# Demonstrating Dynamic Configuration and Execution of Web Processes

Karthik Gomadam, Kunal Verma, Amit P. Sheth, and John A. Miller

Large Scale Distributed Information Systems Lab,
Department of Computer Science,
University of Georgia
{karthik, verma, amit, jam}@cs.uga.edu

**Abstract.** Web processes are next generation workflows on the web, created using Web services. In this paper we demonstrate the METEOR-S Configuration and Execution Environment (MCEE) system. It will illustrate the capabilities of the system to a) Discover partners b) Optimize partner selection using constraint analysis, c) Perform interaction protocol and data mediation. A graphical execution monitor to monitor the various phases of execution will be used to demonstrate various aspects of the system.

## 1 Introduction

The service oriented architecture [6] envisions a dynamic environment where software components could be integrated on the fly based on their declarative descriptions. So far, most of the work in standards of Web services (WS) has been on syntactic standards based on XML, which limits the amount dynamism possible in the such systems. METEOR-S seeks to use semantics in all aspects a Web process lifecycle, especially to support dynamic execution features. Its approach consists of comprehensive modeling and use of semantics that are divided into four types: data (such as that required for input and output message contents), functional (concerning the domain specific capabilities), non-functional (including QoS) and execution (such as that needed for exceptional handling and correctness of execution). MCEE follows the METEOR-S philosophy of using semantics at various stages during the lifecycle of Web processes and is discussed in detail in [1]. This paper demonstrates a real world scenario which is presented in [5]. The rest of the paper is organized as follows. The MCEE architecture is presented in section 2. Section 3 describes the demonstration scenario. Section 4 outlines the unique features of the system. A summary is presented in section 5.

## 2 MCEE Architecture

In this section, we provide a brief overview of MCEE [1]. We will present the design overview and then implementation details.

B. Benatallah, F. Casati, and P. Traverso (Eds.): ICSOC 2005, LNCS 3826, pp. 502 – 507, 2005.

### 2.1 Design Overview

The architecture of our system is illustrated in Figure 1. The different components of the system are:

1. Process manager
2. Proxy
3. Configuration module
4. Execution environment

We discuss each component briefly in rest of this section.

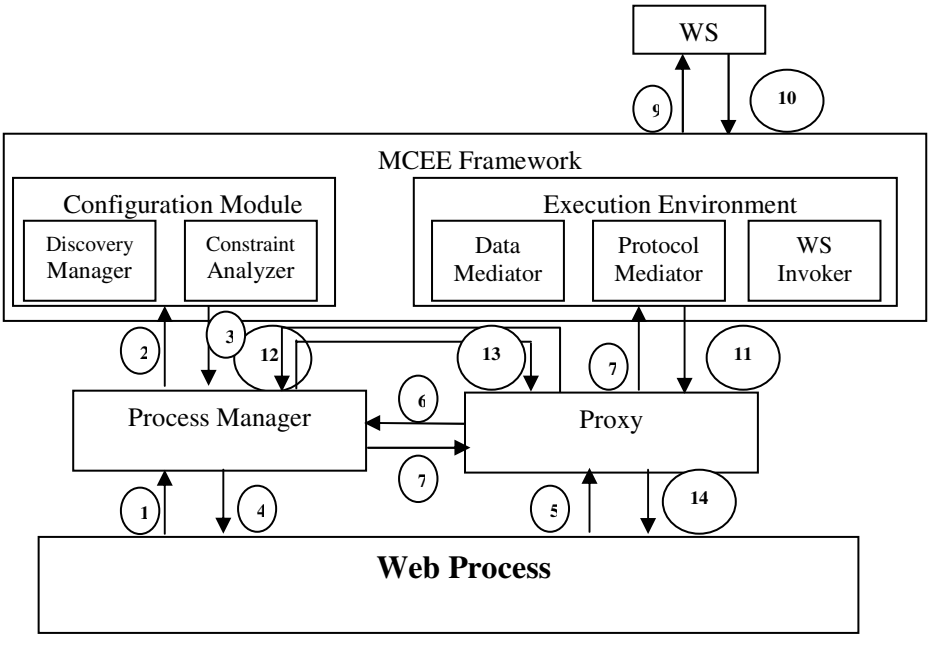


**Fig. 1.** Architectural overview of MCEE

The *configuration module* is responsible for process configuration. The configuration module can be called by the process manager during a) process configuration b) process reconfiguration. During both cases the configuration module performs Web service discovery and constraint analysis. Web service discovery is realized by the Discovery manager component in the configuration module. Service discovery is based on the data, functional and non-functional descriptions of the service requirements captured in the semantic template using the annotations with respect to the corresponding ontologies. The constraint analyzer component helps in creating a set of candidate Web services that satisfy the process constraints. Integer Linear programming is used for solving quantitative constraints and SWRL is used for non-quantitative constraints. A detailed discussion of our earlier work in Web service Quality of Service is presented in [3].

The *execution environment* handles the execution requests initiated by the proxy. The capabilities of the execution environment include a) Data Mediation b) Protocol mediation and c) Web service invocation. The execution environment replies to the proxy with the service output or service exception in the event of service failure. Data mediation is necessary to address issues due to data heterogeneities between the target Web service and the semantic template. WSDL-S allows for specifying data transformations using XSLT or XQuery [2]. The data mediator component is responsible for realizing these data transformations. Interaction protocol heterogeneities are handled by the interaction protocol mediator. The interaction protocol handler in framework is explained in detail in [1].

*Proxies* are Web services generated from the semantic templates for the partner. The proxies initiate the binding request, when they are invoked by the process. The process manager replies to the binding request by returning the service discovered for the template. The proxy then sends an execution request to the execution environment. If the service cannot be successfully executed, the proxy initiates the reconfiguration request. The execution request is illustrated in messages 8 and 9 in Figure 1.

The *process manager* is a Web service that handles three different requests a) *Configuration request,* b) *Binding request and* c) *Reconfiguration request. Configuration requests* are initiated by the Web process execution engine and are sent to the process manager. It forwards the configuration requests to the configuration module which configures the Web process. *Binding requests* are initiated by the proxy and are sent to the process manager, to get the binding information about the Web services discovered. *Reconfiguration requests* are initiated by the proxy and are sent to the process manager, to notify of service failure. The process manager then reconfigures the process, by halting the execution of other proxies and by forwarding the reconfiguration request to the configuration module. The reconfiguration algorithm is discussed in detail in [1]. Messages 1, 2, 3 and 4 in Figure 1 are configuration requests. Messages 6 and 7 in Figure 1 are binding requests. Messages 9 and 10 in Figure 1 are reconfiguration requests.

### 2.2 System Information

The system is implemented using JDK 1.4.2. Web services are written using Java and are deployed in Apache Axis 1.2RC. The discovery module is implemented using jUDDI and UDDI4J. The Web process is orchestrated using the IBM BPWS4J engine and is written in WS-BPEL. Tomcat version 4.1.29 is used for BPWS4J engine and Apache Axis. Tomcat 5.1.3 was used for jUDDI. The system uses the current WS technologies and infrastructure. This allows for interoperability between semantic Web services and Web services as they exist today.

## 3 Demonstration Scenario

We demonstrate a real world use case from the domain of agricultural marketing in India. The scenario is discussed in detail in [5]. The current marketing system has

farmers selling their produce to either a) an Agriculture Produce Market Committee (APMC) b) merchants associated with APMCs or c) brokers associated with APMCs. In rest of the discussion, a farmer is a seller and a buyer can either be an APMC, merchant or broker. In this scenario, we demonstrate the value and utility of dynamic Web processes. This also will demonstrate the capability of the system demonstrated to support and execute such processes. Figure 2 shows an abstract process created to realize this scenario.

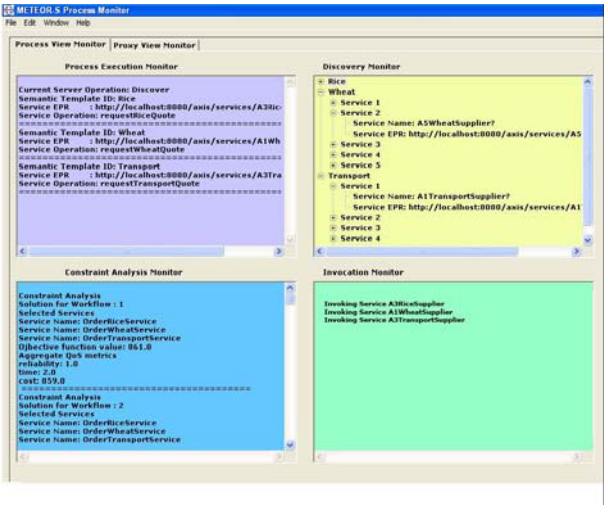


**Fig. 2.** Screen shot of Process Execution Monitor

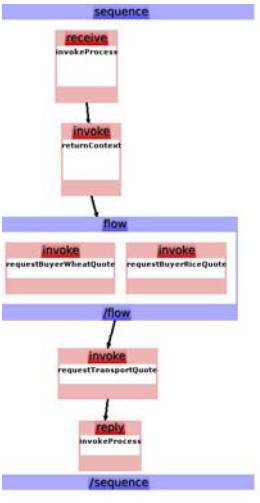


**Fig. 3.** Web Process demonstration scenario

The seller captures the product(s) to be sold, the input and the output types and his constraints in a semantic template. In the demonstration example the seller wants to sell rice and wheat. Constraints on part of the seller could include a) Payment must be made on the same day as the transaction b) Payment must be in cash c) the transportation company must guarantee delivery. Proxy Web services are created from the semantic templates and a Web process with proxies and the process manager as a partner is deployed. The above three mentioned illustrative constraints will be used in the demonstration of the system.

When the Web process is executed the process sends a configuration request to the process manager. The process manager then discovers potential buyers and chooses a set of buyers who satisfy the constraints of the seller.

The constraints of a buyer must also be considered in choosing buyers. Buyer constraints may include a) Payment will be made only by check b) the transportation company will provide insurance only if shipment is greater than a certain minimum amount.

Each proxy when invoked by the Web process sends a binding request to the process manager. The process manager responds with details of the buyer corresponding to the

semantic template that was used to create the proxy. The proxy then sends an execution request to the execution environment. The execution environment performs data and protocol mediations as needed before invoking the buyer Web service.

We demonstrate how adding dynamism to such a Web process helps a seller optimize his profit. This also ensures that for both the buyer and seller the most compatible business partner is chosen.

The Web process is deployed and executed with a set of ten Web services for each partner. Fig. 3 is a screen shot of the METEOR-S web process execution monitor.

## 4 Innovative Features in the System and Demo

1. We have demonstrated the use of MCEE by using it in a real world scenario.
2. Unique capabilities of MCEE include ability to perform discovery, constraint analysis, data and interaction protocol mediation.
3. The system implementation is agnostic to both Web process language (like BPEL) and Web service implementation language.
4. The demonstration gives an insight for using the WSDL-S specification for creating more dynamic processes.

## 5 Summary

We have demonstrated MCEE and have shown how it is used in configuring dynamic Web processes. While using semantics is a critical aspect of METEOR-S, we also seek to build upon existing standards related to Web services and the Service Oriented Architecture. Our aim is to preserve existing investment in Web services technology and tools; this is shown by the reuse of existing WS tools like BPEL process engine and Apache Axis to create our system. The MCEE system can be seen as layer over the current WS infrastructure, which handles the semantic information added through the extensibility capabilities. Our goal is seamless operation of WS and SWS. For this purpose, we have proposed the WSDL-S specification in collaboration with IBM, and have used it in our system as the basis of semantic annotation.

## References

1. Kunal Verma, Karthik Gomadam, Amit P. Sheth, John A. Miller, Zixin Wu, The METEOR-S Approach for Configuring and Executing Dynamic Web Processes, LSDIS Technical Report, 2005.
2. R. Akkiraju, J. Farrell, J.A.Miller, M. Nagarajan, M. Schmidt, A. Sheth, K. Verma, Web Service Semantics - WSDL-S, Position Paper for the W3C Workshop on Frameworks for Semantics in Web Services, Innsbruck, Austria, 2005.
3. Rohit Aggarwal, Kunal Verma, John A. Miller, William Milnor "Constraint Driven Web Service Composition in METEOR-S", Proceedings of IEEE International Conference on Services Computing (SCC 2004), Shanghai, China, September 2004 , pp. 23-30.

4. Kaarthik Sivashanmugam, Kunal Verma, Amit P. Sheth, Discovery of Web Services in a Federated Registry Environment, Proceedings of IEEE Second International Conference on Web Services, June, 2004, pp. 270-278.
5. Vikram Sorathia, Zakir Laliwala, Sanjay Chaudhary, Towards Agricultural Marketing Reforms: Web Services Orchestration Approach, Proceedings of IEEE International Conference on Services Computing, Orlando, Florida, July 2005, pp 260-267.
6. Francisco Curbera, Rania Khalaf, Nirmal Mukhi, Stefan Tai, and Sanjiva Weerawarana, The Next Step In Web Services, Communications of the ACM, 2003