Kno.e.sis Publications

The Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis)

2006

Semantics Enabled Dynamic Process Configuration

Kunal Verma

Karthik Gomadam
Wright State University - Main Campus

Jonathan Lathem

Amit P. Sheth
Wright State University - Main Campus, amit@sc.edu

John A. Miller
Wright State University - Main Campus

Follow this and additional works at: https://corescholar.libraries.wright.edu/knoesis

Part of the Computer Sciences Commons

Semantics enabled Dynamic Process Configuration

Kunal Verma, Karthik Gomadam, Jon Lathem, Amit P. Sheth and John A. Miller
Large Scale Distributed Information Systems lab,
Department of Computer Science,
University of Georgia,
Athens, GA.
{verma, karthik, lathem, amit, jam}@cs.uga.edu

Abstract

Web processes are the next generation workflows created using Web services. This paper addresses research issues in creating a framework for configuring and executing dynamic Web processes. Our approach is that of a multi-paradigm constraint analysis for process configuration using quantitative and logical constraints. We also present a software architecture and an engineering approach for extending current Web service infrastructure to support dynamic Web processes. An execution environment, extending Apache Axis, one of the most popular SOAP implementations, to support dynamic process configuration is presented. Empirical evaluation of the system is performed to demonstrate the cost benefits of dynamic process configuration over static processes. We also demonstrate the time overheads caused by the addition of semantic processing capabilities to Axis.

1. Introduction

The advent of Web services and service oriented architectures (SOA) [1], based on a loosely coupled distributed computing model, has the promise to create next generation Web processes with more agility and dynamism. One of the original goals of SOA based solutions was dynamic binding of Web services to existing business processes. However, in spite of the large scale acceptance and deployment of Web services, they have been relegated to internal integration projects, and the grand vision of agile and virtual enterprises where partners can be integrated on the fly is yet to be realized. There are two key reasons which have lead to curtailing of the earlier expansive visions – 1) business models, until very recently, have not needed such dynamism, and the enterprises needed to become comfortable with static models first and 2) the current standards of Web services are not very suitable for (semi-) automated discovery and integration as they lack adequate semantics for those tasks. Recent business use cases such as ISEC [6] have shown that businesses are trying to build infrastructures, which will allow them to integrate the most optimal partners with their business processes. Hence, the first factor is being overcome by some enterprise. This paper addresses the second factor, which involves creating a framework for (semi-) automated configuration of Web processes by addressing two technical issues – 1) dynamically selecting optimal partner Web services for a process based on user defined constraints and 2) providing a architecture for dynamically binding the optimal partners at runtime by using the extensibility features of Axis 2.0. We present this paper as a sequel to our earlier paper [11], which introduced the notion of adding semantics to WSDL. That work matured into WSDL-S, an acknowledged W3C member submission. In this paper we use extensibility features of Apache Axis 2.0 to realize dynamic web process composition. This work was done as part of the METEOR-S [11] project, which deals with the complete lifecycle of semantic and dynamic Web processes.

The semantics of autonomously created Web services are explicated by annotating them with ontologies, which represent shared conceptualization of domains. Other well regarded contemporary approaches which deal with Semantic Web Services (SWS) include OWL-S [15], WSMO [24] and SWSF [22]. This paper focuses on using WSDL-S with pre-defined BPEL processes to allow dynamic configuration and execution. The distinguishing factor of this work from previous work on SWS includes comprehensive support of quantitative and logical constraints for runtime configuration of Web processes.

The two key components of our framework are the configuration module and architecture to support runtime configuration. The configuration module uses SWS discovery and constraint analysis based on quantitative and logical constraints to configure Web processes. A key research issue was handling both types of constraints. Correspondingly, a solution using an Integer Linear Programming (ILP) solver for handling quantitative constraints and a Semantic Web Rule Language (SWRL) [21] reasoner for logical constraints has been proposed and implemented. From an architectural point of view, a key requirement was to show how the configuration module could be integrated with popular Web service
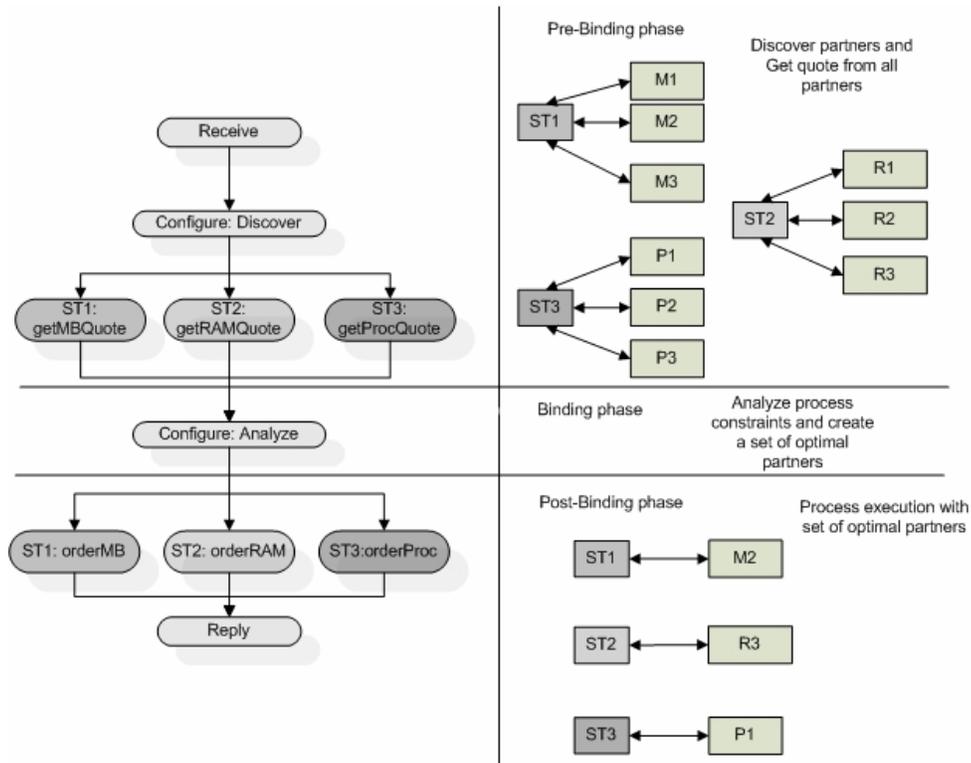
Figure 1: Sample Supply Chain Process

infrastructure. We leveraged the extensibility features of Axis 2.0 to achieve this. The METEOR-S middleware presented in this paper interacts with BPEL engines for enabling runtime configuration and dynamic execution of Web processes.

We will illustrate dynamic process configuration with the help of a dynamic supply chain of a computer manufacturer. In particular, we consider the part procurement component of their supply chain, where the logistics department generates a set of process constraints, which must be satisfied while configuring the process. The constraints include the budget, time, business relationships, parts compatibility, etc. In addition, the process must be optimized on the basis of an objective function.

This paper primarily deals with the conceptual and technological issues of achieving dynamic process configuration and has the following research contributions:

- At a conceptual level, dynamic process configuration using a multi-paradigm constraint analysis approach that considers a broader set of constraints than in previous works in adaptive workflows [18][12] .
- At a technological level, a novel architecture that is based using extensibility elements of popular Web service tool – Axis 2.0 is presented and evaluated demonstrating cost based benefits based on real time fluctuations in the currency market.

The rest of the paper is organized as follows. Section 2 presents the motivating scenario. Section 3 discusses our approach for capturing the semantic descriptions of the services. Sections 4 and 5 present the configuration module and the architecture. The empirical evaluation is presented in section 9 and related work is discussed in section 7. Finally the conclusions and future work are outlined in section 8.

2. Motivating Scenario

In this section, we will outline a motivating scenario for dynamic Web processes. Consider a computer manufacturer, who runs a highly flexible and adaptive supply chain. The manufacturer orders different parts (memory, motherboard and processor) from suppliers based on process constraints generated by the logistics department. The process constrains can include both quantitative constraints (like minimize cost) and non-quantitative constraints (like the parts must be compatible with each other). It has a number of suppliers (both domestic and overseas) that it deals with, and it must choose an optimal set of suppliers based on the process constraints. In the scenario described above, the manufacturer would like to obtain quotes from all the suppliers who supply various components. He would then choose the optimal set of partners who satisfy both the quantitative and logical process constraints. The Web process is illustrated in Figure 1.

The current approaches to process composition have predominantly looked at limited support to dynamic binding of partners to a process. In the scenario just described above, the set of optimal partners can change due to various factors. For example, changes in current rates can affect the cost optimality of the current set of partners. This creates a need for more agile process management frameworks that allows for dynamic configuration and reconfiguration of Web processes. Such a process management framework must be able to do the following:

- Handle quantitative process constraints: Some examples are: 1) Minimize the total cost and 2) the parts must be delivered with 7 days.
- Handle qualitative process constraints: Some examples are: 1) supplier for part 1 must be a preferred supplier and 2) the parts must be compatible with each other. The rules for part compatibility and supplier status are represented using domain ontologies and rules.
- Optimally configure the process based on an objective function: For example: cost must be minimized. Configuration refers to finding an optimal set of partners for the process, who satisfy all the constraints.

3. Semantically Representing Web Services

In this section, we will provide a definition of SWS, which is required for various tasks such as discovery, constraint analysis and mediation. This definition is based on the definition of WSDL and proposed SWS specifications – OWL-S and WSDL-S.

$$SWS = < SLM, \underset{i}{Y}\{sopd_i\}, SLP>$$

Where, SWS is defined as a 3-tuple of: a service level metadata tuple (SLM), collection of semantically described operations and a collection of service level policy assertions (SLP). Policy assertions are formally defined in [23] and are used to represent the non-functional properties of Web services.

SLM = <BusinessName, IndustryDomain, ProductCode, Location>

Where, SLM is a 4-tuple of the following: name of the business (BusinessName), industry domain using the NAICS taxonomy (IndustryDomain), product code using DUNS code (Product Code) and the location (Location).

An example of SLM is given in

Table 1.

Table 1: Example of Service Level Metadata

BusinessName	ABC
IndustryDomain	NAICS:443112(Electronics)
ProductCode	DUNS:32101601 (Random access memory RAM)
Location	Athens, GA

$sopd = <Name:Action_O, I:I_O, O:O_O, Pre:Pre_O, Post:Post_O, Ex:Ex_O, OLP>$

Where, a semantically described operation (sopd) is defined as a 7- tuple of the following: operation name mapped to an action concept in a domain ontology $(Name:Action_O)$, input and output messages mapped to concepts in domain ontology $(I:I_O)$ and $(O:O_O)$, collections of pre and post conditions represented using concepts in a domain ontology $(Pre:Pre_O)$ and $(Post:Post_O)$, a collection of exceptions mapped to a domain ontology $(Ex:Ex_O)$ and a collection of policy assertions (OLP).

Table 2: Example of Semantically Defined Operation

$Name:Action_M$	getOrder:Rosetta#requestPurchaseOrder
$I:I_O$	OrderDetails: Rosetta#PurchaseOrderRequest
$O:O_O$	OrderConfirmation:Rosetta#PurchaseOrderConfirmation
$Ex: Ex_O$	{LowInventoryException:SupplyChainOnt# LowInventoryException}
$Pre:Pre_O$	{AccoutExists:Rosetta#CustomerAccountExists}
$Post:Post_O$	{Confirmed:Rosetta#OrderConfirmed}
OLP	{<encryption,=,RSA,,Requirement>,<responseTime,≤,60,sec, Capability>}

An example of a semantically described operation in using an ontology created from RosettaNet PIPs [17] is shown in Table 2.

4. Configuration Module

In this section, we present the configuration module. The configuration module has two main parts – template driven WS discovery and multi-paradigm constraint analysis.

4.1. Template Driven WS Discovery

SWS Discovery is critical in identifying candidate Web services which provide the required functionality. The goals of discovery in METEOR-S are the following:

- Find services that match user's requirements.

- Provide enough information for automated invocation of the service.

The industry standard UDDI provides a keyword based or category based search for Web services. It is not adequate for either of the above requirements because there is no support for operation based discovery, as operations are units of functionality within the service. In order to overcome this problem, we added another layer over UDDI (using UDDI4J [7] over a jUDDI [8] registry), which allows discovery based on a collection of semantically defined operations, and returns the required information for automated invocation of each operation. The user's requirements are captured using a semantic template (ST) [19].

$$ST = < STLM, \underset{i}{U}\{sopt_i\}, SLPT>$$

Where, a semantic template (ST) is defined as a 3-tuple of the following: collection of semantic operation templates (sopt), a collection of service template level policies (STLP), and a collection of service template level metadata (STLM).

STLM = <BusinessName, IndustryDomain, ProductCode, Location>

Where, STLM is a 4-tuple of the following: name of the business (BusinessName), industry domain using the NAICS taxonomy (IndustryDomain), product code using DUNS code (Product Code) and the location (Location).

sopt = <$Action_O$, I_O, O_O, Pre_O, $Post_O$, Ex_O, OLPT>

Where, a semantic operation template (sopt) is an abstract representation of the functionality of an operation. It is similar to an sopd, except that it defined only using ontological concepts. It is defined as a 7-tuple of the following: an action concept in a domain ontology ($Action_O$), input and output messages mapped to concepts in domain ontology (I_O) and (O_O), collections of pre and post conditions represented using concepts in a domain ontology (Pre_O) and ($Post_O$), a collection of exceptions mapped to a domain ontology ($Ex:Ex_O$,) and a collection of policy assertions (OLP).

An example of semantic template is given in Figure 2.

Semantic Template
IndustryCategory = NAICS:Electronics
ProductCategory = DUNS:RAM
Location = Athens, GA
Operation1 = Rosetta#requestPurchaseOrder
   Input = Rosetta#PurchaseOrderDetails
   Output = Rosetta#PurchaseConfirmation
   Non-Functional Requirements
     Encryption = RSA
     ResponseTime < 5 sec
Operation = Rosetta#QueryOrderStatus
   Input = Rosetta# PurchaseOrderStatusQuery
   Output = Rosetta# PurchaseOrderStatusResponse

Figure 2: Example of Semantic Template

The discovery engine returns a ranked set of Web services for a given semantic template. The discovery algorithm used in the paper is a refinement of the algorithms proposed in [14][15]. STLM is used to find Web services, which deal with certain products in a certain area. Then "Action" is used to find out what functionality the service performs based on the product (buy, sell, trade, etc.). Finally, inputs and outputs are matched. At an abstract level discovery returns a ranked list of services, based on weighted combination of different matchers.

MS (sopt, sopd) = MS-A(sopt, sopd) * MS-SLM(sopt, sopd) * (MS-I(sopt, sopd) + MS-O(sopt, sopd))

Where,

MS (sopt, sopd)	Match Score between sopt and sopd
MS-A(sopt, sopd)	Match Score between action concepts of sopt and sopd
MS-SLM(sopt, sopd)	Match Score between service level metadata of sopt and sopd
MS-I(sopt, sopd)	Match Score between inputs of sopt and sopd
MS-O(sopt, sopd)	Match Score between outputs of sopt and sopd

MS-A and MS-SLM are multiplied to ensure that a service that has a match score of 0 for either, is not returned by discovery. The match score, which is computed using ontology based matching presented in [14][15] is a number between 0 and 1.
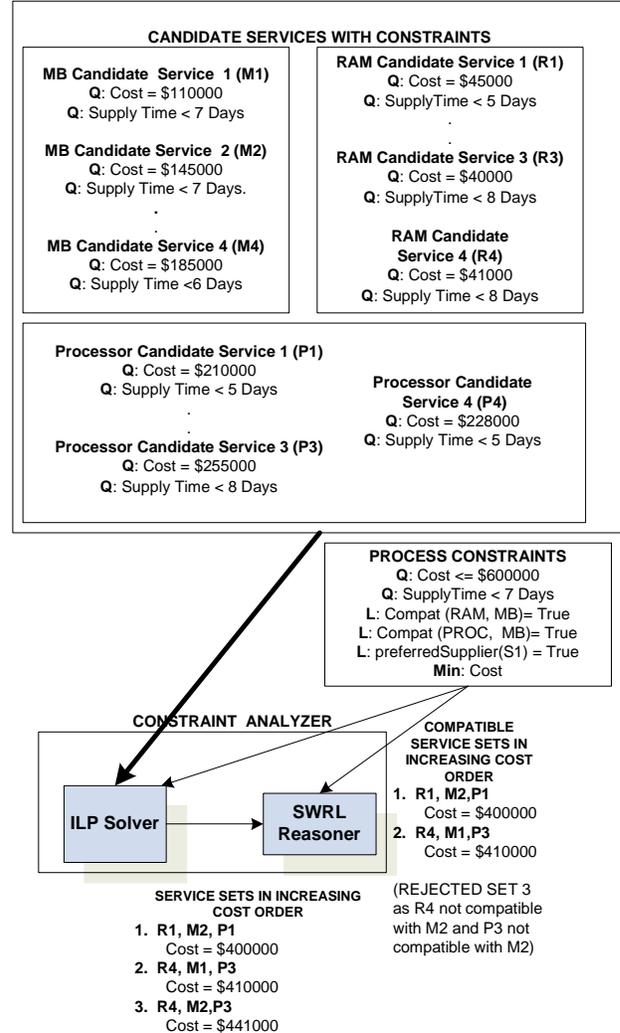
4.2. Multi-paradigm Constraint Analysis

Figure 3: Multi-paradigm Constraint Analysis Module

In order to capture both quantitative and logical constraints, we present an approach for handling both kinds of constraints. This is done using two modules in conjunction – 1) quantitative constraint analysis using ILP and 2) logical constraint analysis using SWRL.

The quantitative constraint analysis and optimization module uses an ILP solver to find an optimal set of services that satisfy the quantitative process constraints. We used the ILP solver of the LINDO API [9] for this

module. The configuration module converts the process constraints and the service constraints into ILP equations. Creating ILP equations for optimizing the process was presented in [1] 0. While our approach is similar to these approaches, we take a broader perspective of the criteria used for selection. Since our focus is on optimizing business process, we propose using domain specific quality of service (QoS) parameters like supply time, cost of the products for service selection, in addition to the generic QoS criteria like time, service invocation cost, reliability etc used by 0. Another significant difference in our approach is that we also allow users to enter constraints or modify them, as unlike the generic QoS criteria, not all domain specific constraints are dependent on the structure of the process. In this section, we will explain how to create the ILP equations for the process and service constraints given in Figure 3.

1. Create a binary variable xij for each candidate service.

$$xij = \{ \frac{1 \text{ if, candidate service } j \text{ is chosen for activity } i}{0, \text{ otherwise}}$$

2. Set the bounds on i and j, where i iterates over the number of activities (M) for which services are to be selected and j iterates over the number of candidate service for each activity - $N(i)$. In Figure 1, $M =3$, as the services have to selected for only three activities - "orderMB", "orderRAM" and "orderProcessor".

3. Set up constraints for the number of services chosen for each task and each service:
One service per task:

$$(\forall i)_{1 \le i \le M} \sum_{j=1}^{N(i)} xij = 1$$

4. Since cost is a generic constraint, the algorithms presented in [2] can be used to aggregate the cost for the complete process and the technique in [26] can be used to generate the ILP equation.

$$\sum_{i=1}^{M} \sum_{j=1}^{N(i)} cost(xij).xij \le 600000$$

5. SupplyTime of a service is independent of the structure of the process, hence this constraint cannot be generated automatically. In such cases, the aggregation type must be specified declaratively.

$$(\forall i)_{1 \le i \le M} (\forall j)_{1 \le j \le N(i)} SupplyTime(xij) \le 7$$

6. Create the objective function. In this case, cost should be minimized:

$$Minimize : \sum_{i=1}^{M} \sum_{j=1}^{N(i)} cost(xij).xij$$

Based on these equations, the ILP solver returns a set of ranked service sets with increasing costs, which are then passed to logical constraint analyzer. As in shown **Error! Reference source not found.**, there are three sets of services returned:

   1. R1, M2, P1 with cost $400000

2. R4, M1, P3 with cost $410000
3. R4, M2, P3 with cost $441000

The logical constraint module uses a SWRL reasoner to handle logical constraints. A detailed explanation of the SWRL reasoner created using SNOBASE is presented in [23]. We chose SWRL to represent such constraints, as it provides a mechanism to use Horn logic like rules, over facts represented in OWL ontologies. This module takes the ranked list provided by the ILP solver and returns the most optimal set that satisfies all the logical constraints. The constraints are expressed as SWRL rules on particular attributes of returned services. The rules are expressed as predicates over the services and any of their attributes, based on domain knowledge captured in OWL ontologies. A supply chain domain ontology capturing the parts, their suppliers and the technology constraints between the parts is shown in Figure 4.
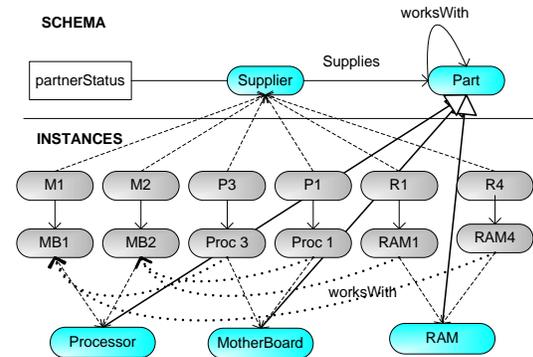

Figure 4: Domain Ontology capturing suppliers, parts and their relationships

There are two aspects of logical constraint analysis – Step 1) creating the rules based on the constraints at design time and Step 2) querying the SWRL reasoner to see if the constraint is satisfied at configuration time. Let us first examine creating the rules. These rules are created with the help of the ontology shown in Figure 4. Here are two sample rules that capture the requirements outlined in the motivating scenario:

1. Supplier 1 should be a preferred supplier. This is expressed in SWRL abstract syntax as: *Supplier (?S1) and partnerStatus (?S1, "preferred") => preferredSupplier (?S1)*

2. Supplier 1 and supplier 2 should be compatible. In plain english, this constraint is "if S1 and S2 are suppliers and they supply parts P1 and P2, respectively, and the parts work with each other, then suppliers S1 and S1 are compatible for parts P1 and P2. This can be expressed in SWRL abstract syntax as: *Supplier (?S1) and supplies (?S1, ?P1) and Supplier (?S2) and supplies (?S2, ?P2) and worksWith (?P1, ?P2) => compatible (?S1, ?S2)*

As illustrated in Figure 3, the ranked list of service sets from the ILP solver is then checked for all the constraints.

We will illustrate this analysis with the help of the first and third sets :

Set 1: R1, M2, P1 with cost $400000.

preferredSupplier(R1) = TRUE (from ontology)

   compatible(R1, M2) = TRUE (since R1 supplies RAM1 and M2 supplies MB2 and RAM1 works with MB2) and

   compatible(P1, M2) = TRUE (since P1 supplies and M2 suppliers MB2 and P1 works with MB2)

Set 3: R4, M2, P3 with cost $441000.

   preferredSupplier(R4) = TRUE (from ontology)

   compatible(R4, M2) = FALSE (since R4 supplies RAM4 and M2 supplies MB2 and RAM4 doesnt work with MB2)
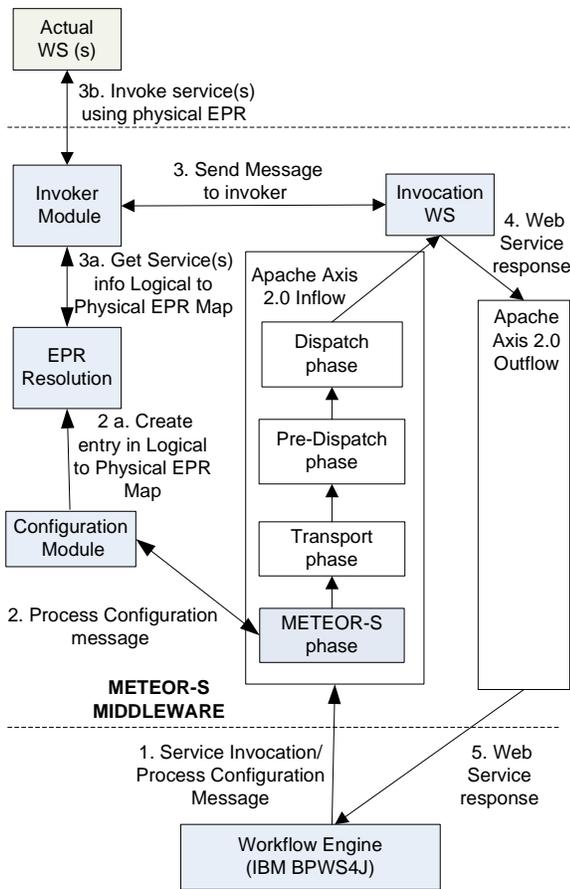
5. System Architecture

Figure 5: System Architecture with METEOR-S phases and modules shown in color

The key to our approach is creating BPEL processes which are deployed with virtual partners. This is achieved by using WSDL-S to create deployable semantic templates. The process communicates with the METEOR-S Middleware using logical endpoints. Logical endpoints are the endpoint references of the semantic templates of the partners of the Web process, which point to the

middleware. In this section we present the system architecture of the METEOR-S middleware which is based on the extensibility features of Axis 2.0. The extensible architecture provided by Axis 2.0, allows users to extend its capabilities by adding new phases or modules. Axis has some pre-defined phases for message handling, where each phase has a well defined role. Details of the Axis 2.0 architecture are available at [2]. Each phase has a number of modules. In order to extend the capabilities of Axis, users can either add phases or add modules to existing phases.

In order to support dynamic process configuration, we have added the METEOR-S phase before the transport-in phase (Figure 5). The METEOR-S phase consists of the Configuration Module and Execution module. The configuration module is responsible for process configuration. Process configuration includes a) Discovering partners satisfying the data and functional semantic requirements captured in the semantic template and b) Creating a set of partners after performing constraint analysis on the non-functional requirements captured in the semantic templates. The execution module is responsible for end point resolution and service invocation. The End Point Resolution module uses the logical end point sent by the process to return the end point of the actual partner chosen for the process. The invoker module is responsible for invoking the partner web services

4.1. Message Flow

We now present the message flow between the Web process and the middleware during configuration and execution.

1. **Service Discovery Request:** The Web process initiates process configuration by sending a Service Discovery request to METEOR-S middleware. The Service Discovery message consists of the semantic templates of the partner services of the process. This message is routed to the Configuration Module, which discovers the services that match the semantic templates.

2. **Pre-Binding Invocation Request:** Whenever a process sends a request before constraint analysis, the middleware invokes the specified operation of all the discovered services associated with the particular semantic template. Generally, such requests are used to obtain metrics to perform constraint analysis. For example,in Figure 1, the process invokes getQuote operation from each supplier service in order to obtain the quotes, which are used for constraint analysis.

3. **Constraint Analysis Request:** The process sends a Constraint Analysis request message to perform the constraint analysis on discovered services. The Constraint Analysis request message is composed of the process constraints, which will be used in optimization as well as

logical constraint analysis. The selected services are stored with the EPR Resolution component using a shared data structure.

4. **Post-binding invocation Request:** The process sends this request to the middleware, requesting to invoke the actual partner service. The message consists of logical endpoint corresponding to the partner's semantic template and the input to the actual service. The invocation component on receiving this message sends an EPR Resolution request to the EPR Resolution component, requesting for the actual end point and the operation to invoke. Upon invoking the partner Web service, the invocation component, returns the result to the invocation web service.

6. Evaluation

In this section we present an empirical evaluation of the METEOR-S Middleware framework for dynamic Web process configuration. The objective of this evaluation is two fold -1) To demonstrate the cost based benefit of using dynamic binding and multi-paradigm constraint analysis and 2) To illustrate additional time overheads caused by discovery and constraint analysis.

The system was implemented in Java 1.4. The BPEL process was deployed and executed using the IBM BPWS4J execution engine. jUDDI configured with MySQL 4.1 database server was used for Web service publication and discovery. We deployed four Web services each to model Mother Board, Memory and Processor suppliers. For process configuration, the constraints discussed in section 4 were used. The tests were run on Intel Pentium 4 PC's running Windows XP and Ubuntu Linux distribution. Apache Tomcat 5.5 was used as the servlet container.

We first present the results of the cost benefits of using dynamic binding. We used cost based analysis described in [20] to simulate the experiments. For our evaluation, changes in currency rates of various countries which had suppliers was one of the main factors affecting the supplier costs. In our experiment we assumed that the suppliers came from China and Taiwan. For our experiments we used currency data for China and Taiwan from x-rates.com. Our experiment demonstrates the cost based advantages of dynamic binding over static binding. We deployed two BPEL processes; one with partners discovered and bound during design time and another with partners captured using Semantic templates. In static binding, the optimal set of partners during design time was selected. In dynamic binding, partner selection and binding are done during the execution time. Figure 6 illustrates the process cost per unit order in these
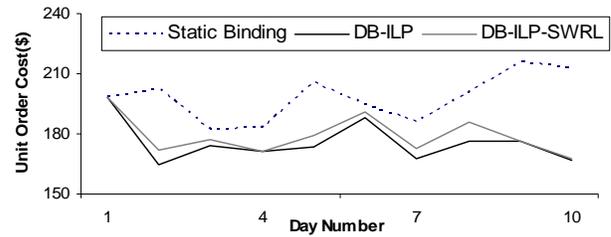
approaches.

Figure 6: Comparison of static and dynamic binding

While evaluating dynamic binding, we illustrate the process cost with (DB-ILP-SWRL)) and without logical constraint analysis (DB-SWRL). As illustrated in Figure 6, although the cost without logical constraint analysis offers the most cost based benefit, there is no guarantee of part compatibility between the various suppliers. Hence we choose the least cost offered by dynamic binding approach along with multi-paradigm constraint analysis. The cost variations in the static binding approach are due to the change in cost of the suppliers. Based on currency fluctuation over ten days, DL-ILP-SWRL had approximately 10% less average cost than static binding.
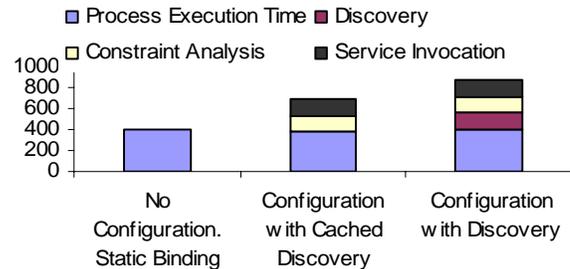

Figure 7: Timing Results

We now show the time overhead at middleware level caused by additional processing. Figure 7 illustrates the execution times between a) No configurations i.e. Static binding b) Configuration with cached discovery and c) configuration with discovery and constraint analysis. In case of static binding we run discovery and constraint analysis to choose the set of partners for the process. The partners are then tightly bound to the process during design time. In case of configuration with cached discovery, the results of discovery for a specific process are cached at the time of first discovery request. The middleware uses the cached set of services for subsequent invocations. In the last case, discovery and constraint analysis are run at every invocation. On an average the computational time for the static process was about 400 ms. The computational time was increased by about 280 ms when the discovery was cached. In the case of performing both discovery and constraint analysis we have an average overhead of about 480 ms. Figure 7 also shows module wise breakdown of the time spent.

7. Related Work

This paper is based on research in the areas of Semantic Web Services, Web process optimization and Semantic Web Service execution environments. In the area of semantic web services, OWL-S, SWSF, WSMO and WSDL-S are different approaches for representing semantic Web services. In this paper, we used a conceptual model based on WSDL-S. The focus of this paper was on configuration using quantitative and logical constraints, which has not been considered these approaches. Medjahed et al [10] present a composability model for web services based on various properties like interfaces of services and quality of service. However that paper does not discuss process optimization.

In the area of Web process optimization, the work presented in Zeng et al. 0 is the closest to this work. It proposed using linear programming to find an optimal set of services for a process. We extend that work by also considering logical constraints, which enable capturing domain specific constraints like preferred suppliers, compatibility constraints etc. In the area of Semantic Web Service Execution Environments, WSMX [5] is a framework for dynamic orchestration of Web services. WSMX differs from this in focus. Our focus is on optimal configuration as opposed to automated orchestration in WSMX.

8. Conclusions and Future Work

The general trend of businesses is to strive for greater automation and agility. Dynamic process configuration is an important step towards creating more agile business processes. In this paper, we have demonstrated how dynamic process configuration can be realized using popular SOA tool – Axis 2.0. At a conceptual level, we presented a multi-paradigm constraint analysis for Web process configuration using both quantitative and logical constraints. We also implemented our approach and evaluated its usefulness with respect to cost and execution time. While our evaluation showed a ten percent decrease in cost over static binding, the overhead introduced was in the order of hundreds of milliseconds. We believe that is not substantial considering that the actual physical processes (ordered items being delivered) run over several days.

Our aim was to compliment research in semantic Web services by presenting an approach for incorporating semantic Web services by extending current SOA infrastructure. We did so by presenting an execution environment that uses the extensibility features of Axis 2.0. Our future work includes adding a data mediation module [13] for addressing data heterogeneities. We invite other SWS researchers to add modules to this implementation. The services used in our evaluation and an online demonstration can be found at: http://lsdis.cs.uga.edu/~gomadam/index.php?page=10

References

[1] R. Aggarwal, K. Verma, J. Miller and W. Milnor, "Constraint Driven Web Service Composition in METEOR-S," SCC, 2004.

[2] Axis 2.0, http://ws.apache.org/axis2/

[3] J. Cardoso, A. P. Sheth, J. A. Miller, J. Arnold, K. Kochut: Quality of service for Workflows and Web Service Processes. Journal of Web Semantics 1(3): 281-308 (2004)

[4] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, S. Weerawarana: The next step in Web services. Communication of the ACM 46(10): 29-34 (2003)

[5] T. Haselwanter, M. Zaremba and M. Zaremba: Enabling Components Management and Dynamic Execution Semantic in WSMX, WIW, June, 2005.

[6] ISEC, Use Case, http://www- 306.ibm.com/software/ ebusiness/jstart/casestudies/niiip_isec.shtml

[7] Java API for UDDI, http://sourceforge.net/projects/uddi4j

[8] jUDDI, Java implementation of UDDI, http://ws.apache.org/juddi/

[9] LINDO API for Optimization, http://www.lindo.com/

[10] B. Medjahed, A. Bouguettaya, A. K. Elmagarmid: Composing Web services on the Semantic Web. VLDB J. 12(4): 333-351 (2003)

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

[12] R. Muller, U. Greiner, E. Rahm, AgentWork: A Workflow System Supporting Rule-Based Workflow Adaptation, Journal of Data and Knowledge Engg, 2004.

[13] M. Nagarajan, K. Verma, .A. Sheth, J. Miller, J. Lathem, Semantic Interoperability of Web Services – Challenges and Experiences, LSDIS Lab Technical Report, 2006.

[14] S. Oundhakar, K. Verma. K.Sivashanmugam, A. Sheth and J. Miller, Discovery of Web Services in a Federated Registry Environment, JWSR, 2 (3), 2005, pp. 1-32

[15] OWL-S, http://www.daml.org/services/owl-s/

[16] M. Paolucci, T. Kawamura, T. Payne and K. Sycara, Semantic Matching of Web Services Capabilities, Proc. of the 1st International Semantic Web Conference, 2002.

[17] Rosetta Net Ontology http://lsdis.cs.uga.edu/ projects/meteor-s/wsdl-/ontologies/rosetta.owl

[18] A.P. Sheth and K. Kochut, Scalable and Dynamic Work Coordination and Collaboration Systems, Workflow Management Systems and Interoperability, Springer, 1995.

[19] K. Sivashanmugam, K.Verma, A. Sheth, J. Miller, Adding Semantics to Web Services Standards, ICWS 2003

[20] J. Swaminathan, S. Tayur, Models for Supply Chains in E-Business, Management Science, 49(10), 2003.

[21] SWRL, http://www.daml.org/2003/11/swrl/

[22] SWSF, http://www.w3.org/Submission/SWSF/

[23] K. Verma, R. Akkiraju, R. Goodwin, Semantic Matching of Web Service Policies, SDWP 2005.

[24] Web Services Modeling Ontology, http://www.wsmo.org

[25] WSDL-S, http://www.w3.org/Submission/WSDL-S/

[26] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, Q. Sheng: Quality driven web services composition. WWW 2003.