-}unused} \le x_{e,2}^{input} + x_{e,2}^{output} + x_{e,2}^{known\text{-}unused},$$

$$x_{f,3}^{input} + x_{f,3}^{known\text{-}unused} \le x_{f,2}^{input} + x_{f,2}^{output} + x_{f,2}^{known\text{-}unused}$$

$$x_{g,3}^{input} + x_{g,3}^{known\text{-}unused} \le x_{g,2}^{input} + x_{g,2}^{output} + x_{g,2}^{known\text{-}unused},$$

$$x_{h,3}^{input} + x_{h,3}^{known\text{-}unused} \le x_{h,2}^{input} + x_{h,2}^{output} + x_{h,2}^{known\text{-}unused}$$

$$x_{i,3}^{input} + x_{i,3}^{known\text{-}unused} \le x_{i,2}^{input} + x_{i,2}^{output} + x_{i,2}^{known\text{-}unused},$$

$$x_{j,3}^{input} + x_{j,3}^{known\text{-}unused} \le x_{j,2}^{input} + x_{j,2}^{output} + x_{j,2}^{known\text{-}unused}$$

$$x_{k,3}^{input} + x_{k,3}^{known\text{-}unused} \le x_{k,2}^{input} + x_{k,2}^{output} + x_{k,2}^{known\text{-}unused},$$

$$x_{l,3}^{input} + x_{l,3}^{known\text{-}unused} \le x_{l,2}^{input} + x_{l,2}^{output} + x_{l,2}^{known\text{-}unused}$$

(6) Binary variables

Number of variables = $3 \cdot |S| \cdot |P| + |S| \cdot |W| = 3 \cdot 3 \cdot 12 + 3 \cdot 4 = 120$

$x_{p,s}^{input}, x_{p,s}^{output}, x_{p,s}^{known\text{-}unused} \in \{0,1\} \quad \forall p \in P = \{a,b,c,d,e,f,g,h,i,j,k,l\}, s \in 1,2,3$

$y_{w,s} \in \{0,1\} \quad \forall w \in W = \{A,B,C,D\}, s \in 1,2,3$.

A.3 Computation Results

CPLEX 10.1 (High Performance Group) found the optimal solution and took less than 0.01 seconds (255 constraints and 120 variables: 12 parameters and 4 Web services).

The optimal solution is: $y_{A,1}^* = y_{B,1}^* = y_{D,2}^* = y_{C,3}^* = 1$.

Appendix B:

Sample files from Web Service Challenge 2008

B.1 Input WSDL File

A sample WSDL file is below. This file includes a detailed descripton of a Web service. The file can include information of all Web services, or a file can include information of a Web service. In the latter case, there exist multiple WSDL files.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:service="http://www.ws-challenge.org/WSC08Services/" targetNamespace="http://www.ws-
challenge.org/WSC08Services/">
 <service name="serv904934656Service">
  <port binding="service:serv904934656SOAP" name="serv904934656Port">
   <soap:address location="http://www.unknownexamplehost.ukn/" />
  </port>
 </service>
 <binding name="serv904934656SOAP" type="service:serv904934656PortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="serv904934656Operation">
   <soap:operation soapAction="http://www.ws-challenge.org/serv904934656" />
   <input>
    <soap:body use="literal" />
   </input>
   <output>
    <soap:body use="literal" />
   </output>
  </operation>
 </binding>
 <portType name="serv904934656PortType">
  <operation name="serv904934656Operation">
   <input message="service:serv904934656RequestMessage" />
   <output message="service:serv904934656ResponseMessage" />
  </operation>
 </portType>
 <message name="serv904934656RequestMessage">
  <part element="service:ComplexElement0" name="ComplexElement0Part" />
  <part element="service:634753311" name="634753311Part" />
 </message>
 <message name="serv904934656ResponseMessage">
  <part element="service:ComplexElement1" name="ComplexElement1Part" />
  <part element="service:617921947" name="617921947Part" />
  <part element="service:2086384287" name="2086384287Part" />
  <part element="service:1987498920" name="1987498920Part" />
 </message>
</definitions>
```

B.2 Input OWL File

A sample OWL file [27] is below. This file includes a detailed descripton of the relationships among parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd="http://www.w3.org/2001/XMLSchema#" xmlns="http://www.ws-challenge.org/wsc08.owl#" xmlns:owl="http://www.w3.org/2002/07/owl#" xml:base="http://www.ws-challenge.org/wsc08.owl">
  <owl:Ontology rdf:about="" />
  <owl:Class rdf:ID="con1988815758" />
  <owl:Class rdf:ID="con1226699739">
    <rdfs:subClassOf rdf:resource="#con1988815758" />
  </owl:Class>
  <owl:Class rdf:ID="con445535565">
    <rdfs:subClassOf rdf:resource="#con1226699739" />
  </owl:Class>
  <owl:Class rdf:ID="con1428646343">
    <rdfs:subClassOf rdf:resource="#con1226699739" />
  </owl:Class>
  <owl:Class rdf:ID="con1653328292">
    <rdfs:subClassOf rdf:resource="#con1226699739" />
  </owl:Class>
  <owl:Class rdf:ID="con664277597">
    <rdfs:subClassOf rdf:resource="#con1226699739" />
  </owl:Class>
  <owl:Class rdf:ID="con464583682">
    <rdfs:subClassOf rdf:resource="#con1226699739" />
  </owl:Class>
  <owl:Class rdf:ID="con772420247">
    <rdfs:subClassOf rdf:resource="#con1226699739" />
  </owl:Class>
  <owl:Class rdf:ID="con1830903175">
    <rdfs:subClassOf rdf:resource="#con445535565" />
  </owl:Class>
  <owl:Class rdf:ID="con2119691623">
    <rdfs:subClassOf rdf:resource="#con1428646343" />
  </owl:Class>
  <owl:Class rdf:ID="con241744282">
    <rdfs:subClassOf rdf:resource="#con1428646343" />
  </owl:Class>
  <owl:Class rdf:ID="con848610623">
    <rdfs:subClassOf rdf:resource="#con1428646343" />
  </owl:Class>
  <owl:Class rdf:ID="con302983909">
    <rdfs:subClassOf rdf:resource="#con1428646343" />
  </owl:Class>
</rdf:RDF>
```

B.3 Input Query File

Below is a sample of query files. A query file includes a composition request. A composition request includes information of input paramters and goal parameters. This file is written in XML [17].

```
<?xml version="1.0" encoding="UTF-8" ?>
<problemStructure>
 <task>
  <provided>
   <instance name="inst472782893" />
   <instance name="inst612055407" />
   <instance name="inst28128416" />
   <instance name="inst1867369562" />
   <instance name="inst474629664" />
   <instance name="inst1729323322" />
   <instance name="inst1376632380" />
   <instance name="inst1747959169" />
   <instance name="inst767307215" />
  </provided>
  <wanted>
   <instance name="inst684358734" />
   <instance name="inst400689885" />
   <instance name="inst433458293" />
   <instance name="inst1992305510" />
  </wanted>
 </task>
<problemStructure>
```

B.4 Output WSBPEL File

Below is a sample WSBPEL file [74]. This file includes the final result of Web service composition. In other words, this file describes how to execute selected Web services in a certain sequence.

```
<?xml version="1.0" encoding="UTF-8"?>
<bpel:process xmlns:bpel="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:service="http://www.ws-challenge.org/WSC08Services/" name="WSC08" targetNamespace="http://www.ws-challenge.org/WSC08CompositionSolution/">
 <bpel:sequence name="main">
  <bpel:receive name="receiveQuery" portType="solutionProcess" variable="query" />
  <bpel:switch name="SolutionAlternatives">
   <bpel:case name="Alternative-Solution0">
    <bpel:sequence>
     <bpel:switch name="Alternative-Services">
      <bpel:case name="Execute-serv212250832Service">
       <bpel:sequence>
        <bpel:invoke name="service:serv212250832Service" portType="service:serv212250832PortType" operation="service:serv212250832Operation" />
       </bpel:sequence>
      </bpel:case>
      <bpel:case name="Execute-serv1667050675Service">
       <bpel:sequence>
        <bpel:invoke name="service:serv1667050675Service" portType="service:serv1667050675PortType" operation="service:serv1667050675Operation" />
       </bpel:sequence>
      </bpel:case>
     </bpel:switch>
     <bpel:switch name="Alternative-Services">
      <bpel:case name="Execute-serv974366889Service">
       <bpel:sequence>
        <bpel:invoke name="service:serv974366889Service" portType="service:serv974366889PortType" operation="service:serv974366889Operation" />
       </bpel:sequence>
      </bpel:case>
      <bpel:case name="Execute-serv281683065Service">
       <bpel:sequence>
        <bpel:invoke name="service:serv281683065Service" portType="service:serv281683065PortType" operation="service:serv281683065Operation" />
       </bpel:sequence>
      </bpel:case>
      <bpel:case name="Execute-serv1736482908Service">
       <bpel:sequence>
        <bpel:invoke name="service:serv1736482908Service" portType="service:serv1736482908PortType" operation="service:serv1736482908Operation" />
       </bpel:sequence>
      </bpel:case>
      <bpel:case name="Execute-serv1043799122Service">
       <bpel:sequence>
        <bpel:invoke name="service:serv1043799122Service" portType="service:serv1043799122PortType" operation="service:serv1043799122Operation" />
       </bpel:sequence>
      </bpel:case>
      <bpel:case name="Execute-serv351115298Service">
```

```
          <bpel:sequence>
            <bpel:invoke name="service:serv351115298Service" portType="service:serv351115298PortType"
operation="service:serv351115298Operation" />
          </bpel:sequence>
        </bpel:case>
        <bpel:case name="Execute-serv1805915141Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv1805915141Service" portType="service:serv1805915141PortType"
operation="service:serv1805915141Operation" />
          </bpel:sequence>
        </bpel:case>
      </bpel:switch>
      <bpel:switch name="Alternative-Services">
        <bpel:case name="Execute-serv1113231355Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv1113231355Service" portType="service:serv1113231355PortType"
operation="service:serv1113231355Operation" />
          </bpel:sequence>
        </bpel:case>
        <bpel:case name="Execute-serv420547531Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv420547531Service" portType="service:serv420547531PortType"
operation="service:serv420547531Operation" />
          </bpel:sequence>
        </bpel:case>
      </bpel:switch>
      <bpel:switch name="Alternative-Services">
        <bpel:case name="Execute-serv1875347374Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv1875347374Service" portType="service:serv1875347374PortType"
operation="service:serv1875347374Operation" />
          </bpel:sequence>
        </bpel:case>
        <bpel:case name="Execute-serv1182663588Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv1182663588Service" portType="service:serv1182663588PortType"
operation="service:serv1182663588Operation" />
          </bpel:sequence>
        </bpel:case>
        <bpel:case name="Execute-serv489979764Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv489979764Service" portType="service:serv489979764PortType"
operation="service:serv489979764Operation" />
          </bpel:sequence>
        </bpel:case>
        <bpel:case name="Execute-serv1944779607Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv1944779607Service" portType="service:serv1944779607PortType"
operation="service:serv1944779607Operation" />
          </bpel:sequence>
        </bpel:case>
      </bpel:switch>
      <bpel:switch name="Alternative-Services">
        <bpel:case name="Execute-serv1252095821Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv1252095821Service" portType="service:serv1252095821PortType"
operation="service:serv1252095821Operation" />
          </bpel:sequence>
        </bpel:case>
        <bpel:case name="Execute-serv559411997Service">
```

```
          <bpel:sequence>
            <bpel:invoke name="service:serv559411997Service" portType="service:serv559411997PortType"
operation="service:serv559411997Operation" />
          </bpel:sequence>
        </bpel:case>
      </bpel:switch>
      <bpel:invoke name="service:serv2014211840Service" portType="service:serv2014211840PortType"
operation="service:serv2014211840Operation" />
      <bpel:invoke name="service:serv1321528054Service" portType="service:serv1321528054PortType"
operation="service:serv1321528054Operation" />
      <bpel:invoke name="service:serv628844230Service" portType="service:serv628844230PortType"
operation="service:serv628844230Operation" />
      <bpel:invoke name="service:serv2083644073Service" portType="service:serv2083644073PortType"
operation="service:serv2083644073Operation" />
      <bpel:switch name="Alternative-Services">
        <bpel:case name="Execute-serv1390960287Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv1390960287Service" portType="service:serv1390960287PortType"
operation="service:serv1390960287Operation" />
          </bpel:sequence>
        </bpel:case>
        <bpel:case name="Execute-serv698276463Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv698276463Service" portType="service:serv698276463PortType"
operation="service:serv698276463Operation" />
          </bpel:sequence>
        </bpel:case>
        <bpel:case name="Execute-serv5592677Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv5592677Service" portType="service:serv5592677PortType"
operation="service:serv5592677Operation" />
          </bpel:sequence>
        </bpel:case>
        <bpel:case name="Execute-serv1460392520Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv1460392520Service" portType="service:serv1460392520PortType"
operation="service:serv1460392520Operation" />
          </bpel:sequence>
        </bpel:case>
        <bpel:case name="Execute-serv767708696Service">
          <bpel:sequence>
            <bpel:invoke name="service:serv767708696Service" portType="service:serv767708696PortType"
operation="service:serv767708696Operation" />
          </bpel:sequence>
        </bpel:case>
      </bpel:switch>
    </bpel:sequence>
  </bpel:case>
 </bpel:switch>
</bpel:sequence>
</bpel:process>
```

Bibliography

[1] Singh, M. P. and Huhns, M. N., 2005, *Service-Oriented Computing: Semantics, Processes, Agents*, England, John Wiley & Sons.

[2] Erl, T., 2004, *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Upper Saddle River, NJ, Prentice Hall.

[3] W3C, "Simple Object Access Protocol 1.2," Retrieved July 15, 2010, from http://www.w3.org/TR/soap/.

[4] XML-RPC.com, Retrieved July 15, 2010, from http://www.xmlrpc.com/.

[5] W3C, "XML Schema," Retrieved August 13, 2009, from http://www.w3.org/XML/Schema.

[6] Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S., 2001, "Web Service Description Language," Retrieved June 25, 2010, from http://www.w3.org/TR/wsdl.

[7] UDDI.XML.ORG, "Universal Description Discovery and Integration," Retrieved June 25, 2010, from uddi.xml.org.

[8] Papazoglou, M. P., 2008, *Web Services: Principles and Technology*, Prentice Hall.

[9] Oh, S.-C., Lee, D. and Kumara, S., 2007, "WSPR: an Effective and Scalable Web Service Composition Algorithm," *International Journal of Web Services Research*, 4(1).

[10] Gu, Z., Xu, B. and Li, J., 2007, "Inheritance-Aware Document-Driven Service Composition", *IEEE International Conference on E-Commerce Technology and on Enterprise Computing, E-Commerce, and E-Services*, Tokyo, Japan, pp. 513-516.

[11] Oh, S.-C., Yoo, J.-W., Kil, H., Lee, D. and Kumara, S., 2007, "Semantic Web-Service Discovery and Composition Using Flexible Parameter Matching", *IEEE International*

Conference on E-Commerce Technology and on Enterprise Computing, E-Commerce, and E-Services, Tokyo, Japan, pp. 533-536.

[12] 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.

[13] Rao, J. and Su, X., 2005, "A Survey of Automated Web Service Composition Methods", *Lecture Notes in Computer Science*, 3387, pp. 43-54.

[14] Yoo, J., Kumara, S., Lee, D. and Oh, S.-C., 2008, "A Web Service Composition Framework Based on Integer Programming with Non-Functional Objectives and Constraints", *IEEE International Conference on E-Commerce Technology and on Enterprise Computing, E-Commerce, and E-Services*, Washington, DC, pp. 347 - 350.

[15] Schach, S., 2006, *Object-Oriented and Classical Software Engineering*, McGraw-Hill.

[16] Berners-Lee, T., Hendler, J. and Lassila, O., 2001, *The Semantic Web*, Scientific American.

[17] W3C, "eXtensible Markup Language (XML)," Retrieved August 13, 2009, from http://www.w3.org/XML/.

[18] W3C, "Hypertext Markup Language (HTML)," Retrieved June 23, 2010, from http://www.w3.org/html/.

[19] W3C, "XML specification DTD," Retrieved July 15, 2010, from http://www.w3.org/XML/1998/06/xmlspec.dtd.

[20] W3C, "Resource Description Framework (RDF)," Retrieved June 23, 2010, from http://www.w3.org/RDF/.

[21] IETF, "Uniform Resource Identifiers (URI): Generic Syntax," Retrieved July 15, 2010, from http://www.ietf.org/rfc/rfc2396.txt.

[22] RFIDJournal.com, "Radio Frequency Identification," Retrieved July 15, 2010, from http://www.rfidjournal.com/.

[23] Brock, D. and Cummins, C., 2003, "EPC Tag Data Specification", *MIT Auto-ID Center White Paper*, Cambridge, Massachusetts Institute of Technology.

[24] GS1, "Global Standards One," Retrieved June 23, 2010, from http://www.gs1.org/.

[25] W3C, "RDF Schema," Retrieved June 23, 2010, from http://www.w3.org/TR/rdf-schema/.

[26] W3C, "World Wide Web Consortium (W3C)," Retrieved July 15, 2010, from http://www.w3.org/.

[27] W3C, "Web Ontology Language (OWL)," Retrieved June 23, 2010, from http://www.w3.org/TR/owl-features/.

[28] Nilsson, N. J., 1998, *Artificial Intelligence: A New Synthesis*, San Francisco, CA, Morgan Kaufmann Publishers.

[29] Bylander, T., 1994, "The Computational Complexity of Propositional STRIPS Planning," *Artificial Intelligence*, 69(1-2), pp. 165-204.

[30] Ghallab, M., Nau, D. and Traverso, P., 2004, *Automated Planning: theory and practice*, Morgan Kaufmann Publisher.

[31] Oh, S.-C., Lee, D. and Kumara, S., 2005, "A Comparative Illustration of AI Planning-based Web Services Composition," *ACM SIGecom Exchanges*, 5(5), pp. 1-10.

[32] Kil, H., Nam, W. and Lee, D., 2008, "Type-Aware Web Service Composition Using Boolean Satisfiability Solver", *IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services*, Washington D.C., pp. 331-334.

[33] McIlraith, S. and Son, T. C., 2002, "Adapting Golog for composition of Semantic Web services", *International Conference on Principles of Knowledge Representation and Reasoning*, Toulouse, France, pp. 482-493.

[34] McIlraith, S., Son, T. C. and Zeng, H., 2001, "Semantic Web Services," *IEEE Intelligent Systems*, 16(2), pp. 46-53.

[35] Narayanan, S. and McIlraith, S., 2002, "Simulation, Verification and Automated Composition of Web Service", *International World Wide Web Conference*, Honolulu, Hawaii, pp. 77-88.

[36] De Giacomo, G., Lesperance, Y. and Levesque, H. J., 2000, "ConGolog, a concurrent programming language based on the situation calculus," *Artificial Intelligence*, **121**(1-2), pp. 109-169.

[37] Levesque, H. J., Reiter, R., Lesperance, Y., Lin, F. and Scherl, R. B., 1997, "Golog: A Logic Programming Language for Dynamic Domains," *Journal of Logic Programming*, 31(1-3), pp. 59-83.

[38] Rao, J., Kungas, P. and Matskin, M., 2003, "Application of Linear Logic to Web Service Composition", *International Conference on Web Services*, Las Vegas, NV, pp. 3-9.

[39] Vossen, T., Ball, M., Lotem, A. and Nau, D., 1999, "On the Use of Integer Programming Models in AI Planning", *International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, pp. 304-309.

[40] Vossen, T., Ball, M., Lotem, A. and Nau, D., 2000, "Applying Integer Programming to AI Planning," *The Knowledge Engineering Review*, 15(1), pp. 85-100.

[41] Kautz, H. and Walser, J. P., 1999, "State-Space Planning by Integer Optimization", *The AAAI Conference on Artificial Intelligence*, Orlando, FL, pp. 526-533.

[42] Kautz, H. and Walser, J. P., 2000, "Integer Optimization Models of AI Planning Problems," *The Knowledge Engineering Review*, 15(1), pp. 101-117.

[43] Van den Briel, M. and Kambhampati, S., 2005, "Optiplan: Unifying IP-based and Graph-based Planning," *Journal of Artificial Intelligence Research*, 24(1), pp. 919-931.

[44] Van den Briel, M., Vossen, T. and Kambhampati, S., 2005, "Reviving Integer Programming Approaches for AI Planning: A Branch-and-Cut Framework", *International Conference on Automated Planning and Scheduling*, pp. 310-319.

[45] Al-Rafai, A. I., 1993, A Priori Interactive Methods for Multiple Objective Linear Programming Problems, *Ph.D. Dissertation*, Department of Industrial Engineering, The University of Oklahoma.

[46] Klein, D. and Hannan, E., 1982, "An Algorithm for the Multiple Objective Integer Linear Programming Problem," *European Journal of Operational Research*, 9(4), pp. 378-385.

[47] Berbner, R., Spahn, M., Repp, N., Heckmann, O. and Steinmetz, R., 2006, "Heuristics for QoS-aware Web Service Composition", *IEEE International Conference on Web Services*, Chicago, IL, pp. 72-82.

[48] Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J. and Chang, H., 2004, "QoS-Aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering*, 30(5), pp. 311-327.

[49] Kritikos, K. and Plexousakis, D., 2009, "Mixed-Integer Programming for QoS-Based Web Service Matchmaking," *IEEE Transactions on Services Computing*, 2(2), pp. 122-139.

[50] Ardagna, D. and Pernici, B., 2006, "Global and Local QoS Guarantee in Web Service Selection," *Lecture Notes in Computer Science*, 3812, pp. 32-46.

[51] Medjahed, B., Bouguettaya, A. and lmagarmid, A. K., 2003, "ComposingWeb services on the Semantic Web," *The VLDB Journal*, 12, pp. 333-351.

[52] Pollock, J. T. and Hodgson, R., 2004, *Adaptive Information*, Hoboken, New Jersey, Wiley.

[53] Sirin, E., Hendler, J. and Parsia, B., 2003, "Semi-Automatic Composition of Web Services Using Semantic Descriptions", *International Conference on Enterprise Information Systems* Angers, France, pp. 17-24.

[54] Aiello, M., Platzer, C., Rosenberg, F., Tran, H., Vasko, M. and Dustdar, S., 2006, "Web Service Indexing for Efficient Retrieval and Composition", *IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services*, Palo Alto, CA, pp. 63-65.

[55] Gu, Z., Li, J. and Xu, B., 2008, "Automatic Service Composition Based on Enhanced Service Dependency Graph", *IEEE International Conference on Web Services*, Beijing, China, pp. 246 - 253.

[56] Oh, S.-C., 2006, Effective Web-Service Composition in Diverse and Large-Scale Service Networks, *Ph.D. Dissertation*, Department of Industrial & Manufacturing Engineering, The Pennsylvania State University.

[57] Yan, Y., Xu, B., Gu, Z. and Luo, S., 2009, "A QoS-Driven Approach for Semantic Service Composition", *IEEE Conference on Commerce and Enterprise Computing*, Vienna, Austria, pp. 523-526

[58] Chen, Z., Wang, H. and Pan, P., 2009, "An Approach to Optimal Web Service Composition Based on QoS and User Preferences", *International Joint Conference on Artificial Intelligence*, Pasadena, CA, pp. 96-101.

[59] 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.

[60] Zhang, R. and Zhang, L. J., 2005, "Web Services Quality Testing," *International Journal of Web Services Research*, 2(2), pp. 1-4.

[61] Salkin, H. M. and Mathur, K., 1989, *Foundations of Integer Programming*, North-Holland, Elsevier.

[62] Bazaraa, M. S., Jarvis, J. J. and Sherali, H. D., 2004, *Linear Programming and Network Flows*, Hoboken, NJ, Wiley.

[63] Brucker, P. J. and Hamacher, H. W., 1989, "K-Optimal Solution Sets for Some Polynomially Solvable Scheduling Problems," *European Journal of Operational Research*, 41, pp. 194-202.

[64] Hamacher, H. W., 1985, "k-Best Solutions To Combinatorial Optimization Problems," *Annals of Operations Research*, 4(6), pp. 123-143.

[65] Hamacher, H. W., 1995, "A Note on K Best Network Flows," *Annals of Operations Research*, 57, pp. 65-72.

[66] Hamacher, H. W., Picard, J.-C. and Queyranne, M., 1984, "Ranking the Cuts and Cut-sets of a Network," *Annals of Discrete Applied Mathematics*, 19, pp. 183-200.

[67] Hamacher, H. W., Picard, J.-C. and Queyranne, M., 1984, "On Finding the K Best Cuts in a Network " *Operations Research Letters*, 2(6), pp. 303-305.

[68] Lawler, E. L., 1972, "A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem," *Management Science*, 18(7), pp. 401-405.

[69] Murty, K. G., 1968, "An Algorithm for Ranking All the Assignments in Increasing Order of Cost," *Operations Research*, 16(3), pp. 682-687.

[70] Ravindran, A., 2008, *Operations Research and Management Science Handbook*, Boca Raton, FL, CRC Press.

[71] Saaty, T. L., 1986, "Axiomatic Foundation of the Analytic Hierarchy Process," *Management Science*, 32(7), pp. 841-855.

[72] Charnesa, A. and Cooperb, W. W., 1977, "Goal Programming and Multiple Objective Optimizations: Part 1," *European Journal of Operational Research*, 1(1), pp. 39-54.

[73] WSC2008, 2008, "Web Service Challenge (WSC)," Retrieved August 20, 2009, from http://cec2008.cs.georgetown.edu/wsc08/.

[74] WSBPEL, "OASIS Web Services Business Process Execution Language," Retrieved August 20, 2009.

[75] CPLEX, Retrieved August 20, 2009, from http://www.ilog.com/products/cplex/.

[76] Eppinger, S. D. and Chitkara, A. R., 2006, "The New Practice of Global Product Development," *MIT Sloan Management Review*, 47(4), pp. 22-30.

[77] Hoffman, B. G., 2006, "Ford Retools Auto Design", *The Detroit News*. Dearborn, MI.

[78] Kotha, S., Olesen, D. G., Nolan, R. and Condit, P. M., 2005, "Boeing 787: Dreamliner," *Harvard Business School Case Study*.

[79] 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.

[80] Szykman, S., 2002, "Architecture and Implementation of a Design Repository System", *ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, ASME DETC2002/CIE-34463, Montreal, Canada.

[81] 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*, DETC1999/DTM-8742, Las Vegas, NV.

[82] 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*, DETC1999/CIE-9025, Las Vegas, NV.

[83] 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.

[84] 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.

[85] Bohm, M. R., Stone, R. B., Simpson, T. W. and Steva, E. D., 2008, "Introduction of a Data Schema to Support a Design Repository," *Computer-Aided Design*, 40(7), pp. 801-811.

[86] Bohm, M. R. and Stone, R. B., 2004, "Product Design Support: Exploring a Design Repository System", *ASME International Mechanical Engineering Congress & Exposition*, IMECE2004-61746, Anaheim, CA.

[87] Bohm, M. R., Stone, R. B. and Szykman, S., 2005, "Enhancing Virtual Product Representations for Advanced Design Repository Systems," *ASME Journal of Computing and Information Science in Engineering*, 5(4), pp. 360-372.

[88] 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*, ASME DETC2005/CIE-85323, Long Beach, CA.

[89] Hirtz, J., Stone, R. B., 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.

[90] Stone, R. B. and Wood, K., 2000, "Development of a Functional Basis for Design," *ASME Journal of Mechanical Design*, 122(4), pp. 359-370.

[91] 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*, DETC2006-99489, Philadelphia, PA.

[92] Campbell, M. I., Cagan, J. and Kotovsky, K., 2000, "Agent-Based Synthesis of Electromechanical Design Configurations," *ASME Journal of Mechanical Design*, 122(1), pp. 61-69.

[93] Campbell, M. I., Cagan, J. and Kotovsky, K., 2003, "The A-Design Approach to Managing Automated Design Synthesis," *Research in Engineering Design*, 14(1), pp. 12-24.

[94] Mittal, S., Dym, C. and Morjara, M., 1986, "PRIDE: An Expert System for the Design of Paper Handling Systems," *Computer*, 19(7), pp. 102-114.

[95] Navinchandra, D., Sycara, K. P. and Narasimhan, S., 1991, "A Transformational Approach to Case-Based Synthesis," *AIEDAM*, 5, pp. 31-45.

[96] Titus, N. and Ramani, K., 2005, "Design Space Exploration Using Constraint Satisfaction", *International Joint Conferences on Artificial Intelligence*, Edinburgh, Scotland, pp. 31-37.

[97] Pahl, G. and Beitz, W., 1996, *Engineering Design: A Systematic Approach*, London, Springer-Verlag.

[98] Yoo, J., Oh, S.-C. and Kumara, S., 2008, "Application of Semantic Web Technology to Scheduling Interoperability", *Industrial Engineering Research Conference*, Vancouver, Canada.

[99] Wooldridge, M., 2002, *An Introduction to MultiAgent Systems*, West Sussex, UK, John Wiley & Sons.

[100] Jennings, N. R. and Wooldridge, M., 1998, *Agent Technology: Foundations, Applications, and Markets*, Springer.

[101] Cohn, K. H., Berman, J., Chaiken, B., Green, D., Green, M., Morrison, D. and Scherger, J. E., 2009, "Engaging Physicians to Adopt Healthcare Information Technology," *Journal of Healthcare Management*, 54(5), pp. 291-300.

[102] Cohn, K. H., 2009, "Changing Physician Behavior Through Involvement and Collaboration," *Journal of Healthcare Management*, 54(2), pp. 80-86.

[103] Orszag, P. R., 2008, "Evidence on the Costs and Benefits of Health Information Technology," Retrieved July 24, 2010, from http://www.cbo.gov/ftpdocs/91xx/doc9168/HealthITTOC.2.1.htm.

VITA

John Jung-Woon Yoo

John Jung-Woon Yoo was born in Seoul, Korea on September 23, 1972. He grew up in the northwestern part of Seoul, Eun-Pyung-Gu Yuk-Chon-Dong, where he attended Yuk-Chon Elementary School, Dae-Sung Middle School, and Chung-Ahm High School. After high school, he attended Korea University on a full scholarship, earning his Bachelor of Science degree in Industrial Engineering in 1996. He earned his Master of Science degree in Industrial Engineering in 1998 from Seoul National University. Before joining Pennsylvania State University, University Park in State College, Pennsylvania for his Ph.D. degree in 2006, he spent six-and-a-half years working professionally for ETRI (Electronics Telecommunications Research Institute) in Daejeon, Korea (the largest national laboratory), and MJL Technology Ltd. in Seoul, Korea as a software developer and designer. Throughout his Ph.D. program, he taught Penn State undergraduate students as an instructor and worked as a teaching assistant for the Department of Industrial & Manufacturing Engineering and as a research assistant for the Department of Information Technology Services. Upon graduation in 2010, he accepted a tenure-track Assistant Professor position in the Department of Industrial and Manufacturing Engineering & Technology at Bradley University in Peoria, Illinois and started a new journey as a researcher and an educator.