LUMINA: USING WSDL-S FOR WEB SERVICE DISCOVERY

by

KE LI

(Under the Direction of John A. Miller)

ABSTRACT

Many businesses are adopting Web Service technologies to provide greater access to their applications. Due to the fast-paced E-Commerce requirements, more and more businesses prefer to only creating their core applications, and outsourcing the non-critical applications, or making use of their partners' applications directly. There is a growing requirement to build complex processes which may include Web Services supplied by the different partners. However, there are two main difficulties in building such Web Processes - 1) the current syntactic search mechanism is ineffective to find out the highly suitable services and 2) there are not many Process designing tools which allow dynamic binding of partner services. In this chapter, we present a solution for both the problems based on WSDL-S, and obtain a satisfying result based on our evaluation.

INDEX WORDS: Semantic Web Service, WSDL-S, Ontology, UDDI, Business Process, Semantic Template

LUMINA: USING WSDL-S FOR WEB SERVICE DISCOVERY

by

KE LI

B.E., Nanjing University of Science & Technology, Nanjing, China, July 2000

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment

of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2005

LUMINA: USING WSDL-S FOR WEB SERVICE DISCOVERY

by

KE LI

Major Professor:	John A. Miller
Committee:	Amit P. Sheth Elieen T. Kraemer

DEDICATION

To my husband Dongtao Jiang

To Baba and Mama

ACKNOWLEDGEMENTS

I am so appreciated for my major advisor, Dr. John. A. Miller, for his guidance and help for my thesis research. Especially, when my thesis research came into the hardest time, Dr. Miller spent a lot time and effort to enable me to finish it on time. I would like to thank Dr. Amit. P. Sheth for his valuable suggestions for my research and assurance in difficult times. Thanks so much to Dr. Eileen T. Kraemer for her encouragement and understanding when I was in my tension-filled days. Special thanks to Kunal Verma, for his help and suggestions in this work, and encouragement during my hardest days. I am thankful to Doug Brower and Zixin Wu for their great cooperation during my research time. As a member of LSDIS, I would also like to acknowledge the help and encouragement given by my fellow-students in METEOR-S group: Cary Pennington, Scott Patterson. I cannot find a word to present my appreciation to Meenakshi Nagaraian. She helped me rehearse my presentation before defense, gave me a lot of valuable suggestions and encouragement. Finally, I sincerely thanks to my parents and my husband for their continuous support and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Web Service technology is adopted by many businesses because it enables them to have business collaboration, both within their organization and with business partners outside. Web Services, based on the SOA (Service Oriented Architecture), are inherently designed for interaction in a loosely coupled environment, and hence are an ideal choice for companies seeking inter or intra business interactions that span heterogeneous platforms and systems. Wide acceptance of Web Services is largely due to the fact that they are built on XML based standards same as SOAP, WSDL and UDDI. The simple Object Access Protocol (SOAP) [44] is a lightweight protocol for exchange of information among Web Services. The Web Service Description Language (WSDL) [13] describes a Web service, including an interface. The Universal Description, Discovery and Integration (UDDI) [1] protocol supports service specialized repositories that enable service providers to advertise Web Services, and service requesters to find and locate Web Services over the Internet. The need has risen to interconnect the various services offered by different businesses to create a complex business process or workflow that spans wider boundaries than ever before.

The idea of building complex business processes has brought importance for tools that can model a business process. Though languages like WS-BPEL (Web Service Business Process Execution Language) offer solutions for integrating Web Services into a business process, they are difficult to learn, involve understanding detailed language syntax and are only part of the solution for designing fully-functional Web processes. Tools that aid developers in easily building Web processes are becoming available; however they do not alleviate several inherent limitations of current Web service technologies that make developing a process overly complicated or error-prone. Moreover, a tool does not exist that introduces semantics to the process design procedure.

A lightweight extension to the WSDL specification – WSDL-S [21] has been jointly proposed by the LSDIS lab and IBM research. The WSDL-S specification allows annotating services and operations by mapping certain key elements to ontological concepts. These enhanced semantic services are published to a UDDI registry and can be dynamically discovered using some ontological concepts. This work enables users to find the highly suitable and appropriate partner services in much less time than manually discovering them using a non-semantic, syntactic based UDDI searching mechanism. It also features the ability of partner services to be bound either at design-time, deployment-time or execution-time (run-time) to further optimize the process and add greater dynamism, as opposed to static-binding of partners.

In this work, we present a solution for designing Semantic Web Processes via WSDL-S. Three tools used to accomplish this project:

Radiant: Annotate the WSDL file using the ontological concepts and publish it to a UDDI registry.

Lumina: Discover the services with the required ontological concepts.

Saros: Design the Semantic Web Service Process.

My main contributions in this work are the following:

1. Design and develop the WSDL-S based semantic discovery tool called Lumina
2. Develop the API for publishing WSDL / WSDL-S files to our enhanced UDDI registry.
3. Design the Semantic Template generator which can be used within Saros.

Radiant is designed and developed by Doug Brewer and Saros is designed and developed by Ranjit Mulye. They are both METEOR-S group members.

CHAPTER 2

BACKGROUND

2.1 Two Problems for Non-Semantic Web Services

During the Web Service Process design procedure, partner services discovery is an important issue. UDDI (Universal Description, Discovery and Integration) plays an important role in the whole Web Service lifecycle. It is a bridge to link service providers with service consumers. UDDI versions 2 & 3 include three parts: Business Entity, Business Service and Technology Model (TModel). Business Entity is used to describe the service provider's information, such as the company's introduction, contact information; Business Service is used for publish a service's information; and TModel is used for giving the interface information of the related service or some detailed information within the service. Service providers use the three elements to describe their service and company information. Consider a scenario in which a process designer wants to create a process by invoking several successive operations. These successive operations may or may not exist in the same service. The designer searches all the suitable services that are described via WSDL in some UDDI registries. It is obvious that this task is hard to accomplish because of two problems: 1.There is no unified method to publish a WSDL file to UDDI, nor there is a unified way to discover a WSDL file; 2. Current syntactic searching mechanisms of UDDI can not handle the complexities of checking for data mapping between outputs and inputs among successive operations that belong to different services.

Much research has been done for the first problem. Best Practice Technical Note [55] proposed a method to map WSDL to UDDI. The authors compared the document types of

WSDL (see Figure 2-1) and the data types of UDDI (see Figure 2-2). Then they gave a solution to map WSDL to UDDI (see Figure 2-3).

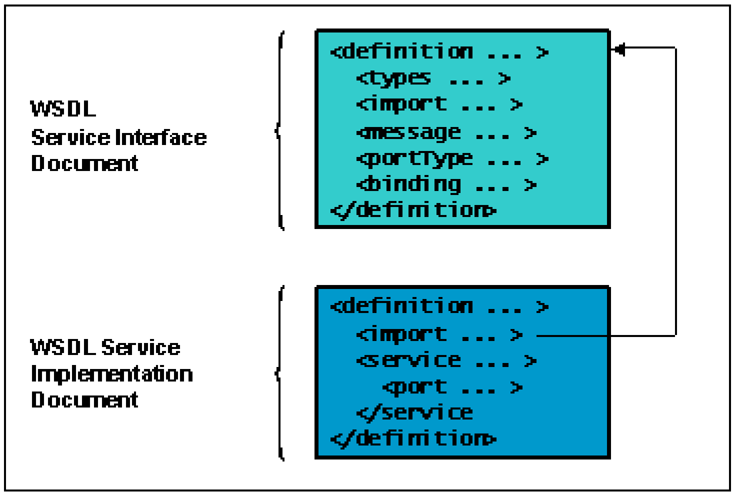


Figure 2-1: Main Elements in WSDL [55]

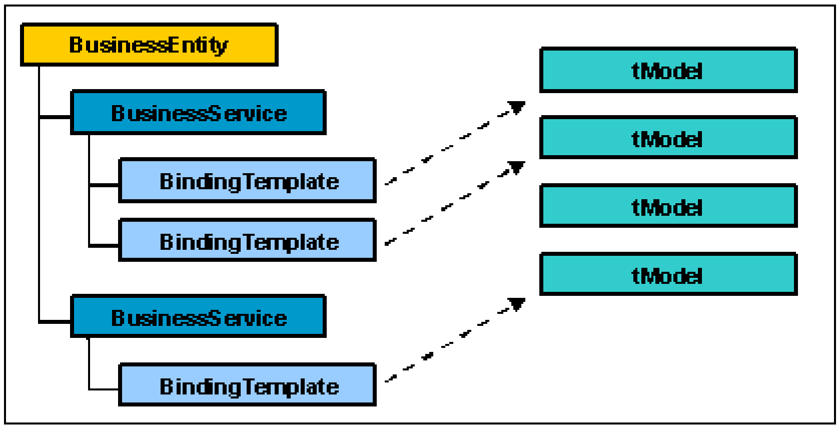


Figure2-2: Main Three Elements in UDDI Data Structure [55]

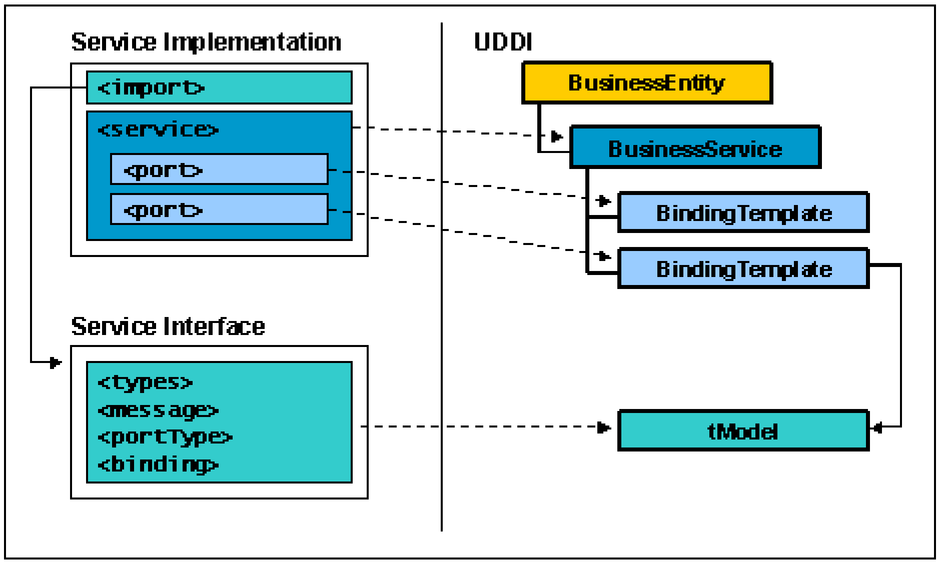


Figure 2-3: WSDL mapping to UDDI [55]

Unfortunately, mapping the whole service interface to a TModel makes it difficult for users to discover a specific operation based on its functionalities.

The second problem mentioned above is the syntactic searching mechanism of UDDI. Currently, most service providers publish their services by supplying the information of service name, description and WSDL file location. More often than not, due to words taking different meanings in different contexts, it becomes difficult to search for a service that provides a desired operation just by searching on the service name and description. Moreover, the syntactic matching mechanism is not enough to solve the data mapping between outputs and inputs among successive services.

These problems can be solved if service properties are described in a proper manner. Considerable work is being applied to utilizing Semantic Web [12] technology for Web Services [16, 19].

2.2 Semantic Web Service

Several academic and industrial organizations are contributing to introduce Semantic Web technology to Web Services and to build suitable frameworks for describing Web Services. The development of a new generation of Web Markup Languages such as RDF (Resource Description Framework) [57], OWL [56] are a great improvement for the Web that enables professionals to create domain ontologies and enables Web Service developers to apply these ontologies to describe their services. To achieve the goal of enabling the users to discover, select, locate, invoke, compose and monitor the Web Services dynamically and automatically, there are four proposals: OWL-S [52], SWSO [53], WSMO [54], and WSDL-S [21]. We briefly review each in the subsequent sections.

2.2.1 OWL-S

OWL-S describes the Web Service based on OWL Description Logic. It is structured by an upper ontology that includes three main classes: Service Profile, Process Model and Service Grounding, shown in Figure 2-4.

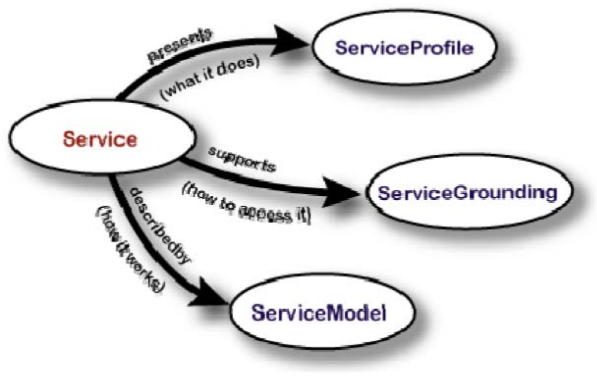


Figure 2-4: OWL-S's Upper Ontology [52]

The Service Profile describes all the information of an “atomic service”. It includes the service provider’s information, functionalities of this service and some non-functional features such as quality rating, reliability, and service category. The specification of service functionalities uses IOPEs (Input, Output, Precondition, and Effect). From the properties that are used to describe a service provider’s information, it is obvious that these properties can be matched to a Business Entity of UDDI.

The Process Model is for real service invocation, composition and data flow interaction, while the Service Profile is for service advertising and discovery. OWL-S separates the process into three types: atomic service, composite process and simple process. Each type of process has the properties described in IOPEs and participants. The instance of IOPE in Service Profile can be pointed to the IOPE instance in Process Model. The controlling abilities of Process Model are very similar to the structure activities supplied by WS-BPEL (Web Service Business Process Exception Language). Moreover, the composite process and the simple process in Process Model have the same meaning as the executed BPEL process and the abstract BPEL process, respectively.

The Service Grounding can map Process Model to any Web Service description language, such as WSDL.

2.2.2 SWSO (Semantic Web Service Ontology)

SWSO supplies two languages: SWSO-FOL (FLOWS) and SWSO-Rules (ROWS). FLOWS (First-order Logic Ontology for Web Service) is developed based on OWL-S. It also includes three parts: Service Descriptor, Process Model and Grounding. The main differences between FLOWS and OWL-S are: 1). FLOWS uses First-order Logic, which enables the representation of different forms of data flow in a composite process where OWL-S uses OWL.

2). FLOWS supplies more description aspects for Process Model than OWL-S. For example, FLOWS provides the following new modules which are not in OWL-S: Ordering Constraints, Occurrence Constraints, State Constraints and Exception Constraints.

2.2.3 WSMO (Web Service Modeling Ontology)

Web Service Modeling Ontology (WSMO) provides a framework (WSMF) and a formal language (WSML) to semantically describe the Web Services. WSMO includes four top level elements: Ontologies, Web Service Descriptions, Goals, and Mediator. See Figure 2-5.

Ontologies are used to provide terminology to describe other WSMO elements.

Web Service Descriptions provide a Web Service's functionality information in IOPEs format and non-functionality information, such as Accuracy, QoS (Quanlity of Service), Security, etc.

Goals are used to advertise Web Services and provide the information that clients may need and be interested in.

Mediator is the core element in WSMO. It solves the interoperability problems between different WSMO elements.

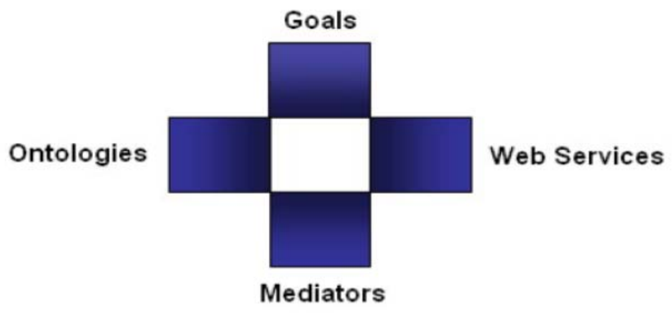


Figure 2-5: Top Level Elements of WSMO [53]

WSMO uses the Goal and the Web Service Description together to support the service advertisement and discovery. A Goal supplies the information from the requester's viewpoint, while a Service Description is used by the service provider to supply the service's capabilities.

The advantage of the separation between the requester's viewpoint and the provider's viewpoint is that the requester's requirements may be comparatively consistent in a time period, while the real implementation of the internal service may vary. The separation may allow the provider to keep this variation in implementation private.

2.2.4 WSDL-S

WSDL-S is jointly proposed by the METEOR-S group at the University of Georgia and IBM Research. Compared with OWL-S and WSMO, WSDL-S is a lightweight extension of current WSDL. WSDL 2.0 has the following constructs to represent a service: interface, message, operation, binding, service and endpoint. Interface, message and operation constructs are used to describe a service abstractly. The other three constructs deal with the implementation, access protocol, and Uniform Resource Locator (URL) of the service. WSDL-S focuses on semantically annotating the abstract definition of a service by using three extensibility elements and two extensibility attributes of WSDL. The two elements are **precondition** and **effect**, which are specified as child elements of the *operation* element. The attributes include: 1. **modelReference**, used to map a WSDL entity (complex type, element, operation, as well as extension elements – precondition and effect) to a concept in some semantic model; 2. **schemaMapping**, added to XSD complex type and elements, for associating the schema elements of a Web Service with semantic models; 3. **category**, used to describe the service categorization information when the service is published to a UDDI registry.

CHAPTER 3

THE METEOR-S SEMANTIC DISCOVERY TOOL – LUMINA

This chapter presents the METEOR-S semantic discovery tool – Lumina – which supports the discovery of Web Services by following the WSDL-S approach via a Graphical User Interface (GUI).

3. 1 Lumina's Functionality

Lumina supplies three discovery models: 1). General UDDI Discovery; 2). WSDL-S Discovery; 3). WSDL Discovery.

3.1.1 General UDDI Discovery

There are five main UDDI Universal Business registries (UBRs): Microsoft, IBM, HP, XMethods and Systinet (See Appendix A). XMethods is a private UDDI registry, which does not supply a Web interface to the service requesters. Each of the other four registries supplies a public Web browser to enable the business, service and TModel search functionalities. Though they are based the same discovery API, - UDDI4J, their search styles are not unified. This is not convenient for the service requesters, because they need to adapt to each registry's search style. Lumina supplies a unified general UDDI search style and allows the service requester to add any UDDI registry information to the registry list. This registry information list can be obtained either from Radiant or from Lumina. UDDI includes three elements: Business Entity (white pages), Business Service (yellow pages) and Technical Model (green pages). Lumina's general UDDI discovery is divided to these three parts correspondingly. The requester can input the following search requirements when discovering each element:

- Business Entity:

    - Business Name, Discovery URL, Categories, TModel Keys

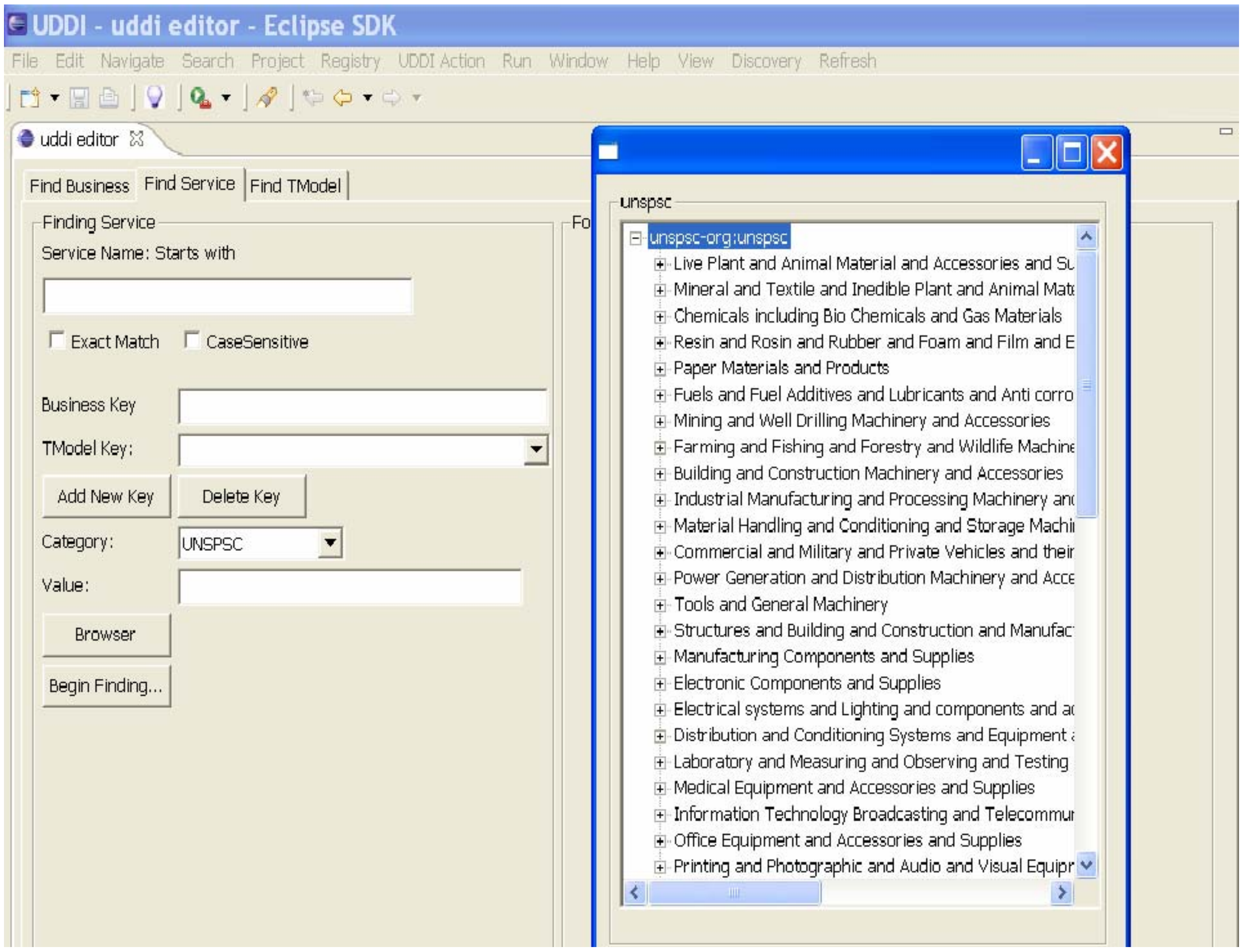
- Business Service:

    - Service Name, Categories, Business Key, TModel Keys

- TModel:

    - TModel Name

When entering the Category information, we supply a taxonomy browser to help the service requester. There are three taxonomies used in our tool: NAICS-1997 [58], ISO3166 [59] and UNSPSC. Figure 3-1 shows the UNSPSC [60] taxonomy.



UDDI - uddi editor - Eclipse SDK
File Edit Navigate Search Project Registry UDDI Action Run Window Help View Discovery Refresh
uddi editor
Find Business
Find Service
Find TModel
Finding Service
Service Name: Starts with
Exact Match
CaseSensitive
Business Key
TModel Key:
Add New Key
Delete Key
Category:
UNSPSC
Value:
Browser
Begin Finding...
Fo
unspsc
unspsc-org:unspsc
Live Plant and Animal Material and Accessories and Su
Mineral and Textile and Inedible Plant and Animal Mat
Chemicals including Bio Chemicals and Gas Materials
Resin and Rosin and Rubber and Foam and Film and E
Paper Materials and Products
Fuels and Fuel Additives and Lubricants and Anti corro
Mining and Well Drilling Machinery and Accessories
Farming and Fishing and Forestry and Wildlife Machine
Building and Construction Machinery and Accessories
Industrial Manufacturing and Processing Machinery and
Material Handling and Conditioning and Storage Machi
Commercial and Military and Private Vehicles and their
Power Generation and Distribution Machinery and Acce
Tools and General Machinery
Structures and Building and Construction and Manufac
Manufacturing Components and Supplies
Electronic Components and Supplies
Electrical systems and Lighting and components and a
Distribution and Conditioning Systems and Equipment
Laboratory and Measuring and Observing and Testing
Medical Equipment and Accessories and Supplies
Information Technology Broadcasting and Telecommu
Office Equipment and Accessories and Supplies
Printing and Photographic and Audio and Visual Equip

Figure 3-1: Category Browser in General UDDI Discovery

3.1.2 WSDL-S Discovery

Based upon the WSDL-S to UDDI mapping structure, Lumina supplies the WSDL-S discovery mode which allows the service request to use ontological concepts to annotate a list of operation functionalities, inputs and outputs. These ontological concepts can either be entered by the requester, or dragged and dropped from the Ontology Navigator in Radiant.

After discovery, the result will be shown in the result panel. The result gives not only the operation's information, but also the related service and the service provider's information. We give the links to get the service and service provider's information.

3.1.3 WSDL Discovery

With the same mapping structure, a WSDL file is mapped to our enhanced UDDI registry. The only difference from WSDL-S discovery is that the requester needs to input the exact operation name, input variables and output variables to proceed with the discovery.

3.2 Architecture of Lumina

Lumina's model part depends on the METEOR-S Discovery API, WSDLS4J and IBM UDDI4J [61]. The METEOR-S Discovery API's class diagram is shown as Figure 3-2. Lumina adopts the Eclipse Plugin techniques to build its GUI: Action, Editor, View, Perspective. The architecture of Lumina is shown in Figure 3-3 and Figure 3-4.

The Lumina's running environment is listed as the following:

- Web Server: Tomcat 5.0.30
- UDDI Registry Implementation: JUDDI 0.9
- Registry Database: MySQL-4.1.12-win32
- Java Compile Environment: JDK 1.5

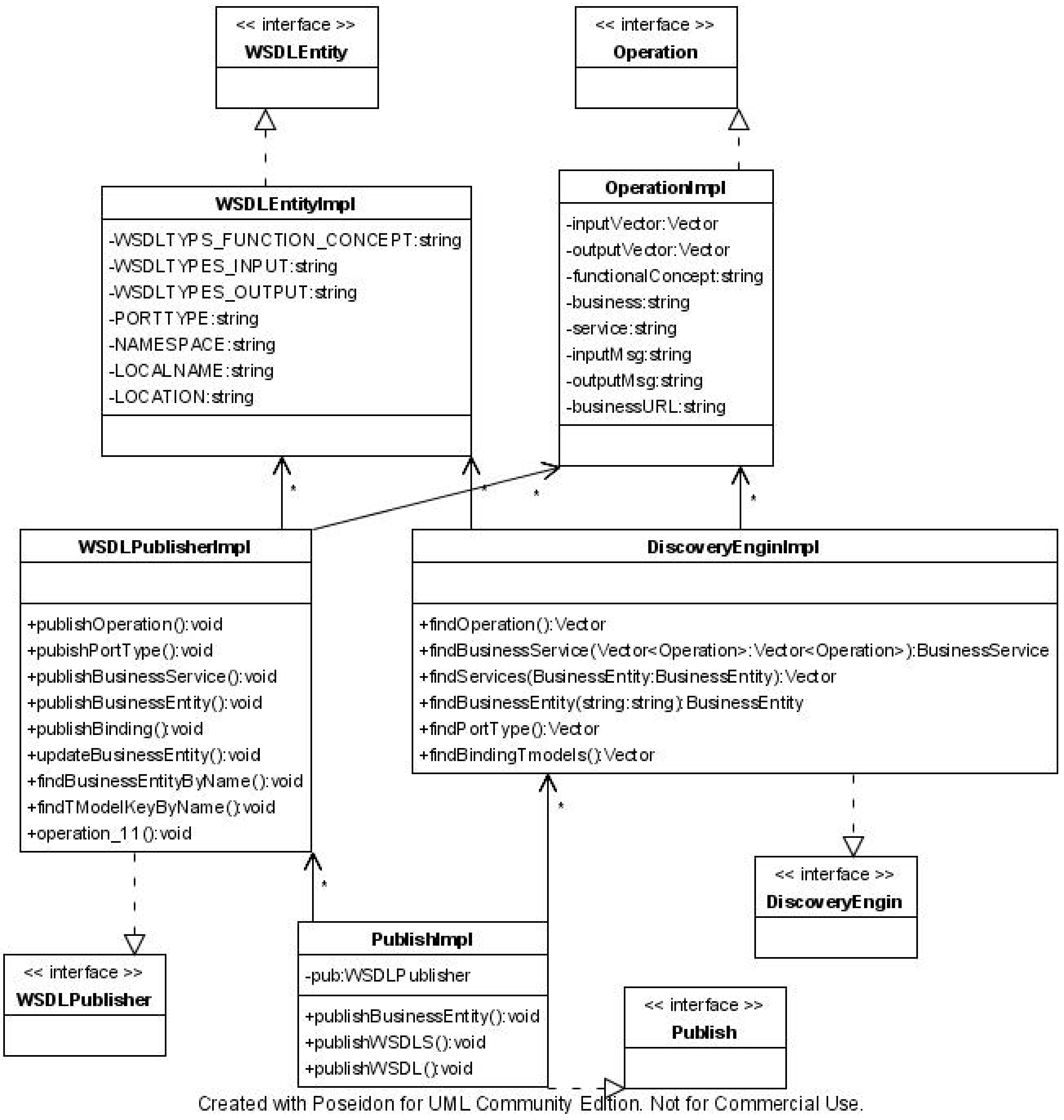


Figure 3-2: UML Class Diagram for METEOR-S Discovery Model

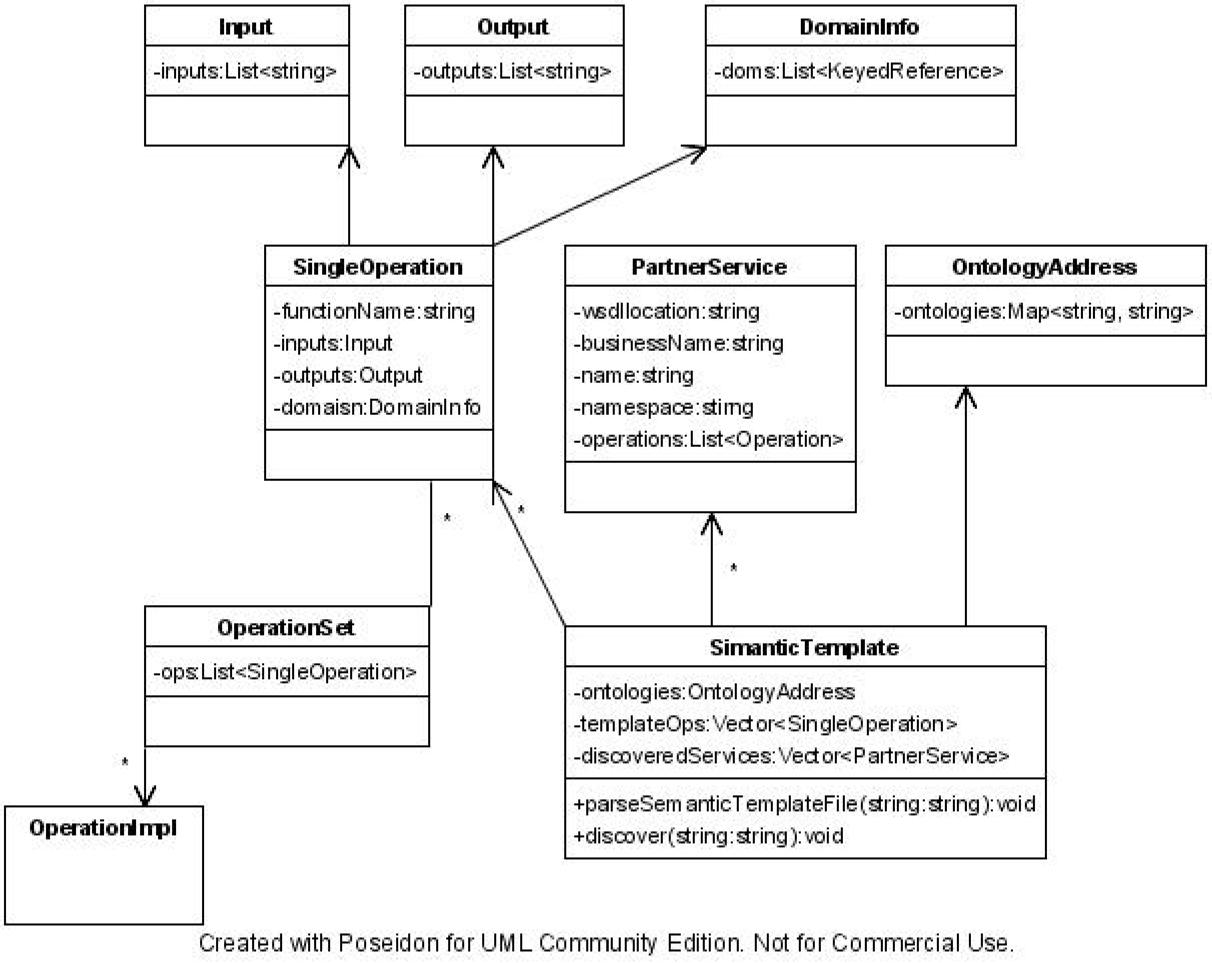

Input
-inputs:List<string>
Output
-outputs:List<string>
DomainInfo
-doms:List<KeyedReference>
SingleOperation
-functionName:string
-inputs:Input
-outputs:Output
-domaisn:DomainInfo
PartnerService
-wsdllocation:string
-businessName:string
-name:string
-namespace:stirng
-operations:List<Operation>
OntologyAddress
-ontologies:Map<string, string>
OperationSet
-ops:List<SingleOperation>
OperationImpl
SimanticTemplate
-ontologies:OntologyAddress
-templateOps:Vector<SingleOperation>
-discoveredServices:Vector<PartnerService>
+parseSemanticTemplateFile(string:string):void
+discover(string:string):void
*
*
*
*
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figure 3-3: UML Class Diagram for Lumina Model Part

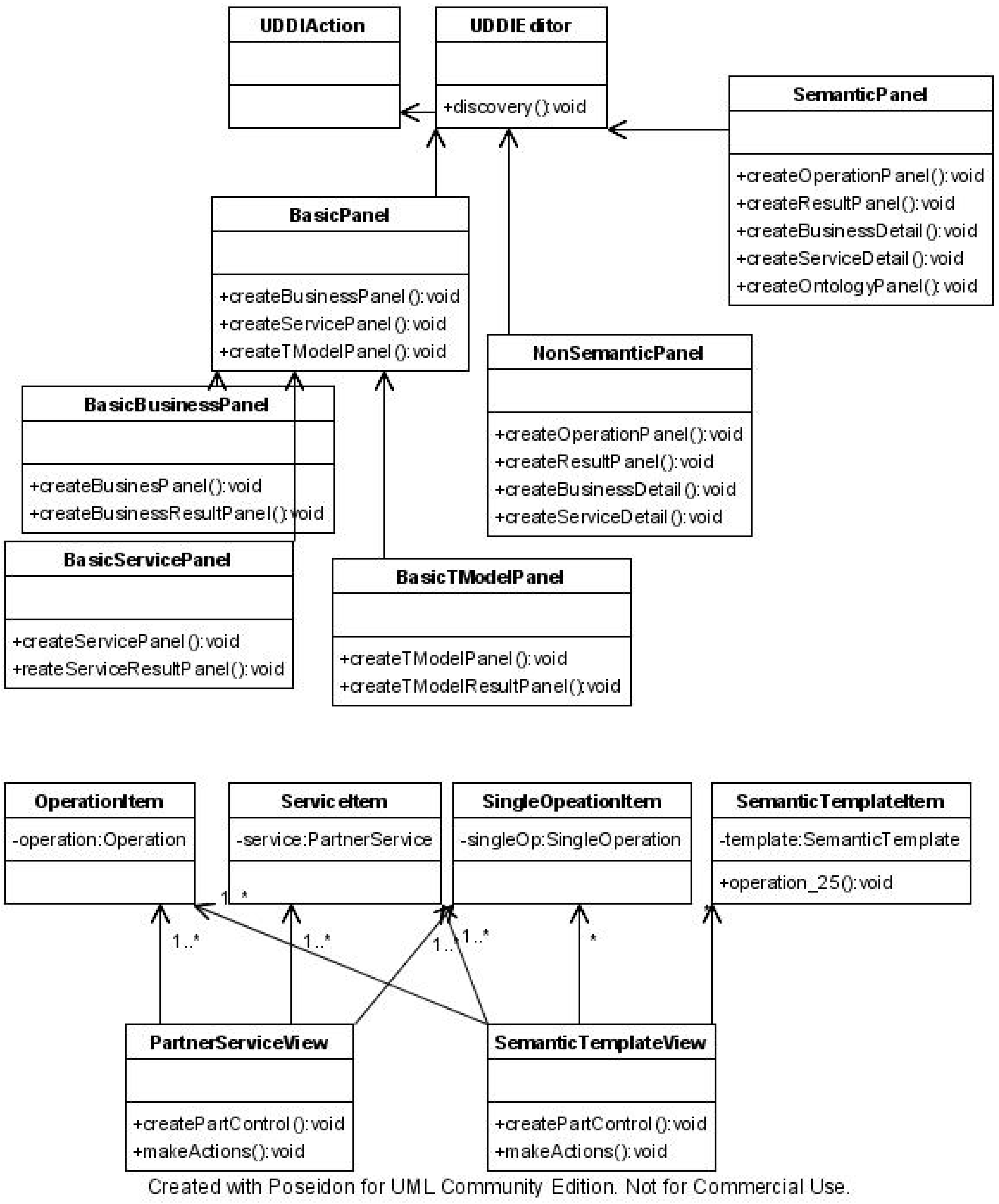

UDDIAction
UDDIEditor
+discovery():void
SemanticPanel
+createOperationPanel():void
+createResultPanel():void
+createBusinessDetail():void
+createServiceDetail():void
+createOntologyPanel():void
BasicPanel
+createBusinessPanel():void
+createServicePanel():void
+createTModelPanel():void
NonSemanticPanel
+createOperationPanel():void
+createResultPanel():void
+createBusinessDetail():void
+createServiceDetail():void
BasicBusinessPanel
+createBusinesPanel():void
+createBusinessResultPanel():void
BasicServicePanel
+createServicePanel():void
+reateServiceResultPanel():void
BasicTModelPanel
+createTModelPanel():void
+createTModelResultPanel():void
OperationItem
-operation:Operation
ServiceItem
-service:PartnerService
SingleOpeationItem
-singleOp:SingleOperation
SemanticTemplateItem
-template:SemanticTemplate
+operation_25():void
1..*
1..*
1..*
1..*
1..*
*
PartnerServiceView
+createPartControl():void
+makeActions():void
SemanticTemplateView
+createPartControl():void
+makeActions():void
Created with Poseidon for UML Community Edition. Not for Commercial Use.

Figure 3- 4: UML Class Diagram for Lumina View Part

CHAPTER 4

DESIGNING SEMANTIC WEB SERVICES: THE WSDL-S APPROACH[1]

[1] Ke Li, Kunal Verma, Ranjit Mulye, Reiman Rabbani, John A. Miller, Amit P. Sheth. Submitted to Semantic Web Processes and Their Applications. J. Cardoso, A. Sheth, Editors. Springer, 2006

ABSTRACT

Many businesses are adopting Web Service technologies to provide greater access to their applications. Due to the fast-paced E-Commerce requirements, more and more businesses prefer to only creating their core applications, and outsourcing the non-critical applications, or making use of their partners' applications directly. There is a growing requirement to build complex processes which may include Web Services supplied by the different partners. However, there are two main difficulties in building such Web Processes - 1) the current syntactic search mechanism is ineffective to find out the highly suitable services and 2) there are not many Process designing tools which allow dynamic binding of partner services. In this chapter, we present a solution for both the problems based on WSDL-S.

1. INTRODUCTION

Many businesses are adopting Web Service technology to expose their business applications, enabling them to have business collaboration, both within their organization and with business partners outside. The adoption of the Service Oriented Architecture (SOA) helps businesses contract-out their non-critical functions. Web Services, based on the SOA, are inherently designed for interaction in a loosely coupled environment, and hence an ideal choice for companies seeking inter- or intra-business interactions that span heterogeneous platforms and systems. Growing wide acceptance of Web Services is largely due to the fact that they are built on XML based standards like SOAP, WSDL and UDDI. Simple Object Access Protocol (SOAP) is a lightweight protocol for exchange of information among Web Services. Web Service Description Language (WSDL) describes a Web service similar as an interface. The Universal Description, Discovery and Integration (UDDI) protocol creates a service lookup platform that enables a service provider to register Web Services, and enables a service consumer to quickly, easily, and dynamically find and locate Web Services over the Internet. The need arisen to interconnect the various services offered by different businesses to create a complex business process or workflow that spans wider boundaries than ever before. The Web Service Business Process Execution Language (WS-BPEL) provides an XML-based language for the formal specification of business processes and business interaction protocols (OASIS).

The idea of building complex business processes has brought importance for tools that can model a business process. Though languages like WS-BPEL offer solutions for integrating Web Services into a business process, they are difficult to learn, involve understanding detailed language syntax and are only part of the solution for designing fully-functional Web processes. Tools that aid developers in easily building Web processes are becoming available; however, they do not alleviate several inherent limitations of current Web service technologies that make developing a process overly complicated or error-prone. Currently there are no Web Service composition tools that include discovery in a seamless fashion. The developer usually needs go to public, semi-public or local registries to look for partner services. Most UDDI registries supply Business Entity, Business Service and TModel as the searching format and adopt the syntactic searching mechanism. Users without enough knowledge about the partner services may spend much time searching for the suitable services, and they are likely to miss some highly suitable services. To solve this problem, we employ Semantic technology not only for the Web process designing task, but also for the whole Web Service lifecycle. A lightweight extension to the WSDL specification – WSDL-S (WSDL-S) is being jointly proposed by the LSDIS lab and IBM research. The WSDL-S specification allows annotating services and operations by mapping certain key elements to ontological concepts. These enhanced semantic services are published to a UDDI registry and can be dynamically discovered using some ontological concepts. This work enables users to find highly suitable and appropriate partner services in much less time than by manual discovery using a non-semantic, syntactic based UDDI searching mechanism. It also features the ability of partner services to be bound at design-time, deployment-time or execution-time (run-time) to further optimize the process and add greater dynamism, as opposed to the static-binding of partners.

In this work, we present a practical approach for designing Semantic Web Process using WSDL-S as a foundation. There are three tools used to accomplish this:

Radiant: Enables the service provider to annotate a WSDL file using ontological concepts and publish it to a UDDI registry.

Lumina: Allows the service requester to discover services with the required ontological concepts.

Saros: Helps the process developer to design a Semantic Web Service Process.

2. BACKGROUND

Different approaches have been proposed for modeling a WS-BPEL process. These approaches aid in designing the process model and later converting it to a WS-BPEL format. UML Activity Diagrams are candidates, but they are unable to model all patterns supported by WS-BPEL (Arkin, Askary, et al. 2005; Wohed1, Aalst et al. 2002). To maximize compatibility with the standards, Saros uses its own abstraction model that resembles the WS-BPEL specification.

There are two ways to design a Web process using today's available tools; the first way is to use a text-editor and type in the BPEL syntax and expressions, and the second is to use a process designer tool.

Several tools are emerging as the need for a streamlined and efficient process development is growing. Some prominent tools in current usage are: IBM WSADIE Designer, Oracle BPEL Designer, CapeScience Orchestrator and Active Webflow. While most of the aforementioned tools have similar features to Saros, e.g., GUI, graphical design, auto-generate BPEL etc., we focused on Saros as it supports semantics in process design via the use of WSDL-S-based Semantic Template.

3. DESIGN SEMANTIC WEB PROCESS USING WSDL-S

Before designing a Semantic Web Process, the Web services have to be annotated with ontological concepts using the extensibility elements and attributes provided with WSDL-S. The annotated files must then be published to an enhanced UDDI registry. This preparatory work supplies the possibility to carry out the partner services discovery at process design time / deployment time / execution time. The following three sections present the WSDL-S tool suite with an emphasis on how they can be used together to conveniently create Semantic Web Processes.

3.1 Service Annotation and Publish using Radiant

WSDL-S supplies an effective approach for describing Web Services by annotating WSDL elements with ontological concepts. Radiant was developed by the METEOR-S group to facilitate this annotation. This tool provides an easy-to-use GUI for a Web Service developer to do the following: 1. Add WSDL-S namespace and other namespaces for all the ontologies used; 2. Drag and drop ontological concepts to the suitable WSDL elements (element, operation, input, output); 3. Add precondition / effect as a child element of operation. Furthermore, it also provides the ability to annotate a Java file with semantic concepts using source code annotations.

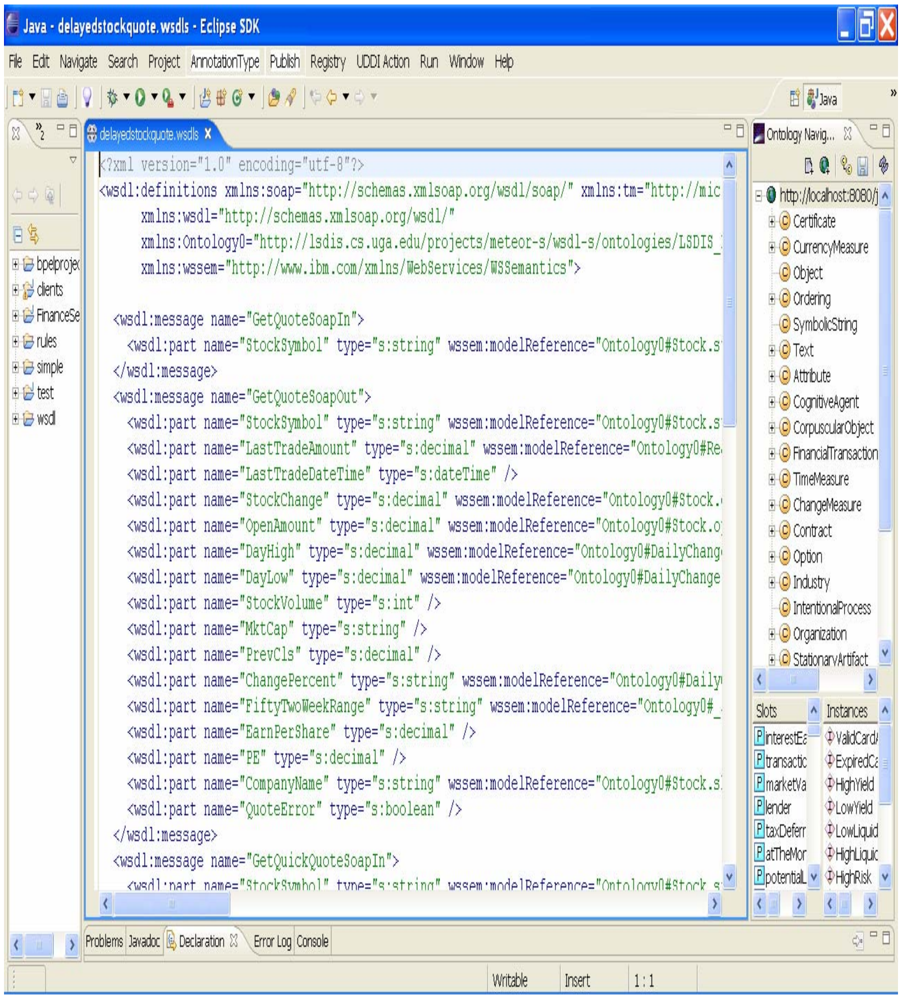
Figure 4-1 is a screen shot of the METEOR-S Radiant Tool for Semantic Annotation which used to annotate WSDL elements with ontological concepts.

As mentioned in Chapter 2, UDDI currently does not support publication or discovery of Semantic Web Services. To enhance UDDI to support semantics, we define an infrastructure to map WSDL-S to UDDI, shown in Figure 4-2, and the mapping details are given in Table 4-1. This is loosely based on UDDI Best Practice (Colgrave and Anuszewski, 2003), which defines a mapping from WSDL to UDDI. As shown in the figure, a WSDL-S service is captured using Business Service entity in UDDI, while portType and each operation within the WSDL-S service is captured using Technical Model.

Table 4-1. **WSDL-S to UDDI Mapping Detail**

WSDL-S	UDDI
Service	**Business Service**
Local name	Name
Service description	Description
Namespace, wsdl location	CategoryBag
portType	**TModel**
Local name	Name
Wsdl location	OverviewDoc
Namespace	CategoryBag
Operation	**TModel**
Local name	Name
Wsdl location	OverviewDoc
Namespace,domain,inputs, outputs,ontological concepts for operation, inputsand outputs, service name,business name	CategoryBag

After annotating a WSDL file to produce a WSDL-S file, the Radiant tool supplies the functionality to publish the service to a UDDI registry. Moreover, during publication using Radiant, users can publish service provider information as a Business Entity and publish services based on WSDL files, see Figure 4-3.



```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://mic
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:Ontology0="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_
      xmlns:wssem="http://www.ibm.com/xmlns/WebServices/WSSemantics">

  <wsdl:message name="GetQuoteSoapIn">
    <wsdl:part name="StockSymbol" type="s:string" wssem:modelReference="Ontology0#Stock.s
  </wsdl:message>
  <wsdl:message name="GetQuoteSoapOut">
    <wsdl:part name="StockSymbol" type="s:string" wssem:modelReference="Ontology0#Stock.s
    <wsdl:part name="LastTradeAmount" type="s:decimal" wssem:modelReference="Ontology0#Re
    <wsdl:part name="LastTradeDateTime" type="s:dateTime" />
    <wsdl:part name="StockChange" type="s:decimal" wssem:modelReference="Ontology0#Stock.
    <wsdl:part name="OpenAmount" type="s:decimal" wssem:modelReference="Ontology0#Stock.o
    <wsdl:part name="DayHigh" type="s:decimal" wssem:modelReference="Ontology0#DailyChang
    <wsdl:part name="DayLow" type="s:decimal" wssem:modelReference="Ontology0#DailyChange
    <wsdl:part name="StockVolume" type="s:int" />
    <wsdl:part name="MktCap" type="s:string" />
    <wsdl:part name="PrevCls" type="s:decimal" />
    <wsdl:part name="ChangePercent" type="s:string" wssem:modelReference="Ontology0#Daily
    <wsdl:part name="FiftyTwoWeekRange" type="s:string" wssem:modelReference="Ontology0#_
    <wsdl:part name="EarnPerShare" type="s:decimal" />
    <wsdl:part name="PE" type="s:decimal" />
    <wsdl:part name="CompanyName" type="s:string" wssem:modelReference="Ontology0#Stock.s
    <wsdl:part name="QuoteError" type="s:boolean" />
  </wsdl:message>
  <wsdl:message name="GetQuickQuoteSoapIn">
    <wsdl:part name="StockSymbol" type="s:string" wssem:modelReference="Ontology0#Stock.s
```

Figure 4-1. **Screen Shot of METEOR-S Radiant Tool**

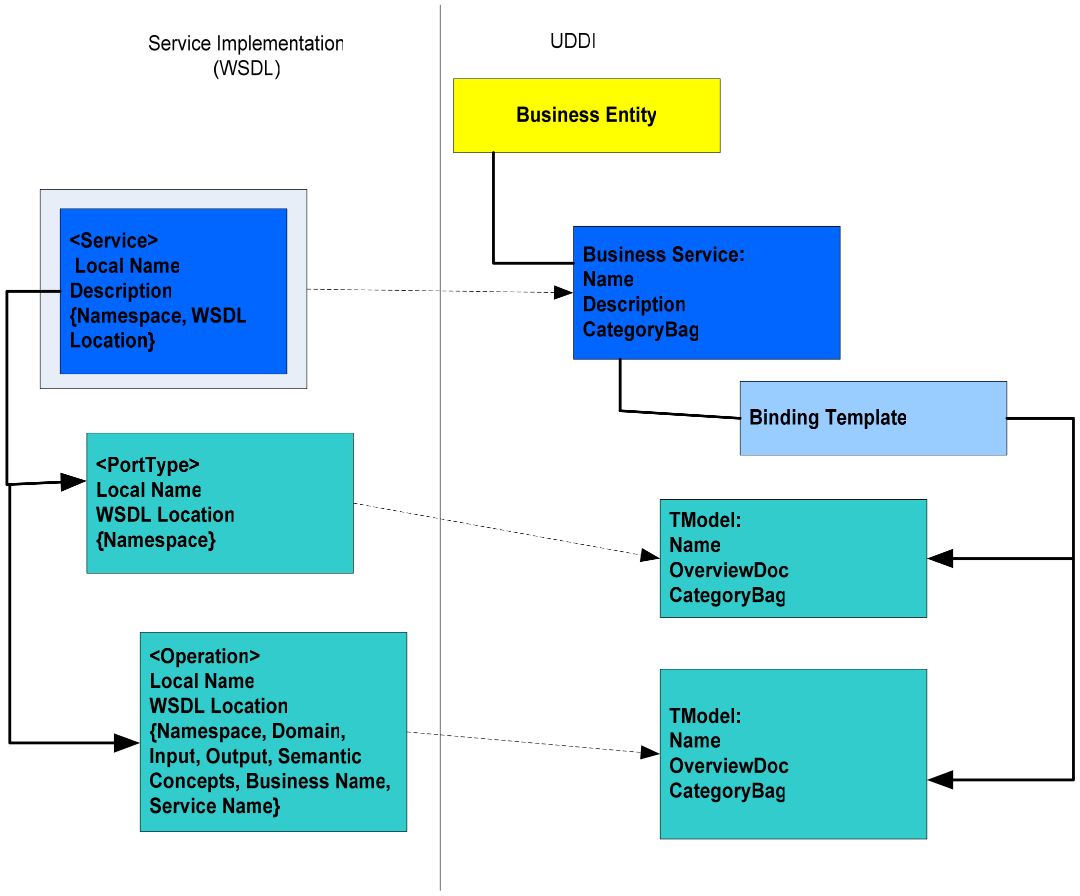
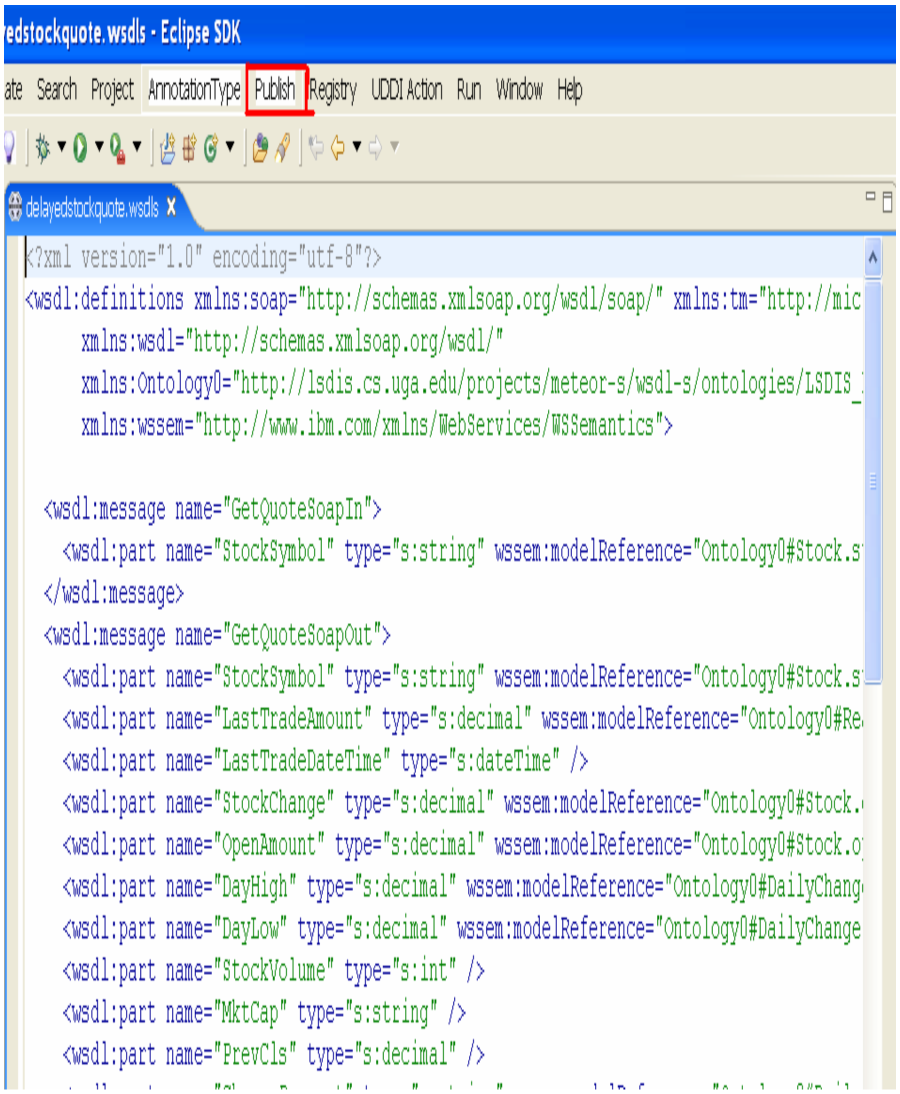

Service Implementation (WSDL)
UDDI
Business Entity
<Service> Local Name Description {Namespace, WSDL Location}
Business Service: Name Description CategoryBag
Binding Template
<PortType> Local Name WSDL Location {Namespace}
TModel: Name OverviewDoc CategoryBag
<Operation> Local Name WSDL Location {Namespace, Domain, Input, Output, Semantic Concepts, Business Name, Service Name}
TModel: Name OverviewDoc CategoryBag

Figure 4-2. **Mapping WSDL-S to UDDI**


edstockquote.wsdls - Eclipse SDK
ate Search Project AnnotationType Publish Registry UDDI Action Run Window Help
delayedstockquote.wsdls

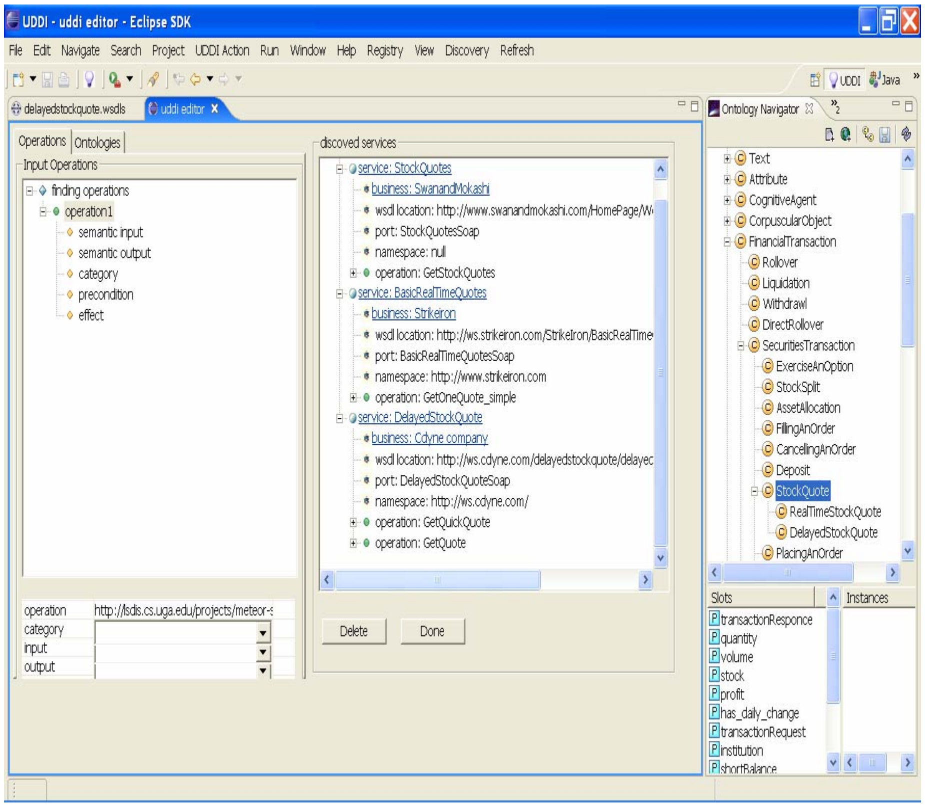
```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://mic
     xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
     xmlns:Ontology0="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_
     xmlns:wssem="http://www.ibm.com/xmlns/WebServices/WSSemantics">

  <wsdl:message name="GetQuoteSoapIn">
    <wsdl:part name="StockSymbol" type="s:string" wssem:modelReference="Ontology0#Stock.s
  </wsdl:message>
  <wsdl:message name="GetQuoteSoapOut">
    <wsdl:part name="StockSymbol" type="s:string" wssem:modelReference="Ontology0#Stock.s
    <wsdl:part name="LastTradeAmount" type="s:decimal" wssem:modelReference="Ontology0#Re
    <wsdl:part name="LastTradeDateTime" type="s:dateTime" />
    <wsdl:part name="StockChange" type="s:decimal" wssem:modelReference="Ontology0#Stock.
    <wsdl:part name="OpenAmount" type="s:decimal" wssem:modelReference="Ontology0#Stock.o
    <wsdl:part name="DayHigh" type="s:decimal" wssem:modelReference="Ontology0#DailyChang
    <wsdl:part name="DayLow" type="s:decimal" wssem:modelReference="Ontology0#DailyChange
    <wsdl:part name="StockVolume" type="s:int" />
    <wsdl:part name="MktCap" type="s:string" />
    <wsdl:part name="PrevCls" type="s:decimal" />
```

Figure 4-3. **Publish Web Services using Radiant**

3.2 Semantic Discovery using Lumina

Based upon the WSDL-S to UDDI mapping structure, the Lumina GUI tool was developed to facilitate Semantic Web Service discovery during process design time or deployment time. The requirements of the user are captured using a semantic template, which captures the abstract functionality of a Web service. The information captured includes a list of abstract operations whose functionality, inputs and outputs are defined using ontological concepts. The discovery engine returns services that are annotated with ontological concepts that are the same or semantically related to concepts in the template.

When discovering a service based on the WSDL-S description, the user can add one or more ontology URLs in the discovery panel and input the ontological concepts to represent input, outputs and operations. After discovery, the result will be shown in the result panel. The result gives not only the operation's detail information, but also the related service and the service provider's information. See Figure 4-4 which shows the annotation of the operation and the result of the discovered services.



operation http://lsdis.cs.uga.edu/projects/meteor-s
category
input
output

service: StockQuotes
business: SwanandMokashi
wsdl location: http://www.swanandmokashi.com/HomePage/W
port: StockQuotesSoap
namespace: null
operation: GetStockQuotes
service: BasicRealTimeQuotes
business: Strikeiron
wsdl location: http://ws.strikeiron.com/StrikeIron/BasicRealTime
port: BasicRealTimeQuotesSoap
namespace: http://www.strikeiron.com
operation: GetOneQuote_simple
service: DelayedStockQuote
business: Cdyne company
wsdl location: http://ws.cdyne.com/delayedstockquote/delayec
port: DelayedStockQuoteSoap
namespace: http://ws.cdyne.com/
operation: GetQuickQuote
operation: GetQuote

Figure 4-4. **WSDL-S Discovery**

Many services have multiple operations within them. With the consideration of economic and connection convenience, the process developers usually would likely to invoke as many operations as possible from one partner service. To deal with this problem, Lumina enables the user to find services that hold more than one required operations. Figure 4-5 shows this by listing two operations in one service.

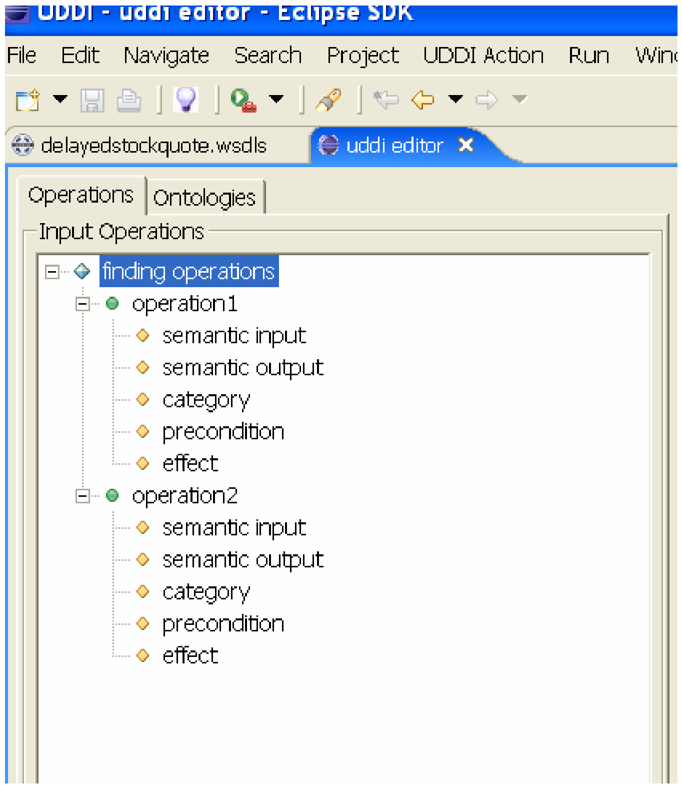


***Figure 4-5.* Searching Service with Multiple Operations**

Moreover, to complement today's non-semantic discovery and give more flexibility for the users to search in different UDDI registries, Lumina provides general UDDI discovery. Using the same mapping structure as WSDL-S to UDDI, Radiant can publish a WSDL file to a UDDI registry directly. This enables Lumina to discover the services described in WSDL based on the keyword searching mechanism. Figure 4-6 shows the general UDDI Discovery format, which has a familiar style to the current UDDI users. Figure 4-7 shows the WSDL Discovery panel. To support the users who wish to use different UDDI registries, Lumina allows the user to add new UDDI registries to the environment and use a uniform searching style to do the discovery. Figure 4-8 shows the registry control panel, which holds the different registries' information and allows users to add or edit a registry.

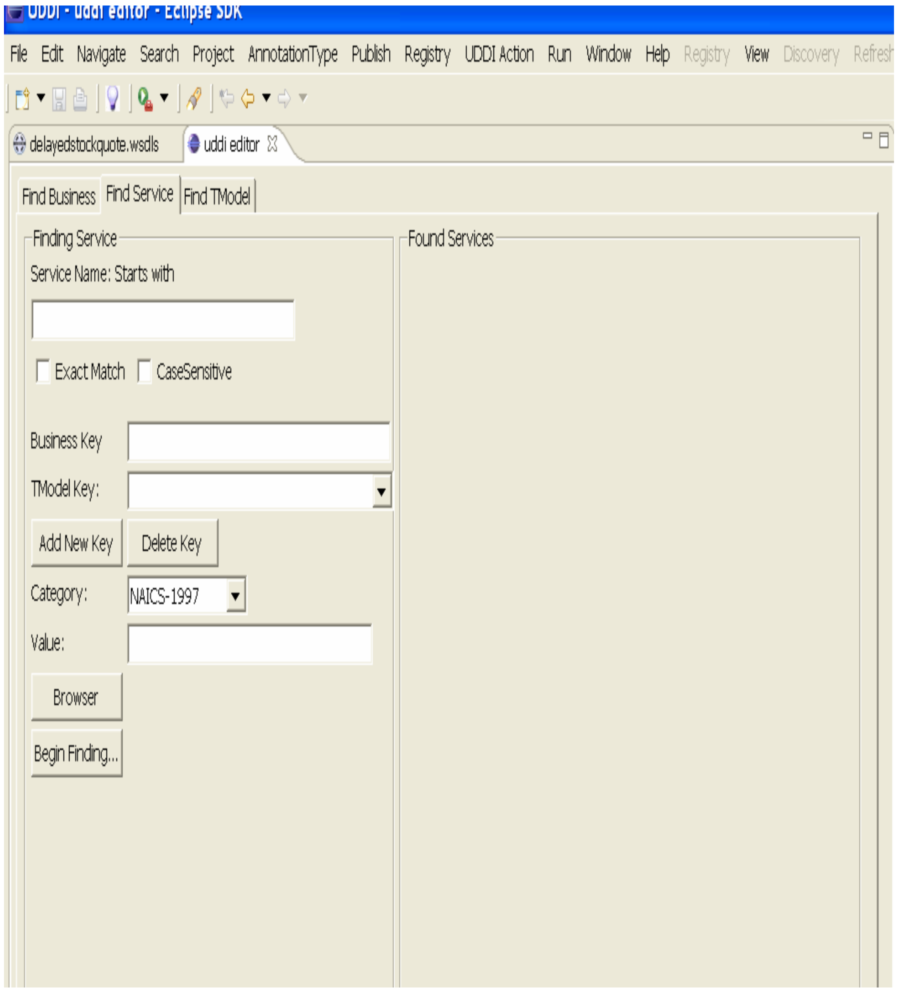


Figure 4-6. **General UDDI Discovery Panel**

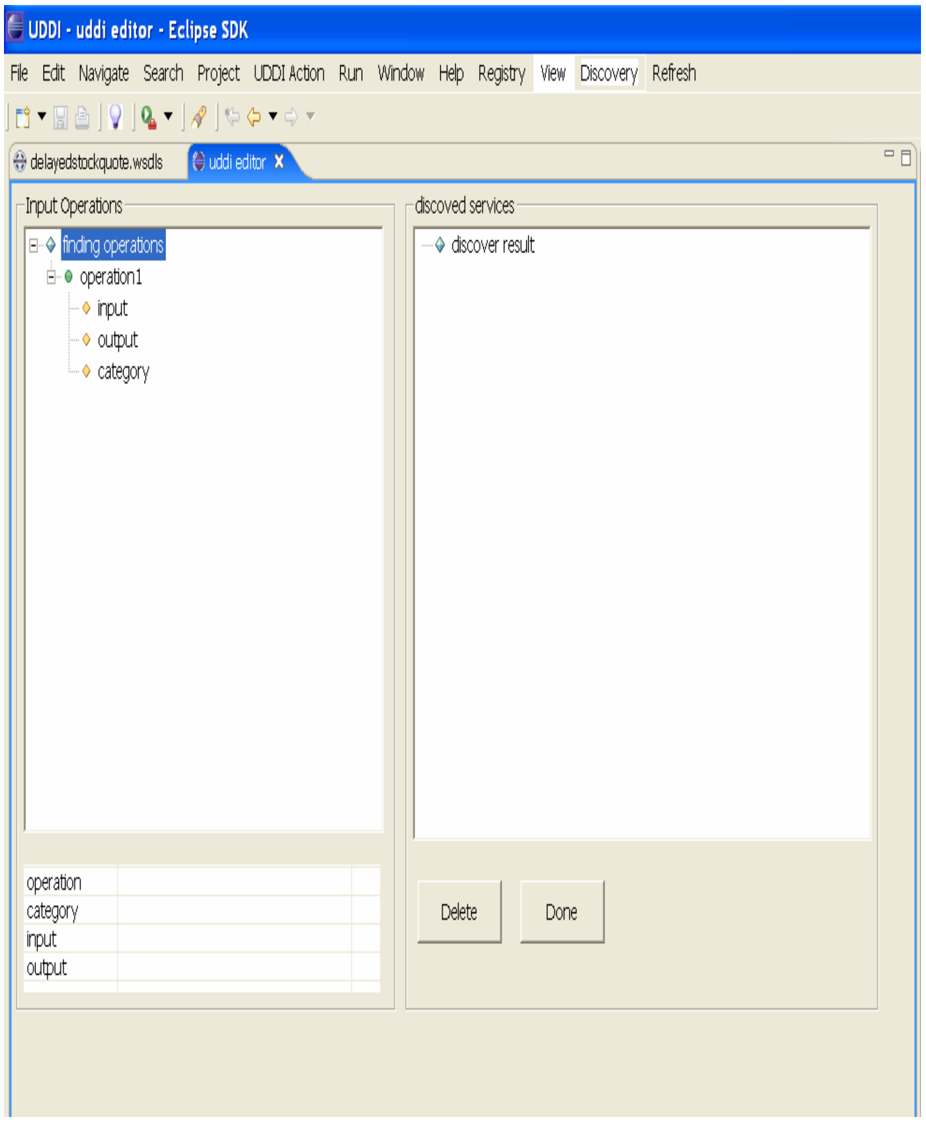


Figure 4-7. **WSDL Discovery Panel**

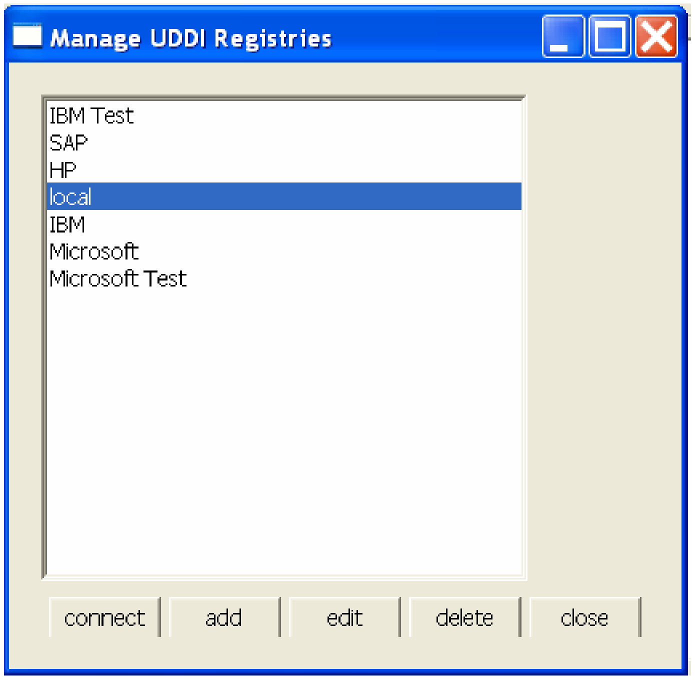


Figure 4-8. **Registry Control Panel**

3.3 Process Design using Saros

3.3.1 Methodology

To support capabilities for dynamic partner selections, the Saros tool makes use of Semantic Template technology. A semantic template can be used to insert a virtual partner into the Web Process. The selection process can be thought of as being constituted of two phases, which are elaborated below.

In the first phase, the process developer generates a semantic template by using the Semantic Template Viewer, which can graphically capture the ontological concepts of the desired virtual partner Web Services, allowing constraints, policies and operational conditions to be added. The developer can add operations to the template and specify the input and output messages for each operation. Each of these entities is annotated by the ontological concepts. The developer can either create a new template or s/he has a choice to load an existing template. Phase two utilizes the core searching mechanism for dynamic partner discovery.

In the second, once the semantic template has been generated, Lumina then processes it for partner discovery. Lumina extracts the semantic information from the template using the METEOR-S WSDLS4J Java implementation and passes it to the discovery module. The discovery module performs semantic search and returns a set of matching ranked results. To further find better matched Web Services, the returned set may be passed through a constraint analyzer module that was developed by the METOER-S lab (Verma, Gomadam, et.al. 2005). An example of some constraints could be TurnAroundTime <= 7 days or Cost<=$5000 or Virtual Partner A, B and C must be compatible with each other. Therefore the process developer can use the semantic template for dynamic partner discovery while s/he is designing the process, so the exact Web Service instance is known beforehand. This phase is explained in Figure 4-9. Alternatively, the semantic template can be used to create a Virtual Partner in the Web Process and the discovery can be deferred until run-time/execution. Both options are available using the Saros and Lumina tools.

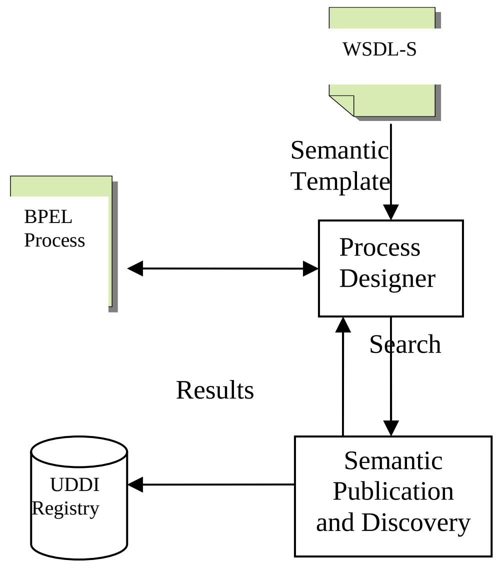


Figure 4-9. **Phase II - Dynamic Partner Discovery**

There are three types of service binding in a Web Process supported by the WSDL-S tool suite as outlined below:

1. Design-time: The set of partners are permanently bound to the Web Process during the design phase.
2. Deployment-time: Uses a virtual partner at design-time, but during deployment the partners are bound to the discovered Web Services. This happens before execution of the process.
3. Run-time: After process instances have started executing, the partners are (re-) discovered and bound.

3.3.2 Architecture of Saros

This section presents an overview of the METEOR-S Process Design Tool – SAROS. The UI of the Process design and development tool consists mainly of three components: Element Palette, Process Canvas, and Element Property Sheet, as illustrated in Figure 4-10. The process designer uses the Model View Controller (MVC) (Reenskaug, 2003) pattern as the underlying architecture. To realize this architecture it uses the Graphical Editing Framework (GEF) toolkit, which is part of the Eclipse tool integration platform. This tool is integrated with Eclipse as an Eclipse plug-in or can be run as an Eclipse application. Figure 4-11 shows a diagram of the tool architecture using the MVC pattern.

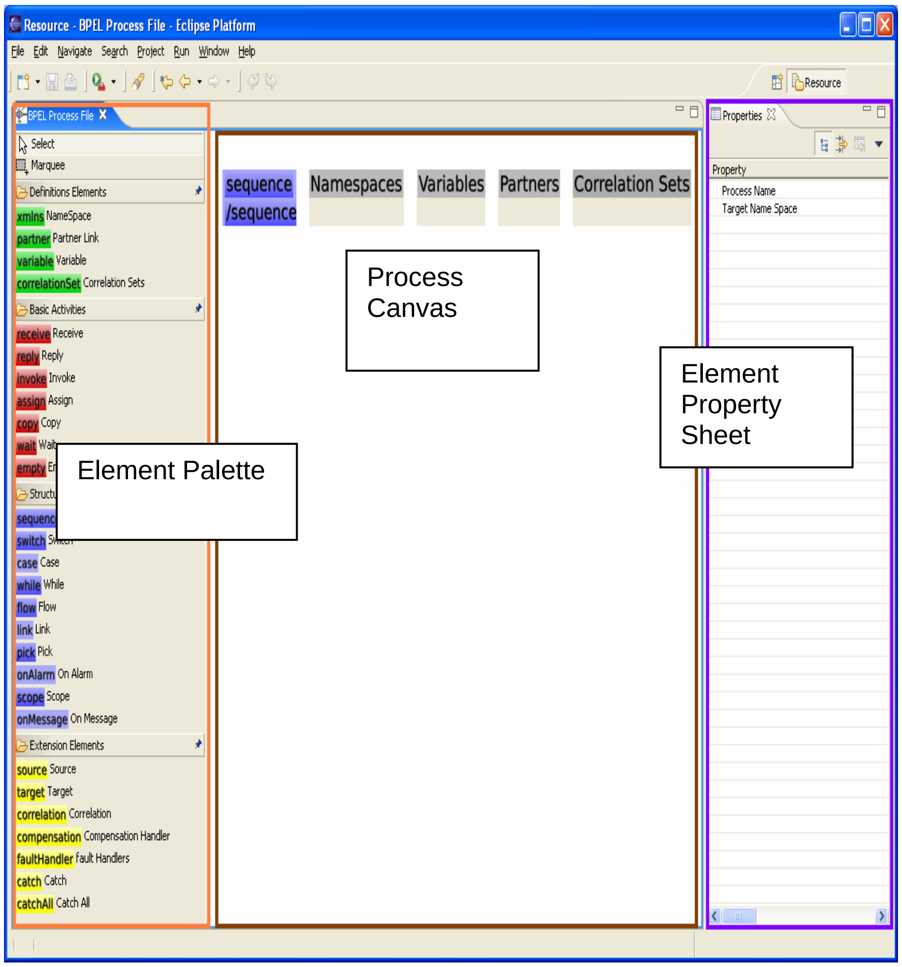


Figure 4-10. **METEOR-S Process Design and Development Tool - Saros**

3.2.1.1 MODEL VIEW CONTROLLER (MCV) ARCHITECTURE

The MVC architecture is a commonly used and effective architecture to build GUI-based systems because it separates the code from the model and view. The “Model” defines the behavior of the application logic, representing the in-memory model of the entire process. The "View" handles the graphical rendering of the model to the UI. The "Controller" is the code that forms the link between the model and the view; it is responsible for handling the editing of BPEL element properties.

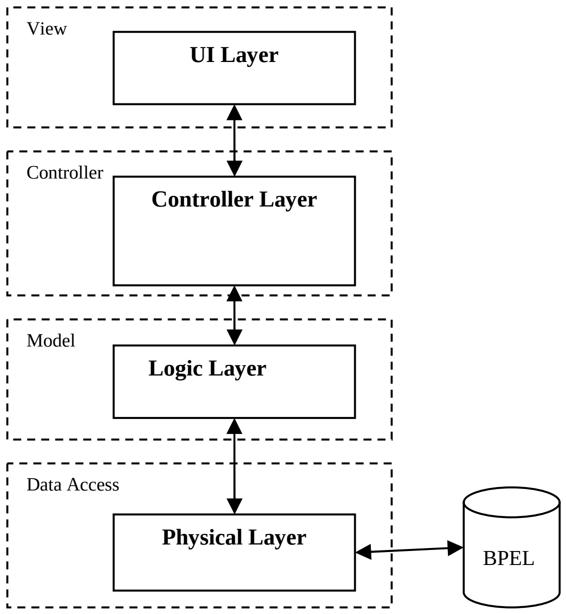


Figure 4-11. **METEOR-S Semantic Process Design and Development Tool Architecture**

3.2.1.2 GRAPHICAL EDITING FRAMEWORK (GEF)

GEF enables developers to create a rich graphical editor for an existing application model (Graphical Editing Framework). The process designer uses the GEF framework to build its graphical user interface because both use the MVC pattern. GEF is fully written in Java and works on all operating systems officially supported by the Eclipse platform (Open platform for tool integration), thereby eliminating any porting issues. GEF depends on Draw2d which is a lightweight toolkit built with the Standard Widget Toolkit (SWT) and offers optimized layout and painting along with providing a native look and feel for the GUI.

3.2.1.3 USABILITY FEATURES

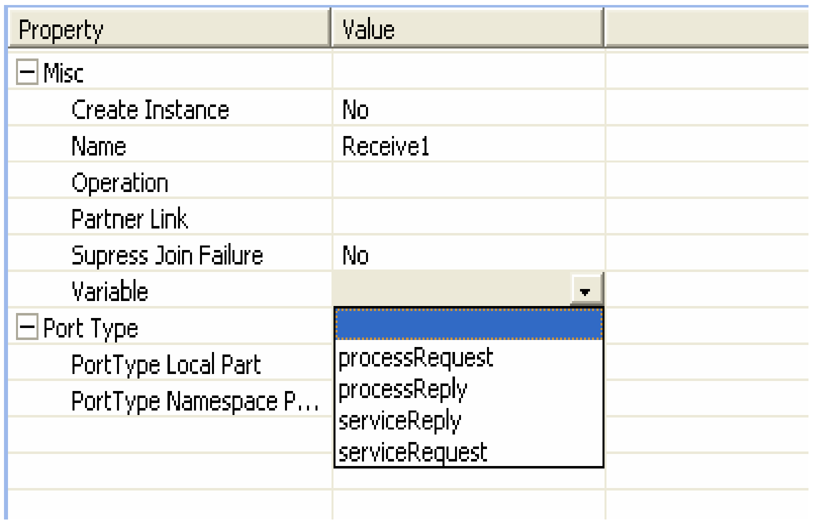
Saros offers an easy to use GUI for process developers to rapidly build Web processes. Process developers are offered support for dragging and dropping of process elements on a process “canvas”. This, combined with ease of element selection and deletion, offers a simple to use GUI. Selecting a particular element opens up a property sheet that allows the user to modify element properties. This approach helps in hiding the unnecessary syntactic details from the developer. Other usability features are outlined below.

- Color coded process activities

The activities have been categorized by functionality so that the developer can quickly understand the canvas visually and intuitively.

- Definition lookup

Many of the basic activities of the business process have properties that refer to definition elements (Variables, Partnerlinks or XML namespaces). To avoid typographical errors, the Process designer tool offers a drop-down box of available choices for the entries. Figure 4-12 provides a sample illustration of this feature.



Property	Value
⊟ Misc	
Create Instance	No
Name	Receive1
Operation	
Partner Link	
Supress Join Failure	No
Variable	
⊟ Port Type	
PortType Local Part	
PortType Namespace P...	

Figure 4-12. **Definition Lookup**

- Avoiding ambiguous process designs

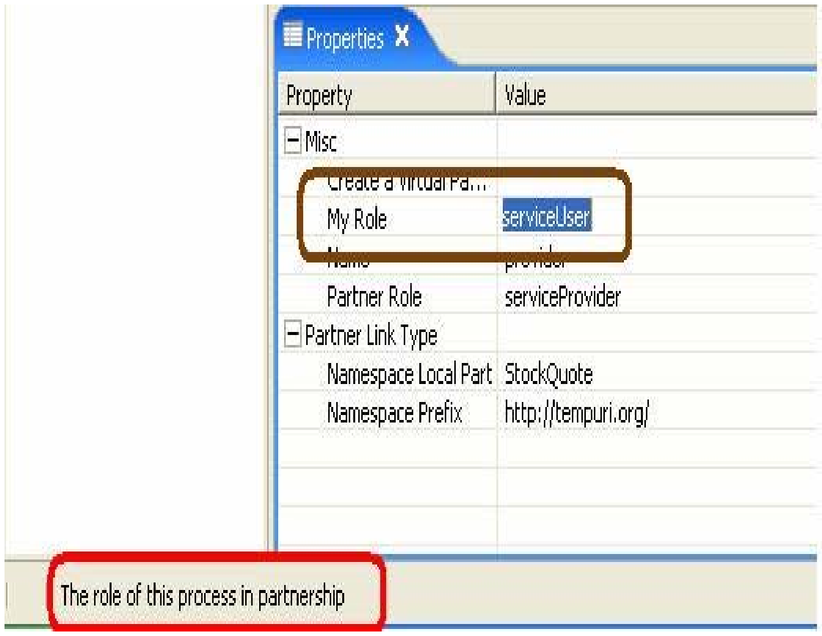
Before allowing addition of a new element to a container type element, Saros validates the insertion. This prevents the process from being in an ambiguous state. For example, the only valid addition to an activity of “Switch” is a case activity, others are not allowed.

- Designing complex processes

Container elements such as sequence, flow, etc. can be nested to any depth. This helps in generating a process with arbitrary complexity but intuitive visualization.

- Intuitive help messages

The tool provides descriptive messages for many of the editable properties of process elements in the status bar. Figure 4-13 shows an illustration of such a help message.



Property	Value
Misc	
Create a Virtual Pa...	
My Role	serviceUser
Name	provider
Partner Role	serviceProvider
Partner Link Type	
Namespace Local Part	StockQuote
Namespace Prefix	http://tempuri.org/

The role of this process in partnership

Figure 4-13. **Status bar Helper Message**

4. SAMPLE USE CASE

In this section, we present a use case in the finance field. Consider this scenario, an investment company provides a service that helps their customers analyze the feasibility of buying some stocks. This service gives the evaluation result based on the following stock ticker information: current price, volume, bid quantity, the highest price in the past one year, earnings per share, and how much money the customer wants to invest. The customer tells the service which stocks s/he is interested in and how much money s/he may invest. The remaining information should depend on other partner services. Following our the WSDL-S approach, we give an outline of 8 steps to present how to use the BPEL Designer tool – Saros – to design such a composite service. These steps also make use of the Lumina discovery tool.

1. **Analyze the business requirements and build a UML diagram**. In this use case, the investment company will use an evaluation service. This service can either belong to the company or belong to the other investment companies. Such an evaluation service needs the stock quote information and the corresponding company financial profile of the past one year. We need two partner services to fill in the above requirements, respectively. Moreover, the two partner services can run simultaneously because they do not depend on each other. The UML diagram is shown in Figure 4-14 according to this use case.

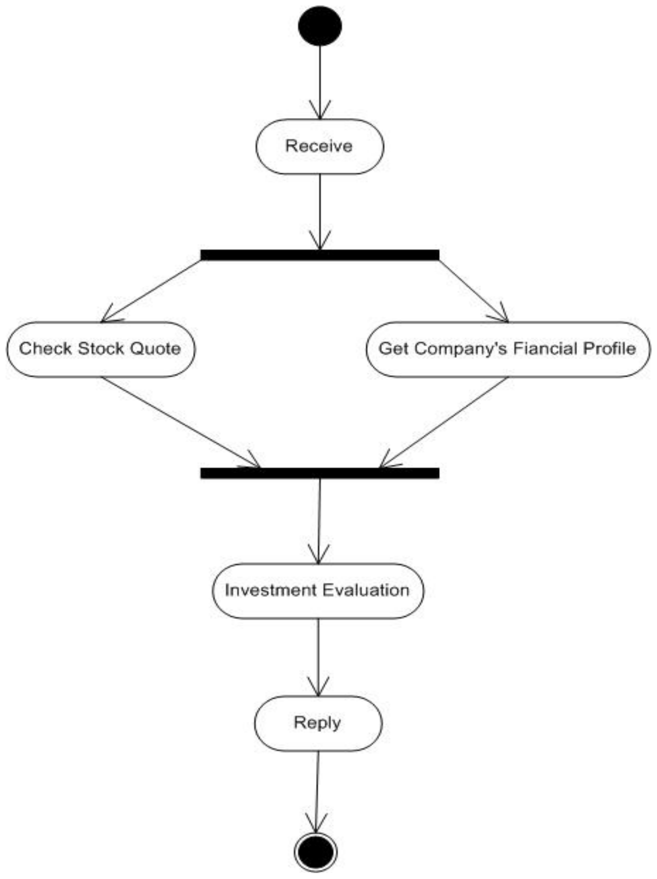


Figure 4-14. **UML Activity Diagram for the User Case**

2. **Fill in the process skeleton:** Map the process sequence in the activity diagram to the sequence element of WS-BPEL, shown in Figure 4-15.

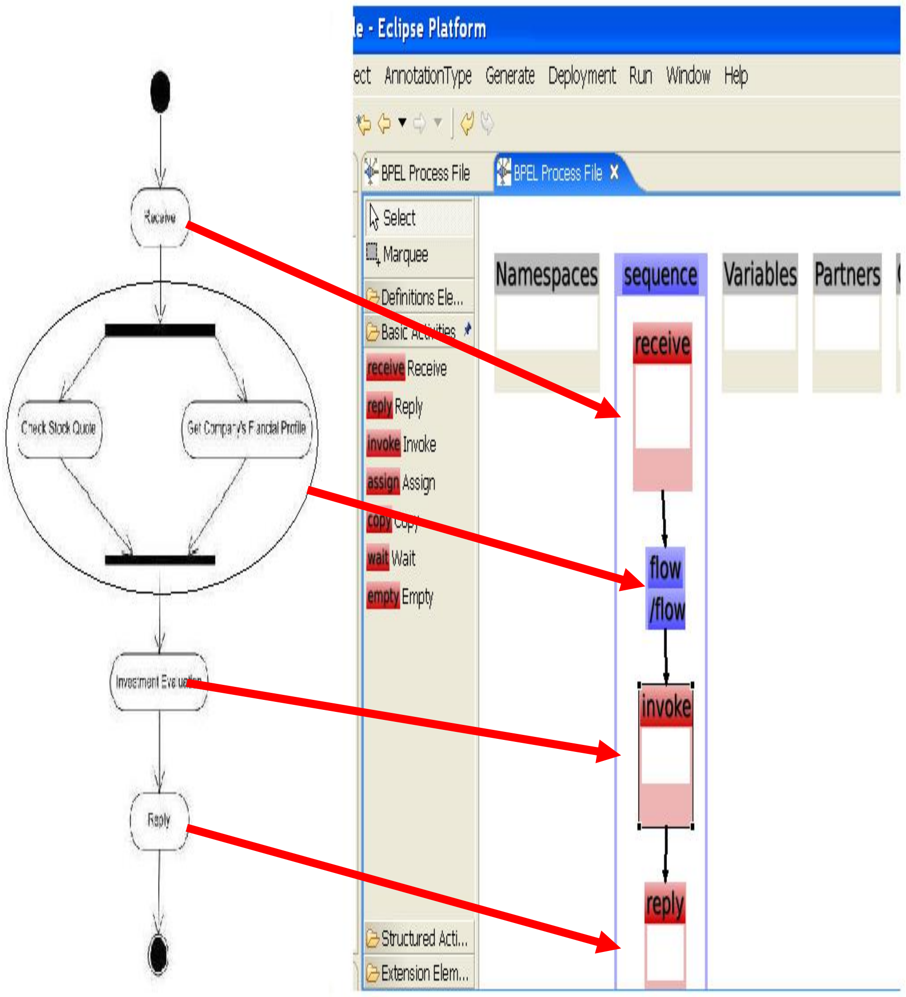


Figure 4-15. **Top Level Sequence**

3. **Fill in the nested constructs / structured activities**: such as Flow, Switch, While, etc. We insert two “Invoke” in the “Flow” for invoking the corresponding two simultaneous partner services, shown in Figure 4-16.

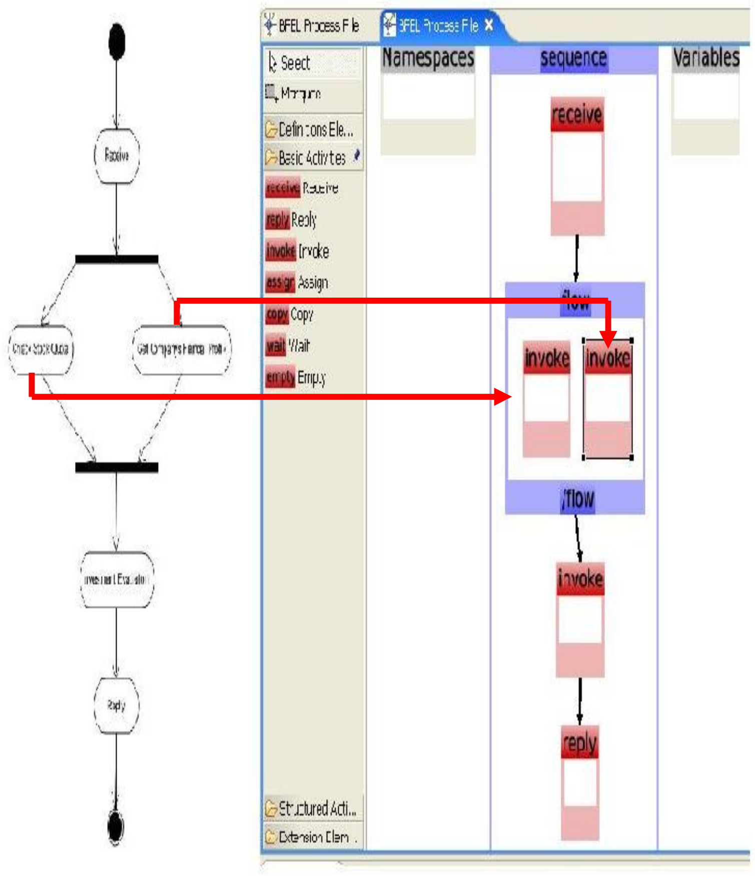


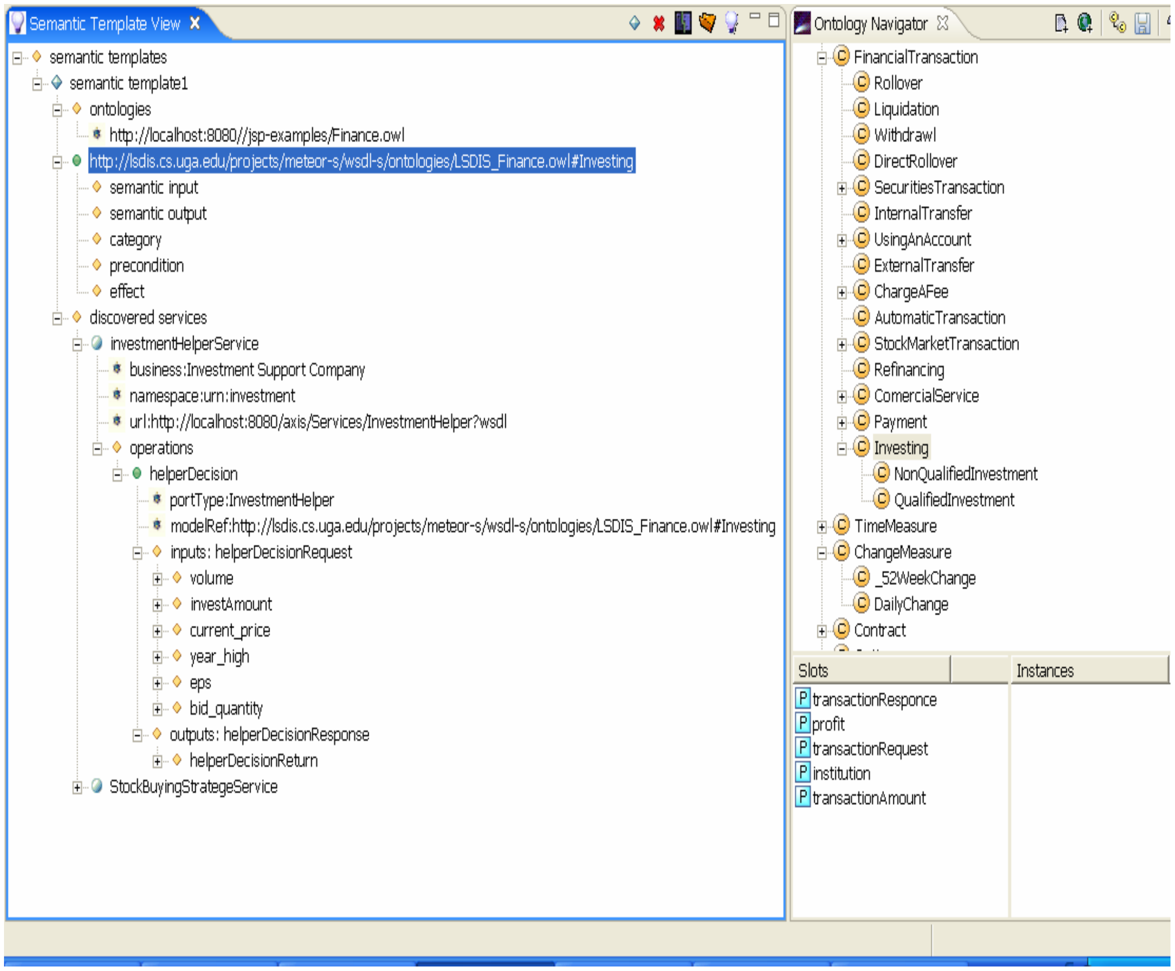
Figure 4-16. **Fill in Nested Constructs**

4. **Identify Partners by using either one of the following methods**:

- Binding the real partners: Based on the business requirements, Lumina can help the developer find highly suitable partner services. It provides two GUIs to accomplish the partner services discovery: UDDI Editor and Semantic Template View. Although their discovery functionalities are the same, they focus on different design aspects. The UDDI Editor provides the flexibility for the developer to do the discovery using both the WSDL-S approach and the general UDDI discovery approach by using Business Entity, Business Service and TModel. The Semantic Template View focuses on building a discovery template based on the business requirements, and its "easy to drag" property enables it to be used with

Saros. Because we focus on the WSDL-S approach, here we use the Semantic Template View in our example.

In this use case, the investment company needs an investment evaluation service. The developer can use the company's own service or search for a more suitable one. The only knowledge about this service is that it can perform investment evaluation. We use the SUMO Finance ontology as the domain knowledge base. Here we choose ontologyNS#investing to annotate the service's operation. Two services are found and the discovered service provides the detailed information at both the service level and the operation level, shown in Figure 4-17.



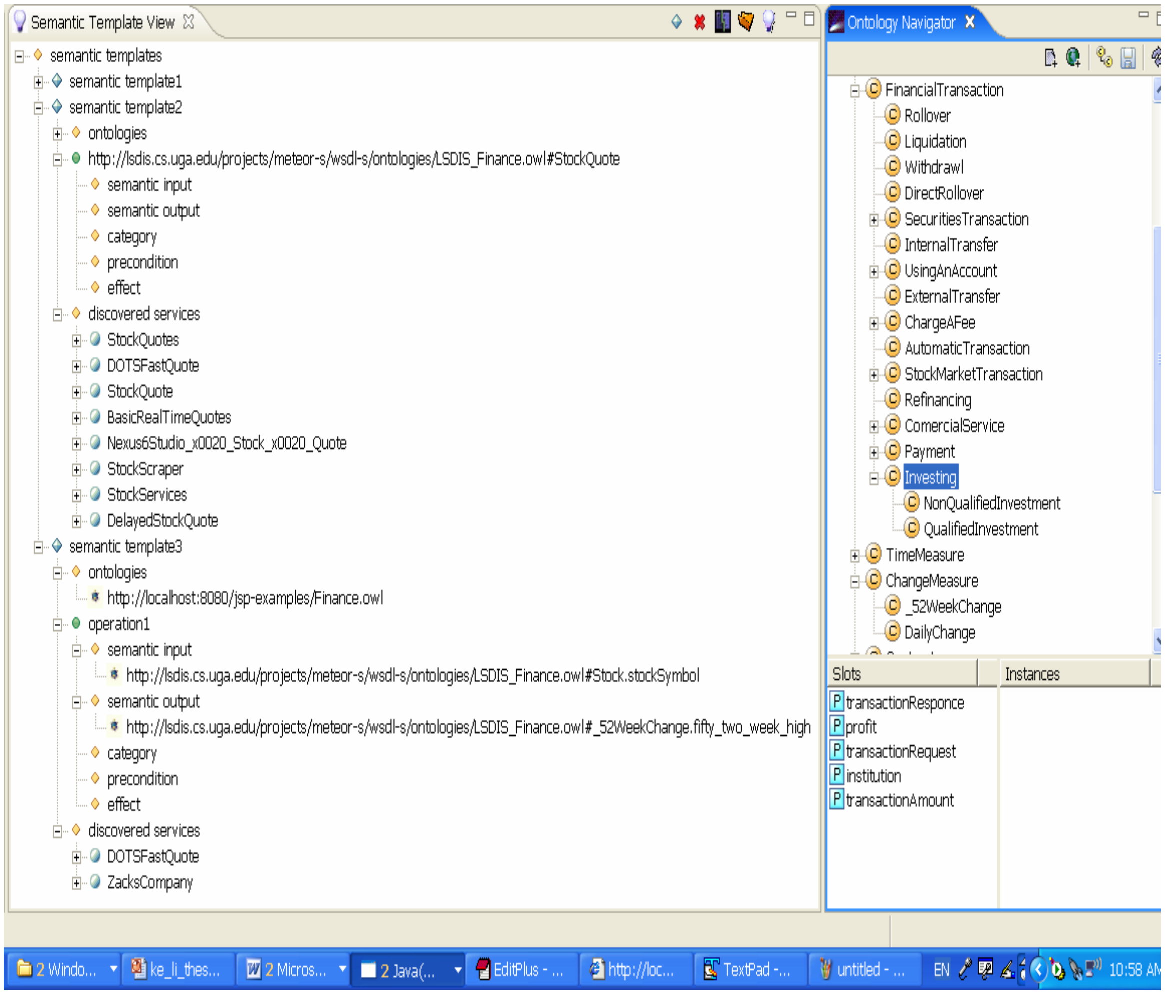
```
semantic templates
  semantic template1
    ontologies
      http://localhost:8080//jsp-examples/Finance.owl
    http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_Finance.owl#Investing
      semantic input
      semantic output
      category
      precondition
      effect
    discovered services
      investmentHelperService
        business:Investment Support Company
        namespace:urn:investment
        url:http://localhost:8080/axis/Services/InvestmentHelper?wsdl
        operations
          helperDecision
            portType:InvestmentHelper
            modelRef:http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_Finance.owl#Investing
            inputs: helperDecisionRequest
              volume
              investAmount
              current_price
              year_high
              eps
              bid_quantity
            outputs: helperDecisionResponse
              helperDecisionReturn
      StockBuyingStrategeService
```

```
FinancialTransaction
  Rollover
  Liquidation
  Withdrawl
  DirectRollover
  SecuritiesTransaction
  InternalTransfer
  UsingAnAccount
  ExternalTransfer
  ChargeAFee
  AutomaticTransaction
  StockMarketTransaction
  Refinancing
  ComercialService
  Payment
  Investing
    NonQualifiedInvestment
    QualifiedInvestment
TimeMeasure
ChangeMeasure
  _52WeekChange
  DailyChange
Contract
```

```
P transactionResponce
P profit
P transactionRequest
P institution
P transactionAmount
```

Figure 4-17. **Discovery the Basic Service Using Operation's Domain Concept**

In the next step, we can use the operation information of the discovered service to search for the other two services: the “stock quote” service and the “company profile” service. There are six attributes in the discovered operation’s input message. The current_price, bid_quantity and volume belong to “stock quote” service; the eps (earning per share) and year_high belong to “company profile” service; the investAmount is provided by the customer. We build the other two semantic templates and add the corresponding output concepts to them respectively.



```
Semantic Template View
semantic templates
  semantic template1
  semantic template2
    ontologies
    http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_Finance.owl#StockQuote
      semantic input
      semantic output
      category
      precondition
      effect
    discovered services
      StockQuotes
      DOTSFastQuote
      StockQuote
      BasicRealTimeQuotes
      Nexus6Studio_x0020_Stock_x0020_Quote
      StockScraper
      StockServices
      DelayedStockQuote
  semantic template3
    ontologies
      http://localhost:8080/jsp-examples/Finance.owl
    operation1
      semantic input
        http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_Finance.owl#Stock.stockSymbol
      semantic output
        http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_Finance.owl#_52WeekChange.fifty_two_week_high
      category
      precondition
      effect
    discovered services
      DOTSFastQuote
      ZacksCompany
```

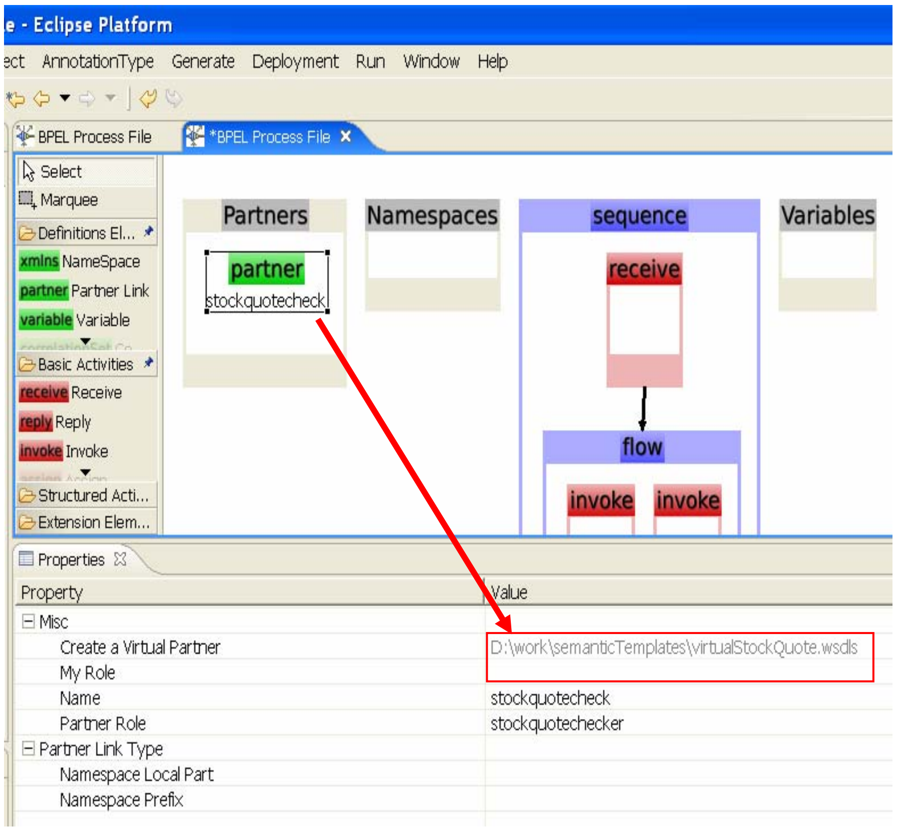
```
Ontology Navigator
FinancialTransaction
  Rollover
  Liquidation
  Withdrawl
  DirectRollover
  SecuritiesTransaction
  InternalTransfer
  UsingAnAccount
  ExternalTransfer
  ChargeAFee
  AutomaticTransaction
  StockMarketTransaction
  Refinancing
  ComercialService
  Payment
  Investing
    NonQualifiedInvestment
    QualifiedInvestment
TimeMeasure
ChangeMeasure
  _52WeekChange
  DailyChange

Slots                     Instances
P transactionResponce
P profit
P transactionRequest
P institution
P transactionAmount
```

***Figure 4-18.* Discovery Partner Services Using Input and Output Attributes' Domain Concepts**

- Binding the virtual partners: Shown in Figure 4-18, there are eight Web services discovered for the “stock quote” service. The developer can pick the most suitable one and insert it into the process if s/he wants to build an executable BPEL process. Alternatively, s/he can make the decision later and build an abstract BPEL process by adding a virtual partner based on the semantic template. The advantages of using the semantic template technology to build an abstract BPEL process fall in two aspects: i) After partner selection has been finalized in the BPEL process, it is possible that more optimal services become available (in regards to various pre-defined quality properties, e.g., cost, time, reliability, accuracy, etc.). ii) The partner functionalities or the user terms change, or a partner becomes unavailable unbeknownst to the process. These problems would render the Web process prone to errors. The Semantic Template View in Lumina enables the developer to design the semantic template graphically with the help of Radiant. Moreover, it can generate the semantic template files or load the semantic template files built previously. Saros can link these semantic template files as the virtual partners. Appendix A shows one semantic template file and Figure 4-19 shows in the box how Saros adds a virtual partner using the file.

5. **Add Namespaces, Variables and Correlation Sets constructs (for executable BPEL process)**: Add the namespace for each partner service. Create all the variables for the message exchange between the services. All the information in this design phase can be found in the discovered services shown by Lumina.
6. **Link Partners to Invoke, Receive and Reply constructs (for executable BPEL process)**: Link the partner to the corresponding activity element by clicking on the element objects property view from a drop down list.
7. **Add the supplementary elements and fill in details**: Add “assign”, “copy”, “link” etc to accomplish the process.

8. **Generate BPEL process**: The BPEL process file and the corresponding process WSDL file are generated by clicking on the save button. The complete BPEL and WSDL files for our scenario are in Appendix B and Appendix C. The completed process view is at Figure 4-20 and Figure 4-21.



Property	Value
⊟ Misc	
Create a Virtual Partner	D:\work\semanticTemplates\virtualStockQuote.wsdls
My Role	
Name	stockquotecheck
Partner Role	stockquotechecker
⊟ Partner Link Type	
Namespace Local Part	
Namespace Prefix	

Figure 4-193. **Add a Virtual Partner**

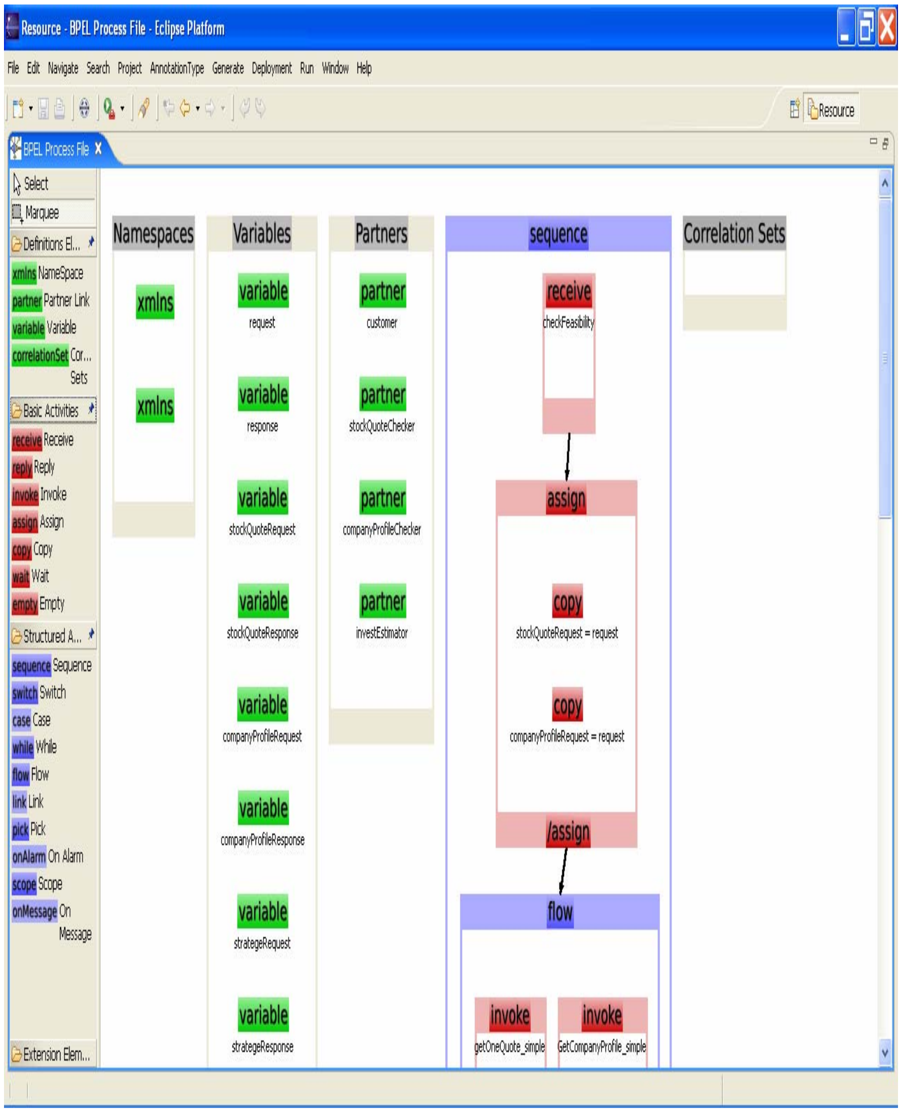


Figure 4-204. **The Complete Process I**

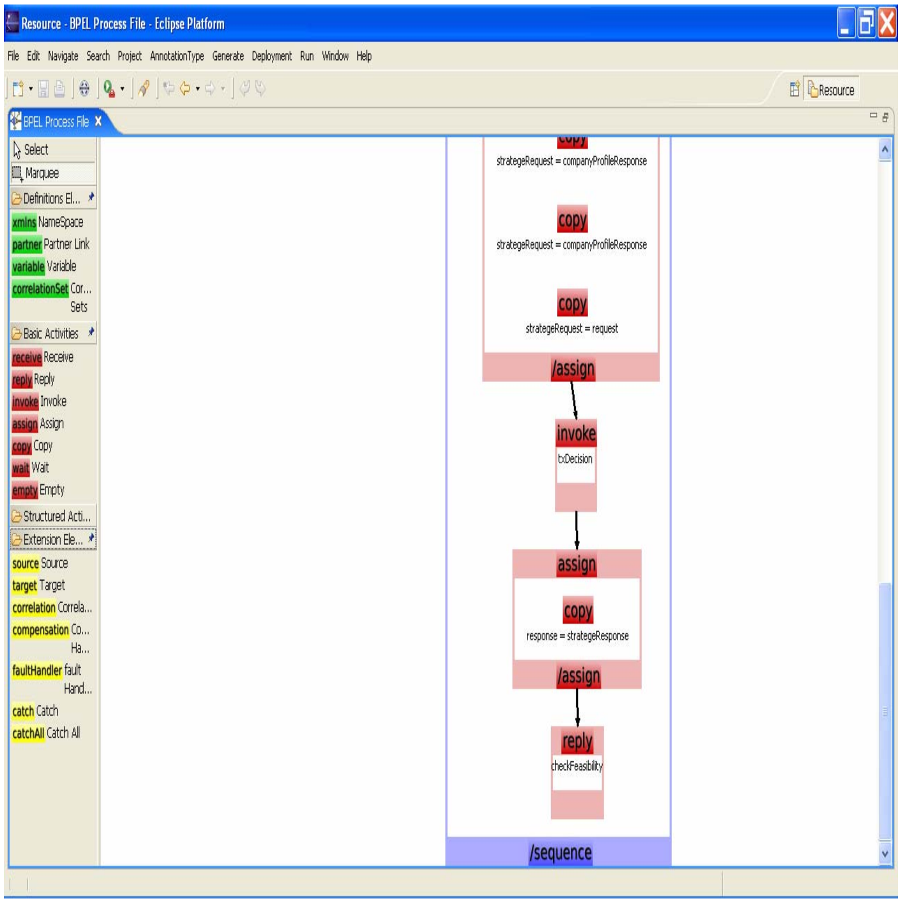


Figure 4-21. **The Complete Process II**

5. RELATED WORK

The methodologies of automatic Web Services composition are mainly separated into two categories: workflow composition and AI planning. To achieve automatic Web Service composition, there are two tasks that should be done. One is to generate an abstract process model by analyzing the process requirements; the other is to dynamically discover and bind the concrete services for the abstract process. The methods to solve the first task are usually related to AI planning and deductive theorem proving.

The workflow composition approach can be further divided into two levels: static workflow composition and dynamic workflow composition. In static workflow composition, the abstract process model is designed by the process developer and the concrete services are discovered, located and combined automatically. EFlow (Casati, Ilnicki, et al. 2000) adopts a graph oriented approach to implement a static workflow composition. There are three types of nodes in the graph: service, decision and event. Arcs represent the dependency relationships between two nodes. The service node gives the service's requirements which are used to discover and bind a concrete service to the process model either at process instantiation time, or at process execution time. Composite Service Definition Language (CSDL) (Casati, Sayal, et al, 2001) is another approach for static workflow composition. It pays attention not only to service level, but also to the operation level. Polymorphic Process Model (PPM) (Schuster, Georgakopoulos, et al. 2000) adopts a state machine to implement service based dynamic composition, while the sub services still follow the static workflow composition approach.

The AI planning problem in Web Service composition can be described using five attributes: the initial state, the final goal, all the available services, the state change functions and all the possible states. Golog is a logic programming language built on top of the situation calculus (a logical language for reasoning about the state change according to actions). Several papers (McIlraith, Son, et al. 2001; Narayanan and McIlraith, 2002; McIraith and Son, 2002) have extended Golog for automatic Web Service composition. The Planning Domain Definition Language (PDDL) (Schuster, Georgakopoulos, et al., 2000) is a language designed specifically to support AI planning. (McDermott, 2002) presented Web Service composition based on PDDL. The author introduced a new value – value of an action – to deal with the closed world assumption in the AI planning. In the closed world assumption, the literal's value is false if it does not exist in the current world. However, this assumption may fail in the automatic Web Service composition procedure, because a new Web Service can be dynamically created, changing the state of the knowledgebase. SWORD (Ponnekanti and Fox, 2002) is another toolkit for Web Service composition based on rule-based plan generation. It adopts the Entity Relationship Model (ER) to describe the set of preconditions and postconditions of a service. SHOP2 (Wu, Sirin, et al. 2003) is a Hierarchical Task Network (HTN) based planner and adopts OWL-S as its description language. A detailed survey of different approaches for workflow composition and AI planning that can be found in (Rao and Su, 2004).

6. CONCLUSION

In this work, we presented the current research and development in the area of Semantic Web Processes. We discussed the importance of Web Services discovery for designing a Web Process, and pointed out that the current non-semantic Web Service technologies do not support automatic Web Service discovery well. WSDL-S jointly proposed by the LSDIS lab and IBM research, is a Semantic Web Service standard candidate which intends to solve the problems of non-semantic Web Service technologies. To indicate the functionalities of WSDL-S in Semantic Web Process design, we applied our WSDL-S based tools – Radiant, Lumina, Saros – to the whole Web Service lifecycle. Following our WSDL-S based approach, the Web Process developer can find the partner services efficiently and effectively. Moreover, by cooperating with the Semantic Template technique in Lumina, Saros enables the developer to design an abstract Web Process by inserting virtual partners. This property facilitates partner search at design time or deployment time, and is helpful to adapt to the Web Service environments which change dynamically. Furthermore, this work highlights some key usability features of the WSDL-S based tool suite that assist process developers in easily designing complex business processes.

The potential of the WSDL-S approach is much more than what we showed in the current tool suite. For example, we do not take “preconditions” and “effects” into consideration. As demonstrated in pervious chapters, the search result can be more accurate by using non-functional constrains, such as cost, reliability, quality, etc. The project of adding WS-Policy to Web Service discovery is on going in the METEOR-S group.

In this work, we used the WSDL-based tool suite to design a Web Process semi-automatically. In some cases, the discovered partner services can not be inserted automatically, because we have yet to complete the data mapping part of this work.

7. QUESTIONS FOR DISCUSSION

Beginner:

1. What are the major entities in UDDI?
2. What is the usage of UDDI for the Web Service Composition?

Intermediate:

1. How is the Semantic Template helpful to build an abstract Web Process?
2. What are the advantages of allowing the developer to put multiple operations within one Semantic Template?
3. Give the mapping structure between WSDL-S and UDDI.

Advance:

1. What is the usage of a Virtual Partner?

Practical Exercises:

1. Download the three tools - Radiant, Lumina and Saros - from LSDIS web site from http://lsdis.cs.uga.edu/projects/meteor-s/downloads/. Use the Sumo-Finance ontology (same

URL as above) to annotate several WSDL files and publish them to an enhanced UDDI registry.

2. Discover the services by using the ontology concepts.
3. Design a Web Process.

8. SUGGESTED ADDITIONAL READING

- Narayanan S., and McIlraith S., "Simulation, Verification and Automated Composition of Web Services" Proceedings of the eleventh international conference on World Wide Web, May 2002, Pages: 77 – 88
- Verma K., Akkiraju R., Goodwin R., Doshi P., and Lee J., "On Accommodating Inter Service Dependencies in Web Process Flow Composition", 2004 AAAI Spring Symposium Series, March 2004, pp 37-43
- Rajasekaran P., Miller J., Verma K., Sheth A., "Enhancing Web Services Description and Discovery to Facilitate Composition", Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition, July 2004, pp. 55-68
- Ranjit Mulye, John A. Miller, Kunal Verma, Karthik Gomadam and Amit P. Sheth, "A Semantic Template Based Designer for Semantic Web Processes," Proceedings of the 3rd International Conference on Web Services (ICWS'05), Orlando, Florida (July 2005)
- Rama Akkiraju, Joel Farell, John A. Miller, Meena Nagarajan, Amit Sheth and Kunal Verma, "Web Service Semantics - WSDL-S," Proceedings of the W3C Workshop on Frameworks for Semantics in Web Service (*W3CW'05*), Innsbruck, Austria (June 2005)

- Rohit Aggarwal, Kunal Verma, John A. Miller and William Milnor, "Constraint Driven Web Service Composition in METEOR-S," *Proceedings of the 2004 IEEE International Conference on Services Computing (SCC'04),* Shanghai, China (September 2004) pp. 23-32
- Sivashanmugam K., Verma K., Sheth A., and Miller J., "Adding Semantics to Web Services Standards", 1st International Conference on Web Services, June 2003, pp. 395-401
- Kunal Verma, Kaarthik Sivashanmugam, Amit P. Sheth, Abhijit Patil, Swapna Oundhakar and John A. Miller, "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services," *Journal of Information Technology and Management (ITM),* Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) pp. 17-39. Kluwer Academic Publishers.

9. REFERENCES

Universal Description, Discovery and Integration (UDDI), *http://www.uddi.org/*

Web Services Business Process Execution Language (WS-BPEL), *http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=WS-BPEL*

METEOR-S: Semantic Web Services and Processes, *http://lsdis.cs.uga.edu/projects/METEOR-S/*

Narayanan S., and McIlraith S., "Simulation, Verification and Automated Composition of Web Services" Proceedings of the eleventh international conference on World Wide Web, May 2002, Pages: 77 - 88

Traverso P., and Pistore M., "Automated Composition of Semantic Web Services into Executable Processes", Proceedings of the 3rd International Semantic Web Conference (ISWC2004), pp. 380-394

W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. QUT Technical report, FIT-TR-2002-06, Queensland University of Technology, Brisbane, 2002

Rajasekaran P., Miller J., Verma K., Sheth A., "Enhancing Web Services Description and Discovery to Facilitate Composition”, Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition, July 2004, pp. 55-68

J. Miller, D. Palaniswami, A. Sheth, K. Kochut, H. Singh, "WebWork: METEOR's Web-based Workflow Management System", *Journal of Information Technology and Management (ITM)*, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) pp. 17-39. Kluwer Academic Publishers.

Nau D., Cao Y.,Lotem A.,and Muñoz-Avila H, "SHOP: Simple Hierarchical Ordered Planner", Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 1999, pp. 968 - 975

GEF: Graphical Editing Framework, *http://www.eclipse.org/gef/*

Verma K., Akkiraju R., Goodwin R., Doshi P., and Lee J., "On Accommodating Inter Service Dependencies in Web Process Flow Composition", 2004 AAAI Spring Symposium Series, March 2004, pp. 37-43

Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001, pp. 34-43

WSDL: Web Service Description Language, *http://www.w3.org/TR/wsdl*

Zarras A., Vassiliadis P., and Issarny V., "Model-Driven Dependability Analysis of Web Services", International Symposium on Distributed Objects and Applications, October 2004, pp. 69-79

Ponnekanti S., and Fox A., "SWORD: A Developer Toolkit for Web Service Composition", In Eleventh World Wide Web Conference (WWW2002, Web Engineering Track), Honolulu, Hawaii, May 2002.

Dogac A., "Exploiting Semantic of Web Services through ebXML Registries", Keynote Talk, 14th International Workshop on Research Issues on Data Engineering, Boston, 2004, *ttp://www.srdc.metu.edu.tr/~asuman/Dogac_RIDE_04_KeynoteAddress.ppt*

Dong X., Halevy A., Madhavan J., Nemes E., and Zhang J., "Similarity Search for Web Services", 30th VLDB Conference, August - September 2004, pp. 372-383

UML: Unified Modeling Language, Object Management Group, *http://www.uml.org/*

Paolucci M., Sycara K., and Kawamura T., "Delivering Semantic Web Services" In Proceedings WWW2003, May 2003, pp. 829

Sivashanmugam K., Verma K., Sheth A., and Miller J., "Adding Semantics to Web Services Standards", 1st International Conference on Web Services, June 2003, pp. 395-401.

Web Service Semantics – WSDL-S, W3C member submission, Version 1.0, 7 November 2005

Aggarwal R., Verma K., Miller J., and Milnor W., "Constraint Driven Web Service Composition in METEOR-S" Proceedings of the 2004 IEEE International Conference on Services Computing (SCC 2004), Shanghai, China (September 2004) pp. 23-32.

Verma K., Sivashanmugam K., Sheth A., Patil A., Oundhakar S., and Miller J., "METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services", Journal of Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) pp. 17-39. Kluwer Academic Publishers

Aggarwal R., Verma K., Miller J., and Milnor W., "Dynamic Web Service Composition in METEOR-S", Technical Report, LSDIS Lab, Computer Science Dept., UGA, May 2004.

Sirin E., Parsia B., and Hendler J., "Composition-driven filtering and selection of semantic Web services", AAAI Spring Symposium on Semantic Web Services, 2004, pp. 129-138

Sivashanmugam K., Miller J., Sheth A., and Verma K., "Framework for Semantic Web Process Composition", International Journal of Electronic Commerce (IJEC), Special Issue on Semantic Web Services and Their Role in Enterprise Application Integration and E-Commerce, Vol. 9, No. 2 (Winter 2004-5) pp. 71-106. M.E. Sharpe, Inc.

Reenskaug T., "The Model-View-Controller (MVC) Its Past and Present", http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern.pdf

OWL-S: Semantic Markup for Web Services, *http://www.daml.org/services/owl-s/1.0/owl-s.html*

Eclipse: Open platform for tool integration, *http://www.eclipse.org/*

Oracle BPEL Process Manager, http://www.oracle.com/technology/products/ias/bpel/index.html

IBM WebSphere: *http://www-306.ibm.com/software/websphere/*

A. Sheth, “Semantic Web Process Lifecycle: Role of Semantics in Annotation, Discovery, Composition and Orchestration”, Invited Talk WWW 2003 Workshop on E-Services and the Semantic Web, Budapest, Hungary, May 2003, http://www.ics.forth.gr/isl/essw2003/talks/seth_essw_semanticwebprocess.htm

K. Sivashanmugam, A. Sheth, J. Miller, K.Verma, R. Aggarwal, P. Rajasekaran, “Metadata and Semantics for Web Services and Processes”, 2003, Book Chapter, Datenbanken und Informationssysteme: Festschrift zum 60- Geburtstag von Gunter Schlageter, Benn et al Eds, Praktische Informatik I, Hagen, pp. 245-272.

A. Patil, S. Oundhakar, A. Sheth and K. Verma, “METEOR-S Web service Annotation Framework”, World Wide Conference, In the Proceedings of the 13th W3C Confernece, New York, USA, 2004, pp. 553-563.

K. Sivashanmugam, K. Verma and A. Sheth, “Discovery of Web Services in a Federated Registry Environment”, 2004, Proceedings of IEEE Second International Conference on Web Services, San Diego, California, USA, pp. 270-278.

Dumas M., and ter Hofstede A. H. M, “UML Activity Diagrams as a Workow Speci_cation Language”, *Lecture Notes in Computer Science*, vol. 2185, pp. 76–90, 2001

WS-Policy: Web Services Policy Framework, http://www-128.ibm.com/developerworks/library/specification/ws-polfram/

SWT: The Standard Widget Toolkit, http://www.eclipse.org/swt/

BPWS4J: The IBM Business Process Execution Language for Web Services Java^{TM} Run Time, *http://www.alphaworks.ibm.com/tech/bpws4j*

ActiveBPEL, Open Source BPEL Server, http://www.activebpel.org/

Kochut K., Sheth A., and Miller J., “ORBWork: A COBRA-Based Fully Distributed Scalable and Dynamic Workflow Enactment Service for METEOR”, Technical Report #UGA-CS-TR-98-006, Department of Computer Science, University of Georgia, 1998

METEOR: Managing End-To-End OpeRations, http://lsdis.cs.uga.edu/Projects/past/METEOR/

Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/soap/

Peltz C, “Web Services Orchestration and Choreography”, Web Services Journal, Volume 03 Issue 07, July 2003, pages 30-35

Doshi P., Goodwin R., Akkiraju R., and Verma K., “Dynamic Workflow Composition using Markov Decision Processes”, International Journal of Web Services Research, 2005, pp. 1-17

RosettaNet: *http://www.rosettanet.org*

Azami M., RosettaNet Ontology, *http://lsdis.cs.uga.edu/~azami/pips.html*

Kitamura Y., and Mizoguchi R.,"Functional Ontology for Functional Understanding", Twelfth International Workshop on Qualitative Reasoning (QR-98), Cape Cod, USA, AAAI Press, 1998, pp.77-87

Gardner D., Knuth K.H., Abato M., Erde S.M., White T., DeBellis R., and Gardner, Common data model for neuroscience data and data model interchange. J. Am. Med. Informatics Assoc. 8(1): 17-33, 2001

Kunal Verma, Karthik Gomadam, Amit P. Sheth, John A. Miller, Zixin Wu, "The METEOR-S Approach for Configuring and Executing Dynamic Web Processes", LSDIS METEOR-S project Technical Report . Date: 6-24-05

Ranjit Mulye, John A. Miller, Kunal Verma, Karthik Gomadam and Amit P. Sheth, "A Semantic Template Based Designer for Semantic Web Processes," Proceedings of the 3rd International Conference on Web Services (ICWS'05), Orlando, Florida (July 2005)

Rama Akkiraju, Joel Farell, John A. Miller, Meena Nagarajan, Amit Sheth and Kunal Verma, "Web Service Semantics - WSDL-S," *Proceedings of the W3C Workshop on Frameworks for Semantics in Web Service (*W3CW'05*),* Innsbruck, Austria (June 2005)

Kunal Verma, Kaarthik Sivashanmugam, Amit P. Sheth, Abhijit Patil, Swapna Oundhakar and John A. Miller, "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services," *Journal of Information Technology and Management (*ITM*),* Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) pp. 17-39. Kluwer Academic Publishers

X. Su and J. Rao. A Survey of Automated Web Service Composition Methods. In Proceedings of First International Workshop on Semantic Web Services and Web Process Composition, July 2004

H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. In Proceeding of 12th International Conference on Advanced Information Systems Engineering (CAiSE), Stockholm, Sweden, June 2000. Springer Verlag.

D. McDermott. “Estimated-regression planning for interactions with Web services”. In Proceedings of the 6th International Conference on AI Planning and Scheduling, Toulouse, France, 2002. AAAI Press

S. McIlraith and T. C. Son. Adapting Golog for composition of Semantic Web services. In Proceedings of the 8th International Conference on Knowledge Representation and Reasoning(KR2002), Toulouse, France, April 2002.

S. McIlraith, T. C. Son, and H. Zeng. Semantic Web services. IEEE Intelligent Systems, 16(2):46–53, March/April 2001

B. Medjahed, A. Bouguettaya, and A. K. Elmagarmid. Composing Web services on the Semantic Web. The VLDB Journal, 12(4), November 2003

S. Narayanan and S. McIlraith. Simulation, verification and automated composition of Web service. In Proceedings of the 11th International World Wide Web Conference, Honolulu, Hawaii, USA, May 2002. ACM. presentation available at http://www2002.org/presentations/narayanan.pdf

D. Wu, E. Sirin, J. Hendler, D. Nau, and B. Parsia. Automatic Web services composition using SHOP2. In Workshop on Planning for Web Services, Trento, Italy, June 2003.

John Colgrave and Karsten Januszewski, Technical Note Using WSDL in a UDDI Registry, Version 2.0.2, 2003

Assaf Arkin, Sid Askary, Ben Bloch, Francisco Curbera, Yaron Goland, Neelakantan Kartha, Canyang Kevin Liu, Satish Thatte, Prasad Yendluri, Alex Yiu, Web Service Business Execution Process Language Version 2.0 (WS-BPEL), http://www.oasis-open.org/committees/download.php/14616/wsbpel-specification-draft.htm.

Petia Wohed1, Wil M.P. van der Aalst Marlon Dumas, Arthur H.M. ter Hofstede, "Pattern Based Analysis of BPEL4WS", http://is.tm.tue.nl/staff/wvdaalst/publications/p175.pdf, QUT Technical report, FIT-TR-2002-04, Queensland University of Technology, Brisbane, 2002.

F. Casati, S. Sayal, M. Shan. Developing e-services for composing e-services. 2001. Submitted for publication andavailable on request.

10. APPENDIX

10.1 Appendix A: Semantic Template for the "Stock Quote" Service

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
    xmlns:targetNamespace="semantic template2"
    xmlns:wssem="http://www.ibm.com/xmlns/
                        WebServices/WSSemantics"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:ontology0="http://lsdis.cs.uga.edu/projects/meteor-
                             s/wsdl-s/ontologies/LSDIS_Finance.owl"
    xmlns:location0="http://localhost:8080/jsp-
                             examples/Finance.owl">
  <wsdl:message name="input1">
  </wsdl:message>
  <wsdl:message name="output1">
    <wsdl:part name="part0"
                  wssem:modelReference=
                            "ontology0#StockQuote.price"/>
    <wsdl:part name="part1"
                   wssem:modelReference=
                           "ontology0#StockQuote.volume"/>
  </wsdl:message>
  <wsdl:portType name="portType">
    <wsdl:operation wssem:modelReference=
                                   "ontology0#StockQuote">
      <wsdl:input message="input1"/>
      <wsdl:output message="output1"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:service>
  </wsdl:service>
</wsdl:definitions>
```

10.2 Appendix B: The BPEL File for the Use Case

```
<process  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
        name="simpleStockStratege"
        targetNamespace="urn:simpleStockStratege"
```

```
       xmlns:tns="urn:simpleStockStratege">
<partnerLinks>
  <partnerLink name="customer"  partnerLinkType=
      "tns:checkStockTrasactionFeasibilityPLT" myRole="caller"/>
  <partnerLink name="stockQuoteChecker"
                      xmlns:ns1="http://www.strikeiron.com"
                      partnerLinkType="ns1:stockQuoteCheckPTL"
                      myRole="stockQuoteChecker" />
  <partnerLink name="companyProfileChecker"
                    partnerLinkType="tns:companyProfileCheckerPTL"
                    partnerRole="companyProfileChecker"/>
  <partnerLink name="investEstimator"
                      xmlns:ns2="urn:investment"
                      partnerLinkType="ns2:investEstimationPTL"
                      partnerRole="investEstimator"/>
</partnerLinks>
<variables>
  <variable name="request" messageType="tns:input"/>
  <variable name="response" messageType="tns:output"/>
  <variable name="stockQuoteRequest"
                 xmlns:ns3="http://www.strikeiron.com"
                 messageType="ns3:GetOneQuoteSoapIn"/>
  <variable name="stockQuoteResponse"
```

```
                xmlns:ns4="http://www.strikeiron.com"
                messageType="ns4:GetOneQuoteSoapOut"/>
  <variable name="companyProfileRequest"
                xmlns:ns5="http://www.strikeiron.com"
                messageType="ns5:GetCompanyProfileSoapIn"/>
  <variable name="companyProfileResponse"
                xmlns:ns6="http://www.strikeiron.com"
                messageType="ns6:GetCompanyProfileSoapOut"/>
  <variable name="strategeRequest" xmlns:ns7="urn:investment"
                messageType="ns7:txDecisionRequest"/>
  <variable name="strategeResponse"
                xmlns:ns8="urn:investment"
                messageType="ns8:txDecisionResponse"/>
</variables>


<sequence>
  <receive name="receive"
          partnerLink="customer"
          portType="tns:CheckStockTransactionFeasibility"
         operation="checkFeasibility"
         variable="request" createInstance="yes">
  </receive>
```

```
<assign >
  <copy>
    <from variable="request" part="symbol"/>
    <to variable="stockQuoteRequest" part="TickerSymbol"/>
  </copy>
  <copy>
    <from variable="request" part="symbol"/>
    <to variable="companyProfileRequest" part="ticker"/>
  </copy>
</assign>
<flow>
    <invoke name="invokeStockQuote"
              partnerLink="stockQuoteChecker"
              xmlns:ns9="http://www.strikeiron.com"
              portType="ns9:BasicRealTimeQuotesSoap"
              operation="getOneQuote_simple"
              inputVariable="stockQuoteRequest"
              outputVariable="stockQuoteResponse">
    </invoke>
    <invoke name="invokeCompanyProfileCheck"
               partnerLink="companyProfileChecker"
               xmlns:ns10="http://www.strikeiron.com"
               portType="ns10:ZacksCompanySoap"
```

```
                operation="GetCompanyProfile_simple"
                inputVariable="companyProfileRequest"
                outputVariable="companyProfileResponse">
     </invoke>
   </flow>
  <assign >
    <copy>
      <from variable="stockQuoteResponse" part="Last"/>
      <to variable="strategeRequest" part="current_price"/>
    </copy>
    <copy>
      <from variable="stockQuoteResponse" part="BidQuantity"/>
      <to variable="strategeRequest" part="bid_quantity"/>
    </copy>
    <copy>
      <from variable="stockQuoteResponse" part="Volume"/>
      <to variable="strategeRequest" part="volume"/>
    </copy>
    <copy>
      <from variable="companyProfileResponse"
              part="Est_EPS_F1"/>
      <to variable="strategeRequest" part="eps"/>
    </copy>
  <copy>
    <from variable="companyProfileResponse"
               part="W52_High_Price"/>
    <to variable="strategeRequest" part="year_high"/>
  </copy>
  <copy>
    <from variable="request" part="investAmount"/>
    <to variable="strategeRequest" part="investAmount"/>
  </copy>
</assign>
<invoke name="invokeStratege"
             partnerLink="investEstimator"
             xmlns:ns11="urn:investment"
             portType="ns11:StockBuyingStratege"
            operation="txDecision"
            inputVariable="strategeRequest"
            outputVariable="strategeResponse">
</invoke>
<assign >
  <copy>
    <from variable="strategeResponse"
               part="txDecisionReturn"/>
    <to variable="response" part="result"/>
```

```
     </copy>
    </assign>
    <reply name="reply"
            partnerLink="customer"
            portType="tns:CheckStockTransactionFeasibility"
            operation="checkFeasibility"
         variable="response">
    </reply>
   </sequence>
</process>
```

10.3 Appendix C: The Process WSDL File for the Use Case

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:simpleStockStratege"
              xmlns:tns="urn:simpleStockStratege"
              xmlns:plnk=
             "http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:message name="output">
    <wsdl:part name="result" type="xsd:boolean"/>
  </wsdl:message>
  <wsdl:message name="input">
  <wsdl:part name="symbol" type="xsd:string"/>
  <wsdl:part name="investAmount" type="xsd:double"/>

</wsdl:message>
<wsdl:portType name="CheckStockTransactionFeasibility">
  <wsdl:operation name="checkFeasibility">
    <wsdl:input message="tns:input"/>
    <wsdl:output message="tns:output"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:service name="simpleStockStrategeBP">
</wsdl:service>
  <plnk:partnerLinkType
            name="checkStockTrasactionFeasibilityPLT">
     <plnk:role name="caller">
       <plnk:portType name="CheckStockTransactionFeasibility"/>
      </plnk:role>
  </plnk:partnerLinkType>

  <plnk:partnerLinkType name="stockQuoteCheckPTL">
     <plnk:role name="stockQuoteChecker">
         <plnk:portType xmlns:n1="http://www.strikeiron.com"
```

```
                              name="n1:BasicRealTimeQuotesSoap"/>
        </plnk:role>
    </plnk:partnerLinkType>


    <plnk:partnerLinkType name="companyProfileCheckerPTL">
        <plnk:role>
            <plnk:portType xmlns:n2="http://www.strikeiron.com"
                              name="n2:ZacksCompanySoap"/>
        </plnk:role>
    </plnk:partnerLinkType>


    <plnk:partnerLinkType name="investEstimationPTL">
        <plnk:role>
            <plnk:portType xmlns:n3="urn:investment"
                              name="n3:StockBuyingStratege"/>
        </plnk:role>
    </plnk:partnerLinkType>
</wsdl:definitions>
```

CHAPTER 5

EVALUATION

In this chapter, we choose three aspects to evaluate our work: 1. efficient and effective services discovery; 2. accurate discovery of the specific operations; 3. semi-automatic BPEL process design.

To prove that our work is useful in a real situation, we use real third party domain ontologies in our evaluation experiments. We obtained a corpus of 424 Web Services form SALCentral.org and XMethods.com. However, due to lack of the relevant domain ontologies, we only use the services in the “money”, “weather” and “airport” categories. We also employ the “loanApprover.wsdl” and “loanAssessor.wsdl” files from ActiveBpel sample scenario, and build two investment strategy services to fit our sample scenario.

5.1 Efficient and Effective Services Discover

There are five public UDDI registries: IBM, Microsoft, HP, XMethods, and Systinet (see Appendix A).

These registries do not supply a unified WSDL to UDDI mapping format, and their syntactic search mechanism limits the search range of the suitable Web Services. To explain this issue, we pick the Microsoft UDDI registry and the XMethods registry as the comparison representatives to our enhanced UDDI registry – Lumina. Because each Web Service provider has his/her own way to publish the service, we respect this situation and use those existing published services directly. Our experimental scenario is to find all the suitable Web Services which implement a “Stock Quote” application. The direct way to search for these services is based on the services names. The XMethods registry gives the direction to publish WSDL files using TModels, while the Microsoft registry does not have such a constraint. We need to take care of both the Business Service and TModel in the Microsoft registry. Table 5-1 shows the precision of the search results by using different possible keywords in a “Stock Quote” service.

$$\text{Precision} = \frac{number\_of\_correctly\_dis\mathrm{cov}ered\_services}{number\_of\_dis\mathrm{cov}ered\_services}$$

$$\text{Recall} = \frac{number\_of\_correctly\_dis\mathrm{cov}ered\_services}{number\_of\_all\_correct\_services}$$

The XMethods registry is small enough that we can count all the “Stock Quote” services inside it. The total number of “Stock Quote” services is 13, while we only found 9 of them based on the keywords above, thus the recall is 69%. The Microsoft registry is so huge that it includes 1000 Business Services and 1000 TModels, so we cannot figure out how many “Stock Quote” services are in this registry to give the recall value.

Table 5-1: Precision of “Stock Quote” Web Services Discovery Result on the Regular UDDI Registries

keywords	XMethods	Microsoft	
	TModel	Service	TModel
Stock quote	3/3 = 100%	17/17 = 100%	28/28 = 100%
stockquote	3/3 = 100%	4/4 = 100%	18/18 = 100%
stock	7/7 = 100%	64/70 = 91%	18/18 = 100%
delayed	1/1 = 100%	5/5 = 100%	1/1 = 100%
Real time	1/3 = 33%	0/1 = 0	0/0
realtime	0/0	0/1 = 0	0/0

From the data results shown in the Table 5-1, these registries cannot produce satisfying discovery result due to the following reasons:

- Based on the same concept, the discovery result is different because of using different keywords.
- The discovery result set is likely missing some matching services because those services use some other words to represent services names rather than the keywords in common sense.
- Some registries do not have a unified structure for publishing WSDL files. This problem confuses service requesters when they are discovering WSDL file based services.

In our enhanced registry – Lumina, we annotated 8 “Stock Quote” Web Services which come from the services in the corpus. These services include 11 operations in all. We use the SUMO-Finance ontology to annotate these services. In Table 5-2, we list the service names, the operations within them and the corresponding ontological concept to each of the operation.

Table 5-3 gives the search result based on the three ontological concepts: StockQuote, RealtimeStockQuote, DelayedStockQuote.

This is a prelimilary research experiment. The number of the services in our enhanced UDDI registry is not large enough to give a conclusive evaluation of the WSDL-S based discovery approach as it only uses 27 relevant services form XMethods and SALCentral. However, we are planning to add many services from the Microsoft UBR in our next evaluation.

Table 5-2: The "Stock Quote" Services in the Enhanced UDDI Registry

Services	Operations	Ontological Concepts
DelayedStockQuote	GetQuote	OntologyNS#DelayedStockQuote
	GetQuickQuote	OntologyNS#RealTimeStockQuote
BasicRealTimeQuotes	GetOneQuote	OntologyNS#RealTimeStockQuote
StockQuotes	GetStockQuotes	OntologyNS#DelayedStockQuote
StockScraper	GetQuote	OntologyNS#StockQuote
StockServices	GetQuotes	OntologyNS#StockQuote
	GetQuickQuotes	OntologyNS#DelayedStockQuote
StockQuote	GetQuote	OntologyNS#StockQuote
DOTSFastQuote	GetStockInfo	OntologyNS#DelayedStockQuote
Nexus6Studio_x0020_Stock_x0020_Quote	GetQuickQuote	OntologyNS#StockQuote
	GetDetailedQuote	OntologyNS#StockQuote

Table 5-3: Precision of Lumina Discovery Result Using StockQuote Ontological Concepts

Ontological Concepts	**Enhanced UDDI Registry**	
	TModel (Operation)	**Service**
Ontology# StockQuote	11 / 11 = 100%	8 / 8 = 100%
Ontology# DelayedStockQuote	9 / 9 = 100%	7 / 7 = 100%
Ontology# RealTimeStockQuote	7 /7 = 100%	6 / 6 = 100%

Table 5-4: Recall of Lumina Discovery Result Using StockQuote Ontological Concepts

Ontological Concepts	Enhanced UDDI Registry	
	TModel (Operation)	Service
Ontology# StockQuote	11 / 11 = 100%	8 / 8 = 100%
Ontology# DelayedStockQuote	9 / 9 = 100%	7 / 7 = 100%
Ontology# RealTimeStockQuote	7 /7 = 100%	6 / 6 = 100%

All the operations which are highly related to the ontological concepts are found. The only work the user needs to do is in two steps: 1. add the ontology URL; 2. choose the operation's ontological concepts. This procedure takes no more than 5 minutes. Moreover, the discovery result supplies not only the "Service" level information, but also the "Operation" level information. In the other tested UDDI registries, we spent hours to search for these services.

5.2 Accurate to Discover the Specific Operations

In most Semantic Web Service work, there is an "Atomic Process" concept which represents a service that expects one message and returns one message in response. However, WSDL permits multiple operations, even multiple portTypes may exist in a single WSDL file. If the service provider only publishes the WSDL file as a "Service", the service requester cannot know the information of the operations within the service. If the service provider publishes each operation as a "Service", then the relationship that links the operations to the service is lost. In our work, we publish each operation as a "TModel", publish the service information to a "Business Service", and use the "BindingTemplate" structure to bind all the "TModel" instances to the "Service". Lumina supports the discovery based on the "operation" level. The advantages of this approach are the following:

1. In service composition, we need not only to match the operation concept, but also to match the input and output concepts for successive operations.

2. The service requester is able to find multiple operations within one service.

The first advantage is obvious. The second advantage can solve many pragmatic problems. Think about such a scenario, an online book seller only sells book to the customers who have logged in. To simplify the scenario, we consider two steps to design the Web Process: first log in, and then buy the books. Of course, the operations related to the two steps should come from the same partner service.

There are twenty seven Web Services which include fifty one operations in our enhanced UDDI registry. In the following two experiments, we will assign different requirements to search for the specific operations within these Web Services. The twenty seven Web Services belong to eight categories: stock quote finder (eight services), currency information / converter (four services), loan service (two services), loan rate calculation (two services), weather forecast (three services), airport / flight information (three services), company profile (one service), credit card validation (two services) and stock investment service (two services).

In the first experiment, we assign the requirements based on“currency” ontology. This is a small ontology and has two classes: the “Currency” and the “CurrencyRate”. Table 5-5 shows the searching requirements and the related discovery results.

In the second experiment, we use the “Weather Domain” in“travel” ontology. We give the discovery results in Table 5-6.

Table 5-5: Web Services Based on the Currency Ontology

Searching Requirements	Searching Results
Operation: ontologyNS#CurrencyRate Input: /n Output: /n	Currencyws: GetRate CurrencyConvertor: ConvertionRate CurrencyExchangeService: getRate

Operation: ontologyNS#Currency
Input: /n
Output: /n
	Country: GetCurrencies;
	GetCurrencyByCountry;
	GetCurrencyCode;
	GetCurrencyCodeByCurrencyName

Operation: ontologyNS#Currency.country
Input: /n
Output: /n
	Country: GetCountries;
	GetCountryByCountryCode;
	GetCountryByCurrencyCode;
	GetISOCountryCodeByCountryName

Operation: ontologyNS#Currency Input: ontologyNS#Currency.country Output: /n	Country: GetCurrencyByCountry
Operation: ontologyNS#Currency Input: ontologyNS#Currency.name Output: ontologyNS#Currency.code	Country: GetCurrencyCodeByCurrencyName
Operation: ontologyNS#Currency.country Input: ontologyNS#Currency.code Output: /n	Country: GetCountryByCurrencyCode;
Operation: ontologyNS#Currency.country Input: ontologyNS#Currency.country_iso_code Output: ontologyNS#Currency.country	Country: GetCountryByCountryCode;

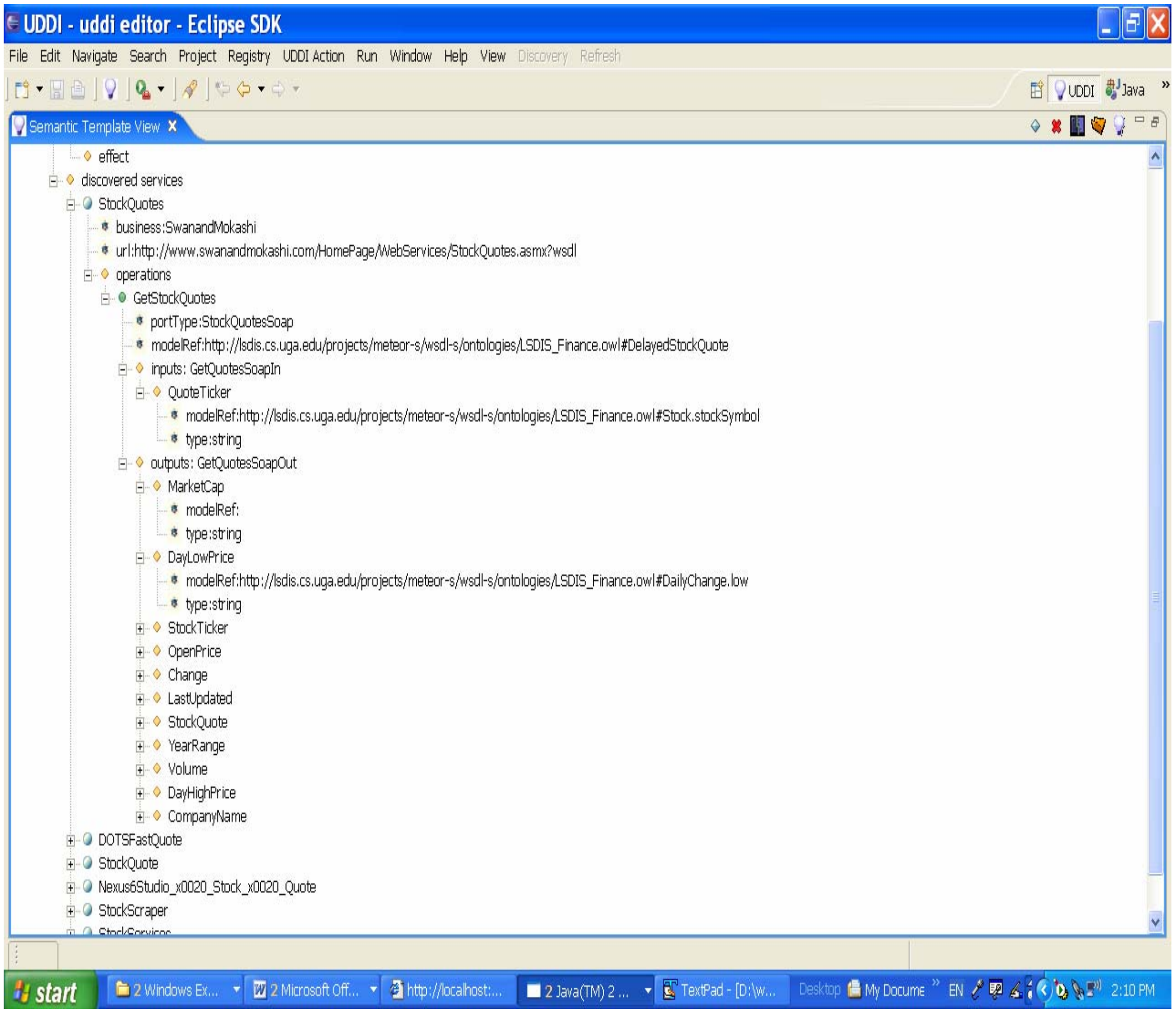
Table 5-6: Web Services Based on the "Weather Domain" in the Travel Ontology

Searching Requirements	Searching Results
Operation: ontologyNS#Forcast Input: /n Output: /n	ForecastByZip: GetForecastByZip WeatherFetcher: GetWeather DOTSFastWeather: GetWeatherByCityState GetWeatherByZip
Operation: ontologyNS#Forcast Input: ontologyNS#ZipCode Output: /n	ForecastByZip: GetForecastByZip DOTSFastWeather: GetWeatherByZip WeatherFetcher: GetWeather
Operation: ontologyNS#Forcast Input: ontologyNS#City Output: /n	DOTSFastWeather: GetWeatherByCityState
Operation: ontologyNS#Forcast Input: ontologyNS#City Output: ontologyNS#Weather.hasPercipitation	DOTSFastWeather: GetWeatherByZip

From the two experiments, we demonstrate "operation" level discovery. The service requester gives the ontological concepts to describe the *operation*, *input* and *output*, and then the discovery result will be limited to include only the highly suitable services. The more requirements the requester gives, the more accurate the result is. This feature is quite useful in Web Services composition. Moreover, this discovery result can further be improved by adding some non-functional constrains in WS-Policy.

5.3 Semi-automatic Design of the BPEL Processes

It is difficult to design a BPEL process even with the help of some GUI based BPEL design tools. The reason is that the partner service discovery result usually only supplies a WSDL file location, but little detailed information needed to wire it into in the process. In contrast to the general UDDI registries, Lumina supplies the detailed information for both the service and its operations. All the partner service information needed to fill in the process can be found from our discovery results. Figure 5-1 shows a sample discovery result. In this work, the developer can choose the partner services from our discovered services result according to some non-functional requirements or the data mapping between input and output message types of the successive operations. In our future work, by adopting the DataMapping technique and WS-Policy, we can implement automatically Web Process composition instead of the current semi-automatic composition.



```
effect
discovered services
  StockQuotes
    business:SwanandMokashi
    url:http://www.swanandmokashi.com/HomePage/WebServices/StockQuotes.asmx?wsdl
    operations
      GetStockQuotes
        portType:StockQuotesSoap
        modelRef:http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_Finance.owl#DelayedStockQuote
        inputs: GetQuotesSoapIn
          QuoteTicker
            modelRef:http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_Finance.owl#Stock.stockSymbol
            type:string
        outputs: GetQuotesSoapOut
          MarketCap
            modelRef:
            type:string
          DayLowPrice
            modelRef:http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_Finance.owl#DailyChange.low
            type:string
          StockTicker
          OpenPrice
          Change
          LastUpdated
          StockQuote
          YearRange
          Volume
          DayHighPrice
          CompanyName
  DOTSFastQuote
  StockQuote
  Nexus6Studio_x0020_Stock_x0020_Quote
  StockScraper
```

Figure 5-1: Sample Discovery Result

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this work, we presented the current research and development in the area of Semantic Web Processes. We discussed the importance of Web Services discovery for designing a Web Process, and pointed out that the current non-semantic Web Service technologies do not support automatic Web Service discovery well. WSDL-S, jointly proposed by the LSDIS lab and IBM research, is a Semantic Web Service standard candidate which intends to solve the problems of non-semantic Web Service technologies. To indicate the functionalities of WSDL-S in Semantic Web Process design, we applied our WSDL-S based tools – Radiant, Lumina, Saros – in the whole Web Service lifecycle. Following our WSDL-S based approach, the Web Process developer can find the partner services efficiently and effectively. Moreover, by cooperating with the Semantic Template technique in Lumina, Saros enables the developer to design an abstract Web Process by inserting virtual partners. This property facilitates partner search at design time or deployment time, and is helpful to adapt to Web Service environments which change dynamically. Furthermore, this work highlights some key usability features of the WSDL-S based tool suite that assist process developers in designing complex business processes.

By carrying out several experiments, we evaluated our work in three ways: 1. efficient and effective discovery of services; 2. accurate discovery of the specific operations; 3. facilitation semi-automatic design of BPEL processes.

The potential of the WSDL-S approach is much more than what we showed in the current tool suite. For example, we do not take “preconditions” and “effects” into consideration. As demonstrated in pervious chapters, the search result can be more useful by using non-functional constraints, such as cost, reliability, quality, etc. The project of adding WS-Policy to Web Service discovery is ongoing in the METEOR-S group.

In this work, we used the WSDL-based tool suite to design a Web Process semi-automatically. In some cases, the discovered partner services can not be inserted automatically, because we have yet to complete the data mapping part of this work.

Future research directions that we are beginning to explore are in the following:

- Develop a constraint analyzer to extend our WSDL-S based tool suite.
- Adopt data mapping techniques to implement Web Process composition automatically.
- Extend the discovery, composition by using "preconditions" and "effects" to achieve a more accurate result.
- Develop a process monitor to trace the process at execution time.
- Organize Saros to run in the same workspace as Lumina, and supply the Drag and Drop functionality to further ease the work of the process developer.

REFERENCES

[1] Universal Description, Discovery and Integration (UDDI), http://www.uddi.org/

[2] Web Services Business Process Execution Language (WS-BPEL), http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=WS-BPEL.

[3]METEOR-S: Semantic Web Services and Processes,
http://lsdis.cs.uga.edu/projects/METEOR-S/.

[4] Narayanan S., and McIlraith S., "Simulation, Verification and Automated Composition of Web Services" Proceedings of the eleventh international conference on World Wide Web, May 2002, pp. 77 – 88.

[5] Traverso P., and Pistore M., "Automated Composition of Semantic Web Services into Executable Processes", Proceedings of the 3rd International Semantic Web Conference (ISWC2004), pp. 380-394.

[6] W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. QUT Technical report, FIT-TR-2002-06, Queensland University of Technology, Brisbane, 2002

[7] Rajasekaran P., Miller J., Verma K., Sheth A., "Enhancing Web Services Description and Discovery to Facilitate Composition", Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition, July 2004, pp. 55-68

[8] J. Miller, D. Palaniswami, A. Sheth, K. Kochut, H. Singh, "WebWork: METEOR's Web-based Workflow Management System", *Journal of Intelligent Information Systems: Integrating Artificial Intelligence and Database Technologies (JIIS),* Special Issue on Workflow Management Systems, Vol. 10, No. 2 (March/April 1998) pp. 185-215. Kluwer Academic Publishers.

[9] Nau D., Cao Y.,Lotem A.,and Muñoz-Avila H, "SHOP: Simple Hierarchical Ordered Planner", Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 1999, Pages 968 – 975.

[10] GEF: Graphical Editing Framework, *http://www.eclipse.org/gef./*

[11] Verma K., Akkiraju R., Goodwin R., Doshi P., and Lee J., "On Accommodating Inter Service Dependencies in Web Process Flow Composition", 2004 AAAI Spring Symposium Series, March 2004, pp 37-43.

[12] Berners-Lee, T., Hendler, J., Lassila, O., “The Semantic Web”, Scientific American, May 2001, pp. 34-43.

[13] WSDL: Web Service Description Language, http://www.w3.org/TR/wsdl.

[14] Zarras A., Vassiliadis P., and Issarny V., “Model-Driven Dependability Analysis of Web Services”, International Symposium on Distributed Objects and Applications, October 2004, pp. 69-79.

[15] Ponnekanti S., and Fox A., “SWORD: A Developer Toolkit for Web Service Composition”, In Eleventh World Wide Web Conference (WWW2002, Web Engineering Track), Honolulu, Hawaii, May 2002.

[16] Dogac A., “Exploiting Semantic of Web Services through ebXML Registries”, Keynote Talk, 14th International Workshop on Research Issues on Data Engineering, Boston, 2004, http://www.srdc.metu.edu.tr/~asuman/Dogac_RIDE_04_KeynoteAddress.ppt.

[17] Dong X., Halevy A., Madhavan J., Nemes E., and Zhang J., “Similarity Search for Web Services”, 30th VLDB Conference, August – September 2004, pp. 372-383.

[18] UML: Unified Modeling Language, Object Management Group, http://www.uml.org/.

[19] Paolucci M., Sycara K., and Kawamura T., “Delivering Semantic Web Services” In Proceedings WWW2003, May 2003, pp. 829.

[20] Sivashanmugam K., Verma K., Sheth A., and Miller J., “Adding Semantics to Web Services Standards”, 1st International Conference on Web Services, June 2003, pp. 395-401.

[21] Miller J., Verma K., Rajasekaran P., Sheth A., Aggrawal R., and Sivashanmugam K., “WSDL-S: A Proposal to W3C WSDL 2.0 Committee”, http://lsdis.cs.uga.edu/projects/WSDL-S/wsdl-s.pdf

[22] Aggarwal R., Verma K., Miller J., and Milnor W., “Constraint Driven Web Service Composition in METEOR-S” Proceedings of the 2004 IEEE International Conference on Services Computing (SCC 2004), Shanghai, China (September 2004) pp. 23-32.

[23] Verma K., Sivashanmugam K., Sheth A., Patil A., Oundhakar S., and Miller J., “METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services”, Journal of Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) pp. 17-39. Kluwer Academic Publishers.

[24] Aggarwal R., Verma K., Miller J., and Milnor W., “Dynamic Web Service Composition in METEOR-S”, Technical Report, LSDIS Lab, Computer Science Dept., UGA, May 2004.

[25] Sirin E., Parsia B., and Hendler J., "Composition-driven filtering and selection of semantic Web services", AAAI Spring Symposium on Semantic Web Services, 2004, pp. 129-138.

[26] Sivashanmugam K., Miller J., Sheth A., and Verma K., "Framework for Semantic Web Process Composition", International Journal of Electronic Commerce (IJEC), Special Issue on Semantic Web Services and Their Role in Enterprise Application Integration and E-Commerce, Vol. 9, No. 2 (Winter 2004-5) pp. 71-106. M.E. Sharpe, Inc.

[27] Reenskaug T., "The Model-View-Controller (MVC) Its Past and Present", http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern.pdf.

[28] OWL-S: Semantic Markup for Web Services, http://www.daml.org/services/owl-s/1.0/owl-s.html.

[29] Eclipse: Open platform for tool integration, http://www.eclipse.org/.

[30] Oracle BPEL Process Manager,
http://www.oracle.com/technology/products/ias/bpel/index.html.

[31] IBM WebSphere: http://www-306.ibm.com/software/websphere/.

[32] A. Sheth, "Semantic Web Process Lifecycle: Role of Semantics in Annotation, Discovery, Composition and Orchestration", Invited Talk WWW 2003 Workshop on E-Services and the Semantic Web, Budapest, Hungary, May 2003,
http://www.ics.forth.gr/isl/essw2003/talks/seth_essw_semanticwebprocess.htm.

[33] K. Sivashanmugam, A. Sheth, J. Miller, K.Verma, R. Aggarwal, P. Rajasekaran, "Metadata and Semantics for Web Services and Processes", 2003, Book Chapter, Datenbanken und Informationssysteme: Festschrift zum 60- Geburtstag von Gunter Schlageter, Benn et al Eds, Praktische Informatik I, Hagen, pp. 245-272.

[34] A. Patil, S. Oundhakar, A. Sheth and K. Verma, "METEOR-S Web service Annotation Framework", World Wide Conference, In the Proceedings of the 13th W3C Confernece, New York, USA, 2004, pp. 553-563.

[35] K. Sivashanmugam, K. Verma and A. Sheth, "Discovery of Web Services in a Federated Registry Environment", 2004, Proceedings of IEEE Second International Conference on Web Services, San Diego, California, USA, pp. 270-278.

[36] Dumas M., and ter Hofstede A. H. M, "UML Activity Diagrams as a Workow Speci_cation Language", *Lecture Notes in Computer Science*, vol. 2185, pp. 76–90, 2001.

[37] WS-Policy: Web Services Policy Framework,

http://www-128.ibm.com/developerworks/library/specification/ws-polfram/.

[38] SWT: The Standard Widget Toolkit, http://www.eclipse.org/swt/.

[40] BPWS4J: The IBM Business Process Execution Language for Web Services Java^{TM} Run Time, http://www.alphaworks.ibm.com/tech/bpws4j.

[41] ActiveBPEL, Open Source BPEL Server, http://www.activebpel.org/.

[42] Kochut K., Sheth A., and Miller J., "ORBWork: A COBRA-Based Fully Distributed Scalable and Dynamic Workflow Enactment Service for METEOR", Technical Report #UGA-CS-TR-98-006, Department of Computer Science, University of Georgia, 1998.

[43] METEOR: Managing End-To-End OpeRations,
http://lsdis.cs.uga.edu/Projects/past/METEOR/.

[44] Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/soap/.

[45] Peltz C, "Web Services Orchestration and Choreography", Web Services Journal, Volume 03 Issue 07, July 2003, pp. 30-35.

[46] Doshi P., Goodwin R., Akkiraju R., and Verma K., "Dynamic Workflow Composition using Markov Decision Processes", International Journal of Web Services Research, 2005, pp. 1-17.

[47] RosettaNet: http://www.rosettanet.org.

[48] Azami M., RosettaNet Ontology, http://lsdis.cs.uga.edu/~azami/pips.html.

[49] Kitamura Y., and Mizoguchi R.,"Functional Ontology for Functional Understanding", Twelfth International Workshop on Qualitative Reasoning (QR-98), Cape Cod, USA, AAAI Press, 1998, pp.77-87.

[50] Gardner D., Knuth K.H., Abato M., Erde S.M., White T., DeBellis R., and Gardner, Common data model for neuroscience data and data model interchange. J. Am. Med. Informatics Assoc. 8(1): 17-33, 2001.

[51] Kunal Verma, Karthik Gomadam, Amit P. Sheth, John A. Miller, Zixin Wu, "The METEOR-S Approach for Configuring and Executing Dynamic Web Processes", LSDIS METEOR-S project Technical Report . Date: 6-24-05.

[52] OWL-S: Semantic Markup for Web Services, http://www.daml.org/services/owl-

s/1.1/overview/.

[53] Semantic Web Service Ontology (SWSO), http://www.w3.org/Submission/SWSF/.

[54] Web Service Modeling Ontology (WSMO), http://www.w3.org/Submission/WSMO/.

[55] Technical Note Using WSDL in a UDDI Registry, Version 2.0.2, http://www.oasis-open.org/committees/uddi-spec/doc/tn/uddi-spec-tc-tn-wsdl-v2.htm.

[56] OWL Web Ontology Language, http://www.w3.org/TR/owl-features/.

[57] Resource Description Language (RDF), http://www.w3.org/RDF/.

[58] Taxonomy: NAICS – North Amecian Industrial Classification System.
http://www.census.gov/epcd/www/naics.html;

[59] ISO 3166 Code, http://userpage.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

[60] UNSPCE Classification, http://www.zycus.com/resource-centre/unspsc-classification.html

[61] UDDI4J, http://uddi4j.sourceforge.net/

[62] Assaf Arkin, Sid Askary, Ben Bloch, Francisco Curbera, Yaron Goland, Neelakantan Kartha, Canyan Kevin Liu, Satish Thatte, Prasad Yendluri, Alex Yiu, Web Service Business Execution Process Language Version 2.0 (WS-BPEL),
http://www.oasis-open.org/committees/download.php/14616/wsbpel-specification-draft.htm.

[63] Petia Wohed1, Wil M.P. van der Aalst Marlon Dumas, Arthur H.M. ter Hofstede, "Pattern Based Analysis of BPEL4WS", http://is.tm.tue.nl/staff/wvdaalst/publications/p175.pdf, QUT Technical report, FIT-TR-2002-04, Queensland University of Technology, Brisbane, 2002.

[64] F. Casati, S. Sayal, M. Shan. Developing e-services for composing e-services. 2001. Submitted for publication andavailable on request.

[65] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. In
Proceeding of 12th International Conference on Advanced Information Systems En-
gineering (CAiSE), Stockholm, Sweden, June 2000. Springer Verlag.

APPENDIX A – UDDI Universal Business Registries

Microsoft UDDI	• General Info : http://uddi.microsoft.com • Live Inquiries: http://uddi.microsoft.com:80/inquire • Live Publish : https://uddi.microsoft.com:443/publish • Test Inquiries: http://test.uddi.microsoft.com:80/inquire • Test Publish : https://test.uddi.microsoft.com:443/publish • Test Inq (v2) : http://uddi.rte.microsoft.com:80/inquire • Test Pub (v2) : https://uddi.rte.microsoft.com:443/publish
IBM UDDI	• General Info : https://www-3.ibm.com/services/uddi/protect/registry.html • Live Inquiries: http://www-3.ibm.com:80/services/uddi/inquiryapi • Live Publish : https://www-3.ibm.com:443/services/uddi/protect/publishapi • Test Inquiries: http://www-3.ibm.com:80/services/uddi/testregistry/inquiryapi • Test Publish : https://www-3.ibm.com:443/services/uddi/testregistry/protect/publishapi • Test Inq (v2) : http://www-3.ibm.com:80/services/uddi/v2beta/inquiryapi • Test Pub (v2) : https://www-3.ibm.com:443/services/uddi/v2beta/protect/publishapi
HP UDDI	• General Info :http://uddi.hp.com • Live Inquiries: http://uddi.hp.com/ubr/inquire • Live Publish : https://uddi.hp.com/ubr/publish
XMethods UDDI	• General Info : http://xmethods.net • Live Inquiries: http://uddi.xmethods.net/inquire • Live Publish : None (XMethods UDDI is private)
Systinet	• General Info : http://systinet.com • Inquiries : http://www.systinet.com:80/wasp/uddi/inquiry/ • Publish : http://www.systinet.com:443/wasp/uddi/publishing/