A Schematic Approach for Dynamic Web Service Composition Using Web Ontology Language

[1]Nannapaneni Anusha, [2]M.Rajasekaran

[1]PG Scholar, [2]Assistant Professor
[1]Department of Computer Science, SRM University, Chennai, Tamilnadu, India

***Abstract*—The Composition of existing web services to deliver a new functionality is an important requirement in many business domains. Service composition extends the notion of service discovery by enabling composition of existing web services to meet the requirements of a given high-level task description. The proposed work focuses on developing an ontology-based dynamic web service composition approach that uses Web Ontology Language (OWL) to register, query and dynamically invoke composed web services. The composition logic is defined using workflow constructs that supports reasoning at runtime. The invocation of a composed web service is done using a simple application where a service client discovers and invokes a composed web service at runtime. The service invocation procedure involves resolving the components of the composed web service, querying using the semantic description of the components, selecting the best matched web service for each component, and executing the composed web service according to the composition logic. The approach used for publication, discovery and composition of semantic web services is METEOR-S(Managing End-To-End OpeRations for Semantic Web Services).**

***Keywords*—service composition, service discovery, web ontology language, semantic web services.**

I. INTRODUCTION

Business integration is possible with standardized technologies that enable efficient interoperability among heterogeneous computer systems. These technologies are known as Web services. Simple Object Access Protocol (SOAP) hides the implementation details and could be used for interaction among distributed systems by using Extensible Markup Language(XML).

In Service Oriented Architecture (SOA), a service is defined as a reusable component for use in a business process. A service converts a set of inputs to a set of outputs. The inputs may be business data in a consistent state while the output can be information or business data in another consistent state. Services are interoperable and are location independent. Web services use SOAP messages to communicate with other Web services. The SOAP messages are described using Web Service Description Language (WSDL) and can be transmitted over a standard communication protocol such as Hyper-Text Transfer Protocol (HTTP).

Due to the increasing interest in knowledge over data, and the rising popularity of the Semantic Web as well as Web Services, there have been significant interests in developing technologies that support Semantic Web Services [2]. The Semantic Web industry is experiencing a need for identifying and developing technology that will provide a firm and long-term foundation to support Web services in the future. This foundation should support the universal approaches that are technically feasible for service deployment. The foundation should have the features of flexibility and extensibility, and consistency with the vision of the Semantic Web. Web Services based on industry standards of Universal Description, Discovery and Integration (UDDI), WSDL and SOAP focus only on operational and syntactic details for implementation and execution which makes service publication, discovery and composition very restricted.

II. METHODOLOGY

Web process composition can be done using two approaches. One approach is using an abstract process containing abstract services as a starting point. An abstract service is a placeholder for a set of services matching the template that can be constructed for the abstract service [3]. In some cases, the set may have cardinality greater than one, for example, multiple competing services which match the template may be available. In this way, the topology of the service process is largely fixed; however, actual service selection may be highly dynamic. An alternative approach to composition is to not start with a basic abstract process, but rather form a set of goals and build the whole process. Several AI researches are investigating the use of planning agents for this purpose. In the near term, having a well-designed abstract process as a starting point (i.e., having most of process topology pre-designed), is a useful and pragmatic initial step.

There are two types of business processes: an executable process which models the actual behavior of the participant in a business interaction and abstract processes which use process descriptions [1] for business protocols. Business interaction models include sequences of messages from one party to another, both asynchronous and synchronous in long-running, stateful interactions involving multiple parties. A business protocol is a formal specification of the message exchange procedure used in business interactions. The process descriptions are used to indicate the message exchange between the parties involved in the protocol without revealing their internal behavior. A process can define a business protocol, using the notion of abstract process.

The key approach is in allowing users to capture high level specifications as abstract processes. Use Semantic Web technologies to represent the requirements for each service in the process. We build on earlier work on automated discovery of Semantic Web Services [2] based on the user's requirements. 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. During optimization we select an optimal set of Web service which is one that best satisfies the constraints and minimizes (or maximizes) an objective function.

III. System Architecture

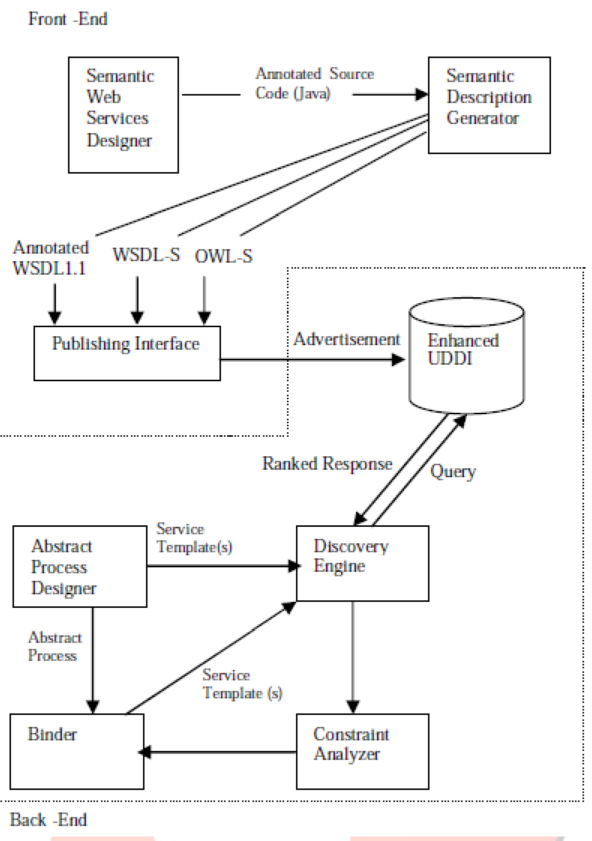

Fig. 1. System Architecture

Service Composition Using OWL

In the front-end, the semantic Web service designer is used to create semantic Web services by annotating source code with ontology concepts [1]. This annotated source code can be converted into annotated WSDL, WSDL-S or OWL-S document by the semantic description generator [1]. Annotated WSDL uses the extensible elements provided in WSDL to add semantics to it. WSDL-S defines its own constructs to represent semantics. The descriptions can then be published into enhanced UDDI which was created by adding a layer above UDDI v2 [2] to add semantics to it.

In this work, we present the Constraint Driven Web Service Composition in METEOR-S (Managing End-To-End Operations for Semantic Web Services), which is a comprehensive framework for the composition of Web services. The METEOR-S back-end allows the manufacturers to bind services based on the given abstract process, requirements and the process constraints. We implemented the enhanced UDDI, discovery engine, constraints analyzer, optimizer and binder. We were able to achieve Web service composition starting from an abstract process based on the constraints. We were also able to bind the optimal service set to the abstract process.

METEOR-S allows us to design and abstractly represent the functionality of the required services using Service Templates (ST). The Discovery Engine is an interface over UDDI registries to provide semantic publication and discovery. The constraint analyzer module produces approved (includes optimal, near optimal and ranked) sets based on business process constraints and objective functions. The binder binds a set of services to the abstract process and generates an executable process.

Abstract Process Designer

Abstract process designer involves creating a representation of Web processes. Design of abstract processes involves the following tasks.

1. Creating the flow of the process using the control flow constructs [4].
2. Representing the requirements of each service in the process by specifying service templates, which allow the process designer to either bind to a known Web service or specify a semantic description of the Web service.
3. Specifying process constraints for optimization.

Discovery Engine

Given a service template, the discovery engine [2] will return a set of service advertisements which match the template. The discovery engine also searches for the transformations required to make a service advertisement match the template.

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 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. The constraints analyser has three sub-modules: the constraint representation module, the cost estimation module and the optimization module.

The constraint representation module allows us to represent the business constraints in ontologies. A business constraint is defined as any constraint that affects the selection of a Web service for a process.

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 the following:

- Service Dependencies
- Querying and cost estimation
- Process constraints

Service Dependencies: It is possible for the selection of one service to depend on another. 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.

Querying and Cost Estimation: Let us consider the supply chain for the manufacturer. 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

Actual values for cost, supply time and other such factors can be obtained either from the UDDI, or by querying internal databases/third parties (like consumer reports) or getting quotes from the suppliers Web services.

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. The process level QOS is calculated as the aggregation of QOS of all the services in the process.

The constraints specified by the user are stored in the Service Template. The service providers can specify an operation in the service which can be invoked to get the QOS Metrics or constraints of the service. The Optimizer module retrieves constraints for the services matching the Service Template from either the UDDI or by invoking an operation of the service specified by the provider. 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 and objective function, 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.

BINDER

After sending the service templates to the discovery engine, discovering and optimizing, the last stage in METEOR-S Constraint Driven composition deals with binding the abstract process to the optimal set of services (which match the service templates and satisfy the given constraints) to generate an executable process [3].

IV. 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 [4].

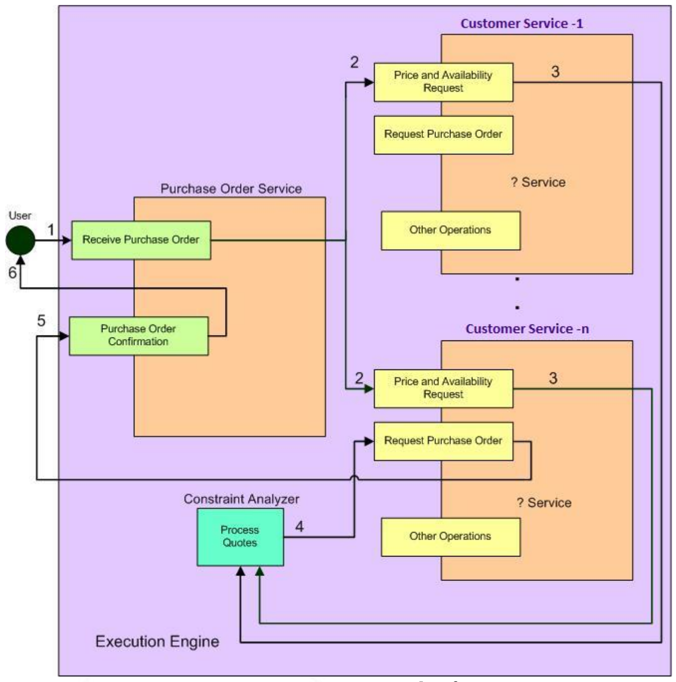

Fig. 2. Example of METEOR-S

Above Figure shows the internal working of the METEOR-S backend at the Web service level for a supply chain example. The PriceAndAvailabilityRequest is sent to multiple candidate services and responses are received. The response quotes are then processed and the most optimized candidate is chosen and the PurchaseOrderRequest is then sent.

V. Conclusion

In this work, we have presented an approach for achieving constraint driven Web service composition. This work builds on the METEOR-S Web Service Composition Framework by adding the abstract process designer, constraint analyzer, optimizer and binder modules. We have extended the workflow QOS model to allow for global optimization and composition of Web processes.

METEOR-S adds the advantage of taking an abstract process as a starting point and automatically binding services to it. To have dynamism in process composition, METEOR-S helps to provide the plug-and-play support for dynamically selecting Web services by enhancing discovery of relevant Web services using Semantics. It also reduces manual intervention during Web process composition. It provides the facility of choosing the optimal set automatically or having the user choose the best set from a list. Constraint analysis gives a better service and choice to the clients by making sure that the services satisfy the constraints and also by making sure that the optimal set of services is the one that is used to create the executable process.

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 work to comprehensively address the issue of composing business processes from an abstract process using business and process constraints. We have described the METEOR-S package including the METEOR-S Web service annotation framework, scheduling, discovery, constraint analysis, optimization and composition.

References

[1] J. Luo et al., "Adding OWL-S Support to the Existing UDDI Infrastructure," Proc. IEEE Int'l Conf. Web Services (ICWS '06), pp. 153-162, 2006.

[2] F. Colasuonno et al., "jUDDI+: A Semantic Web Services Registry Enabling Semantic Discovery and Composition," Proc. IEEE Eighth Int'l Conf. E-Commerce Technology and IEEE Third Int'l Conf. Enterprise Computing, E-Commerce, and E-Services (CEC/EEE '06), 2006.

[3] Micillo.R.A, Venticinque.S, Mazzocca.N, Aversa.R.,"An Agent Based Approach for Distributed Execution of Composite Web Services,"Proc. IEEE 17th Int'l Conf.Workshop on Enabling Technologies:Infrastructure for Colloborative Enterprises,2008.

[4] Cheng.H.J, Ou-Yang.C, Juan.Y.C.,"A Pattern Based Approach To Workflow Structure Analysis," Proc. IEEE 18th Int'l Conf.Industrial Engineering and Engineering Management (IE&EM),2011.