Dynamic Web Service Composition in METEOR-S

Rohit Aggarwal, Kunal Verma, John Miller, William Milnor[1]

LSDIS Lab, University of Georgia, Athens, 30602
{aggarwal, verma, jam, milnor}@cs.uga.edu

Abstract

Creating Web processes using Web services technology gives us the opportunity for selecting new services which best suit our need at the moment. Doing this automatically would require us to quantify our criteria for selection. In addition, there are challenging issues of correctness and optimality. We present a Dynamic Web Service Composition tool in METEOR-S, a system for dynamic composition of Web services, which allows the process designers to design processes, based on business and process constraints. Our approach is to reduce dynamic composition of Web services to a constraint satisfaction problem. It uses a multi-phase approach for constraint analysis. This work was done as part of the METEOR-S framework, which aims to support the complete lifecycle of semantic Web processes.

1. INTRODUCTION

Research initiatives in the areas of workflows, information systems and databases are being directly employed by businesses to model, design and execute their critical processes. With the growth of the process centric paradigm, a greater level of integration is seen across functional boundaries, leading to higher productivity. There is however, a growing need for dynamic integration with other business partners and services. Several architectures have been postulated for more flexible and scalable process environments; for example, [Sheth et al., 1999] discusses process portal, process vortex and dynamic trading processes as three points in a space recognizing various types of business relationships, as well as more technical issues of scale and dynamism. The growth of Web services and service oriented architecture (SOA) offer an attractive basis for realizing such architectures.

The focus of this paper is on a limited form of dynamic composition of Web services. We refer to the process of automatically binding services to abstract processes as dynamic composition. The key to our approach is in allowing users to capture high level specifications as abstract processes. We use Semantic Web [Berners-Lee et al., 2001] technologies to represent the requirements for each service in the process. We build on earlier work in Semantic Web Services for automated discovery of Web services [Cardoso and Sheth, 2002, Verma et al., 2004, Paolucci et al., 2002] based on the users requirements. A requirement for automated discovery is annotation [Patil et al., 2004]. After discovery, the candidate services must be selected on the basis of process and business constraints. We present a multi-phase approach for constraint representation, cost estimation and optimization for constraint analysis and optimal selection of Web services.

Consider a supply-chain management scenario where a manufacturer gets a large order and it must procure parts from suppliers based on its inventory levels. Currently, the manufacturer would either be tightly integrated to a few suppliers or he would go to an exchange to find suppliers. So far, the high cost of integration made it impractical for the manufacturer to integrate a number of suppliers into their processes and dynamically choose the suppliers based on their quotes. With the help of industry wide acceptance of standards like Business Process Execution Language for Web Services (BPEL4WS) [Andrews et al., 2003], Web Service Description Language (WSDL) and Simple Object Access Protocol (SOAP), Web Services offer the potential of low cost and immediate integration with other applications. However, initial research and most of the industrial technology development efforts thus far have been focused on static integration using Web services [Curbera et al., 2002]. In this paper, we present the Dynamic Web Service Composition in METEOR-S, which is a comprehensive framework for the composition of dynamic Web services. METEOR-S back-end would allow the manufacturers to dynamically compose the services based on the requirements and the process constraints.

This work has been done as a part of the METEOR-S [METEOR-S, 2003] project in the LSDIS Lab at the University of Georgia, which aims to create a comprehensive framework for dynamic web processes. Section 2 discusses the METEOR-S framework while section 3 explains the architecture of the METEOR-S

[1] We would like to acknowledge Dr. Amit Sheth's important contributions to this paper. He could not become a co-author of this paper since he is the program co-chair of the conference.

backend. Section 4 introduces the abstract process designer which involves creating a representation of Web processes. Our Semantic Web service discovery algorithm is presented in section 5. Section 6 discusses the constraint analyser followed by the run-time module in section 7. In section 8, we compare our approach to other related work and finally in section 9, we present conclusions and future work.

2. METEOR-S

The METEOR (Managing End-To-End OpeRations) project in the LSDIS Lab focused on workflow management techniques for transactional workflows [Sheth, 1996]. Its follow-on project, which incorporates workflow management for semantic Web services, is called METEOR-S (METEOR for Semantic Web Services). A key feature in this project is the usage of semantics for the complete lifecycle of semantic Web processes, which represent complex interactions between semantic Web services.

The main stages of creating semantic Web processes have been identified as development, annotation, discovery, composition and orchestration. A key research direction of METEOR-S has been exploring different kinds of semantics, which are present in these stages. We have identified data, functional, Quality of Service and execution semantics as different kinds of semantics and are working on formalizing their definitions. A detailed explanation of the underlying conceptual foundation of METEOR-S is present in [Sheth, 2003].

From an architectural point of view, we divide METEOR-S in two main parts – the front-end and the back-end. The back-end, which is the focus of this paper, covers the abstract process design, discovery, constraint analysis and execution stages. The main components of the back-end are the 1) Abstract Process Designer, 2) Discovery Engine, 3) Constraint Analyser and 4) Execution Engine. The front-end of METEOR-S which covers annotation and publication is discussed in [Rajasekaran et al., 2004]

We provide a representational framework for accommodating Data semantics, Functional semantics, Quality of Service and Execution Semantics to support activities in the complete Web process lifecycle. For background, we will provide brief descriptions of data, functional and QoS semantics in this section. Because of the complexity of the Execution Semantics, we defer its discussion to our ensuing work in [Verma et al., 2004c]

2.1. Data Semantics

For Web services to communicate with each other, they should understand the semantics of each others data. Use of semantics in multi-databases was discussed in [Sheth and Kashyap, 1992], which presented methodologies to find semantic similarities between disparate objects. Inputs, outputs and exceptions of Web services in a domain can be represented using OWL [McGuinness et al., 2004] ontologies. In fact, ontologies can be replaced (with compromises, of course) with standard vocabularies or taxonomies. For example, ebXML Core Component Dictionary [ebXML, 2001] or RosettaNet Technical Dictionary [RosettaNet, 2002] can be used to represent input/output/exception data in Web services.

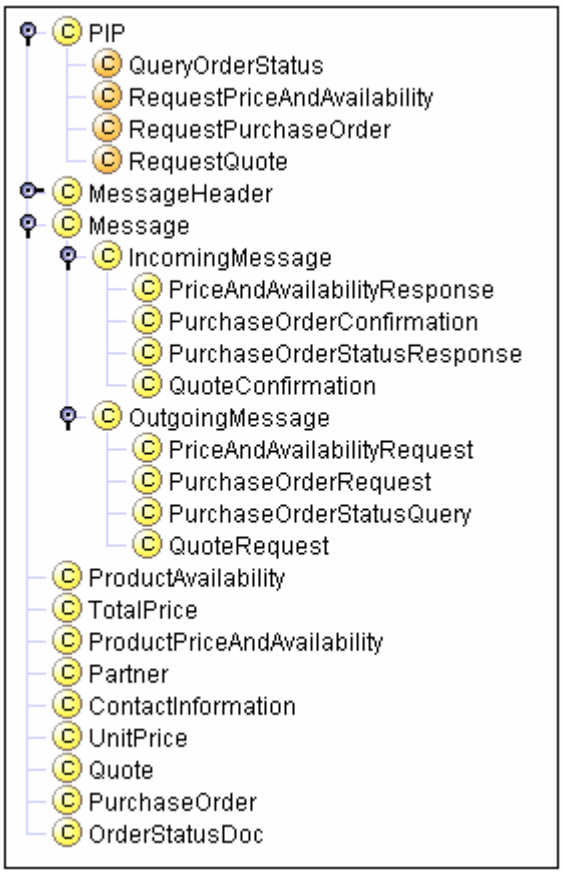

Figure 1: Snapshot of a part of RosettaNet Ontology

In Figure 1, we have shown a snapshot of RosettaNet Ontology [Azami, 2004] which we are creating using RosettaNet PIP's in OWL.

2.2. Functional Semantics

The functional semantics of a Web service operation is a combination of its data semantics, and classification of its operations functionality as well as its pre-conditions and post-conditions. Let s be a service and o be one of its operations then, we define functional semantics of an operation of a Web service as $F(s, o) = <F_c(s,o), I(s,o), O(s,o), E(s,o), P(s,o), Po(s,o)>$, where

$F_c(s,o)$	Functional classification of operation 'o' in terms of ontological concepts
I(s,o)	Inputs of operation 'o' in terms of ontology concepts
O(s,o)	Outputs of operation 'o' in terms of ontology concepts
E(s,o)	Exceptions throwable during execution of operation 'o' in terms of ontology concepts
P(s,o)	Pre-conditions of operation 'o' in terms of ontological concepts
Po(s,o)	Post-conditions of operation 'o' in terms of ontological concepts

We have shown a custom sub-ontology in Figure 2 which is an extension of the of the RosettaNet ontology shown in Figure 1. Functions of a web service can be defined as a set of related operations which can be mapped to concepts in the ontology in order to get functional semantics of a Web service.

Functional Semantics

$F_c(s,o)$	#OrderBattery
I(s,o)	#PurchaseOrderRequest
O(s,o)	#PurchaseOrderConfirmation
E(s,o)	InvalidContactInformation
P(s,o)	ContactInformation <> null
Po(s,o)	

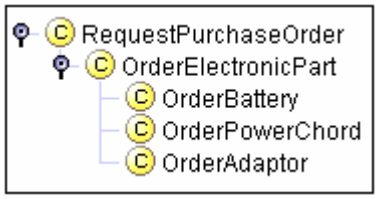

Figure 2: Custom sub-ontology of RosettaNet

2.3. Quality of Service (QoS) Specifications

The Quality of Service (QoS) specifications of a Web service characterize performance and other qualitative/quantitative aspects of Web services. In order for the suppliers of services to understand each others QoS terms, a common understanding must be reached on the meaning of the terms. Ontologies can be used to represent and explicate the semantics of these parameters. [Cardoso and Sheth, 2003; Cardoso et al., 2004; Zeng et al., 2003] have described generic QoS metrics based on time, cost, availability and reliability. We have created an extensible ontology to represent the generic metrics, as well as domain specific QoS metrics. We define the QoS of an operation j of a Web service i as follows:

$QoS(s_i, o_j) = <T(s_i,o_j), C(s_i,o_j), R(s_i,o_j), A(s_i,o_j), DS_1(s_i,o_j), DS_2(s_i,o_j),..., DS_N(s_i,o_j)>$ where,

T(s,o)	Execution time of Web service 's' when operation 'o' is invoked
C(s,o)	Cost of Web service 's' when operation 'o' is invoked
R(s,o)	Reliability of Web service 's' when oper 'o' is invoked
A(s,o)	Availability of Web service 's' when oper 'o' is invoked
$DS_i(s,o)$	Service/operation level domain specific QoS metrics

Each metric specification consists of a quadruple.

$QoS_q(s,o)$ = <name, comparisonOp, val, unit>, where 'name' is the parameter name, comparisonOp is a comparison operator, 'val' is a numerical value, and unit is the metric unit. For example QoS can be represented as follows:

Name	Val	Unit
Time	60	Seconds
Cost	100	Dollars
Reliability	.9	
Availability	.8	

For details on QoS metrics see [Cardoso et al., 2004].

The overall semantics of an operation are defined as:

$OP(s_i,o_j) = <F(s_i,o_j), QoS(s_i,o_j)>$, where

$F(s_i,o_j)$ and $QoS(s_i,o_j)$ are functional semantics and QoS specifications of the required operation as defined in sections 2.2 and 2.3.

Services can be represented as Service Advertisements (SA) which are the actual services/WSDL files published in the registry.

3. Architecture

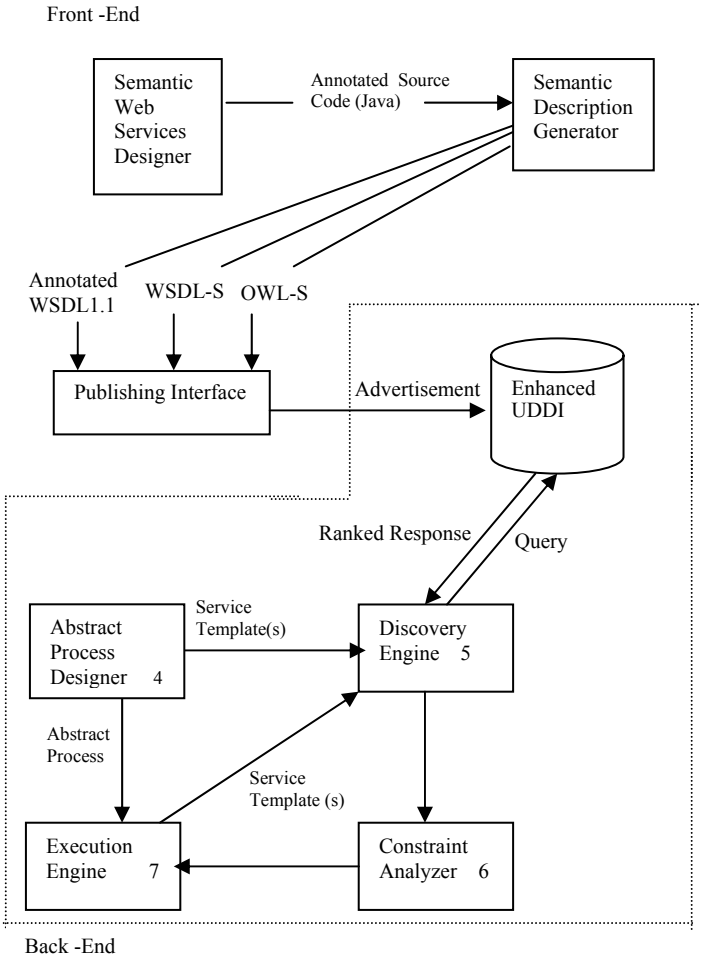

Figure 3: Architecture of METEOR-S

Figure 3 shows the architecture of the METEOR-S which is made up of a front-end [Rajasekaran et al., 2004] and back-end. The abstract process designer module has been discussed in [Sivashanmugan et al., 2003]. It allows us to design and abstractly represent the functionality of the required services using service templates. The Discovery Engine [Verma et al., 2004a] is an interface over UDDI [UDDI, 2002] registries to provide semantic publication and discovery. The constraint analyzer module produces approved (includes optimal, near optimal and ranked) sets based on business and process constraints. The execution engine binds the optimal set of services to the abstract process and executes them. We discuss all the modules of METEOR-S backend in the following sections.

4. Abstract Process Designer

This stage involves creating a representation of Web processes. We have chosen BPEL4WS as it is the de facto industry standard and provides a rich set of constructs for modelling workflow patterns [Wohed, 2002]. Design of abstract processes involves the following tasks.

1. Creating the flow of the process using the control flow constructs provided by BPEL4WS
2. Representing the requirements of each service in the process by specifying service templates, which allow the process designer to either specify semantic description of the Web service or a binding to a known Web service. Semantic descriptions allow process designers to automatically discover and bind Web services to the process.
3. Specifying process constraints for optimization.

Let us examine the creation of the abstract process with the help of an example. Consider the process of a distributor for processing customer orders. It starts by receiving the order from a customer. Then the order is processed and potential suppliers are selected. This process also includes a step, where potential suppliers may be contacted for quotes. After getting the quotes, the best candidates are chosen on the basis of process and business constraints and the orders are sent to them. This process can be designed by first deciding the flow of the different activities involved. This can be done by creating the process flow in BPEL4WS. The abstract process flow is shown in Figure 5.

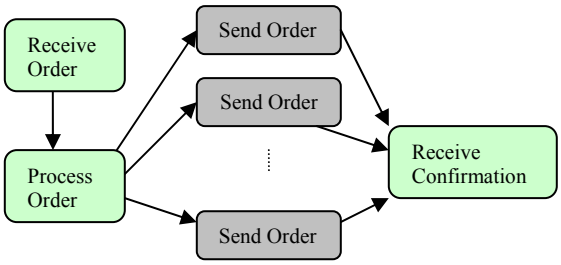

Figure 5: Abstract Distributor Process

The designer can then decide which services to bind statically and which to bind dynamically. The internal distributor services for processing the order and selecting the suppliers do not change so they may be statically bound to the process. However, the suppliers to be contacted depend on the order, so the supplier services should be able to be dynamically bound to the process. This can be done by specifying service templates for the suppliers. We have defined specifications for augmenting BPEL4WS with service templates for each invocation of a Web service in [Sivashanmugan et al., 2004]. A service template is created by using functional semantics as well as QoS specifications of all the operations of the Web service in the process. Service template (ST) is defined as <SL(ST), OP(ST,o_1), …, OP(ST,o_m)>, where

SL(ST)	Service Level Parameters
OP(ST,o_i)	Operation Semantics

SL(ST) = <B(ST), L(ST), D(ST)>, where

B(ST)	Business Name of the service provider
L(ST)	Geographic Location of the service
D(ST)	Domain of the service

The Location of the service is specified using the ISO 3166 Geographic Code System and the domain is specified using the NAICS taxonomy [NAICS, 2002]. Also, our system is compatible with any other standard that can be used to specify the location and domain.

The user can specify any of these attributes for discovery. Here is an example of a service template for the service that supplies batteries in Georgia, which provides operation for ordering batteries.

Service Template (ST)

Feature	Weight	Constraint
L(ST)	1	Georgia
D(ST)	1	Battery Supplier
F_c(ST,o)	1	#OrderBattery
I(ST,o)	.8	#PurchaseOrderRequest
O(ST,o)	1	#PurchaseOrderConfirmation
R(ST,o)	1	> 0.9
A(ST,o)	.7	>0.8

The last phase of abstract process creation involves setting the process constraints. Further details are provided in section 6.2.3.

Process Optimization

Feature	Goal	Value	Unit	Aggregation
Cost	Optimize		Dollars	Σ
supplytime	Satisfy	< 7	Days	MIN
partnerStatus	Optimize			MIN

Table 1

5. Discovery Engine

UDDI is the current standard for Web service discovery and publication. Semantic search in UDDI was first proposed by [Paolucci et al., 2002]. We have implemented our own algorithms to support semantic querying for services annotated using METEOR-S specifications. Service providers can annotate their services to create service advertisements and publish them using MWSDI [Verma et al., 2004a]. Given a service template, the discovery engine will return a set of service advertisements which match the template. In this section, we briefly describe service advertisements and the algorithm for discovery.

WSDL is the industry standard for describing Web services. It is, however, primarily syntactic in nature, and does not explicate the semantics of the service provider. DAML-S [Ankolekar et al., 2002] (now replaced by OWL-S [Ankolenkar et al., 2003]), presented semantic representation of Web services using an ontology based mark-up language. In an effort to be closely aligned to industry standards, we proposed semantic annotation of WSDL [Sivashanmugan et al., 2003; Rajasekaran et al, 2004]. The service advertisements which include the specifications developed in Section 2 are annotated WSDL files and published in our enhanced UDDI registry.

$SA = <SLP(SA), OP(SA,o_1), \ldots, OP(SA,o_n)>$

More details about semantic annotation of WSDL can be found in [Patil et al., 2004, Rajasekaran et al., 2004, Sivashanmugam et al., 2003]. To find the most optimal service set, we have a three phase selection process. The first phase is automated service discovery, followed by constraint analysis and then optimization based on user constraints as shown in Figure 6.

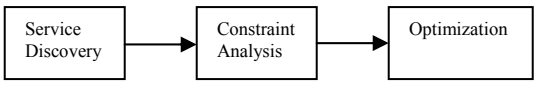

Figure 6: Three Phases of Selection Process

A detailed description of our discovery algorithm has been provided in our ensuing work in [Verma et al, 2004c]. The remaining two phases are discussed in the following section.

6. Constraint Analyzer

The constraint analyzer dynamically selects services from candidate services, which are returned by the discovery engine. Dynamic selection of services raises the issues of correctness and optimality. Our approach is to represent all criteria that affect the selection of the services as constraints or objectives. This converts the problem to a constraint satisfaction/optimization problem. Any candidate set of services for the process which satisfies the constraints is a feasible set. In order to achieve this, we use the three modules as shown in Figure 2. The layers are the constraint representation module, the cost estimation module and the optimization module. We discuss these modules in detail in the next sub sections.

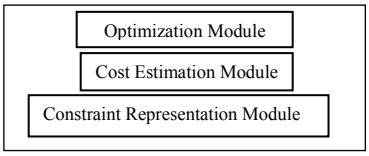

Figure 7: Architecture of Constraint Analyzer Module

6.1. Constraint Representation Module

The constraint representation module allows us to represent the business constraints in ontologies and databases. A business constraint is defined as any constraint that affects the selection of a Web service for a process. For example, some suppliers may be preferred suppliers for one part, but secondary suppliers for another part. There may exist a number of such business constraints for a particular process. Depending on the particular instance of the process, some constraints may be more important than others. For example, a secondary supplier may be chosen over a preferred supplier if it is cheaper. For illustration purposes, let us consider an example of representing business constraints. We have developed an electronics part ontology representing relationships between electronic items such as network adapters, power cords and batteries. The ontology is used to capture the suppliers for each part, their relationships with the manufacturer and the technology constraints in their parts. Let us express the following facts in the domain ontology (Figure 2) derived from the Rosetta Net ontology (Figure 1).

Fact	OWL expression
Supplier1 is an instance of network adaptor supplier Supplier1 supplies #Type1 Supplier1 is a preferred supplier.	<NetworkAdaptorSupplier rdf:ID="Supplier1"> <supplies rdf:resource="#Type1"/> <supplierStatus>preferred</supplierStatus> </NetworkAdaptorSupplier>

Type1 is an instance of NetworkAdaptor Type1 works with Type1Battery	<NetworkAdaptor rdf:ID="Type1"> <worksWith> <Battery rdf:ID="Type1Battery"> </worksWith></ NetworkAdaptor >

With the help of such statements the required business and technological constraints, which will be critical in deciding the suppliers, can be encoded in the ontology. In future, we will use SWRL [Horrocks et al., 2003] will provide more descriptive rules for specifying constraints.

6.2. Cost Estimation Module

The cost estimation module queries the information stored in the cost representation module for estimating costs for various factors which affect the selection of the services for the processes. The factors which affect service selection are

- Process constraints
- Querying and cost estimation
- Service Dependencies

6.2.1. Service Dependencies

It is possible for the selection of one service to depend on another [Verma et al., 2004b]. These dependencies may be based on a number of criteria like business constraints, technological constraints or partnerships. One type of service captures the notion that the selection of one service will affect choices of other services. See [Verma et al., 2004c] for details

6.2.2. Querying and Cost Estimation

Let us consider the supply chain for the manufacturer we mentioned in the introduction. Here are some of the factors which may affect the selection of the suppliers for a particular process.

- Cost for procurement
- Delivery time
- Compatibility with other suppliers
- Relationship with the supplier
- Reliability of the supplier's service
- Response time of the supplier's service

Depending on the manufacturer's preferences at process execution, all the factors can be more or less important. For example, at a certain point of time a manufacturer may only want to deal with preferred suppliers, while at other times he may choose the lowest cost alternative. In order to be able to set priorities between these factors, the cost estimation module queries the constraint representation and assigns costs on the basis of the results of the queries. The values for the parameters can either be actual or estimated.

The process designer does not need to specify multiple logical queries (e.g., find preferred suppliers or find non-preferred suppliers) to get the results. Using the cost estimation module to quantify the criteria gives us more power than just using conjunction and disjunction which would require query rewriting for any changes in the constraints.

For some of the factors like cost and supply time, actual values may be used as constraints. The cost, supply time and other such factors can be obtained by either getting quotes from the suppliers Web services or by querying internal databases. For other factors, the constraint representation module can be queried to estimate values. The cost estimation module estimates a cost on the basis of the query results. We have assigned costs to various criteria to illustrate this concept. The ontology is queried to get the relationship status of the supplier using the following query in DQL [Fikes et al., 2002].

supplierName supplierStatus ?status

If we query for Type1, based on the ontology, the "preferred" will be returned. The cost function for getting the value of for this parameter is:
preference(Si, Sj) = [0,1], where preferred partners are close to 1, secondary are about .5 and other partners are near 0.

6.2.3. Process Constraints

We refer to any constraints that apply to only that particular process as process constraints. The constraints are set on either the actual values or the estimated values. We model process constraints as constraints on Quality of Service specifications which were discussed in section 2.3. The process level QoS is calculated as the aggregation of QoS of all the services in the process. Algorithms for aggregation QoS of services in a process are discussed in [Cardoso and Sheth, 2003; Cardoso et al., 2004; Zeng et al., 2003]. In this implementation, the user has to specify the aggregation operators for QoS parameters.

$QoS(p) = \langle T(p), C(p), R(p), A(p), DS_1(p), DS_2(p), \ldots\ldots DS_N(p) \rangle$

T(p)	Execution time of the entire Web process
C(p)	Cost of invoking all the services in the process
R(p)	Cumulative reliability of all services in process
A(p)	Cumulative availability of all services in process
$DS_i(p)$	Cumulative scores for Domain specific QoS parameters.

Each metric specification consists of a 5- tuple set.
QoS_i (p) = {name, comparisonOp, val, unit, aggregationOp}, where 'name' is the name of the QoS parameter, 'val' is a numerical value, 'comparisonOp' is a

comparison operator, 'unit' is the unit of measurement and 'aggregationOp' is aggregation operator. For most metrics, the process QoS can be calculated using the aggregation operators' summation, multiplication, maximum or minimum. However, in some cases, the user has to define user defined function for aggregation.

In order to illustrate this, let us consider the following scenario for the manufacturer, where the manufacturer has to procure the parts and quantities shown in Table 2.

No.	Part	Quantity
1	Batteries	1000
2	Power Chord	900
3	Network Adaptor	1200

Table2

6.3. Constraint Optimizer

This is the final stage before execution. The cost estimation module quantifies the process QoS parameters for all candidate services in the process. The process constraints are directly converted to constraints for an Integer Linear Programming Solver called LINDO [LINDO]. For illustration, let us consider the equations generated for a sample process. S_i is the i^{th} service in the registry. O_{ij} is the j^{th} operation in S_i. ST_k is the k^{th} service template and OT_{kl} is the l^{th} operation in ST_k. For service template k, the constraints are:

$$x_{kli} = \begin{cases} 1 \text{ if operation } O_{ij} \text{ in service } S_i \text{ matches} \\ \text{operation } OT_{kl} \text{ in Service template } ST_k \\ 0 \text{ otherwise} \end{cases}$$

$$\min \left(w_1 * \sum_{kli} c_{ij} x_{kli} + w_2 * \sum_{kli} p_{ij} x_{kli} \right)$$

subject to:

$$\sum_{kli} x_{kli} = 3 \quad \text{and} \quad \sum_{i} x_{kli} = 1$$

Where c_{ij} and p_{ij} are the cost and partner preference respectively of operation j in service i. w_1 and w_2 are used to weight the cost and preference according to the choice of the process designer.

The objective function for optimization, which is a linear combination of the parameters, is extracted from the Service Template defined by the user.

These constraints, when fed into the LINDO Integer Linear Programming solver, will produce a number of feasible sets which would be ranked from optimal to near optimal solutions. The ranking is done on the basis of the value of the objective function. The value of each individual constraint like time, cost, and partner preference is also provided for feasible sets. The process designer is given the option of selecting the feasible set to be sent to the run-time module.

7. Run Time Module

This stage deals with converting the abstract process and the service templates to an executable Web process once an optimal set has been chosen and finally its execution. The BPWS4J [Curbera et al., 2004] engine provides a runtime environment to execute Web processes represented in BPEL4WS. Users can specify the process flow and data dependencies between the Web services using BPEL4WS and deploy the process on the engine. Currently, the engine only supports design time or deployment time binding as it requires the users to specify the location of the WSDL files of all services in the process during deployment. We have added a layer over the engine, that allows us to achieve dynamic binding. Details of this module have been discussed in [Verma et al., 2004b]. Dynamic binding involves sending the semantic templates to the discovery engine and discovering and binding the services to the process at runtime.

7.1. Binder

This module uses the abstract process and the optimal set of services (which match the service templates and satisfy the given constraints) to construct an executable BPEL file. We are using the BPWS4J API to parse the abstract BPEL file and make changes to it. The abstract BPEL file contains placeholders for the actual service details to be filled in. Assuming the user gives the following abstract BPEL and service template:

Abstract BPEL	Service template
<invoke name="invoke" partner="supplier" portType="?" operation="?" inputVariable="?" outputVariable="?"/>	Operation = #OrderBattery Input:#PurchaseOrderRequest Output:#PurchaseOrder Confirmation

A service advertisement will be returned by the system along with the location of the WSDL corresponding to the service. The WSDL file will be used to extract portType, namespace, etc. and would be inserted at appropriate locations in the BPEL. Hence using the service advertisement and the WSDL location we can construct the Executable BPEL:

```
<invoke name="invoke"
partner="supplier" portType="sup:BatterySupplier"
operation="SendOrder" inputVariable="purchaseOrderRequest"
outputVariable="purchaseOrderConfirnation"/>
```

We are using WSDL4J API [WSDL4J, 2003] to extract Web service details like portType, namespace, etc. which are then inserted into the BPEL file. The final BPEL file is then displayed to the user through the GUI

so that any assignments which were not handled by the BPWS4J parser can be corrected by the user. The final executable BPEL file is then sent to the BPWS4J execution engine to be executed.

8. Related Work

Semantics has been proposed as key to increasing automation in applying Web services and managing Web processes that take care of interactions between Web services to support business processes within and across enterprises [Ankolenkar et al., 2003; Kifer and Martin, 2002; Bussler et al., 2002]. Academic approaches like WSMF, OWL-S and METEOR-S have tried to approach this solution by using ontologies to describe Web services. This approach is consistent with the ideas of the Semantic Web, which tries to add greater meaning to all entities on the Web using ontologies.

Automated discovery of Web services requires accurate descriptions of the functionality of Web services, as well as an approach for finding Web services based on the functionality they provide. [Wroe et al., 2003] has discussed classification of services based on their functionality. Another approach tries to define the functionality of a Web service as the transformation of inputs to outputs [Paolucci et al., 2002]. Creating process ontologies was discussed in [Klein and Bernstein, 2001]. Our discovery algorithm considers functional and data semantics as well as QoS specifications.

Highly intertwined with semantics (and considered in this proposal as part of semantic specification) is the issue of Quality of Service (QoS), pursued from academic setting in [Cardoso and Sheth, 2003; Cardoso et al., 2004; Zeng et al., 2003], and in industry setting under the Web Service Policy framework [Box et al., 2003].

Use of automation in composing or orchestrating Web processes is predicated on having sufficient machine processable information about the process requirements as well as the available Web services. Thus, Web services need semantic annotation and process requirements need to be specified at a high level. These requirements may be specified as goals [Bussler et al., 2002], application logic (e.g. using extended Golog [McIlraith and Son, 2002]) or hierarchal planning constructs [Wu et al., 2003]. None of the above approaches for automated composition have considered a comprehensive framework for composition that would optimize selection of Web services on the basis domain specific QoS in presence of service dependencies.

We believe that the ability to choose services dynamically is crucial to the success of the service oriented architecture. OWL-S is a markup language anchored in an OWL ontology for automatic composition of Web services. It has not yet developed formalisms for optimization on the basis of QoS. An effort that comes closest to our research is Self-Serv [Benatallah et al., 2003], which provides an environment for creation of processes. They have, however, not considered issues like handling dependencies between Web services in a process. Another relevant work [Zeng et al., 2003] proposed a linear programming approach to optimize service selection across the process using generic QoS parameters. While they focus solely on optimization on generic QoS issues, we provide a comprehensive framework, which optimizes service selection based on multi dimensional criteria such as domain constraints, inter-service dependencies and QoS.

9. Conclusion and Future Work

In this paper, we have presented an approach to for achieving dynamic Web service composition. This work builds on the METEOR-S Web Service Composition Framework by adding the constraint analyzer module, which uses an Integer Linear Programming Solver for process optimization based on process and business constraints. We have extended the workflow QoS model in [Cardoso et al., 2004] to allow for global optimization and dynamic composition of Web processes. We believe that our cost estimation module adds great flexibility to our system by allowing us to quantify selection criteria. We believe this is the first paper to comprehensively address the issue of dynamic business process integration using business and process constraints.

In the future, we plan to achieve runtime binding of Web services. We also intend to develop a model for incorporating inter-service dependencies.

REFERENCES

[Andrews et al., 2003] Andrews et al., Business Process Execution Language for Web Services Version 1.1, available at http://www-106.ibm.com/developerworks/webservices /library/ws-bpel/ (2003).

[Ankolekar et al., 2002] A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara, "DAML-S: Web service Description for the Semantic Web," in Proceedings of the 1st International Semantic Web Conference (2002).

[Ankolenkar, 2003] 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. The DAML Services Coalition, "DAML-S: Web Service Description for the Semantic Web", The First International Semantic Web Conference (ISWC), Sardinia (Italy), (2002).

[Azami, 2004] M. Azami, RosettaNet Ontology, http://lsdis.cs.uga.edu/~azami/pips.html , 2004

[Benatallah et al., 2003] Boualem Benatallah, Quan Z. Sheng, Marlon Dumas: The Self-Serv Environment for Web Services Composition. IEEE Internet Computing 7(1): 40-48 (2003).

[Berners Lee et al., 2001] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web, Scientific American, 284(5):34--43, May 2001.

[Box et al., 2003] Box et al., Web Services Policy Framework (WSPolicy), availab)le at http://www-106.ibm.com/developerworks/library/ws-polfram, (2003).

[Bussler et al., 2002] Bussler, C., Fensel, D. and Maedche, A. A Conceptual Architecture for Semantic Web Enabled Web Services SIGMOD Record, Special Issue Semantic Web and Databases (2001).

[Cardoso and Sheth, 2003] Jorge Cardoso, Amit P. Sheth: Semantic E-Workflow Composition. Journal of Intelligent Information Systems 21(3): 191-225 (2003).

[Cardoso et al., 2004] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, Quality of Service for Workflows and Web Service Processes, Journal of Web Semantics (accepted) (2004).

[Curbera et al., 2002] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana: IEEE Internet Computing: Spotlight - Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. IEEE Distributed Systems Online 3(4): (2002)

[Curbera et al., 2004] Curbera et al., IBM Business Process Execution Language for Web Services Java™ Run Time available at http://www.alphaworks.ibm.com/tech/bpws4j

[Fikes et al., 2002] Fikes et al., DAML Query Langauge, available at http://www.daml.org/2002/08/dql, (2002).

[Horrocks et al., 2003] Horrocks et al., A Semantic Web Rule Language Combining OWL and RuleML, http://www.daml.org/2003/11/swrl/, 2003

[Kifer and Martin, 2002] Michael Kifer and David Martin, Bring Services to the Semantic Web and Semantics to the Web services, SWSC, (2002).

[Klein and Bernstein., 2001] M. Klein, and A. Bernstein. "Searching for Services on the Semantic Web using Process Ontologies", in The First Semantic Web Working Symposium (SWWS-1). 2001.

[LINDO] LINDO API version 2.0, Lindo Systems Inc. http://www.lindo.com/

[McGuinness et al., 2004] McGuinness et al., Web Ontology Language (OWL). Web-Ontology (WebOnt) Working Group http://www.w3.org/2001/sw/WebOnt/, 2002

[McIlraith and Son, 2002] McIlraith, S. and Son, T., Adapting Golog for Composition of Semantic Web Services, Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002), Toulouse, France, April, (2002).

[METEOR-S, 2002] METEOR-S:Semantic Web Services and Processes, http://swp.semanticweb.org, (2002).

[NAICS, 2002] North American Industry Classification System, US Census Beureau, 2002

[Paolucci et al., 2002] Paolucci, M. and Kawamura, T. and Payne, T.R. and Sycara, K. (2002) Importing the Semantic Web in UDDI. Proceedings of Web Services, E-Business and Semantic Web Workshop, CAiSE 2002., pages 225-236, (2002).

[Patil et al., 2004] A. Patil, S. Oundhakar, A. Sheth, K. Verma, METEOR-S Web service Annotation Framework, To appear in the proceedings of the 13th International World Wide Conference, (2004).

[Rajasekaran et al, 2004] Enhancing Web Services Description and Discovery to Facilitate Orchestration, Submitted to SWSWPC, 2004 (In conjunction with ICWS'2004)

[Sheth and Kashyap, 1992] A. Sheth, V. Kashyap: So Far (Schematically) yet So Near (Semantically). DS-5 1992: 283-312

[Sheth et al., 1999] A. Sheth, W.M.P. van der Aalst, and I.B. Arpinar, Processes Driving the Networked Economy: Process Portals, Process Vortexes, and Dynamically Trading Processes, IEEE Concurrency, 7 (3), pp. 18-31, (1999).

[Sheth, 2003] A. Sheth, "Semantic Web Process Lifecycle: Role of Semantics in Annotation, Discovery, Composition and Orchestration," Invited Talk, WWW 2003 Workshop on E-Services and the Semantic Web, Budapest, Hungary, May 20, (2003).

[Sheth, 1996] A. Sheth, K. Kochut, J. Miller, D. Worah, S. Das, C. Lin, D. Palaniswami, J. Lynch, I. Shevchenko: Supporting State-Wide Immunisation Tracking Using Multi-Paradigm Workflow Technology. VLDB 1996: 263-273

[Sivashanmugam et al., 2003]K. Sivashanmugam, K. Verma, A. Sheth, J. Miller: Adding Semantics to Web Services Standards, Proceedings of 1st International Conference of Web Services, 395-401, (2003).

[Sivashanmugam et al., 2004]K. Sivashanmugam, J. Miller, A. Sheth, and K. Verma, Framework for Semantic Web Process Composition, International Journal of Electronic Commerce (to appear), (2004).

[Verma et al., 2004a] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar and J. Miller, METEOR–S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services, Journal of Information Technology and Management (to appear), (2004).

[Verma et al., 2004b] K. Verma, R. Akkiraju, R. Goodwin, P. Doshi, J. Lee, On Accommodating Inter Service Dependencies in Web Process Flow Composition, AAAI Spring Symposium PP: 37-43 on Semantic Web Services.

[Verma et al., 2004c] METEOR-S – An Environment for creating Semantic Web Processes, K. Verma, R. Aggarwal, J. Miller, A. Sheth, VLDB Journal, 2004

[Wohed, 2002] P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Pattern-Based Analysis of BPEL4WS, QUT Technical report, FIT-TR-2002-04, Queensland University of Technology, Brisbane, 2002, available at http://tmitwww.tm.tue.nl/staff/wvdaalst/Publications/p175.pdf, (2002).

[Wroe et al., 2003] C. Wroe, R. Stevens, C. Goble, A. Roberts, M. Greenwood, A suite of DAML+OIL Ontologies to Describe Bioinformatics Web Services and Data. in International Journal of Cooperative Information Systems special issue on Bioinformatics, March 2003 .ISSN:0218-8430, (2003).

[WSDL4J,2003] WSDL4J Project,http://www-124.ibm.com/developerworks/projects/wsdl4j/

[Wu et al., 2003] D. Wu, B. Parsia, E. Sirin, J. Hendler, and D. Nau. Automating DAML-S web services composition using SHOP2. In Proceedings of 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, (2003).

[Zeng et al., 2003] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, Q. Sheng: Quality driven web services composition. WWW 2003: 411-421, (2003).