THE METEOR-S FRAMEWORK FOR SEMANTIC WEB PROCESS COMPOSITION

by

KAARTHIK SIVASHANMUGAM

(Under the direction of Dr. John. A. Miller)

ABSTRACT

The Web services have been recognized to have the potential to revolutionize e-commerce. The potential for businesses to be able to interact with each other on the fly is very appealing. To date, however, the activity of creating Web processes using Web services has been handled mostly at the syntactic level. Current composition standards focus on building the process based on the interface description of the participating services. The limitation of a rigid approach is that it does not allow businesses to dynamically change partners and services. We enhance the current process composition techniques by using Semantic Process Templates to capture the semantic requirements of the process. The semantic process templates can act as configurable modules for common industry practices maintaining the semantics of the participating activities, control flow, intermediate calculations, conditional branches and exposing it in a industry accepted interface. The templates are instantiated to form executable processes according to the semantics of the corresponding templates. The use of ontologies in template definition allows much richer description of activity requirements and a more effective way of locating services to carry out the activities in the executable Web process. During discovery of services we consider not only functionality, but also the quality of the services. Our unique approach combines the expressive power of the present Web service composition standards and the advantages of the semantic Web techniques for template definition and service discovery. The prototype implementation of the framework for building the templates and generating the processes is discussed.

INDEX WORDS: Web Service Composition, Semantic Web Processes, Semantic Web Service Discovery, Semantic Process Templates

THE METEOR-S FRAMEWORK FOR SEMANTIC WEB PROCESS COMPOSITION

by

KAARTHIK SIVASHANMUGAM

B.E., Anna University, India, 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

2003

THE METEOR-S FRAMEWORK FOR SEMANTIC WEB PROCESS COMPOSITION

by

KAARTHIK SIVASHANMUGAM

DEDICATION

To my father B. Sivashanmugam, mother P.N. Kamala and brother S. S. Dhandapani

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. John. A. Miller for his guidance, foresight, and his assurance in difficult times. Dr. Miller has been very generous with his time and wisdom. I would also like to thank Dr. Amit P. Sheth and Dr. Hamid R. Arabnia for their valuable suggestions and being a part of my committee. Special thanks to my friends P. Giridhar, K. Ramesh Kumar and G. Srikumar and for their support and encouragement. Finally, thanks to Kunal Verma and other LSDIS student members for helping me in this work.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

There has been significant excitement over the promise of Web services. The existence of standardized software components over the Internet that can be accessed, described and registered using XML based protocols could lead to powerful applications spanning the Internet. From the perspective of e-commerce, the idea of creating dynamic business processes on the fly (described as “dynamic trading processes” in [1]), would allow corporations to enable full-scale business process integration [2] further leveraging the power of the World Wide Web. There has been a flurry of activity in the area of Web services and in ways to assemble these Web services to create Web processes. Web Processes represent next generation technology for carrying out core business activities, such as e-commerce and e-services, and are created from the composition of Web Services [3] or other software components. Web processes encompass the ideas of both inter and intra organizational workflow. While there has been significant progress in this area, there are a number of factors that prevent the wide scale deployment of Web services and creation of Web processes. The most inherent problems concern describing and discovering Web services. The current solutions and standards take a structural approach to describing Web services using XML based descriptions [4]. The main problem with this approach is that, it is not possible to explicitly define the meaning intended by the Web service provider. "Formally self-described" [5, 6, 7, 8] semantic Web services are a solution to semantically describe and discover Web services. Most of the composition standards build on top of Web service description standards. Hence semantically describing a service could result in composing a process whose individual components are semantically described. When all the tasks involved in a Web process are semantically described, we may call such process as Semantic Web Processes [9].

As part of the METEOR-S project at the Large Scale Distributed Information Systems (LSDIS) Lab at the University of Georgia, we are using techniques from the semantic Web [10], semantic Web services [11] and earlier research in workflow management as part of the METEOR project [12, 13] to deal with the problems of semantic Web service description, discovery and composition. In particular, the METEOR-S project associates semantics to Web Services, covering input/output, functional/operational descriptions, execution and quality, and exploits them in the entire Web process lifecycle encompassing semantic description/annotation, discovery, composition and enactment (choreography and orchestration) of Web services. The current emphasis of the METEOR-S project has been on semantic annotation of Web services [14], semantic discovery infrastructure for Web services (MWSDI: METEOR-S Web Service Discovery Infrastructure) [15] and semantic composition of Web services (MWSCF: METEOR-S Web Service Composition Framework). This paper focuses on MWSCF.

MWSCF aims at using the power of Web services to allow corporations to create processes that mirror today's dynamic and ever-changing business needs. Corporations can expose their application software as Web services so that other corporations can dynamically find and invoke them. In order to precisely define a business or workflow process, several process specification languages have been created. These standards are based on WSDL descriptions. Most of these languages have focused on representing the semantics of user requirements, so that partners and Web services can be dynamically discovered before the executable process is created. We have defined Semantic Process Templates (SPTs), which allow us to semantically define each activity involved in a process. With an SPT, a process can be generated that will have a concrete Web service implementation for each activity preserving their semantics. The main focus of this paper is one aspect – that of creating semantically enriched process templates, which can be refined into concrete executable processes based on the users requirements. The most challenging problem in creating a process template is to capture the semantics of the activities in the process, so that relevant Web services can be bound to activities of the process. We extend our previous efforts on semantic Web service description and discovery to describe the process template and to discover relevant Web services and to generate the executable process based on the discovered services. Using the approach stated and framework discussed in this paper, semantic Web processes can be more effectively designed. This is a straightforward approach that is compatible with present industry standards for Web services.

CHAPTER 2

FRAMEWORK FOR SEMANTIC WEB PROCESS COMPOSITION [i]

[i] Kaarthik Sivashanmugam, John A. Miller, Amit P. Sheth, Kunal Verma. Submitted to *International Journal of Electronic Commerce*

1. Introduction

There has been significant excitement over the promise of Web services. The existence of standardized software components over the Internet that can be accessed, described and registered using XML based protocols could lead to powerful applications spanning the Internet. From the perspective of e-commerce, the idea of creating dynamic business processes on the fly (described as “dynamic trading processes” in [1]), would allow corporations to enable full-scale business process integration [2] further leveraging the power of the World Wide Web. There has been a flurry of activity in the area of Web services and in ways to assemble these Web services to create Web processes. Web Processes represent next generation technology for carrying out core business activities, such as e-commerce and e-services, and are created from the composition of Web Services [3] or other software components. Web processes encompass the ideas of both inter and intra organizational workflow. While there has been significant progress in this area, there are a number of factors that prevent the wide scale deployment of Web services and creation of Web processes. The most inherent problems concern describing and discovering Web services. The current solutions and standards take a structural approach to describing Web services using XML based descriptions [4]. The main problem with this approach is that, it is not possible to explicitly define the meaning intended by the Web service provider. "Formally self-described" [5, 6, 7, 8] semantic Web services are a solution to semantically describe and discover Web services. Most of the composition standards build on top of Web service description standards. Hence semantically describing a service could result in composing a process whose individual components are semantically described. When all the tasks involved in a Web process are semantically described, we may call such process as Semantic Web Processes (SWP) [9].

As part of the METEOR-S project at the Large Scale Distributed Information Systems (LSDIS) Lab at the University of Georgia, we are using techniques from the semantic Web [10], semantic Web services [11] and earlier research in workflow management as part of the METEOR project [12, 13] to deal with the problems of semantic Web service description, discovery and composition. In particular, the METEOR-S project associates semantics to Web Services, covering input/output, functional/operational descriptions, execution and quality, and exploits them in the entire Web process lifecycle encompassing semantic description/annotation, discovery, composition and enactment (choreography and orchestration) of Web services. The current emphasis of the METEOR-S project has been on semantic annotation of Web services [14], semantic discovery infrastructure for Web services (MWSDI: METEOR-S Web Service Discovery Infrastructure) [15] and semantic composition of Web services (MWSCF: METEOR-S Web Service Composition Framework). This paper focuses on MWSCF.

MWSCF aims at using the power of Web services to allow corporations to create processes that mirror today's dynamic and ever-changing business needs. Corporations can expose their application software as Web services so that other corporations can dynamically find and invoke them. In order to precisely define a business or workflow process, several process specification languages have been created. These standards are based on WSDL descriptions. Most of these languages have focused on representing the semantics of user requirements, so that partners and Web services can be dynamically discovered before the executable process is created. We have defined Semantic Process Templates (SPTs), which allow us to semantically define each activity involved in a process. With an SPT, a process can be generated that will have a concrete Web service implementation for each activity preserving their semantics. The main focus of this paper is one aspect – that of creating semantically enriched process templates, which can be refined into concrete executable processes based on the users requirements. The most challenging problem in creating a process template is to capture the semantic capabilities of activities in the process, so that relevant Web services can be bound to activities of the process. We extend our previous efforts on semantic Web service description and discovery to describe the process template and to discover relevant Web services and to generate the executable process based on the discovered services. Using the approach stated and framework discussed in this paper, semantic Web processes can be more effectively designed. This is a straightforward approach that is compatible with present industry standards for Web services.

The key features and contributions of this paper are the following:

- A comprehensive framework for composition of SWPs
- Using process templates to store semantics of each activity in an SWP, and
- Process generation and dynamic discovery of services based on semantics of each activity in the SWP during process generation.

The rest of the paper is organized as follows. Section 2 discusses three of the most popular composition standards. We also present a sample of a typical Web process that will be used for illustrative purposes. Section 3 presents a detailed discussion of the components making up our composition framework. It also briefly summarizes the steps involved in composing a semantic Web process. Key features of MWSCF are highlighted in section 4. Section 5 discusses related work. Conclusions and future work are presented in section 6.

2. Background study

In this section, we first present an overview of the present composition standards. Later, we provide a sample Web process and explain the implementation with respect to each of the presented composition standards.

2.1. Overview of Present Standards for Process Specification

To enable enterprise application integration (EAI) and business process integration within and across organizations, the importance of the languages for Web service composition has increased. So have the number of proposals for the Web service composition standards proposed by different vendors, organizations and consortia. These standards propose XML based open standards for assembling a number of Web services to form a business process. The standards that are currently being considered for building processes using Web service composition include (among others) BPEL4WS [16], BPML [17] and DAML-S [5]. In spite of the fact that all of these standards aim to solve the problems related to process description, they differ in many aspects. Comparing these standards requires an in-depth study of application scenarios of the composed processes and the support of these standards for each of the scenarios. [18] compares these standards using a framework with a set of patterns that are representative of the recurring situations found in different workflow management systems and in the context of enterprise application integration. Comparison based on other criteria is available in [19, 20, 21, 22, 23, 24, 25]. The following sub-sections provide an overview of three major proposed standards; interested readers are referred to the above comparative studies for further information.

2.1.1 BPEL4WS

The Business Process Execution Language for Web Services {BPEL4WS) [16] is a language to specify business processes and business interaction protocols. It superseded XLANG [26] and WSFL [27] as a standard for Web services flow specification. The model and XML-based grammar provided by BPEL define the interactions between a process and its partners using Web services interfaces. BPEL also defines the states and logic of coordination between these interactions and systematic ways of dealing with exceptional conditions. The business interaction protocols are called abstract processes. They are used to specify public and visible message exchange between different parties involved in a business protocol, but they do not reveal the internal behavior or the implementation of the involved parties. The executable processes on the other hand are like workflow descriptions represented using basic and structured activities specifying a pattern of execution of Web services. The process model defined by BPEL is based on the WSDL service description model. The services (described as partners in BPEL) that are invoked by the process and the services that invoke the process are represented using their WSDL description. An executable process can be a Web service by itself and the interface of that process can be represented using WSDL.

2.1.2 BPML

The Business Process Modeling Language (BPML) [17] is based on an abstract model and grammar for expressing *abstract* and *executable* business processes. Using BPML, enterprise processes, complex Web services and multi-party collaborations can be defined. A process in BPML is a composition of activities that perform specific functions. The process directs the execution of these activities. It can also be a part of another composition by defining it as a part of a parent process or by invoking it from another process. Each activity (both simple and complex) in the process has a *context*, which defines common behavior for all activities executing in that context. Hence a process can be defined as a type of complex activity that defines its own context for execution. The BPML specification defines 17 *activity* types, and three *process* types. The different process types are *nested processes* which are defined to execute within a specific context and whose definitions are a part of context definition, *exception processes* to handle exceptional conditions in executing a parent process and *compensation processes* to provide compensation logic for their parent processes. Each process definition may specify any of the three ways of instantiating a process: in response to an input message, in response to a raised *signal*, or invoked from an *activity*. BPML specifications support importing and referencing service definitions given in WSDL. It also suggests standardizing BPML documents by using RDF for semantic meta-data, XHTML and the Dublin Core metadata to improve human readability and application processability.

2.1.3 DAML-S

The DAML-based Web Service Ontology (DAML-S) [5] is an initiative to provide an ontology markup language expressive enough to semantically represent capabilities and properties of Web services. DAML-S is based on DAML+OIL and the aim is to discover, invoke, compose, and monitor Web services. It defines an upper ontology appropriate for declaring and describing services by using a set of basic classes and properties. In DAML-S, each service can be viewed as a *Process* and its *Process Model* is used to control the interactions with the service. Using the *Process Ontology*'s sub-ontologies, *Process Ontology* and *Process Control Ontology*, it aims to capture the details of the Web service operation. The *Process Ontology* describes the inputs, outputs, preconditions, effects, and component sub-processes of the service. *Process Control Ontology* is used to monitor the execution of a service. However, the current version of DAML-S does not define the *Process Control Ontology*. DAML-S also categorizes three types of processes. The first type is an *atomic process*, which do not have any sub-processes and can be executed in a single step. The second type is a *simple process*, which is not invocable as it is used as an abstraction for representing an atomic or composite process. A c*omposite process* is the third type, which is decomposable into sub-processes. A composite process uses several control constructs to specify how inputs are accepted and how outputs are returned.

There is a need to compare and analyze the features of these languages in detail to frame a single and powerful and interoperable standard for composing processes. Web processes should be dynamic and flexible enough to adapt to the changes in demands from customers or market forces. To meet this requirement, BPEL and BPML, abstract the service references in the process from actual service implementations. This helps in selecting a correct service implementation for each activity during process deployment (deployment-time binding) or execution (execution-time binding). However, the present composition standards like BPEL and BPML are lacking in an important aspect, semantically representing the activity components of a process. BPML only suggests adding semantics to the process description.

We handle this problem by capturing semantics of the involved activities in the process template. These semantic descriptions are not bound to Web service implementations. Before deployment of the process, the services that satisfy the semantic requirements are discovered and bound to the activities in the process template. Using the interfaces and message types supported by the services, an executable process is generated which is used for deployment. Any process management system will demand a powerful discovery algorithm integrated into the process management system. This demand becomes critical when the size and the number of services available in the Web are taken into consideration. Our approach supports precise semantics based discovery of Web services. The following section details a sample business process. Composition of this process using our framework is discussed in the subsequent sections.

2.2 Sample Web Process

Let us consider an example for a typical Web process in the e-business domain, an electronic toy manufacturer processing a distributor's order. The manufacturer hosts an application (*getOrderPriceAndDeliveryDate*) where the distributor can query for the price and date of delivery by specifying an order. The manufacturer, upon receiving the order details, processes the order and returns the details that the distributor is querying for. Based on the details returned by the application, the distributor can place the actual order (*placeOrder*) keeping the returned details as the agreement between the manufacturer and the distributor. The applications, *getOrderPriceAndDeliveryDate* and *placeOrder*, have several intermediate steps, which are to be carried out by services within and outside the Manufacturer’s organization. For brevity, we have considered only the *getOrderPriceAndDeliveryDate* application in our example. In the *getOrderPriceAndDeliveryDate* application, when the distributor places an order, the manufacturer checks the inventory to verify if it has enough goods to satisfy the order. In case there is enough stock then the manufacturer contacts its delivery partner for a date of delivery and its agreement database or accounts department to fix a price for the order. Based on the price returned by the delivery partner and the fix price methods, the price for the order is finalized. Then, the delivery date and the finalized price are returned to the distributor. In the other case, when there is not enough stock in the inventory, the manufacturer contacts its supplier partner for the required components. Then the manufacturer contacts its delivery partner to arrange for delivery of components to its manufacturing site and later to deliver the products to the distributor. The price and delivery date are returned to the distributor. The following figure depicts this process in detail.

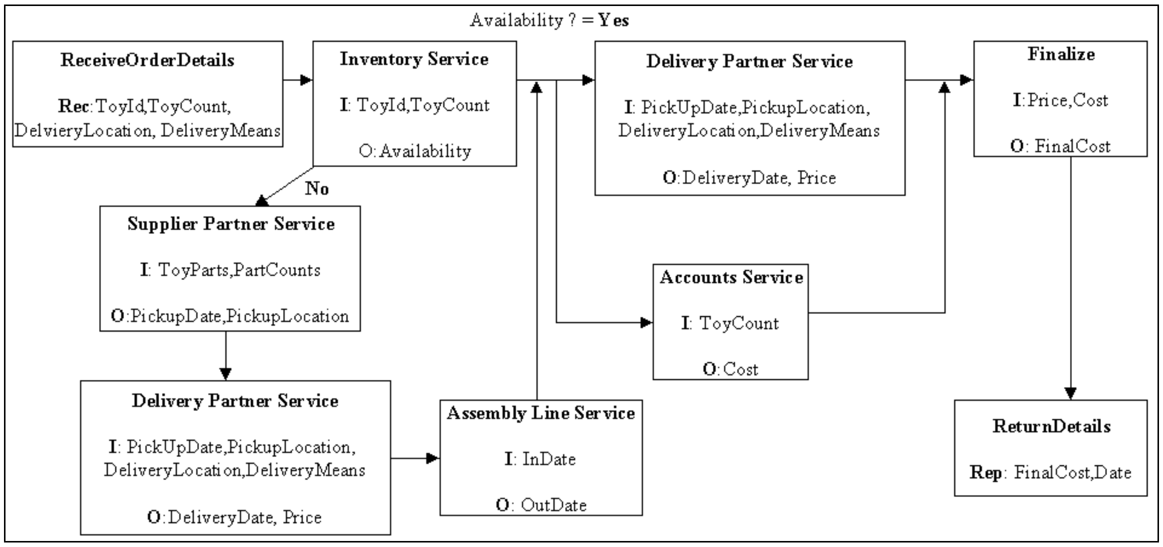


Figure 1: Sample Web Process[ii] (*getOrderPriceAndDeliveryDate*)

This above-mentioned procedure for order processing can be implemented as an SWP. This process may span across several organizations. In this example, the process design will

[ii] I, O respectively represent inputs and outputs of each acivity. Rec and Rep are the inputs to be received by the process and output produced by the process

happen in the Manufacturer's organization. When the process is statically composed as a workflow, then the supplier partner and delivery partner services are decided before hand and integrated in the process along with manufacturer's intra organizational services such as fix price, assembly line, inventory, etc. Due to the dynamic nature of the business, hard coding the business logic and participating services may not be efficient. This process should be able to be integrated with any potential delivery partner or supplier partner. The present standards support this kind of design, but they impose a restriction on the interface provided by these potential partners. In BPEL, it is assumed that if a service needs to be a part of a process instance, then it should provide the interface specified by the WSDL's port type[iii] construct in the process definition. We attempt to solve this problem by specifying the process using templates during process design. Such templates are independent of the service description and process definition standards. Hence any service that satisfy the semantic requirements of the activity can be used to carry out that activity in the process. Before execution, the process templates are used to instantiate an executable process in any of the process definition standards and executed accordingly.

The idea of customizable processes and using process templates has been discussed earlier in [1]. It proposes three architectures/modalities for managing inter-organizational business processes. One of the architectures envisions *Process Portal* hosted by an enterprise or an organization for its customers. It manages a variety of customizable processes in which a subscribing company or a trading partner might do an individual activity. The second architecture is the *Process Vortex* for specialized markets where interactions are controlled by some third party. The business processes in the process vortex are designed to incorporate different trading models and they are available as templates that can be used to customize processes. The *Dynamic Trading Process* architecture defined as the third architecture is a virtual market place for different products spanning across multiple industries. In this architecture, processes can be

[iii] *PortType* construct in WSDL is used to group operations

constructed based on customer's needs. It supports flexible and dynamic trading processes that are composed upon requests from customers and are based on the QoS requirements specified by the customer. The framework discussed in our present work can be used to design processes in any of these modalities. In our system, the processes can be defined using semantic templates and the users of the process or provider of the process can customize and generate executable process.

The next section explains the METEOR-S composition framework and demonstrates how it can be used to design the *getOrderPriceAndDeliveryDate* SWP.

3. METEOR-S Web Service Composition Framework (MWSCF)

This section describes the MWSCF. There are four major components in MWSCF: the Process Builder, the Discovery Infrastructure (MWSDI), XML repositories and the Process Execution Engine. The process designer constitutes a builder and a process generator. It provides a graphical user interface to design/open process templates and provides a process generator, which uses MWSDI and data in XML repositories to convert the template into an executable process. The METEOR-S Web Service Discovery Infrastructure (MWSDI) is used to access a community of Web service registries and semantically search for Web services. The generated executable process is then executed using a process execution engine. The XML repositories in the architecture are used to store ontologies, activity templates and process templates. The details of each of these components are discussed in the following sections. Figure 2 shows the overall architecture of MWSCF.

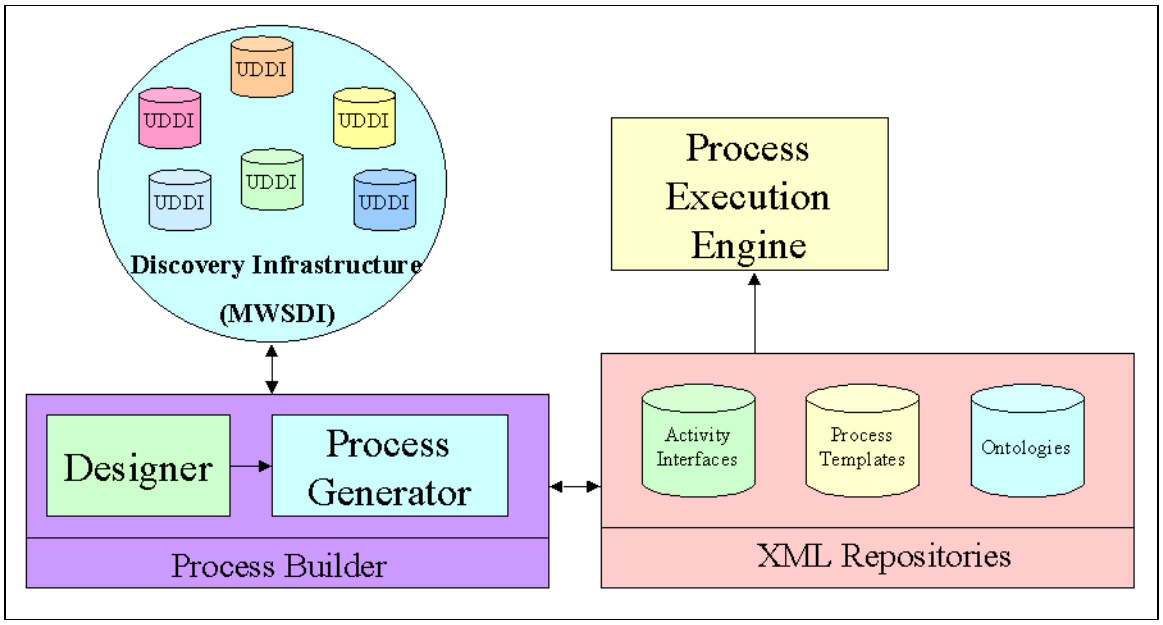


Figure 2: Web Service Composition Framework

3.1 METEOR-S Web Service Discovery Infrastructure (MWSDI)

Web services are advertised in registries. The initial focus of Universal Description, Discovery and Integration [28] specifications was geared towards working with a Universal Business Registry (UBR), which is a master directory for all publicly available Web services. However, the new version of the UDDI specification [29] recognizes the need for existence of multiple registries and the need for interactions among them. A large number of registry/repository implementations for electronic commerce, each focusing on registering services of interest to respective sponsoring groups, are also anticipated. Hence, the challenge of dealing with hundreds of registries (if not thousands) during service publication and discovery becomes critical. Searching for a particular Web service would be very difficult in an environment consisting of hundreds of registries. This search would involve locating the correct registry in the first place and then locating the appropriate service within that registry.

Finding the right services would be easier if the registries were categorized based on domains with each registry maintaining only the Web services pertaining to that domain. If the registries are specialized like this, search for services in that domain can be carried out in a relevant registry. In addition, adding semantics to the domain-registry association will help in efficiently locating the right registries based on query requirements. In MWSDI, we use a specialized ontology called the Registries Ontology, which maintains relationships between all domains in MWSDI, and associates registries to them (see [15] for details).

Improving service discovery also involves adding semantics to the Web service descriptions and registering these descriptions in the registries. Adding semantics to Web service descriptions can be achieved by using ontologies that support shared vocabularies and domain models for use in the service description. Using domain specific ontologies, the semantics implied by structures in service descriptions, which are known only to the writer of the description (provider of web service), can be made explicit. Hence, while searching for a Web service, we can refer to relevant domain specific ontologies to enable semantic matching of services. MWSDI provides support for this kind of matching by relating both Web service descriptions and user requirements to ontologies. It also provides an infrastructure for accessing multiple registries. The registries are provided by different registry operators and they may support their own domain specific ontologies for their registries. The registries may also want to offer their own version of semantic publication and matching algorithms. Along with that, each operator may also provide value added services for the registry users.

We have implemented MWSDI to demonstrate this scalable infrastructure of Web service registries for semantic publication and discovery of services. It is implemented as a P2P network of UDDI registries. The MWSDI prototype system allows different registries to register in a P2P network and categorize registries based on domains. These registries in turn support domain specific ontologies and provide value added services for performing registry operations. MWSDI supports semantic publication of services. The inputs and outputs of the services are semantically annotated and these annotations are captured in UDDI. To perform semantic discovery according to the original implementation, the users can annotate the inputs and outputs of the service requirements and the discovery process in a UDDI will result in the services that match these semantic requirements. This discovery algorithm has been extended later in [14] that also supports annotating each operation in a WSDL file with a concept in functional ontology along with the annotation of preconditions and effects of that operation. Hence during discovery the service requirements are semantically annotated by associating it with concepts in ontologies that represent operation, inputs, outputs, preconditions and effects of the service. The discovery mechanism supported in MWSCF is based on all these kinds of annotations in addition to input and output semantics. Using the MWSDI with the semantic publication and discovery algorithms can significantly improve upon the current standards in Web service registration and discovery. MWSDI provides the flexibility to search for Web services based on ontologies. The user when designing a process may specify the discovery details of an activity by using the *registry ontology* (discussed in [15]). MWSDI architecture has been implemented on a cluster of SUN workstations as peer-to-peer network using the JXTA [30] framework. Xindice [31], a native XML database that comes with JWSDP [32] is used for implementing UDDI registries in MWSDI. UDDI4J [33] is used for accessing UDDI registries during publication and discovery.

3.2 Process Builder

The process Builder implemented in Java has a designer that assists in composing semantic Web processes. WSDL4J [34] has been used for processing WSDL files. The Jena tool kit has been used to building and processing ontologies. The builder supports three different approaches to specify an activity. Each activity in the process can be specified using a

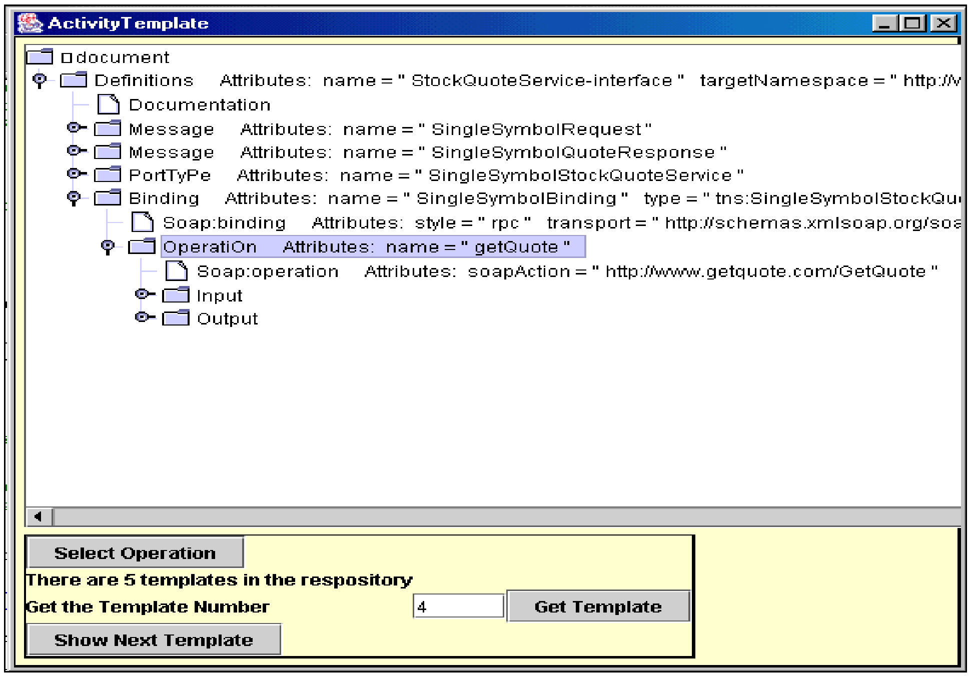
- Web service implementation,
- Web service interface, or
- Semantic activity template.

3.2.1 Specifying an Activity using a Web Service

Static composition of a process is done by specifying activities using concrete Web service implementations. This type of composition is discussed in one of our previous projects called SCET [35]. MWSCF allows linking an activity to a WSDL file and a relevant operation in it. For example, in the process discussed in section 2.2, the process designer knows the details of the intra organizational services like *checkInventory*, *fixPrice*, etc. These services can be used to carry out respective activities in a process. The process designer can link the activities to a WSDL file and an operation in it. Even if the service interface or implementation changes, as long as the URL of the WSDL and the name of the operation do not change, MWSCF can associate the activity with that operation. During process generation, the *portType* and *message* details are extracted from the WSDL and used in the generated executable process.

3.2.2 Specifying an Activity using a Web Service Interface

An activity can also be specified using a Web service interface. If an activity is linked to a Web service interface, during process generation, a concrete service that implements the interface could be used to carry out the activity. Only the services that implement the interface are discovered. Discovering services that implement interfaces has been suggested in [36]. UDDI is a registry and not a repository and hence it does not allow publishing the interface definitions. Popular or industry specific interfaces can be published in UDDI using *tModels* that will have references to the interface definitions. All the services that implement that interface will indicate the conformance to that interface by binding the *tModel* using *binding template* constructs in UDDI. During discovery process, the *tModel* that represent an interface, can be used and all the services that implement this interface can be discovered. In MWSCF, during the creation of process templates, process designer can specify activities using Web service interfaces. During process generation a service can be selected from the list of services that implement the interface. The *portType* and *message* details of the implementing services are retrieved and used during process generation. In MWSCF, these interfaces are stored in a XML repository. The interfaces in the interface repository are identified using the same *id* as that of the *tModels* in UDDI that represent each of these interfaces. The user while designing a process can browse (shown in figure 3) through these interfaces and select an interface and an operation in the interface to link to an activity. The identifier (same as *tModel id*) of the interface is used during a UDDI search to retrieve services that implement the interface.



```
document
  Definitions   Attributes:  name = " StockQuoteService-interface "   targetNamespace = " http://v
    Documentation
    Message   Attributes:  name = " SingleSymbolRequest "
    Message   Attributes:  name = " SingleSymbolQuoteResponse "
    PortTyPe   Attributes:  name = " SingleSymbolStockQuoteService "
    Binding   Attributes:  name = " SingleSymbolBinding "   type = " tns:SingleSymbolStockQu
      Soap:binding   Attributes:  style = " rpc "   transport = " http://schemas.xmlsoap.org/soa
      OperatiOn   Attributes:  name = " getQuote "
        Soap:operation   Attributes:  soapAction = " http://www.getquote.com/GetQuote "
        Input
        Output
```

There are 5 templates in the respository

Get the Template Number: 4

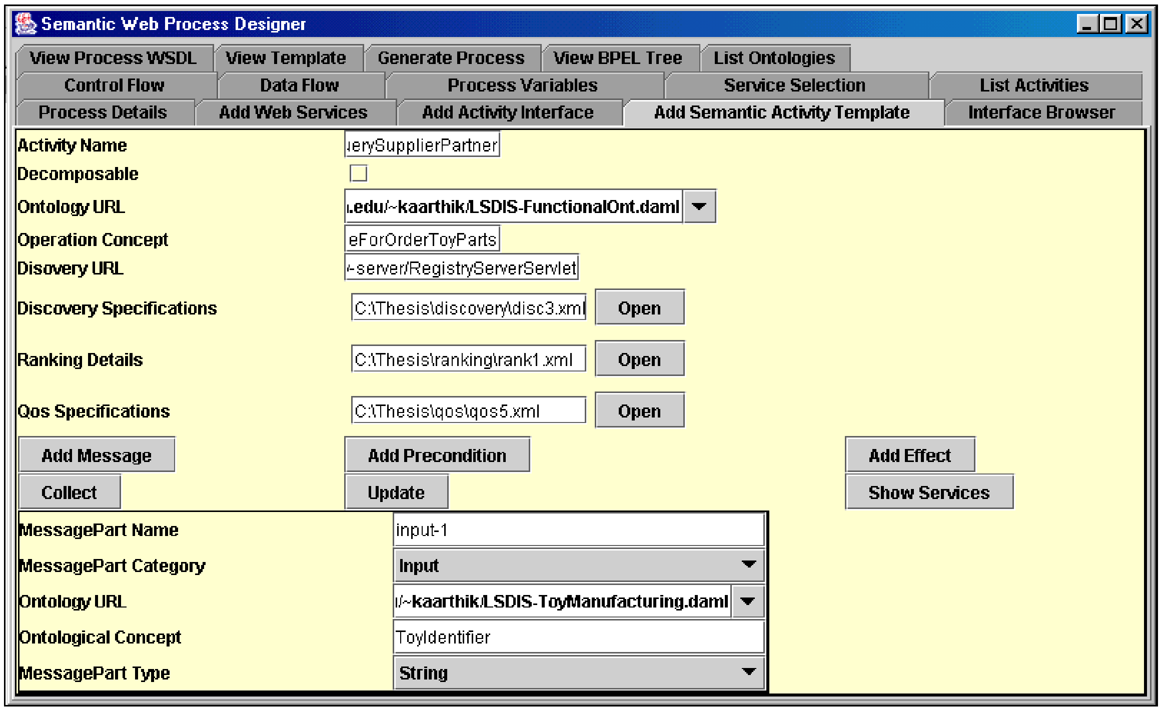
Figure 3: Browser to Web Service Interfaces

The user in addition to specifying the interface and an operation also specifies the discovery details and QoS requirements for that activity. The discovery details are the details based on which UDDI registries can be queried. For example, the UDDI supports searching for services based on name of the business or services (keywords, wildcard character and qualifiers), categorization (in taxonomies), characterization (technical fingerprint), etc. These details in addition to the interface details are used during discovery of Web services for an activity. In another example, if a *tModel* representing a technical fingerprint is specified in the discovery details of an activity, then a *tModelBag* is constructed using this *tModel* and the *tModel* that represents the interface. The constructed *tModelBag* is used during service search. The MWSDI specific details can also be represented in this. The discovery details are given in an XML representation of the API supported by UDDI specifications.

The QoS requirements may also be given in an XML file. The QoS details are used in ranking the discovered services. The process designer can use this ranking to select an appropriate service to carry out an activity. The details are given in section 3.2.7. If there is a standard interface for the *Delivery* service discussed in sample web process in section 2.2, it can be cached in the XML repository. Then during the process template design, the activity for *delivery* service can be specified using that interface.

3.2.3 Specifying an Activity using a Semantic Activity Template

In the third approach, the requirements for an activity are given using its semantic characteristics. In the previous approach (section 3.2.2), the activity is specified using an interface, meaning that an operation in the interface to carry out the activity and the data type (complex or simple) for input and output of the activity is stated as the activity requirements. When the activity is specified using a semantic activity template, the activity requirements given as the semantics of the IO along with the operational semantics of the activity are specified. The functional semantics of an activity, its IO, its preconditions and effects are represented using ontological concepts. The services that conform to these semantic characteristics are discovered and ranked. The users can select a service from the list of discovered services. This approach of specifying the activities assumes that all services are semantically annotated by which each operation in a WSDL file, its inputs, and outputs are mapped to ontological constructs in addition to having additional tags for preconditions and effects of the operations. With this methodology, capabilities of each operation in a WSDL file can be captured. These semantic details in a WSDL file could be published in a UDDI. Hence given a set of requirements based on these semantic details, the services that match these requirements could be more precisely found. A detailed discussion is presented in [14]. The data types of input and output are optional and could be used as a weighted component by the match algorithm. As in the previous approach, the discovery details and QoS requirements could be specified for the semantic activity template too. These discovery details are combined with the semantic requirements and search is performed within a registry. The QoS requirements may be used to rank the resulting services. The following figure shows the user interface window to specify an activity using semantic activity template.



Activity Name	ierySupplierPartner
Decomposable	☐
Ontology URL	.edu/~kaarthik/LSDIS-FunctionalOnt.daml
Operation Concept	eForOrderToyParts
Disovery URL	-server/RegistryServerServlet
Discovery Specifications	C:\Thesis\discovery\disc3.xml
Ranking Details	C:\Thesis\ranking\rank1.xml
Qos Specifications	C:\Thesis\qos\qos5.xml

MessagePart Name	input-1
MessagePart Category	Input
Ontology URL	/~kaarthik/LSDIS-ToyManufacturing.daml
Ontological Concept	ToyIdentifier
MessagePart Type	String

Figure 4: Semantic Activity Template Specification Interface

In our example discussed in section 2.2, the activity for *Delivery* Service can be specified as a semantic template. The input and output of the activity can be represented using a standard vocabulary or ontology. A domain ontology for the domain *Cargo Services* could be used for this requirement annotation. Our work like most other work in semantic Web research is based on rich framework for ontology engineering and re-use. The *Cargo Services* ontology may encompass concepts like *Air Cargo Services*, *Cargo Insurance*, *Maritime Cargo Services*, *Rail Cargo Services*, *Trucking* (taken from Yahoo directories). Hence using such ontology, the semantics of an activity are specified. During service discovery, the service implementations that used this ontology and annotated their descriptions will be semantically compared with the requirements and ranked.

3.2.4 Process Composition

When composing a generic process template the user gives the list of activities and control flow constructs to link the activities. After finalizing the template, the user can save the template for future use or can find service implementations for each activity in the template. Though the templates could be used during run-time to find services for each activity, at present MWSCF only support deployment time binding. This is because the initial implementation of MWSCF uses an engine that supports deployment time binding. Though the builder supports different approaches to design, the focus of this work is on the third approach where the process is composed by defining each activity using semantic templates. When all the activities in a process template are specified using semantic templates then it is called a semantic process template.

3.2.5 Semantic Process Template

The semantic process template is a collection of activities, which can be linked using control flow constructs. A sample process template in XML format is shown in figures 5 and 6.

The process templates in MWSCF have BPEL-like syntax. For representing control flow the template, uses the BPEL constructs. First step in template creation is to create a WSDL that represents the description for the desired process is generated using a WSDL editor. This WSDL is then linked to the process template. The template can be explained as follows:

- *Process-template* is the root element that will enclose the entire template definition. It has different attributes that represent different ontologies or other name spaces.
- The control constructs (like <*sequence*>, <*flow*>, <*switch*> etc.) are used to represent control flow in the template. They do not need translation and they are used as it is during the process generation phase.
- In the example discussed in section 2.2, the inputs and outputs of the respective activities *Receive Order Details* and *Return Details* are represented as messages in the WSDL file (shown in section 3.4) of the process. The *receive* and *reply* constructs in the template are linked to the WSDL description (*process-wsdl-operation*) and an *operation* in the description that the process designer creates. The messages that are to be received and returned by the process are also captured in that WSDL file. They will be translated to containers[iv] in the executable process.

[iv] called Variable is BPEL version 1.1

```
<?xml version = "1.0" encoding = "UTF-8"?>
<process-template name = "sample process"
                LSDIS-ToyManufacturing = "uuid: f4a6574b-49f4-a657-f908-45fa62354d84"
                LSDIS-OrderPlacement = "uuid: f4 a9574b-49f4-a957-f956-45fa62355c94"
                LSDIS-FunctionalOnt = "uuid: f4a5904b-49f4-c563-f590-45fa65289d77"
                LSDIS-CargoServices = "uuid: f4a8884b-49f4-b634-f888-45fa62098d43"
                xsd = "http://www.w3.org/2001/XMLSchema">
  <sequence name = "sequence-1">
    <receive name = "receive-1“process-wsdl-operation = “ getDetails "/>
    <invoke-activity name = “InventoryCheckActivity" type = "ServiceImpl"
                wsdl-URL = "http://lsdis.cs.uga.edu/proj/meteors/wsdls/all_services.wsdl"
                operation-name = "checkInventory"/>
    <switch name = "switch-1">
      <case condition = "(inventory-availability,' = ','no')">
        <sequence name = "sequence-3">
        <invoke-activity name = “QuerySupplierPartner"  type = "SemanticTemplate"
                semantic-spec = "semantic-1"
                qos-spec = "qos-1"
                ranking-weights = "ranking-1“
                discovery-URL  = "http://lsdis.cs.uga.edu:7001/registry-server/RegistryServerServlet"/>
        <invoke-activity name = “DeliveryPartnerService" type = "WSInterface"
                tModel-id = "uuid:f4a6574b-49f4-a657-f908-45fa62354d84"
                operation-name = "arrangeDelivery"
                qos-spec = "qos-2"
                discovery-details = "discovery-1“
                discovery-URL  = "http://lsdis.cs.uga.edu:7001/registry-server/RegistryServerServlet"/>
        <invoke-activity name = “AskAssemblyLine" type = "ServiceImpl"
                wsdl-URL = " http://lsdis.cs.uga.edu/proj/meteors/wsdls/all_services.wsdl "
                operation-name = "assemblyService"/>
        </sequence>
      </case>
    </switch>
    <sequence name = "sequence-2">
      <flow name = "flow-1">
        <invoke-activity name = "DeliveringToysToDistributor" type = "ServiceImpl"
                 wsdl-URL = " http://lsdis.cs.uga.edu/proj/meteors/wsdls/all_services.wsdl "
                operation-name = “deliveryService“/>
        <invoke-activity name = “FixPrice" type = "ServiceImpl"
                wsdl-URL = "http://lsdis.cs.uga.edu/proj/meteors/wsdls/all_services.wsdl"
                operation-name = "fixPrice"/>
      </flow>
      <invoke-activity name = “FinalizePrice"  type = "ServiceImpl"
                wsdl-URL = "http://lsdis.cs.uga.edu/proj/meteors/wsdls/all_services.wsdl"
                operation-name = "finalizePrice"/>
    </sequence>
    <reply name  =  "reply-1" process-wsdl-operation = “getDetails "/>
  </sequence>
```

Figure 5: Sample Process Template Listing 1

```
  <criteria>
    <semantic-spec name = "semantic-1">
            <input name = "input-1"
                 onto-concept = "LSDIS-ToyManufacturing:ToyIdentifier"/>
            <input name = "input-2“
                 onto-concept = "LSDIS-OrderPlacement:OrderCount"/>
            <operation name = "operation-name"
                 operation-concept = "LSDIS-FunctionalOnt:getDetailsForOrderToyParts"/>
            <output name = "output-1"
                 onto-concept = "LSDIS-CargoServices:PickupDate“/>
            <output name = "output-2"
                 onto-concept = "LSDIS-CargoServices:PickupLocationIdentifier“/>
    </semantic-spec>
    <discovery-spec name = "discovery-1">
            <find_service>
              <categoryBag>
                <keyedReference  tModelKey = "uddi:ubr.uddi.org:categorization:geo3166-2"
                        keyName = "GEO:United States"
                        keyValue = "US"/>
             </categoryBag>
            </find_service>
    </discovery-spec>
     <qos-spec name = "qos-1">
            <delaytime unit = "milliseconds" qualifier = "LT"> 90 </delaytime>
    </qos-spec>
     <qos-spec name = "qos-2">
            <delaytime unit = "milliseconds" qualifier = "LT"> 200 </delaytime>
    </qos-spec>
    <ranking-weights name = "ranking-1">
            <semantic-matching-weight> 75 </semantic-matching-weight>
            <qos-weight>  25 </qos-weight>
            <input-semantics-weight>30 </input-semantics-weight>
            <output-semantics-weight> 20 </output-semantics-weight>
            <operational-semantics-weight> 50 </operational-semantics-weight>
            <delay-time-weight> 100 </delay-time-weight>
    </ranking-weights>
    <protocol-variable name = "inventory-availability"  type = "xsd:string"/>
  </criteria>
</process-template>
```

Figure 6: Sample Process Template Listing 2

- The *invoke-activity* elements in the process template are translated into corresponding *invoke* elements in the generated processes. The *invoke-activity* elements in the template are of three *types*:
    1. *ServiceImpl*, if the activity is specified using a concrete implementation,
    2. *WSInterface*, if the activity is specified using a Web service interface, and
    3. *SemanticTemplate*, if the activity is specified using a semantic activity template.

  Based on the discovery[v] (*semantic-spec*, *discovery-URL*, *tModel id*, *discovery-spec*) and QoS (*qos-spec*) criteria given for each of the activities, the relevant services to carry out each activity are discovered and selected (discussed in section 3.2.6). The *invoke* elements in executable process will have other details like *portType*, *operation*, *input* and *output containers*, etc. that are extracted from the WSDL description of the service that is selected to carry out the activity.
- Input and output *container* details of the *invoke* elements in the executable processes are generated from the data flow details provided by the user (shown in figure 11).
- The business protocol data (like *inventory-availability*) that are used in process control (like conditional statements) need not be explicitly assigned in the process template. Instead, during process generation, the user can map output from any of the participating service to that variable. These details are translated into <*assign*> and <*copy*> tags in the final generated executable process.
- Other details given by the user for discovery (*semantic-spec, discovery-spec* and *qos-spec*) and ranking (*ranking*-details) for activities during process design are also present in the process template under the *criteria* element. The *discovery-spec*

[v] *ServiceImpl* type does not need discovery as *wsdl-URL* and the *operation-name* to invoke are given in the template itself

element in the template (figure 6) refers to a XML representation of the API used to query for services that are categorized using geo3166-2 taxonomy. Since it is given in conjunction with an activity of type *WSInterface*, when finding Web services for the activity *DeliveryPartnerService*, all the services that implement the interface and categorized using the taxonomy are discovered and ranked. The *semantic-spec* element is used to give the semantics of the activity. *qos-spec* is used to specify the QoS criteria of the activities. The *ranking-weights* element is used to assign weights to rank the discovered services. Detailed discussion on discovery and ranking is given in the following section.

3.2.6 Service Ranking and Selection

Service selection is a crucial aspect of composition. Hence the discovery algorithms in our system are supplemented with a good ranking scheme. The service selection is based on the discovery details for the activity provided by the user. The user could specify the discovery URL for each activity. This discovery URL could point to a market place registry, a private enterprise registry, a domain registry, or a Universal Business Registry. The service discovery will be carried out in that registry. Registry selection can also be based on the registries ontology as proposed in MWSDI. Since discovery could result in a large number of candidate services, we have implemented a ranking mechanism that will help the process composer to select an appropriate service. The ranking of services for each activity can be based on the semantic matching of activity requirements with the service specifications and on the satisfiability of service in terms of QoS requirements of the activity. Our approach requires that each service registered with UDDI is linked to a semantically annotated WSDL description and that the WSDL descriptions are linked to WSEL [27] (Web Service Endpoint Language) files that have the QoS details of all the operations in the service. WSEL is an XML format for the description of non-operational and behavioral characteristics of service endpoints, like quality-of-service or security properties. This specification is under development and at present no specification exists for this language. WSFL [27] specification envisions the need for this language and suggests using it in conjunction with an activity to describe endpoint properties and enable better matchmaking. We have taken this idea and linked each WSDL file to a WSEL file, which has the QoS specifications of the operations in the WSDL file.

If the activity is specified as a semantic activity template then the overall ranking of a service is the weighted arithmetic mean of the two dimensions. The first dimension is based on the *Semantic Matching*. The second dimension is the *QoS criteria matching*. Semantic matching can be considered in ranking only if the activity is specified semantically. However, the QoS based ranking can be done for both interface and semantics based specification of an activity. Matching semantics is done against the *operations*, *inputs*, *outputs*, *preconditions* and *effects* of the activity requirements with the service details.

$$M_S = \frac{\sum_{i=1}^{5} M_i W_i}{\sum_{i=1}^{5} W_i}$$

M_S : Semantic Matching Value

i : Index of the assigned weights (0 – 5)

M_i : Semantic Matching Value of ith criterion

W : Weight of the ith criterion

Figure 7: Formula for Calculating Semantic Matching Value

The semantic matching ranking is based on the weights assigned by the user to the individual parts of the requirements. The user can assign the weights for each of the parts in the activity requirement namely operations, inputs, outputs, preconditions and effects. The assigned weights are normalized before calculation (at least one weight must be non-zero). During discovery the semantic criteria of the activity are matched against the semantic details of the services registered in UDDI. The weights corresponding to the matched semantics are used to rank the services. The formula to calculate the ranking value for semantic matching is given in figure 7.

Let us consider a sample calculation using the semantic specifications of *QuerySupplierPartner* given in the template in figure 6. The semantic requirements of that activity are given in *semantic-spec* element named *semantic-1*. It specifies that the candidate Web services for that activity should have an operation that conforms to the concept *getOrderDetailsForOrderToyParts* in the ontology represented by name space *LSDIS-FunctionalOnt* and takes two inputs and produces two outputs. It also specifies the semantics of the inputs and outputs using ontological constructs. From these specifications it can be understood that one of the inputs of the *QuerySupplierPartner* should conform to the *OrderCount* concept (class) in the *LSDIS-OrderPlacement* ontology. The template refers to four different ontologies, namely, *LSDIS-ToyManufacturing*, *LSDIS-OrderPlacement*, *LSDIS-FunctionalOnt*, *LSDIS-CargoServices* that are published in UDDI and identified by *a tModel id*. The same *tModel id* is used to index the ontologies in the ontology repository.

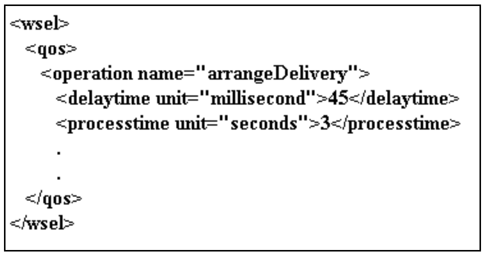
For *QuerySupplierPartner*, the inputs are annotated as *LSDIS-ToyManufacturing:ToyIdentifier* and *LSDIS-OrderPlacement:OrderCount*. Similarly the outputs and operation are also annotated. So the user can assign weights to inputs, outputs and operations. The semantics of preconditions and effects are not specified in the semantic requirement specification of *QuerySupplierPartner* and hence weights are also not assigned for the precondition and effects. In this case the weights are assigned only to three parts (i=3) operations, inputs and outputs. The weights assigned in the template are 30, 20 and 50 respectively. If there is an operation in a WSDL file that takes two inputs and produces two outputs with the following semantics:

1. The functionality of the operation is mapped to *LSDIS-FunctionalOnt:getDetailsForOrderToyParts*. The functional semantics of the service exactly matches with the functional semantics of the activity. Hence the semantic matching value M_1 is 1.
2. One of the inputs is annotated using *LSDIS-ToyManufacturing:ToyIdentifier* and the other using *LSDIS-OrderPlacement:OrderCount*. In this case the input semantics of the service exactly match to the inputs semantics of the activity. The semantic matching value M_2 is 1.
3. One of the outputs is annotated with *LSDIS-CargoServices:PickupDate* and the other output is annotated with *LSDIS-CargoServices:PickupLocationDetails* which is a concept in the ontology that is 2 levels up in the hierarchy created by subClassOf relationships. One of the outputs match exactly and the other output does not match exactly. For non-exact matches the semantic matching value is calculated using a linear function that decides the semantic matching value based on the subClassOf hierarchy. The final semantic matching value M_3 is the average of semantic matching values of the two outputs. In the example the value of M_3 is $(1+0.8)/2 = 0.9$. The subClassOf hierarchy and linear function used is this implementation can be extended with a better function that can be used to characterize two concepts separated in an ontology by a number of named relationship properties.
4. The value of M_S is $(1 * 50 + 1 * 30 + 0.9 * 20) / (50+30+20) = 0.98$.

The next dimension in ranking is based on the QoS requirements of each activity. Each activity can be linked to requirements specification that defines the QoS parameters of that activity regardless of the different design approaches discussed in section 3.2. For simplicity of discussion, our present work considers five different QoS parameters, which are the subset of QoS details that we have identified in one of our previous work [37]. The different QoS specification parts used in ranking are:

- Task Delay Time
- Task Process Time
- Task Realization Cost
- Task Reliability Measure

In addition to QoS the WSEL file can also support representing boundary values[vi] and possible values for the input and output parameters of each operation. Our present discovery method does not take boundary values into consideration. The following figure shows a sample WSEL file.



```
<wsel>
  <qos>
    <operation name="arrangeDelivery">
      <delaytime unit="millisecond">45</delaytime>
      <processtime unit="seconds">3</processtime>
      .
      .
  </qos>
</wsel>
```

Figure 8: Sample WSEL Details of a Service

When specifying activity in a process the designer can link a QoS requirement specification to it. The QoS requirements are specified with qualifiers. A sample QoS

[vi] For example if a service 'TakeOrder' has an operation that takes order for some product X and if the maximum order that it can take is for 1000, it can specify that the value for the input parameter "numberOfProducts" should be less than 1000. Also if the input takes any enumerated value set it could be specified in the WSEL file. These details help users in properly selecting appropriate services.

requirements specification is shown in the template in figure 6 under *qos-spec* element. Each of the QoS parameter in the requirements description is given a weight. For every service that is discovered, the QoS compatibility is checked and using the weight for each QoS parameter the ranking is calculated. The formula for calculating QoS matching of an activity is shown below:

$$M_Q = \frac{\sum_{i=1}^{4} M_i W_i}{\sum_{i=1}^{4} W_i}$$

M_Q : QoS Matching Value

i : Index of the assigned weights (0 – 4)

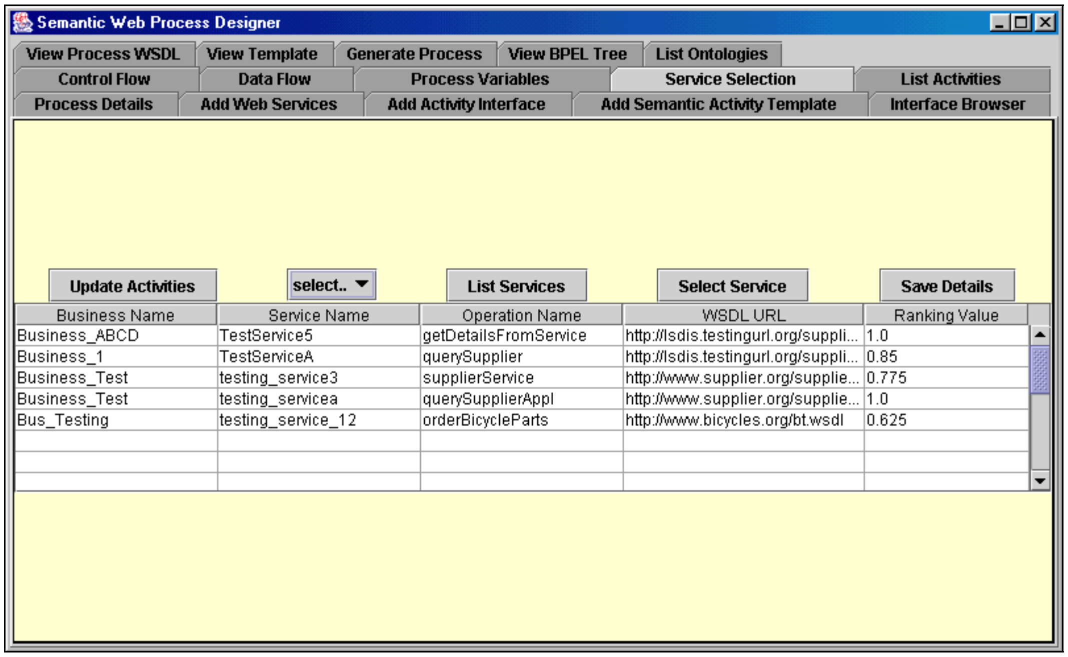
M_i : QoS Matching Value of i^{th} criterion

W : Weight of the i^{th} criterion

Figure 9: Formula for Calculating QoS Matching Value

Considering the sample template and the activity named *QuerySupplierPartner* in it, the *qos-spec* indicates that there is only one QoS requirement and that *delay-time* for the activity should be less than 90 milliseconds. The QoS matching value of the candidate services are calculated using a matching function. If for a service the matching value is 0.9 then the value of M_Q is calculated as (0.9 * 25) / 25 = 0.9

After calculating the ranking values (M_S and M_Q) for the Semantic Matching and QoS dimensions, weights assigned by the user for each of these dimensions are used to calculate the overall ranking. The overall ranking value is the weighted arithmetic mean of the ranking values in each dimension. In the example activity *QuerySupplierPartner*, the weights for Semantic matching and QoS matching are respectively 75 and 25. The overall ranking value is hence calculated as (0.98 * 75 + 0.9 * 25) / (75 + 25) = 0.96. The services are ranked based on the calculated value and the process designer can select one from the list of services (see figure 10):



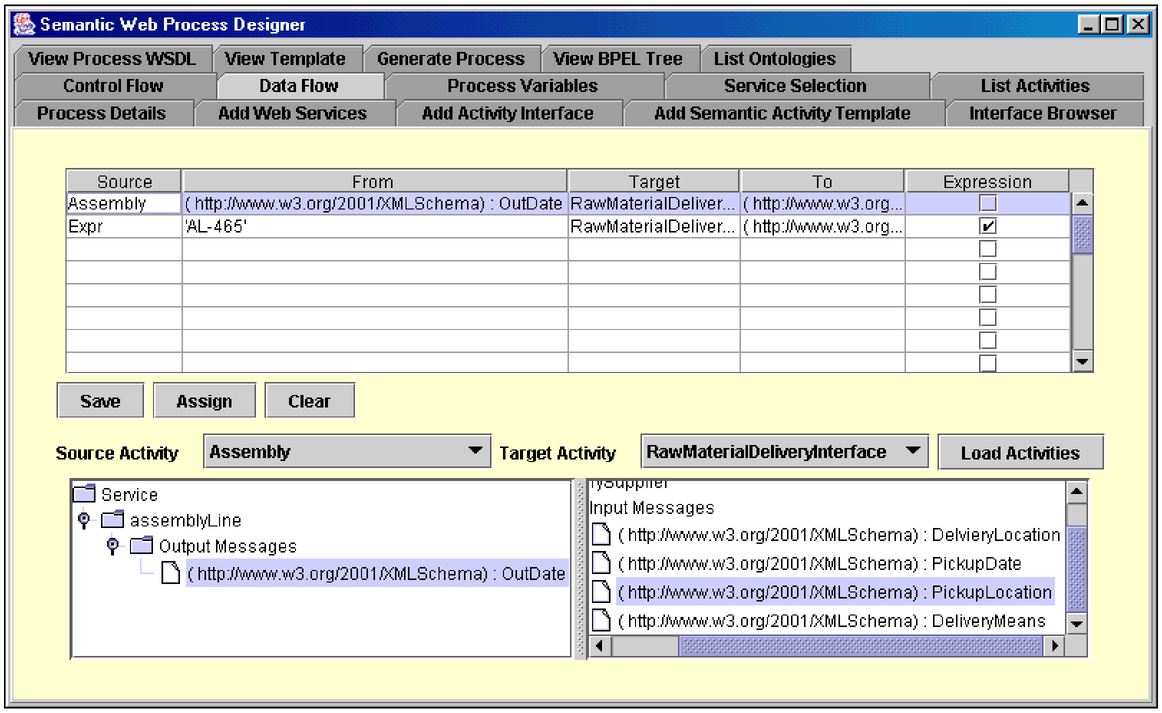
Business Name	Service Name	Operation Name	WSDL URL	Ranking Value
Business_ABCD	TestService5	getDetailsFromService	http://lsdis.testingurl.org/suppli...	1.0
Business_1	TestServiceA	querySupplier	http://lsdis.testingurl.org/suppli...	0.85
Business_Test	testing_service3	supplierService	http://www.supplier.org/supplie...	0.775
Business_Test	testing_servicea	querySupplierAppl	http://www.supplier.org/supplie...	1.0
Bus_Testing	testing_service_12	orderBicycleParts	http://www.bicycles.org/bt.wsdl	0.625

Figure 10: Service Ranking Display Window for an Activity

3.2.7 Process Generation

After designing the template, the process designer can generate an executable process. This involves finding services pertinent to each activity in the process, retrieving their WSDL file, and extracting relevant information to generate the process. The discovery of services is done for each activity independent of other activities. The builder helps in finding Web services for each activity in the process. The user can then link services to incorporate data flow in the process. For this purpose, after obtaining data link requirements between two services, the process generator assists the user in establishing explicit data flow link between output parameters of one service to the input parameters of the other.

The interface in MWSCF that assists in establishing data flow is shown in the figure 11. If the process creator wants to specify the data flow link between two activities, the respective Web service descriptions can be fed to the interface that can be used to establish the data flow in the process. The process creator has to explicitly state the mapping between output parameters of one service to the input parameters of the other.



Source	From	Target	To	Expression
Assembly	(http://www.w3.org/2001/XMLSchema) : OutDate	RawMaterialDeliver...	(http://www.w3.org...	☐
Expr	'AL-465'	RawMaterialDeliver...	(http://www.w3.org...	☑
				☐
				☐
				☐
				☐
				☐
				☐

Source Activity: Assembly
Target Activity: RawMaterialDeliveryInterface

Service
assemblyLine
Output Messages
(http://www.w3.org/2001/XMLSchema) : OutDate

rySupplier
Input Messages
(http://www.w3.org/2001/XMLSchema) : DelvieryLocation
(http://www.w3.org/2001/XMLSchema) : PickupDate
(http://www.w3.org/2001/XMLSchema) : PickupLocation
(http://www.w3.org/2001/XMLSchema) : DeliveryMeans

Figure 11: Interface to Establish Data Flow

The port type data extracted from the WSDL of the process (created by process creator), the data flow requirements obtained from the user and the control flow constructs in the templates are used to generate an executable process. The WSDL descriptions of the participating Web services are also retrieved. These WSDL files will be used during deployment time for binding. During the process generation phase the process WSDL file[vii] is updated with the service link

[vii] WSDL file describing the process. It is designed by the process designer. It has the details of the inputs, outputs and operations of the process

details. A sample executable BPEL process that is generated from the template shown in section 3.2.5 can be found in the appendix.

3.3 XML Repositories

In MWSCF, we have a pool of XML repositories that are used for managing (storing, searching and reusing/sharing) the following

- Ontologies which are used during annotation of services and annotation of semantic activity requirements,
- Semantic process templates which can be opened and edited using the process designer during process design, and
- Activity/service interfaces.

All data (ontologies, interface definitions and process templates) are based on XML and since they are meant for sharing and re-use, using XML repositories will be of much help in this regard. The ontologies in the repositories are identified using the *tModelId* of the published *tModels* that represent these ontologies. In another XML repository, the process templates are categorized and stored. The categorization (taxonomy or ontology) is stored in the same XML repository as that of the process templates. Each category in the categorization has an identifier. The XML repository for storing process templates has different collections based on these identifiers and the process template is categorized by storing the template in a particular collection that represents a category. There is also an interface repository that stores the WSDL interface definitions that can be browsed and selected to link to an activity during the process design. Using these repositories during process design, service discovery or process generation gives the user a powerful environment to compose a process. Xindice is used for implementing these XML repositories.

3.4 Execution Engine

The generated executable process can be executed in any respective execution engine. As the work to develop our own execution engine is underway, at present MWSCF has been implemented and tested using BPWS4J orchestration server. Deploying a process in BPWS4J [38] engine requires a BPEL file containing the process definition, a WSDL description[viii] for the process and the other WSDL descriptions of the web services that a part of the process. The WSDL descriptions of the participating services are needed for deployment time binding. Using the WSDL files of the participating services and other details, an in-memory model [39] of the process is created. Since the present release of BPWS4J does not allow deploying a process without a BPEL file, we have used the BPWS4J printer class to write the in-memory model of the process into a BPEL file. This file is then validated using the tool validator that comes with BPWS4J. Once it is validated, it is deployed in the BPWS4J engine and it can be invoked like any other Web service.

3.5. Summary of Steps in Semantic Process Composition

This section summarizes how Semantic Web Processes are designed, composed and executed.

1. The WSDL description for the desired process is generated using a WSDL editor. In the example discussed in section 2.2, the activities *Receive Order Details* and *Return Details* will be represented in the WSDL file of the process. The process template

[viii] created by the process composer (shown in Figure 14)

will be linked to this file. The process template shown in section 3.2.5 is linked to this file and the operation *getOrderPriceAndDeliveryDate*. This WSDL file can be annotated and published as a Web service.

```
  <?xml version="1.0" encoding="UTF-8"?>
  <definitions
    name="sample"
    targetNamespace="http://lsdis.cs.uga.edu"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://lsdis.cs.uga.edu/sample.wsdl"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <message name="getDetailsResponse">
      <part name="OrderPrice" type="xsd:double"/>
      <part name="PossibleOrderDeliveryDate" type="xsd:date"/>
    </message>
    <message name="getDetailsRequest">
      <part name="ToyId" type="xsd:string"/>
      <part name="ToyCount" type="xsd:int"/>
      <part name="LocationIdToDeliver" type="xsd:string"/>
    </message>
    <portType name="samplePortType">
      <operation name="getDetails">
            <documentation>Distributor's can use this file to input some order on toys to get the price and possible price for
            the order</documentation>
        <input message="tns:getDetailsRequest"/>
        <output message="tns:getDetailsResponse"/>
      </operation>
    </portType>
    <service name="getDetailsBusinessProcess">
   <documentation>This service is a business process in itself..clients just need to call this service to execute the
            process</documentation>
    </service>
   <slt:serviceLinkType name="sampleProcessSLT">
    <slt:role name="anyRole">
     <slt:portType name="tns:samplePortType"/>
    </slt:role>
   </slt:serviceLinkType>
</definitions>
```

Figure 12: Process WSDL

2. A process template is created or opened in the process designer.

3. Activities are added to the template and control flow constructs are added to the templates (if needed).
4. Each activity is semantically annotated and two XML files representing discovery details and QoS specifications are linked to them.
5. Services are discovered (using discovery criteria), ranked (using semantic matching and QoS criteria) and selected for each activity.
6. Data flows between services are established.
7. The executable process is generated by the process generator using the WSDL of the process, the process template and the WSDL files of the participating services.
8. The process is validated, deployed and it is ready for invocation.

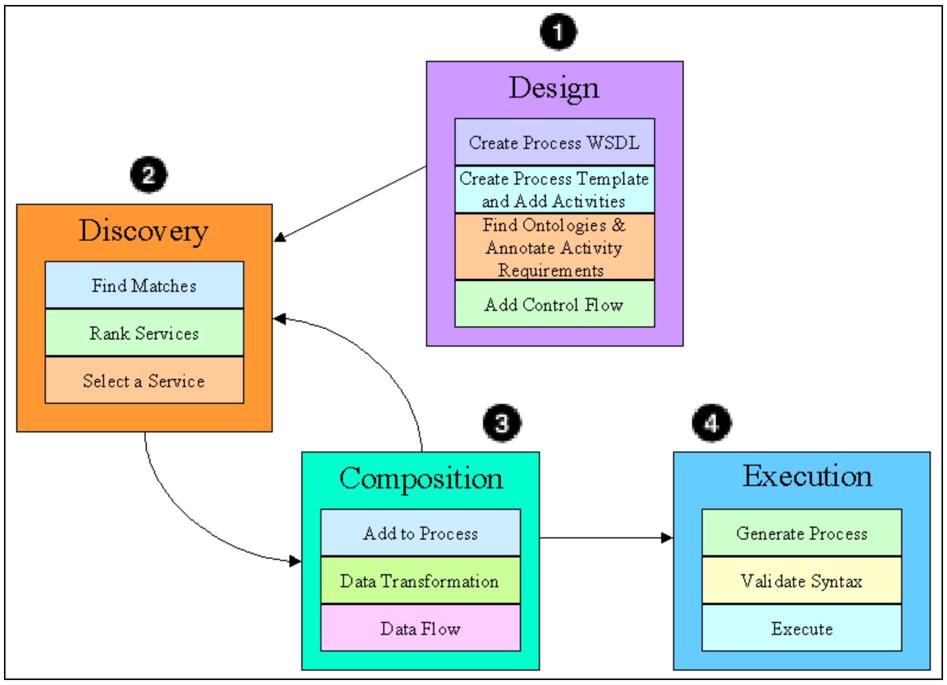


Figure 13: Steps in Semantic Web Process Composition

The above-mentioned steps considered only specifying the activities using Semantic templates. However, any type of activity specification technique discussed in section 3.2 is allowed. Figure 13 shows different phases in the semantic Web process composition. In step 1 (design phase), the figure shows only designing a process using Semantic Activity Templates. Other approaches (specifying an activity using a Web service interface or a concrete Web service implementation) are not shown. Steps 2 and 3 depict that for each activity a service is to be selected and the data flow is established if needed. These steps are done repeatedly until all the activities are bound to a Web service implementation and data flow is established. After this, the actual executable process is generated.

4. Features of MWSCF

The increasing pressure from the market and competitors force the companies to always strive for efficiency and improvement in their processes. This creates the need for an environment that helps building, analyzing and executing processes that are integrated deeply into the business itself. This environment has to provide interactive features for building processes based on business requirements and application semantics. As a step towards this aim, we have completed an initial prototype implementation of the framework as described in section 3 that has the following features:

4.1 Semantically Enriched Service and Process Descriptions

This framework supports building processes using activities that are semantically described. This helps in building the overall process, preserving the semantics of each activity. Each activity will be achieved using a Web service. Binding a service to an activity may not be easy considering the number of services that will be available in the future and their heterogeneity. Present standards like BPEL and BPML provide limited forms of dynamic binding. These standards include *portType* descriptions in the process definitions. Hence, for an activity, only the services that implement the *portType* can be used. This type of hard coded binding is not always suitable for ever-changing e-business applications. Using semantic descriptions not only helps in efficiently finding relevant Web services to execute each activity, it also helps in reasoning about them. Using semantic templates optimized flow or process execution could be calculated and QoS trade-off analysis could be done.

4.2 Configurable Processes

MWSCF provides an architecture where the processes are represented using templates, which are not bound to concrete service implementations. This feature is present in almost all process modeling languages. Our designer tool, in addition, allows users to build a process template and configure each activity in the process template to build an executable process according to the requirements. Using the builder, users can open an existing process template and configure activities in that process. This improves the usability of generic processes for different situations and, in addition, it also enables personalization of the processes based on the configuration parameters. This will help in the success of new business models like outsourcing of Web processes. Off-the-shelf ready-made template can be bought and used instead of building it from the scratch. Common processes can be made available for rent/lease using value added service features like easy configuration, friendly interface for configuration, etc. For example, let us consider a typical composite Web service example "Conference Booking". This is a Web process by itself as it involves several activities like booking travel tickets, booking hotel, registering for the conference, etc. An organization that specializes in this type of business can create an abstract Web process template. The clients of the organization can use this template to feed in parameters like QoS of each activity, ranking and discovery details, etc. to build the actual process. The organization can then execute this configured process in its own service execution engine and return the results back to user. Considering another example, in intra-enterprise process that takes an order and delivers some product to the customer. Depending on the price that the customer is willing to pay for the delivery service QoS parameters of the task that is involved in product delivery can be configured so that the execution engine selects a relevant service that handles the product delivery order satisfying the required QoS criteria.

Comprehensive Framework for Web Service Composition

The MWSCF provides a comprehensive framework for Web Service composition. The MWSCA extends full life-cycle support for achieving semantic Web processes by providing an infrastructure to assist a user starting from service publication and discovery (MWSDI) to designing a process (Process Designer) and executing it (Execution Engine). The designer interface that we have developed as a part of MWSCF provides to easy-to-use interface. Using the built-in powerful discovery technique in the process builder, users can effectively compose a process. The framework provides interfaces where users can view the ranking of services and select the services that are appropriate for their requirements. A unique aspect of the framework is that it automatically retrieves the WSDL files for deployment and extracts information from them to generate the process. The generated executable process will preserve the semantics specified in the process template. All these user interfaces and features implemented, as a part of MWSCF will help in rapid and efficient development of Web processes.

Adaptability to any Process Specification Standard

The present implementation of MWSCF allows constructing process template with BPEL-like syntax. However, the generated executable process can be made independent of BPEL syntax. The process generator is implemented as a separate module independent of other modules in the MWSCF. Hence a pluggable process generator module can replace the existing module that generates BPEL process. If the new process generator module is able to generate executable processes in other specifications based on the template, then it can be used in the framework to generate a process following a different standard. In fact, the best case is when these modules co-exist to help the user generate processes in any standard.

5. Related Work

There have been few research efforts and commercial tools to automate business processes to dynamically compose Web services and to use semantic Web techniques in Web service composition. In this section, we will discuss the related work that deals with Web service composition and other research efforts that are related to our work.

The main focus of our paper is creating semantic process templates. There are a few papers that discuss process templates. For example, [40] discusses reusable process skeletons that implement some conversational logic. It argues that the notion of process templates and service libraries would help to speed up and ease the development of processes that incorporate B2B capability. In our work, we have discussed creating templates and storing them in a repository. To ease the discovery and usage of these templates we have categorized them. Taxonomies of processes have been discussed in [41]. It discusses classifying process instances depending on their characteristics, such as outcome of the process instance, duration of execution, etc. Our work discusses classifying the process templates based on their semantic capabilities.

There are several systems that help in Web service composition. [42] discusses a platform where Web services are declaratively composed and executed in a peer-to-peer environment. It does not perform dynamic discovery. The user has to discover services and then define composition using state-chart diagrams. Then the composition can be executed in a P2P fashion with the help of peer software components coordinators for each constituent Web service. [43] discusses a case based reasoning mechanism for discovery of services to form a composite Web service. The participating services are selected based on the relationships between the services and the constraints involved. eFlow [44] is another system that allows composing, customizing and deploying e-services. The process is modeled as a graph that defines the control and data flow. The nodes represent services in the process. A node could be either an ordinary node which is statically bound to a service or it could be a generic node which is specified with a configuration parameter and a list of services out of which one is bound to the node during run time. They have the notion of process templates and categorizing them in a hierarchy. eFlow also supports dynamic discovery technique using a mediator and the associated rules. Unlike our system neither the discovery mechanism nor the process templates in these systems is semantic.

The fundamental assumption in our work of annotating process definition with ontologies is that the candidate services are semantically described and discovered. The need for semantics in service description has been discussed in quite a few of previous publications [5, 14, 45, 46]. DAML-S is a popular initiative in this direction to describe service capabilities using ontologies. A DAML-S based prototype for semi-automatic composition of services has been discussed in [47]. It provides a composer that will aid the user to select services for each activity in the composition and to create flow specifications to link them. Upon selecting a service, the services that can produce output that could be fed as the input of the selected service are listed after filtering based on profile descriptions. The user can manually select the service that he wants to fit in at a particular activity. After selecting all the services, the system generates a composite process in DAML-S. The execution is done by calling each service separately and passing the results between services according the flow specifications. Step-by-step composition of Web services using DAML-S descriptions has also been discussed in one of our previous papers [48]. It discusses using ontologies to solve the problems in discovery of Web services and to resolve the structural and semantic heterogeneity among these services. The discovery methodology and the set of algorithms discussed in this related work supports discovery of services based on functional requirements and operational metrics. Though all of these related work attempt to solve the same problem (semantic discovery and composition) as that our work, the approach is different. Our work, instead of DAML-S, is based on the industry standards for Web services namely WSDL, UDDI and BPEL4WS. Our work does not discuss sequential composition where the service for an activity in the process is decided before deciding for the next activity. In our work, the entire composition can be defined as a template of semantically annotated activities, and the discovery of services for the activities need not be sequential.

We believe that our composition methodology is better than other present frameworks, because the richness needed in representing services and data in the e-business domain is captured well using ontologies. The use of ontologies to aggregate the products, services, processes and practices within the industry to realize successful net markets has been discussed in [49]. It argues that elements of commerce and relationships between them are used to model market places and identifies ontologies as the means to do that. It states that ontological engineering is the prime requisite for information and services aggregation. It encourages developing and using domain/industry specific ontologies. Representing the products and services using ontologies will help to understand them from the different viewpoints and roles with in that domain/industry. Another related work [50] discusses developing a Universal Business Language. UBL aims to define Business Information Entities at a semantic level. It is something similar to an ontology with a few restrictions [51]. Like ontologies, UBL is aimed to model the real world focusing on a domain (business) to enable semantic interoperability. Our paper discusses using standard vocabularies/ontologies to markup process templates for better interoperability and process generation. [52] identifies semantics as one of the important aspect that B2B protocol standards aims to standardize. It also lists *business content* or *vocabulary* as one of the facets of semantics in B2B standards. In another related work, [53] states that interoperating services need to agree upon vocabularies, document formats and conversation definitions. They add that, in addition to this, agreement has to be there between various horizontal and vertical industry segments to use the standard vocabularies and conversations. [8] proposes Web Services Modeling Framework (WSMF) to enable flexible and scalable e-commerce using Web Services. It discusses a conceptual model for developing, describing and composing Web services. It advocates using semantic Web techniques to deal with the problems of heterogeneity and scalability in e-commerce. It also discusses different types and approaches for scalable mediation between trading partners in e-commerce. We realize the importance of the mediation mechanism to deal with inherent heterogeneity in an open and flexible environment. This itself is a separate research direction. Hence we have deferred that for future work. However, our work reiterates the need for using semantics both in service description and process description to help make the vision of universal interoperability in e-business a reality.

6. Conclusions and Future Work

Web services have created a major wave in the IT industry. Several standards are being proposed, consortia have been created and academic research has increased rapidly. The obvious reasons are the immense power of Web services with regards to e-business and the commercial value behind them. The Web services are indeed useful for easier, faster integration, good in terms of return of investment (ROI), establishing friction free markets, rapid value added assembly of services, etc. However, some of the inherent problems of e-business like scalability and semantic interoperability are not solved by the service-oriented architecture provided by Web services. The convergence of ideas, findings, and results from various initiatives like ebXML, Semantic Web and Web services can bring about better solutions.

Our work is on applying Semantic Web techniques to design e-business processes. Applying [54] Semantic Web findings to Web services technologies and use them for e-business could revolutionize the existing business models and the way they are carried out. This idea has received much attention and support both from academia and industry. For example, [55] addresses these problems with a framework that aims to align concepts known from the Semantic Web and the ebXML initiative and [56] discusses using Web services for implementing business processes and the need for Business Process Management to use Web services for dynamic e-businesses.

Using Semantic Process Templates in MWSCF provides following advantages:

- It helps in rapid process composition. The discovery of services can be performed later and can be delegated to the system.
- It is not necessary to build a process from scratch. Templates can be configured and reused.
- Process design is flexible as it is independent of web service *portTypes*. The change in partner interface without change in the semantics of the interface does not affect the process template or the discovery of services.
- Process re-design is greatly facilitated.
- Ready made templates can act as business/reference models and could be re-used by different organizations that want to implement same process with different services.

Our work does not take into consideration the B2B protocol standards that are crucial for inter-enterprise collaboration and achieving public processes. Our work focuses on the private processes of the enterprises that need to incorporate external services. Typically business transactions occur using some B2B protocol standard [52] that defines the message formats exchanged, sequencing, security, etc. There are several domain specific (Rosetta Net [57]) and domain independent standards (ebXML BPSS [58]]) that focus on different aspects of the B2B protocol. In addition, there are Web services standards (WSCI[ix] [59], WS-CS [60]) to define message exchange in a business collaboration and choreograph the activities in the collaboration. These standards aim to improve the stateless synchronous/uncorrelated asynchronous model of interaction supported by WSDL. A multi-party collaboration is carried out either by using a global controller that coordinates various activities on behalf of the involved parties, or using a mechanism [61] that links a data and messages from visible public processes to the private organizational processes. In any case this is outside the scope of a process specification language like BPML or BPEL[x] and as well as outside the scope of our work too. We foresee the convergence of standards like BPEL, WSCI, ebXML to realize a powerful e-commerce model. This convergence in conjunction with Semantic Web techniques will revolutionize the way businesses are done over the Internet.

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

- Specifying collaboration in the process template and using an algorithm [62] to check the compatibility of a service to engage in a conversation.
- Building own engine for better control over the process and performing sub-conversation in a process.
- Extending process templates to capture the behavior of the intended process at high level and providing direct mapping to the formalisms like state charts or Petri nets etc. to enable process verification [63] and simulation [64, 65].
- Using a template in conjunction with other template [66] and a related analysis.

[ix] The convergence of complementary standards happened in the case of WSCI and BPML aims to provide comprehensive view of role of businesses in a collaboration and the flow of activities that characterize each business

[x] Abstract processes in BPEL are aimed to describe business protocols that specify the sequencing of messages exchanged by one particular partner with its other partners to achieve a business goal . However, an abstract BPEL process defines the business protocol from the perspective of a single entity in the collaboration, while real world business collaboration need a peer-to-peer conversational model.

- Incorporating important e-business aspects like negotiation [67], Service Level Agreements [68] and contracts [69] in template design and service discovery.
- Specifying goal definition as a part of process template. Goal definitions will represent a business goal in highlevel representation in some standard format like UML, and
- Investigating possibilities to represent preconditions and effects in a more expressive way.

CHAPTER 3

CONCLUSIONS

Web services have created a major wave in the IT industry. Several standards are being proposed, consortia have been created and academic research has increased rapidly. The obvious reasons are the immense power of Web services with regards to e-business and the commercial value behind them. The Web services are indeed useful for easier, faster integration, good in terms of return of investment (ROI), establishing friction free markets, rapid value added assembly of services, etc. However, some of the inherent problems of e-business like scalability and semantic interoperability are not solved by the service-oriented architecture provided by Web services. The convergence of ideas, findings, and results from various initiatives like ebXML, Semantic Web and Web services can bring about better solutions.

Our work is on applying Semantic Web techniques to design e-business processes. Applying [54] Semantic Web findings to Web services technologies and use them for e-business could revolutionize the existing business models and the way they are carried out. This idea has received much attention and support both from academia and industry. For example, [55] addresses these problems with a framework that aims to align concepts known from the Semantic Web and the ebXML initiative and [56] discusses using Web services for implementing business processes and the need for Business Process Management to use Web services for dynamic e-businesses.

Using Semantic Process Templates in MWSCF provides following advantages:

- It helps in rapid process composition. The discovery of services can be performed later and can be delegated to the system.
- It is not necessary to build a process from scratch. Templates can be configured and reused.
- Process design is flexible as it is independent of web service *portTypes*. The change in partner interface without change in the semantics of the interface does not affect the process template or the discovery of services.
- Process re-design is greatly facilitated.
- Ready made templates can act as business/reference models and could be re-used by different organizations that want to implement same process with different services.

Our work does not take into consideration the B2B protocol standards that are crucial for inter-enterprise collaboration and achieving public processes. Our work focuses on the private processes of the enterprises that need to incorporate external services. Typically business transactions occur using some B2B protocol standard [52] that defines the message formats exchanged, sequencing, security, etc. There are several domain specific (Rosetta Net [57]) and domain independent standards (ebXML BPSS [58]]) that focus on different aspects of the B2B protocol. In addition, there are Web services standards (WSCI[xi] [59], WS-CS [60]) to define message exchange in a business collaboration and choreograph the activities in the collaboration. These standards aim to improve the stateless synchronous/uncorrelated asynchronous model of interaction supported by WSDL. A multi-party collaboration is carried out either by using a global controller that coordinates various activities on behalf of the involved parties, or using a mechanism [61] that links a data and messages from visible public processes to the private

[xi] The convergence of complementary standards happened in the case of WSCI and BPML aims to provide comprehensive view of role of businesses in a collaboration and the flow of activities that characterize each business

organizational processes. In any case this is outside the scope of a process specification language like BPML or BPEL[xii] and as well as outside the scope of our work too. We foresee the convergence of standards like BPEL, WSCI, ebXML to realize a powerful e-commerce model. This convergence in conjunction with Semantic Web techniques will revolutionize the way businesses are done over the Internet.

[xii] Abstract processes in BPEL are aimed to describe business protocols that specify the sequencing of messages exchanged by one particular partner with its other partners to achieve a business goal . However, an abstract BPEL process defines the business protocol from the perspective of a single entity in the collaboration, while real world business collaboration need a peer-to-peer conversational model.

REFERENCES

[1] Sheth A., van der Aalst, Arpinar B., Processes driving the networked economy, IEEE CONCURRENCY 7: (3) 18-31 JUL-SEP 1999.

[2] Bussler C., B2B Integration Concepts and Architecture, ISBN 3-540-43487-9, Springer. 2003

[3] Benatallah B., Dumas M., Fauvet M. C. and Rabhi F.A. Towards Patterns of Web Services Composition. In S. Gorlatch and F. Rabhi (Eds): "Patterns and Skeletons for Parallel and Distributed Computing". 2002. Springer Verlag (UK).

[4] Christensen E., Curbera F., Meredith G., Weerawarana S., Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001.

[5] Ankolenkar A., Burstein M., Hobbs J. R., Lassila O., Martin D. L., McDermott D., McIlraith S. A., Narayanan S., Paolucci M., Payne T. R. and Sycara K., DAML-S: Web Service Description for the Semantic Web, The First International Semantic Web Conference (ISWC), Sardinia (Italy), June, 2002.

[6] Sheth A. and Meersman R., Amicalola Final Report: SIGMOD Record Special Issue on Semantic Web, Database Management and Information Systems, December 2002.

[7] Handschuh S., Sollazzo T., Staab S., Frank M., and Stojanovic N., Semantic Web Service Architecture - Evolving Web Service Standards toward the Semantic Web. The 15th International FLAIRS Conference, Special Track on Semantic Web, Florida, May 14-16, 2002.

[8] Fensel D. and Bussler C., The Web Service Modeling Framework WSMF, Technical report, Vrije Universiteit Amsterdam, 2002.
http://informatik.uibk.ac.at/users/c70385/wese/wsmf.paper.pdf

[9] METEOR-S: Semantic Web Services and Processes, Applying Semantics in Annotation, Quality of Service, Discovery, Composition, and Execution.
http://lsdis.cs.uga.edu/proj/meteor/SWP.htm, http://swp.semanticweb.org

[10] W3C, Technology and Society Domain, Semantic Web Activity, Semantic Web, http://www.w3.org/2001/sw/

[11] Semantic Web Enabled Web Services, http://swws.semanticweb.org

[12] METEOR Project on Workflow and Semantic Web Process, http://lsdis.cs.uga.edu/proj/meteor/meteor.html

[13] Sheth A., Kochut K., "Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration Systems, in Workflow Management and Interoperability, A. Dogac et al Eds., Springer Verlag, 1999, pp. 35-59.

[14] Sivashanmugam, K., Verma, K., Sheth, A., Miller, J., Adding Semantics to Web Services Standards, Proceedings of the 1st International Conference on Web Services (ICWS'03), Las Vegas, Nevada (June 2003) pp. 395 – 401.

[15] 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 (to appear).

[16] Andrews T., Curbera F., Dholakia H.,Goland Y., Klein J., Leymann F., Liu K., Roller D., Smith D., Trickovic I., Weerawarana S., Ed: Satish Thatte, Business Process Execution Language for Web Services, Version 1.1. May 2003.
ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf

[17] Arkin A., Business Process Modeling Language, http://www.bpmi.org/specifications.esp

[18] van der Aalst W.M.P., Dumas M., ter Hofstede A.H.M., and Wohed P. Pattern-Based Analysis of BPML (and WSCI). QUT Technical report, FIT-TR-2002-05, Queensland University of Technology, Brisbane, 2002.

[19] Peltz C., Web Service Orchestraction:a review of emerging technologies, tools and standards. Jan 2003.
http://devresource.hp.com/drc/technical_white_papers/ WSOrch/WSOrchestration.pdf

[20] Shapiro R., A Comparison of XPDL, BPML and BPEL4WS, Rough Draft, 2002
http://xml.coverpages.org/Shapiro-XPDL.pdf

[21] Comparison of DAML-S and BPEL4WS (initial draft), 2002
http://www.ksl.stanford.edu/projects/DAML/Webservices/DAMLS-BPEL.html

[22] Web Service Choreography Interface (WSCI) 1.0 Specification, FAQs.
http://wwws.sun.com/software/xml/developers/wsci/faq.html

[23] Tolksdorf R., A Dependency Markup Language for Web Services, Web, Web-Services, and Database Systems, NODe 2002 Web and Database Systems 2002, Erfurt, Germany, October 7-10, 2002.

[24] Haberl S., Business Process Description Languages
http://www.cis.unisa.edu.au/~cissh/research/webflow/bpdl.html

[25] Business Process Standards for Web Services.
http://www.webservicesarchitect.com/content/articles/BPSFWSBDO.pdf

[26] 'XLANG: Web Services for Business Process Design',
http://www.gotdotnetcom/team/xml_wsspecs/xlang-c/default.htm.

[27] 'Web service flow language (WSFL) 1.0',
http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf.

[28] Universal Description, Discovery and Integration of Web Services, http://www.uddi.org/

[29] UDDI Spec Technical Committee Specification, 2002. http://uddi.org/pubs/uddi-v3.00-published-20020719.htm

[30] Gong L., "JXTA: A Network Programming Environment", IEEE Internet Computing, (5)3:88--95, May/June

[31] Apache Xindice, http://xml.apache.org/xindice/

[32] Java Web Services Developer Pack. http://java.sun.com/webservices/webservicespack.html

[33] UDDI4J Overview, http://www-124.ibm.com/developerworks/oss/uddi4j/

[34] Web Services Description Language for Java Toolkit, http://www-124.ibm.com/developerworks/projects/wsdl4j/

[35] Chandrasekaran S., Miller J. A., Silver G., Arpinar I. B. and Sheth A. P., "Performance Analysis and Simulation of Composite Web Services," Electronic Markets: The International Journal of Electronic Commerce and Business Media, Special Issue on Web Services, Ronald Klueber and Heiko Ludwig (Guest Editors) Vol. 13, No. 2 (Spring 2003) pp. 18-30. Taylor and Francis Publishing.

[36] Using WSDL in a UDDI Registry, Version 1.08, http://www.oasis-open.org/committees/uddi-spec/doc/bp/uddi-spec-tc-bp-using-wsdl-v108-20021110.pdf

[37] Cardoso, J., Sheth A. and Miller J. Workflow Quality of Service. International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference (ICEIMT/IEMC'02), Valencia, Spain, Kluwer Publishers.

[38] Business Process Execution Language for Web Services JavaTM Run Time, https://www.alphaworks.ibm.com/tech/bpws4j

[39] Business Process with BPEL4WS: Learning BPEL4WS, Part 3, Activities and the in-memory model, http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol3.html

[40] Sayal M., Casati F., Dayal U., Shan M-C., Integrating Workflow Management Systems with Business-to-Business Interaction Standards, 18th International Conference on Data Engineering (ICDE'02), p. 0287.

[41] Casati F. and Shan M., Semantic Analysis of Business Process Executions, SpringerLink:Lecture Notes in Computer Science 2287, p. 287

[42] Sheng Q. Z., Benatallah B., Dumas M., Mak E. O, SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment, Proceedings of 28th International Conference on Very Large Data Bases 2002, Eds. Philip A. Bernstein; Yannis E. Ioannidis; Raghu Ramakrishnan; Dimitris Papadias, Hong Kong China, 20-23 Aug. 2002, Morgan Kaufmann Publishers, USA, pp. 1051-1054.

[43] Limthanmaphon B. and Zhang Y., Web Service Composition with Case-Based Reasoning. In Proc. Fourteenth Australasian Database Conference (ADC2003), Adelaide, Australia. Conferences in Research and Practice in Information Technology, 17. Schewe, K.-D. and Zhou, X., Eds., ACS. 201-208.

[44] Casati F., Ilnicki S., Jin L., Krishnamoorthy V., and Shan M.-C. eFlow: a platform for developing and managing composite e-services. Technical report, Hewlett Packard, 2000.

[45] Supercharging WSDL with RDF: Managing structured Web service metadata, http://www-106.ibm.com/developerworks/library/ws-rdf/?dwzone=ws

[46] Dumas M., O'Sullivan J., Heravizadeh M., Edmond D. and ter Hofstede A.. Towards a semantic framework for service description In Proc. of the IFIP Conference on Database Semantics, Hong Kong, April 2001. Kluwer Academic Publishers.

[47] Sirin E., Hendler J., Parsia B., "Semi-automatic Composition of Web Services using Semantic Descriptions." Accepted to "Web Services: Modeling, Architecture and Infrastructure" workshop in conjunction with ICEIS2003, 2002.

[48] Cardoso J., Sheth A., Semantic e-Workflow Composition, Journal of Intelligent Information Systems (to appear), 2003.

[49] Smith H., The Role of Ontological Engineering in B2B Net Markets, August 2000. http://www.ontology.org/main/papers/csc-ont-eng.html.

[50] Burdett D., Avoiding EDI's Mistakes With Web Services Semantic Interoperability, EAI Journal Volume 4, Number 12 (December 2002), pages 8-11.

[51] Obrst L., Park J., Yim P., Semantics, Ontologies & UBL, http://ubl.cim3.org/~lcsc/tempMeetingResources/for_2002-04-02_a/Semantics_Ontologies_n_UBL_outline_1a.ppt.

[52] Bussler, C.: B2B Protocol Standards and their Role in Semantic B2B Integration Engines. In: Bulletin of the Technical Committee on Data Engineering. March 2001,Vol. 24, No.1. IEEE Computer Society.

[53] Sahai A., and Machiraju V., Enabling a Ubiquitous e-Service Vision on the Internet, e-Services Journal, 1(1), 2002.

[54] Trastour D., Bartolini C., Preist C., Semantic web support for the business-to-business E-Commerce Lifestyle, Autonomous Agents and Multi-Agent Systems.AAMAS'02.

[55] Hofreiter B., Huemer C. and Winiwarter W., Towards Syntax-Independent B2B, ERCIM News No. 51, Special Theme: Semantic Web, October 2002.

[56] Leymann F., Roller D., and Schmidt M.-T., Web services and business process management, IBM Systems Journal, New Developments in Web Services and E-commerce, Volume 41, Number 2, 2002.

[57] RosettaNet Home, http://www.rosettanet.org/RosettaNet/Rooms/DisplayPages/LayoutInitial

[58] ebXML Business Process Specification Schema, Version 1.01, http://www.ebxml.org/specs/ebBPSS.pdf

[59] Web Service Choreography Interface 1.0 , http://wwws.sun.com/software/xml/developers/wsci/wsci-spec-10.pdf

[60] Conversation Support for Web Services, http://www.alphaworks.ibm.com/tech/cs-ws

[61] Bussler C.: The Role of B2B Protocols in Inter-enterprise Process Execution. In Proceedings of Workshop on Technologies for E-Services (TES 2001) (in cooperation with VLDB2001). Rome, Italy, September 2001.

[62] Wombacher A. and Mahleko B., Finding Trading Partners to Establish Ad-Hoc Business Processes, Proceedings of the Tenth International Conference on Cooperative Information Systems 2002 (CoopIS '02), 2002

[63] Hull R., Benedikt M., Christophides V., Su J., E-Services: A Look Behind and Curtain, in Proc. of ACM SIGMOD/PODS 2003.

[64] Bosilj V., Stemberger M. and Jaklic J., "Simulation Modelling Toward E-Business Models Development", International Journal of Simulation Systems, Science & Technology, Special Issue on: Business Process Modelling, Vol. 2, No. 2, 16-29. (2001).

[65] Silver G., Maduko A., Jafri R, Miller J. A. and Sheth A. P., "Modeling and Simulation of Quality of Service for Composite Web Services," Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI'03), Orlando, Florida (July 2003) pp. -. (to appear)

[66] Edmond D. and ter Hofstede A.H.M.. Service composition for electronic commerce. In Proceedings of PACIS-2000 (Pacific Asia Conference on Information Systems), Hong Kong, June 2000.

[67] Benyoucef M. and Keller R. K., An Evaluation of Formalisms for Negotiations in E-Commerce, In Proceedings of the Workshop on Distributed Communities on the Web, pages 45-54, Quebec City, QC, Canada, June 2000. Springer. LNCS 1830.

[68] Sahai A, Durante A, Machiraju V. Towards Automated SLA Management for Web Services. HPL-2001-310. http://www.hpl.hp.com/techreports/2001/HPL-2001-310R1.pdf

[69] Angelov S., Grefen P, An Approach to the Construction of Flexible B2B E-Contracting Processes; CTIT Technical Report 02-40; University of Twente, 2002.

APPENDIX A: GENERATED PROCESS

The following listing shows an executable process generated from the template shown in figures 5 and 6. During process generation 2 files are created. The first file has the executable BPEL process and the second file is a WSDL file (sample-utils.wsdl in this example) that has definitions for all the service link types that are given in *partner* elements.

A sample executable process that is generated is as follows:

```
<process name = "sample"
       xmlns:NS1 = "http://lsdis.cs.uga.edu"
        xmlns:tns-utils = "http://lsdis.cs.uga.edu//sample-utils.wsdl"
         xmlns:NS2="http://lsdis.cs.uga.edu:7001/axis/services/sample/axis/services/sample"
         xmlns:NS3="http://decatur.cs.uga.edu:8080/axis/services/sampleSupplierService7/axis/serv
         ices/ sampleSupplierService7"
         xmlns:NS4="http://decatur.cs.uga.edu:8080/axis/services/sampleDelvieryService23/axis/se
         rvices/sampleDelvieryService23"
      xmlns = "http://schemas.xmlsoap.org/ws/2002/07/business-process/">
  <partners>
   <partner name = "caller" serviceLinkType = "NS1:sampleProcessSLT"/>
   <partner name = "service-provider-1" serviceLinkType = "tns-utils:provider-1-SLT"/>
   <partner name = "service-provider-2" serviceLinkType = "tns-utils:provider-2-SLT"/>
   <partner name = "service-provider-3" serviceLinkType = "tns-utils:provider-3-SLT"/>
   </partners>
 <containers>
     <container name = "request" messageType = "NS1:getDetailsRequest"/>
     <container name = "response" messageType = "NS1:getDetailsResponse"/>
```

```
    <container name = " InventoryCheckActivity -request"
                    messageType = "NS2:checkInventoryRequest"/>
    <container name = " InventoryCheckActivity -response"
                    messageType = " NS2:checkInventoryResponse"/>
    <container name = "DeliveringToysToDistributor-request"
                              messageType = " NS2:arrangeDeliveryRequest"/>
    <container name = "DeliveringToysToDistributor -response"
                              messageType = " NS2:arrangeDeliveryResponse"/>
    <container name = "FixProce-request" messageType = " NS2:fixPriceRequest"/>
    <container name = "FixPrice-response" messageType = " NS2:fixPriceResponse"/>
    <container name = "FinalizePrice-request" messageType = " NS2:finalizePriceRequest"/>
    <container name = "FinalizePrice-response" messageType = " NS2:finalizePriceResponse"/>
    <container name = "QuerySupplierPartner-request"
                              messageType = " NS3:orderToyPartsRequest"/>
    <container name = "QuerySupplierPartner-response"
                              messageType = " NS3:orderToyPartsResponse"/>
    <container name = "DeliveryPartnerService-request"
                              messageType = " NS4:arrangeDeliveryRequest"/>
    <container name = "DeliveryPartnerService-response"
                              messageType = " NS4:arrangeDeliveryResponse"/>
    <container name = "AskAssemblyLine -request"
                              messageType = " NS2:assemblyServiceRequest"/>
    <container name = "AskAssemblyLine -response"
                              messageType = " NS2:assemblyServiceResponse"/>
  </containers>
  <sequence name = "sequence-1">
```

```
<receive name = "receive" partner = "caller" portType = "NS1:samplePortType"
         operation = "getDetails" container = "request"
         createInstance = "yes"/>
<assign>
  <copy>
      <from container = "request" part = "ToyId"/>
      <to container = "InventoryCheckActivity -request" part = "ToyIdentifier"/>
  </copy>
  <copy>
      <from container = "request" part = "ToyCount"/>
      <to container = “InventoryCheckActivity -request" part = "NoOfToys"/>
  </copy>
</assign>
<invoke name = "InventoryCheckActivity" partner = "service-provider-1"
         portType = "NS2:IntraOrgServicesPortType"
        operation = "checkInventory" inputContainer = " InventoryCheckActivity -request"
        outputContainer = "InventoryCheckActivity -response">

<assign>
  <copy>
      <from container = " InventoryCheckActivity -response" part = "ToysCountToOrder"/>
      <to container = "QuerySupplierPartner-request" part = "OrderCount"/>
  </copy>
  <copy>
      <from container = "request" part = "ToyId"/>
      <to container = "QuerySupplierPartner -request" part = "IdentifierForToy"/>
      </copy>
    </assign>
    <switch name  =  "switch-1">
      <case  condition  =  "bpws:getContainerData('InventoryCheckActivity-response',  'return')  =
'no'">
        <sequence name  =  "sequence-3">
            <invoke name = "QuerySupplierPartner" partner = "service-provider-2"
                     portType = "NS3:SupplierPartnerPT"
                     operation = "orderToyParts"
                     inputContainer = "QuerySupplierPartner -request"
                     outputContainer = "QuerySupplierPartner -response">

            <assign>
              <copy>
                    <from container = "QuerySupplierPartner-response" part = "PickUpDate"/>
                    <to container = "DeliveryPartnerService-request" part = "CollectDate"/>
              </copy>
              <copy>
                    <from container = "QuerySupplierPartner-response"
                            part = "PickUpLocationIdentifier"/>
                    <to container = "DeliveryPartnerService-request" part = "CollectLocationId"/>
              </copy>
              <copy>
                    <from expression = "'AA-465'"/>
                    <to container = "DeliveryPartnerService-request" part = "DeliveryLocationId"/>
              </copy>
```

```
           <copy>
                  <from expression = "'AIR-CARGO'"/>
                  <to container = "DeliveryPartnerService -request" part = "DeliveryMeans"/>
           </copy>
          </assign>
       <invoke name = "DeliveryPartnerService" partner = "service-provider-3"
                   portType = "NS4:DeliveryPartnerPortType"
                   operation = "arrangeDelivery"
                   inputContainer = "DeliveryPartnerService-request"
                   outputContainer = "DeliveryPartnerService-response">
          <assign>
           <copy>
                  <from container = "DeliveryPartnerService -response" part = "DeliveryDate"/>
                  <to container = "AskAssemblyLine -request" part = "InDate"/>
           </copy>
          </assign>
       <invoke name = "AskAssemblyLine" partner = "service-provider-1"
                  portType = "NS2: IntraOrgServicesPortType"
                   operation = "assemblyService"
                   inputContainer = "AskAssemblyLine-request"
                   outputContainer = "AskAssemblyLine-response">
     </sequence>
   </case>
</switch>
<assign>
  <copy>
```

```
            <from container = "request" part = "DeliveryLocation"/>

            <to container = "DeliveringToysToDistributor -request" part = "DeliveryLocationId"/>

        </copy>

        <copy>

            <from container = "request" part = "DeliveryMeans"/>

            <to container = "DeliveringToysToDistributor -request" part = "DeliveryMeans"/>

        </copy>

        <copy>

            <from expression = "'AL-465'"/>

            <to container = "DeliveringToysToDistributor -request" part = "PickUpLocationId"/>

        </copy>

        <copy>

            <from container = "AskAssemblyLine-response" part = "OutDate"/>

            <to container = "DeliveringToysToDistributor -request" part = "CollectDate"/>

        </copy>

        <copy>

            <from container = "request" part = "ToyCount"/>

            <to container = "FixPrice-request" part = "ToyCount"/>

        </copy>

      </assign>

    <sequence name = "sequence-2">

      <flow name = "flow-1">

        <invoke name = "DeliveringToysToDistributor" partner = "service-provider-1"

              portType = "NS2:IntraOrgServicesPortType "

              operation = "deliveryService"

              inputContainer = "DeliveringToysToDistributor -request"
            outputContainer = "DeliveringToysToDistributor-response">
      <invoke name = "FixPrice" partner = "service-provider-1"
            portType = "NS2: IntraOrgServicesPortType"
            operation = "fixPrice" inputContainer = "FixPrice-request"
            outputContainer = "FixPrice-response">
   </flow>
  <assign>
   <copy>
          <from container = "FixPrice-response" part = "Price"/>
          <to container = "FinalizePrice-request" part = "DeliveryCost"/>
   </copy>
   <copy>
          <from container = "DeliveringToysToDistributor-response" part = "Cost"/>
          <to container = "FinalizePrice-request" part = "ToysCost"/>
   </copy>
  </assign>
  <invoke name = "FinalizePrice" partner = "service-provider-1"
         portType = "NS2:IntraOrgServicesPortType "
         operation = "finalizePrice" inputContainer = "FinalizePrice-request"
         outputContainer = "FinalizePrice-response">
</sequence>
  <assign>
   <copy>
          <from container = "DeliveringToysToDistributor-response" part = "DeliveryDate"/>
          <to container = "response" part = "PossibleOrderDeliveryDate"/>
   </copy>
```

```
      <copy>
              <from container = "FinalizePrice-response" part = "FinalCost"/>
              <to container = "response" part = "FinalCost"/>
      </copy>
    </assign>
  <reply name = "reply-1" partner = "caller" portType = "NS1:samplePortType"
          operation = "getDetails" container = "response"/>
  </sequence>
</process>
```

APPENDIX B: INSTALLATION GUIDE

The implementation of the framework uses several tools and packages namely Jena, JXTA, Xindice, JWSDP, UDDI4J, WSDL4J, BPWS4J. A short summary of their use and links to download and installation instructions are provided for each of these tools.

Jena DAML-API [Version 1.6.1]

- Provides support for loading DAML ontologies into jena RDF models and performing several functions such as loading an ontology, traversal, etc.
- DOWNLOAD & INSTALLATION: http://www.hpl.hp.com/semweb/download.htm

Project JXTA [Version 65e]

- Provides a set of platform independent protocols that are used to allow devices interconnected in a network to communicate and collaborate using Peer-to-Peer communication.
- DOWNLOAD & INSTALLATION: http://www.jxta.org/project/www/download.html

Xindice [Version 1.1b]

- Database designed to store XML data.
- DOWNLOAD & INSTALLATION: http://xml.apache.org/xindice/download.html

Java Web Services Developer Pack [Version 1.2]

- Used to test and deploy XML applications and Web applications
- The framework uses
    - Java WSDP Registry Server –used to manually or automatically advertise and discover Web services
    - Apache Tomcat container- provides an environment for development and deployment of Java application

- DOWNLOAD & INSTALLATION:http://java.sun.com/webservices/downloads/webservicespack.html

UDDI4J [Version 2.0]

- Provides class libraries that help in interacting with UDDI (Universal Description, Discovery and Integration) registry.
- DOWNLOAD & INSTALLATION:

http://www-124.ibm.com/developerworks/oss/uddi4j/

WSDL4J [Version 1.4]

- Used for creating, representing and manipulating WSDL documents.
- DOWNLOAD & INSTALLATION:

http://www-124.ibm.com/developerworks/projects/wsdl4j

BPWS4J [Version 1.1]

- Provides a platform for creating, validating and executing BPEL4WS processes
- DOWNLOAD & INSTALLATION:
  http://alphaworks.ibm.com/aw.nsf/download/bpws4j

APPENDIX C: USER GUIDE

This user's guide is designed to help understand the functionalities and operations of METEOR-S Process Composition tool. Kindly refer to the *installation guide* for the software that needs to be installed for operating this tool.

After installing the respective software, for running the tool, the user needs to type the following command in the directory that contains the classes for the tool.

```
java –classpath %BCP%;%UDDI%;%JENA%;%WSD%;XIN%;. MainWindow
```

BCP - includes the jar files of BPWS4J

JENA – includes jar files of Jena

WSD – includes jar files of WSDL4J

XIN – includes jar files of Xindice

UDDI – includes jar files of UDDI4J

Before using the tool, it is required to start the start the required UDDI registries (if needed).

Operating the Tool

It should be noted that this tool could be used for "Template Design" and "Process Composition".

Template Design

The template can be designed in three ways.

- Concrete Web Service: Static composition of a process is done by specifying activities using concrete Web service implementations.
- Interface: Activities can be specified by a Web Service interface. Only the services that implement the interface can be discovered using the *tModel id.*

- Semantic Template: Requirements for the activity are given using its semantic characteristics.

Concrete Web Service:

1. Select the ADDWEBSERVICES tab from the GUI.
2. The following fields appear on the GUI:
    - Activity Name: Enter the name of the activity
    - WSDL URL: Enter the URL of the WSDL file pertaining to the activity.
    - Operation Name: Enter the name of the operation that you want to associate to an the activity.
3. After entering corresponding details in these fields click on the SAVE tab.

Interface:

Interface Implementation can be done by two methods.

Method1: To be used when the *tModel id* for a particular type of interface is known.

1. Select the ADDACTIVITYINTERFACE tab from the GUI.
2. The following fields appear on the GUI:
    - Activity Name: Enter the name of the activity.
    - *tModel* id: Enter the id of the *tModel* for a particular type of interface.
    - Operation Name: Enter the name of the operation in the interface.
    - Discovery URL: Enter the access URL of the UDDI registry to be used.
    - Discovery Specification: Specify additional features for basic discovery supported by UDDI.

- Ranking Details: Enter the URL of the XML file that contains the weights for QoS criteria.
- QoS: Enter the URL of the XML file that contains QoS requirements.

3. After entering corresponding details in these fields click on the SAVE tab.

Method2: To be used when you want to select a particular type of interface from a list of interfaces in the repository.

1. Select the INTERFACEBROWSER tab from the GUI.
2. The following fields appear on the GUI:
    - Select: Select a single interface from the lists different interfaces.
    - Operation Name: Enter the name of the operation in the interface.
    - Discovery URL: Enter the access URL of the UDDI registry to be used
    - Discovery Specification: Specify additional features for basic discovery supported by UDDI.
    - Ranking Details: Enter the URL of the XML file that contains the weights for QoS criteria.
    - QoS: Enter the URL of the XML file that contains QoS requirements.
3. After entering corresponding details in these fields click on the SAVE tab.

Semantic Template:

1. Select the ADDSEMANTICACTIVITY tab from the GUI.
2. The following fields appear on the GUI:
    - Activity Name: Enter the name of the activity.
    - Decomposable: under construction. You can ignore this.
    - Ontology URL: Enter the URL of the required functional Ontology.

- Operation Concept: Enter the Operation concept from the required ontology that is associated with the functionality of the service.
- Discovery URL: Enter the access URL of the UDDI registry to be used.
- Discover Specifications: Specify additional details for basic discovery supported by UDDI.
- Ranking Details: Enter the URL of the XML file that contains the weights for Qos criteria.
- Message Part Name: Specify the input and output concepts from the ontologies.
- Message Part Category: under construction.
- Collect: Collect the entered semantics of input / output / precondition / effect for the activity.
- Add: Add the semantics of input / output / precondition / effect to the activity.

3. After entering corresponding details in these fields click on the SAVE tab.

Process Composition

Process Composition assumes that the WSDL file of the process and the template for the process is already present.

The steps for “Process Composition” are as follows.

1. Select the PROCESS NAME tab from the GUI. This step is required to initialize all the required details. The following fields appear on the GUI after selection:
    - Process Name: Enter the name of the Process.
    - URL for WSDL of process: Specify the URL of the WSDL file of the process.
    - Operation Name in process WSDL: Specify the name of the operation for which the process is to be generated.

- Read Details: Click on Read Details after entering all the details.

2. Select SERVICE SELECTION tab from the GUI. This step is required for selecting a particular service for a particular activity. The following fields appear on the GUI after selection:
    - Update Activities: Click on Updates Activities so that all the activities that needed to be discovered are updated.
    - Select: Click on Select, this lists the activities from which on can be selected.
    - List Services: Click on List Services, this lists the services for the selected activity along with the Operation Name, WSDL, Ranking value for each of the services.

        -Operation Name: Displays the operation name of the service selected to be used for the selected activity.

        -WSDL: Displays the WSDL URL for the service selected.

        -Ranking value: Displays the ranking value for the selected activity.
    - Select Services: Select a particular service from the list and click on Select Services.
    - This procedure can be repeated for other activities.
    - After selecting services for all the activities, click on SAVE DETAILS tab which will save all the details in a file called “activities.list”.

3. Select DATA FLOW tab from the GUI. This step is required for establishing the data flow from one activity to another. The following fields appear on the GUI after selection:

- Load Activity: Click on Load Activity, this loads all the activities.
- Source Activity: Click on Source Activity, this loads the output messages of all the activities except for receive where in it loads the input messages.
- Target Activity: Click on Target Activity, this loads all the input messages for all the activities except for reply where in it loads the output messages.
- After selecting the Source Activity and Target Activity click on SELECT AND ASSIGN. This creates a data flow from the source activity to the target activity.
- The data flow is represented in five columns namely Source From Target To Expression.
- If you desire to give a constant as an input to a particular activity, click on the FROM column and delete the details in it and enter the XPATH expression and check in the Expression box.
- Press SAVE and details of the dataflow are saved in a file called SerializedAssignmentDetails.

4. Select PROCESS VARIABLES from the GUI.

   Executable process variables should carry assignments. This is the requirement of the BPEL4WS specifications. Therefore unassigned process variables in the template have to be assigned to values (output of Web services) to generate executable process.

   The following fields appear on the GUI after selection:

   - UPDATE VARIABLES and ACTIVITIY: Click on Update Variables and this updates the variables and activities.

    - SELECT ACTIVITY: Select a particular activity for which you want to assign variable(s). The output messages for this activity are displayed.
    - Select an output message to assign to the variable.
    - SELECT VARIABLE: Select the required variable.
    - ASSIGN: Click on ASSIGN, this assigns the selected output to the selected variable.
    - Repeat the same procedure for other variables and activities
    - SAVE: Click on SAVE, this will save all the assignment details to a file called SerializableVariableAssignmentDetails.

5. Select GENERATE PROCESS

   This generates the process taking into account the template and the WSDL file specified and the Data Flow constructed and the variable assignments done.

   There are few other tabs on the GUI, which have the following functionalities.

   - View BPEL Tree: This tab on the GUI gives a view of the process tree that is generated.
   - List Activities: This tab on the GUI lists all the activities present.
   - List Ontology: This tab on GUI lists all the ontologies present.
   - View Template: This tab on the GUI displays the template used.
   - View Process WSDL: This tab on the GUI displays the WSDL of the process.
   - Control Flow: Under construction.

After the process is generated the output is in the form of two files called the generatedBPEL_final.bpel and utils.wsdl and two intermediate files, namely, the intermediateBPEL.xml and generatedBPEL_initial.bpel. The intermediateBPEL.xml has the skeletal structure of the process. The generatedBPEL_initial.bpel has the generated process but with redundancies with respect to namespaces, containers, partners, etc. The generatedBPEL_final.bpel has the generated process without the redundancies. The utils.wsdl has the service link types of the services used in the generated process.

APPENDIX D: TESTING

The tool implemented performs two main functions, service discovery and process generation. This section describes the testing process that was carried on to carry out these functionalities. To test these functions a process template was created and a process was generated using the template. This template was created in order to generate an executable process that will help in arranging for a trip to a conference. It has four activities getting conference details, booking a hotel, booking a ticket to go to conference location and booking the return ticket. Since the purpose is to demonstrate the working of the tool and not to generate a process with intricate details, a simplified process has been assumed.

The METEOR-S Process Composition tool needs two file as inputs, the process template (shown in figure 18) and the process WSDL (shown in figure 19). The process template, the process WSDL and the generated process (shown in figure 20) are shown in the end of this appendix section.

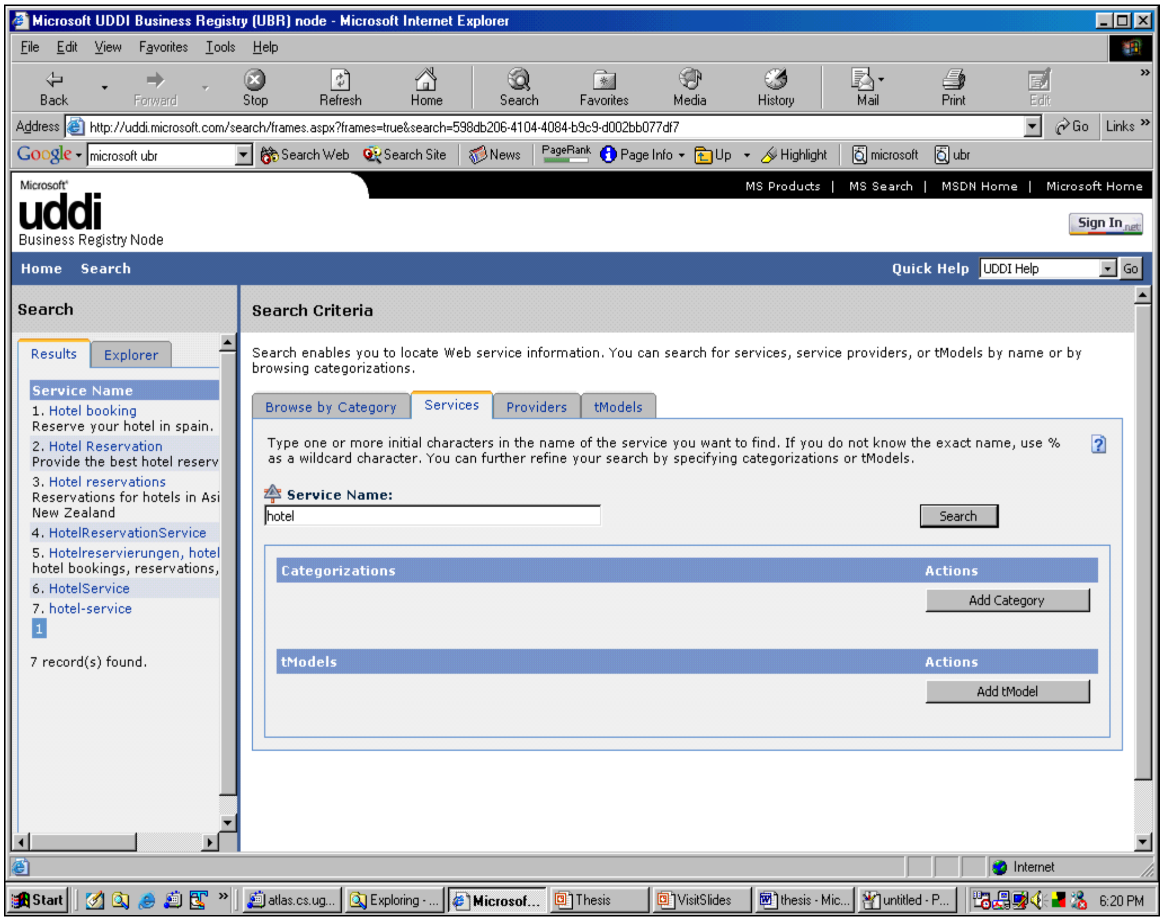
Discovery

This section describes the testing of the discovery mechanism implemented as a part of this work. The process template has an activity to reserve a room in a hotel (for simplicity, the location of the hotel is assumed to be in the location of the conference). If the search for Web services that assist in making room reservation has to be carried out in UDDI it can be done in two ways, keyword based search or category-based search.

Microsoft's UBR (http://uddi.microsoft.com/search/default.aspx) was searched for testing. When the search was keyword based (using 'hotel') it resulted in 7 records. When the search was based on the UNSPC3.1 taxonomy. It resulted in only one service.

The results can be summarized as follows

1. The descriptions of few services were meaningful. Typically descriptions are for human understanding and it is not feasible to use descriptions for automated service discovery or selection.
2. None of the results was linked to a WSDL file. Even if we could locate WSDL files of the services, selection of the most appropriate service based on the semantics of the activity requirements is not possible with the current discovery mechanism supported by UDDI.



Microsoft UDDI Business Registry (UBR) node - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit
Go Links
Search Web Search Site News PageRank Page Info Up Highlight microsoft ubr
MS Products | MS Search | MSDN Home | Microsoft Home
Microsoft uddi Business Registry Node
Sign In
Home Search
Quick Help UDDI Help Go
Results Explorer
Browse by Category Services Providers tModels
Search
Add Category
Add tModel
Internet
Start
atlas.cs.ug... Exploring - ... Microsof... Thesis VisitSlides thesis - Mic... untitled - P...
6:20 PM

Address: http://uddi.microsoft.com/search/frames.aspx?frames=true&search=598db206-4104-4084-b9c9-d002bb077df7

Google: microsoft ubr

Search

Service Name
1. Hotel booking Reserve your hotel in spain.
2. Hotel Reservation Provide the best hotel reserv
3. Hotel reservations Reservations for hotels in Asi New Zealand
4. HotelReservationService
5. Hotelreservierungen, hotel hotel bookings, reservations,
6. HotelService
7. hotel-service

1

7 record(s) found.

Search Criteria

Search enables you to locate Web service information. You can search for services, service providers, or tModels by name or by browsing categorizations.

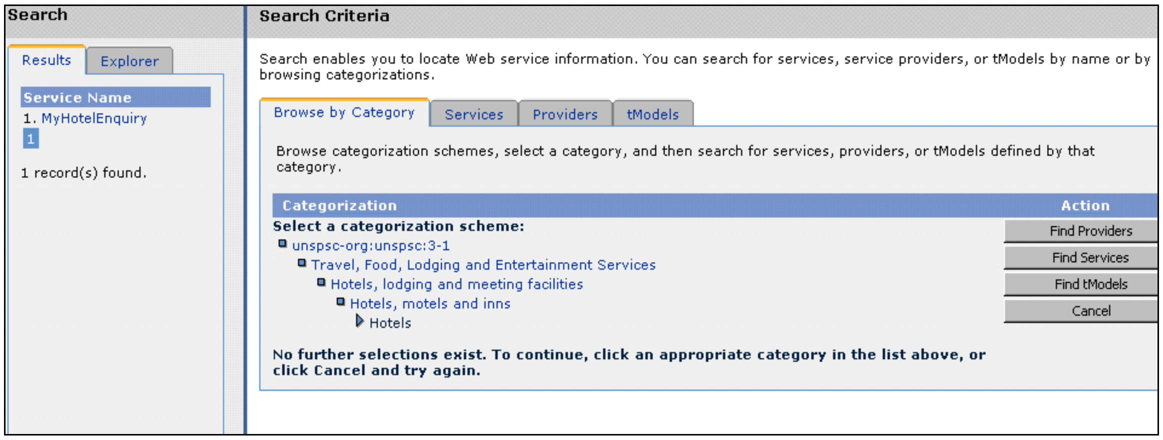
Type one or more initial characters in the name of the service you want to find. If you do not know the exact name, use % as a wildcard character. You can further refine your search by specifying categorizations or tModels.

Service Name: hotel

Categorizations	Actions

tModels	Actions

Figure 14: Search for Hotel Booking services in UBR using keyword


Search

Results | Explorer

Service Name

1. MyHotelEnquiry

1

1 record(s) found.

Search Criteria

Search enables you to locate Web service information. You can search for services, service providers, or tModels by name or by browsing categorizations.

Browse by Category | Services | Providers | tModels

Browse categorization schemes, select a category, and then search for services, providers, or tModels defined by that category.

Categorization | Action

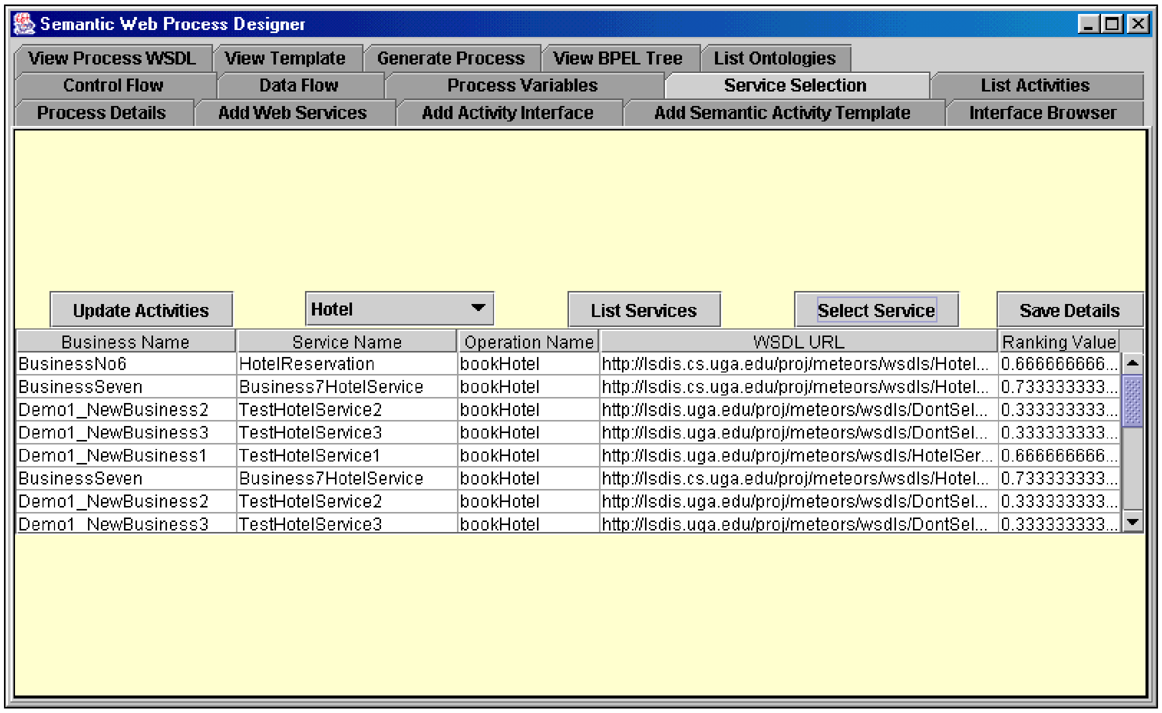
Select a categorization scheme:

- unspsc-org:unspsc:3-1
  - Travel, Food, Lodging and Entertainment Services
    - Hotels, lodging and meeting facilities
      - Hotels, motels and inns
        - Hotels

No further selections exist. To continue, click an appropriate category in the list above, or click Cancel and try again.

Figure 15: Search for Hotel Booking services in UBR using category

In contrast, the discovery mechanism suggested in this paper can be used effectively to discover appropriate Web services based on the semantics of the requirement. For example, the template shown in figure 18 has an activity named *Hotel* that is annotated with *HotelServiceSemantics*. This semantic specification is defined in the *criteria* element of the template. It states that one of the inputs of the service should be compatible with the concept *CheckInDate* in the ontology identified by *HotelOnto* and so on. The METEOR-S Process Composition tool discovery mechanism finds all related services and rank them based on their semantics. The following screenshot shows the service selection step in the process composition using our tool.



Business Name	Service Name	Operation Name	WSDL URL	Ranking Value
BusinessNo6	HotelReservation	bookHotel	http://lsdis.cs.uga.edu/proj/meteors/wsdls/Hotel...	0.666666666...
BusinessSeven	Business7HotelService	bookHotel	http://lsdis.cs.uga.edu/proj/meteors/wsdls/Hotel...	0.733333333...
Demo1_NewBusiness2	TestHotelService2	bookHotel	http://lsdis.uga.edu/proj/meteors/wsdls/DontSel...	0.333333333...
Demo1_NewBusiness3	TestHotelService3	bookHotel	http://lsdis.uga.edu/proj/meteors/wsdls/DontSel...	0.333333333...
Demo1_NewBusiness1	TestHotelService1	bookHotel	http://lsdis.uga.edu/proj/meteors/wsdls/HotelSer...	0.666666666...
BusinessSeven	Business7HotelService	bookHotel	http://lsdis.cs.uga.edu/proj/meteors/wsdls/Hotel...	0.733333333...
Demo1_NewBusiness2	TestHotelService2	bookHotel	http://lsdis.uga.edu/proj/meteors/wsdls/DontSel...	0.333333333...
Demo1_NewBusiness3	TestHotelService3	bookHotel	http://lsdis.uga.edu/proj/meteors/wsdls/DontSel...	0.333333333...

Figure 16: Search for Hotel Booking services based on semantics of requirements

The tool selects the services and ranks based on the semantics of the requirement and the QoS criteria. Using the overall ranking value, a service can be selected. The higher-ranking value implies better match between activity requirement and service specification. This type of service discovery and ranking mechanism is not supported by UDDI. Hence the semantic discovery mechanism can improve upon the existing Web service discovery techniques and can be used effectively in automated Web service discovery.

Process Generation

The second functionality of the tool is process generation. This section explains testing carried our to check the functionality. The template describes various activities and their sequence in the process of arranging for a trip to a conference. The template describes the process shown in the following figure:

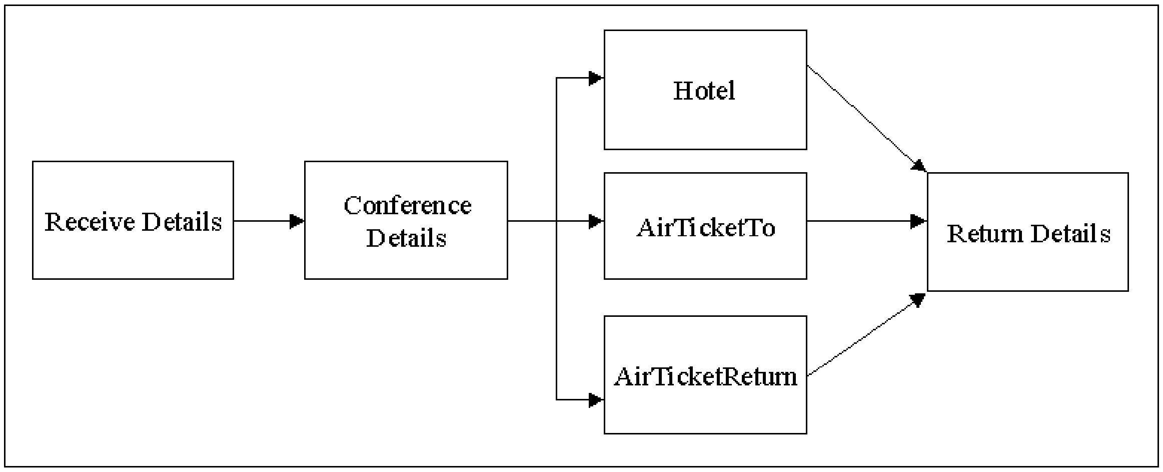


Figure 17: Web Process that is described in the template

Two of the activities *AirTicketTo* and *AirTicketReturn* are specified as activities of type *serviceImpl*. There are two other activities *ConferenceDetails* and *Hotel* of types *WSInterface* and *SemanticActivityTemplate* respectively. After the discovery and service selection, data flow has to be established among the activities. Based on the template, WSDL details and the data flow specified during build time, the process was generated successfully. The process is listed in the end of the appendix. The control flow in the template for this process is different from the control flow specified in the template shown in figure 1 and the process generation was successful. Hence the process generator is working well for different kind of control and data flow constructs.

Template Follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<process-template name="sample process" HotelOnto="HotelOntologyTModel"
FinanceOnto="FinanceOntologyTModel" GeneralTrade="GeneralTradeOntology">
  <sequence name="sequence-1">
    <receive name="receive" processWSDL-operation = "arrange4Conference"/>
        <invoke-activity name="ConferenceDetails" type="WSInterface"
```

```
          tModel-id="uuid:f5e283fd-0bf5-e283-b52d-b8acf6332a17"
          qos-spec="qos-1"
          ranking-weights="ranking-1"
          operation-name="getConferenceDetails"/>
        <flow name="flow-1">
              <invoke-activity name="Hotel" type="SemanticTemplate"
                    semantic-spec="HotelServiceSemantics"
                    ranking-weights="ranking-2"
                    discovery-URL=
                "http://westpoint.cs.uga.edu:8080/registry-server/RegistryServerServlet"/>
                  <invoke-activity name="AirTicketTo" type="ServiceImpl"
                    wsdl-URL=
                "http://lsdis.cs.uga.edu/proj/meteors/wsdls/TravelServiceNo1.wsdl"
                    operation-name="bookAirTicket"/>
                  <invoke-activity name="AirTicketReturn" type="ServiceImpl"
                    wsdl-URL=
                "http://lsdis.cs.uga.edu/proj/meteors/wsdls/TravelServiceNo1.wsdl"
                    operation-name="bookAirTicket"/>
        </flow>
    <reply name="reply" processWSDL-operation = "arrange4Conference"/>
  </sequence>
      <qos-spec name="qos-1">
        <delaytime unit="milliseconds" qualifier="LT">200</delaytime>
    </qos-spec>
    <ranking-weights name="ranking-2">
          <semantic-matching-weight>100</semantic-matching-weight>
```

```
            <input-semantics-weight>30</input-semantics-weight>
            <output-semantics-weight>20</output-semantics-weight>
            <operational-semantics-weight>30</operational-semantics-weight>
        </ranking-weights>
        <ranking-weights name="ranking-1">
            <qos-weight>100</qos-weight>
        </ranking-weights>
        <criteria>
          <semantic-spec name="HotelServiceSemantics">
            <input name="anyInput1" onto-concept="HotelOnto:CheckInDate"/>
            <input name="anyInput2" onto-concept="HotelOnto:CheckOutDate"/>
            <input name="anyInput3" onto-concept="HotelOnto:GuestName"/>
            <input name="anyInput4" onto-concept="FinanceOnto:CrCardNo"/>
            <input name="anyInput5" onto-concept="FinanceOnto:CrCardType"/>
            <input name="anyInput6" onto-concept="FinanceOnto:CrCardExpiryDate"/>
            <output name="anyOutputName"
                onto-concept="GeneralTrade:Confirmation"/>
            <operation name="operationConcept"
                operation-concept="HotelOnto:RoomBooking"/>
          </semantic-spec>
          </criteria>
</process-template>
```

Figure 18: Process Template used to generate the process

Process WSDL follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
 name="ConferenceScenarioNo1"
 targetNamespace="http://lsdis.cs.uga.edu/ConferenceScenarioNo1.wsdl"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:tns="http://lsdis.cs.uga.edu/ConferenceScenarioNo1.wsdl"
 xmlns:slt="http://schemas.xmlsoap.org/ws/2002/07/service-link/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <message name="arrange4ConferenceResponse">
  <part name="HotelConfirmation" type="xsd:string"/>
  <part name="ToTicketConfirmation" type="xsd:string"/>
  <part name="ReturnTicketConfirmation" type="xsd:string"/>
 </message>
 <message name="arrange4ConferenceRequest">
  <part name="AttendeeName" type="xsd:string"/>
  <part name="ConferenceId" type="xsd:int"/>
  <part name="CCNo" type="xsd:string"/>
  <part name="CCType" type="xsd:string"/>
  <part name="CCExpDate" type="xsd:string"/>
 </message>
 <portType name="ConferenceScenarioNo1PortType">
  <operation
    name="arrange4Conference">
    <documentation>Books hotel, travel tickets for a conference</documentation>
```

```
            <input message="tns:arrange4ConferenceRequest"/>
            <output message="tns:arrange4ConferenceResponse"/>
        </operation>
    </portType>
  <slt:serviceLinkType name="sampleConferenceArrangerSLT">
    <slt:role name="anyRole">
      <slt:portType name="tns:ConferenceScenarioNo1PortType"/>
    </slt:role>
  </slt:serviceLinkType>
</definitions>
```

Figure 19: Process WSDL used to generate the process

Generated process is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<process xmlns="http://schemas.xmlsoap.org/ws/2002/07/business-process/"
xmlns:NS1="http://lsdis.cs.uga.edu/ConferenceScenarioNo1.wsdl"
xmlns:NS2="http:/lsdis.cs.uga.edu/ConferenceServiceNo1.wsdl"
xmlns:NS3="http://lsdis.cs.uga.edu/HotelServiceNo1.wsdl"
xmlns:NS4="http://lsdis.cs.uga.edu/TravelServiceNo1.wsdl"
xmlns:tns-utils="http://lsdis.cs.uga.edu/sample-utils.wsdl">

  <partners>
  <partner name="caller" serviceLinkType="NS1:dummySLT"/>
  <partner name="service-provider-1" serviceLinkType="tns-utils:provider-1-SLT"/>
  <partner name="service-provider-2" serviceLinkType="tns-utils:provider-2-SLT"/>
```

```
<partner name="service-provider-3" serviceLinkType="tns-utils:provider-3-SLT"/>
</partners>

<containers>
  <container messageType="NS1:arrange4ConferenceRequest" name="receive"/>
  <container messageType="NS2:getConferenceDetailsRequest"
                    name="ConferenceDetails-request"/>
   <container messageType="NS2:getConferenceDetailsResponse"
                    name="ConferenceDetails-response"/>
  <container messageType="NS3:bookHotelRequest" name="Hotel-request"/>
  <container messageType="NS3:bookHotelResponse" name="Hotel-response"/>
  <container messageType="NS4:bookAirTicketRequest"
                    name="AirTicketTo-request"/>
  <container messageType="NS4:bookAirTicketResponse"
                    name="AirTicketTo-response"/>
  <container messageType="NS4:bookAirTicketRequest"
                    name="AirTicketReturn-request"/>
  <container messageType="NS4:bookAirTicketResponse"
                    name="AirTicketReturn-response"/>
  <container messageType="NS1:arrange4ConferenceRequest"
                    name="response"/>
</containers>

<sequence name="sequence-1">
```

```
  <receive container="receive" createInstance="yes" name="receive"
operation="arrange4Conference" partner="caller"
portType="NS1:ConferenceScenarioNo1PortType"/>

  <assign name="ConferenceDetails">

  <copy><from container="receive" part=" ConferenceId"/><to container="ConferenceDetails-
request" part=" ConferenceId"/></copy><copy><from expression="'2003'"/><to
container="ConferenceDetails-request" part=" Year"/></copy>

  </assign>

  <invoke inputContainer="ConferenceDetails-request" name="ConferenceDetails"
operation="getConferenceDetails" outputContainer="ConferenceDetails-response"
partner="service-provider-1" portType="NS2:ConferenceServiceNo1PortType"/>

  <flow name="flow-1">

    <assign name="Hotel">

    <copy><from container="receive" part=" CCNo"/><to container="Hotel-request" part="
CreditCardNo"/></copy><copy><from container="receive" part=" CCExpDate"/><to
container="Hotel-request" part=" CreditCardExpDate"/></copy><copy><from
container="receive" part=" CCType"/><to container="Hotel-request" part="
CreditCardType"/></copy><copy><from container="receive" part=" AttendeeName"/><to
container="Hotel-request" part=" GuestName"/></copy><copy><from
```

```
container="ConferenceDetails-response" part=" StartDate"/><to container="Hotel-request"
part=" DateIn"/></copy><copy><from container="ConferenceDetails-response" part="
EndDate"/><to container="Hotel-request" part=" DateOut"/></copy>

    </assign>

    <invoke inputContainer="Hotel-request" name="Hotel" operation="bookHotel"
outputContainer="Hotel-response" partner="service-provider-2"
portType="NS3:HotelServiceNo1PortType"/>

    <assign name="AirTicketTo">

    <copy><from container="ConferenceDetails-response" part=" Location"/><to
container="AirTicketTo-request" part=" To"/></copy><copy><from
expression="'HOMELOCATION'"/><to container="AirTicketTo-request" part="
From"/></copy><copy><from container="ConferenceDetails-response" part=" StartDate"/><to
container="AirTicketTo-request" part=" JourneyDate"/></copy><copy><from
container="receive" part=" AttendeeName"/><to container="AirTicketTo-request" part="
PassengerName"/></copy>

    </assign>

    <invoke inputContainer="AirTicketTo-request" name="AirTicketTo"
operation="bookAirTicket" outputContainer="AirTicketTo-response" partner="service-provider-
3" portType="NS4:TravelServiceNo1PortType"/>
```

```
    <assign name="AirTicketReturn">

    <copy><from container="ConferenceDetails-response" part=" Location"/><to
container="AirTicketReturn-request" part=" From"/></copy><copy><from
expression="'HOMELOCATION2'"/><to container="AirTicketReturn-request" part="
To"/></copy><copy><from container="ConferenceDetails-response" part=" EndDate"/><to
container="AirTicketReturn-request" part=" JourneyDate"/></copy><copy><from
container="receive" part=" AttendeeName"/><to container="AirTicketReturn-request" part="
PassengerName"/></copy>

    </assign>

    <invoke inputContainer="AirTicketReturn-request" name="AirTicketReturn"
operation="bookAirTicket" outputContainer="AirTicketReturn-response" partner="service-
provider-3" portType="NS4:TravelServiceNo1PortType"/>

  </flow>

  <assign name="reply">

  <copy><from container="Hotel-response" part=" Confirmation"/><to container="response"
part=" HotelConfirmation"/></copy><copy><from container="AirTicketTo-response" part="
Confirmation"/><to container="response" part=" ToTicketConfirmation"/></copy><copy><from
container="AirTicketReturn-response" part=" Confirmation"/><to container="response" part="
ReturnTicketConfirmation"/></copy>
  </assign>
```

```
    <reply container="response" name="reply" operation="arrange4Conference"
partner="caller" portType="NS1:ConferenceScenarioNo1PortType"/>

  </sequence>

</process>
```

Figure 20: Process Generated based on template shown in figure 18