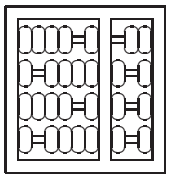

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

A UDDI Extension for Web Service-based Business Process Management Systems

D. Z. G. Garcia *M. B. F. de Toledo*

Technical Report - IC-06-019 - Relatório Técnico

October - 2006 - Outubro

A UDDI Extension for Web Service-based Business Process Management Systems

Diego Zuquim Guimarães Garcia*
Maria Beatriz Felgar de Toledo†

Abstract

The Universal Description Discovery & Integration (UDDI) is a specification for XML registry that enables functionality-based publication and discovery of Web services. This paper presents a UDDI extension to support Quality of Service (QoS) management in the context of Business Process Management Systems (BPMSs). To allow service selection according to functional and non-functional requirements, a BPMS architecture should include brokers to match consumer policies with policies stored in UDDI repositories and monitors to update QoS attributes. The main contributions of this approach are the use of the Web Services Policy Framework specification to describe policies for Web services and a UDDI extension to include QoS attributes.

1 Introduction

As a result of the intensification of electronic commerce and outsourcing of functions there is a need to automate interorganizational business processes. Business processes are at the core of electronic business operations. Organizations have identified process automation as the foundation for enterprise application integration (EAI), on-line service delivery and electronic commerce. Thus, business solutions should include Business Process Management Systems (BPMSs) [5, 18].

Electronic services such as Web services have been announced as the next wave of electronic business applications within and across organizational boundaries. BPMSs allow automated activities within a business process to be executed as Web services that can be dynamically discovered and bound [14, 2].

*Institute of Computing, University of Campinas, 6176 13081-970 Campinas, SP, Brazil, e-mail: diego.garcia@ic.unicamp.br

†Institute of Computing, University of Campinas, 6176 13081-970 Campinas, SP, Brazil, e-mail: beatriz@ic.unicamp.br

The increasing number of provider organizations that offer Web services has prompted research in service description and discovery [10]. Consumer organizations need tools to search for suitable services that can be found all over the world. This poses challenges in discovery mechanisms. Not only techniques to discover services are required but also high-quality information for registered services [12].

The Universal Description Discovery & Integration (UDDI) standard [9] is a step towards meeting these new demands. UDDI supports service description and discovery based on functional aspects. However, consumer requirements may include not only functional aspects of services but also non-functional aspects. Thus, the service discovery depends on the ability to describe and to match service Quality of Service (QoS) offers and demands, in addition to functional capabilities [21].

There is a variety of QoS attributes, such as performance attributes, which are significant for consumers [19]. As observed in [15], policies can be created to describe Web service QoS. Hence, QoS attributes are candidates for policy definition using the Web Services Policy Framework (WS-Policy) specification [3]. WS-Policy offers an XML model for service description. The definition of QoS policies provides a way to distinguish Web services offering equivalent functionalities and to select services meeting functional and non-functional organizational requirements.

Therefore, UDDI must be extended to allow mapping QoS policies into UDDI data structures. Another challenge that rises in this context is to keep QoS attributes up-to-date [17].

The goal of this paper is to propose a UDDI extension that supports QoS management for Web service-based BPMSs. In the proposed approach, QoS information derived from policies are encapsulated inside QoS Policy structures stored in UDDI registries. QoS Policy structures are associated with Binding Template structures representing Web service instances. Each element of a QoS Policy can be associated with Technical Model (*tModel*) structures, which allow specification, reuse and standardization of QoS-related concepts. Furthermore, the extension allows the use of monitoring entities to update QoS attributes.

The paper major contributions are: extending the UDDI information model and API sets to refine service selection, using WS-Policy to specify QoS policies for Web services, using *tModels* to define QoS related-concepts, and supporting business-to-business interactions on the Web.

The rest of the paper is organized as follows. Section 2 presents basic concepts. Section 3 discusses the proposed approach for UDDI extension in the context of Web service-based BPMSs. Section 4 describes BPMS architecture based on the proposed UDDI extension. Section 5 presents implementation aspects and evaluation of the architecture. Section 6 discusses related work. Section 7 presents conclusions.

2 Basic Concepts

This section presents some basic concepts for the better understanding of this paper.

2.1 Business Process Management

Business Process Management Systems (BPMSs) provide control for the definition and coordination of business processes. A business process is a collection of linked activities. Activities are descriptions of work pieces that collectively realize a business objective, typically within the context of an organizational structure. An activity may be manual or automated. Differently from traditional Workflow Management Systems (WfMSs), BPMSs focus on the management of dynamic interorganizational processes. These business-to-business processes comprise activities that may be performed by different business partners [27, 18].

BPMSs support business process automation. The life cycle of business process automation (Figure 1) starts with process design. The process is then registered with a BPMS, which can create process instances. BPMS coordinates instance execution and records information collected during execution. The execution history is analyzed and used to improve process definitions [5, 28].

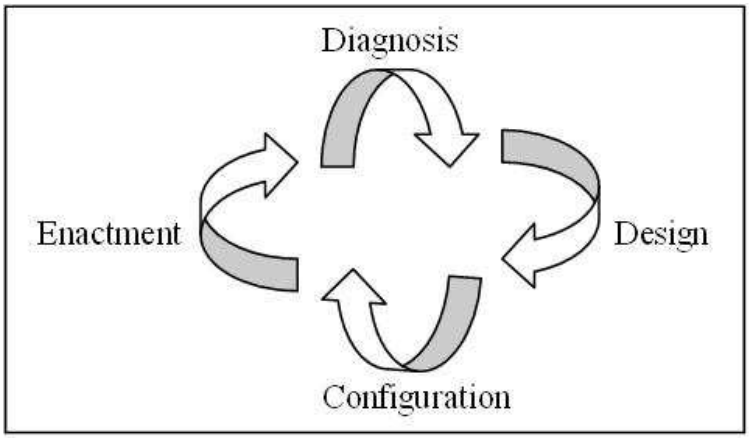

Figure 1: Business Process Management life cycle

2.2 Web Services

Technologies such as Web services may be used as key building blocks in service-oriented architectures (SOAs) [18, 24]. A service-oriented BPMS supports the management of business processes that use electronic services such as Web services to achieve business goals [14, 2].

A Web service is an electronic service identified by a URI (Uniform Resource Identifier). XML (eXtensible Markup Language) standards [21, 1] are used to describe service interfaces and to invoke services through the Web. Figure 2 shows the Web service basic architecture. Web service technology comprises three basic standards [1]:

- Web Services Description Language (WSDL) provides a model and a format for describing the abstract functionality of a Web service as well as the concrete details of a service description.
- Universal Description Discovery & Integration (UDDI) offers a registry (Figure 2) for Web service descriptions that supports the publication and discovery of providers, the Web services they make available and the technical interfaces that consumers may use to access the services.
- SOAP is a protocol intended for exchanging structured information in a decentralized, distributed environment, such as a Web service environment.

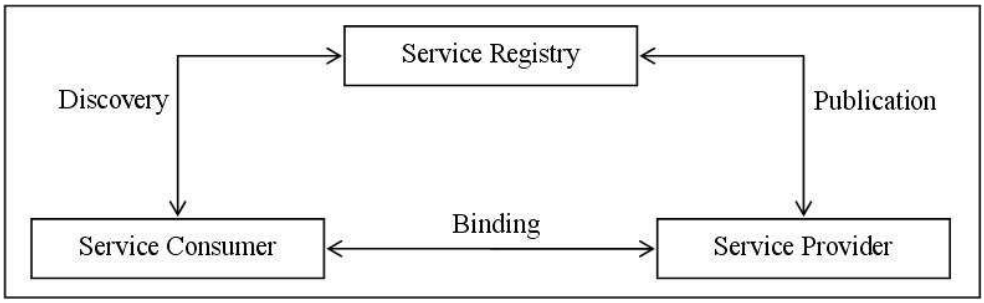

Figure 2: Web service basic architecture

Additional Web service specifications are under development. One example is the Web Services Policy Framework (WS-Policy). It provides a set of constructs that can be extended to specify a broad range of service requirements and capabilities [3]. Although WSDL represents the standard for the Web service functionality definition, efforts are now focusing on languages that can complete such a description by considering aspects that are not directly related to what a Web service does and how a service should be invoked. WS-Policy represents one of these attempts and is a candidate to become a future standard due to its flexibility [4].

Typically, in Web service-based BPMSs, executing a business process requires integration of several loosely coupled services. In this type of environment, the management of policies in areas such as access control or QoS becomes a challenge. Such policies may be related with, for instance, service operations, management of service level agreements, and enforcement and management of high-level business initiatives such as regulations and protection of intellectual property. Current approaches lack

ways to define concepts for a specific area and the ability to monitor/enforce policies [24].

3 QoS-based UDDI Extension

This section presents a UDDI extension to allow the definition of QoS attributes for Web services.

Web service-based BPMSs require registries where providers may publish services that will be discovered during process enactment. Service discovery demands a service description comprising service capabilities, interface, classification and QoS [21, 1].

Based on XML and XML Schema, UDDI provides an interoperable infrastructure for integrating information in Web service environments for both publicly available services and services exposed internally within organizations [9]. However, the current UDDI standard still lacks facilities regarding QoS description and discovery. As WS-Policy offers a means to specify QoS, service providers may specify their services using WSDL and WS-Policy standards and publish provided services. UDDI should be extended to include QoS policies and the enhanced service description may be used in service discovery. These extensions to UDDI registry and query mechanisms enhance search flexibility. Web services can be searched by functional characteristics with required QoS attributes as constraints.

3.1 Information Model

UDDI information model [9] is composed of data structure instances expressed in XML. They are persistently stored in UDDI registries. The extended UDDI information model (Figure 3) aggregates the *qosPolicy* structure and its relationships. The data structure types are described below.

- *businessEntity*: top-level data structure within UDDI that contains descriptive information about the provider organization, such as name, contact and classification information. Each *businessEntity* may offer various *businessServices*.
- *businessService*: represents a logical Web service that may have multiple implementations. It includes descriptive information about a Web service, such as name, description and classification.
- *bindingTemplate*: represents a service implementation and provides the information needed to bind with the service. Each *bindingTemplate* contains information such as access point and transport protocol.
- *tModel* (Technical Model): represents unique concepts in UDDI, such as category systems and specifications. Examples include *tModels* based on WSDL,

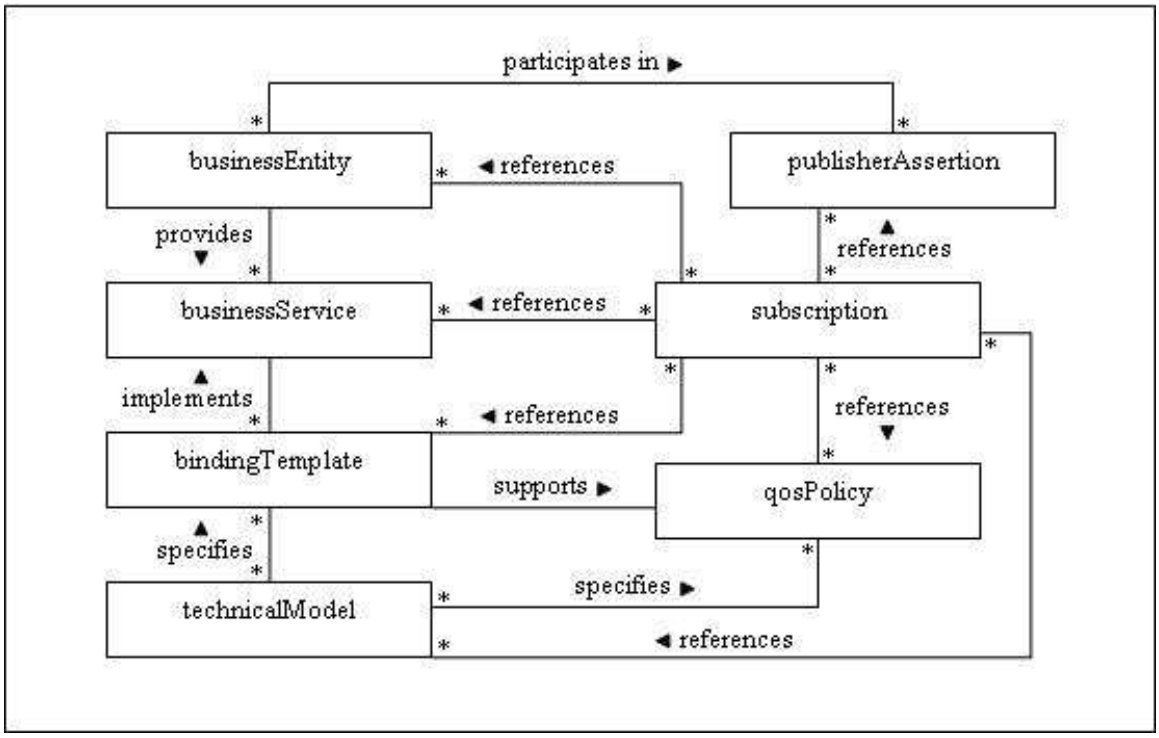

Figure 3: Extended UDDI information model

XML Schema Definition and other documents that outline and specify service interfaces.

- *publisherAssertion*: defines a coupled community of *businessEntities*. It can be used by associated organizations to export their relationships, for instance subsidiary companies, private supply chains and industry consortia.

- *subscription*: describes a request to keep track of activities in a UDDI registry according to preferences provided with the request. Subscribers can register themselves for receiving information about changes in any of the UDDI structures.

- *qosPolicy*: represents a QoS policy for a service implementation. The provided information describes attributes of a Web service represented by the *bindingTemplate* structure. References to *tModels* can be used to describe that a policy conforms to particular specifications.

3.2 Attributes

The *qosPolicy* structure supports the description of QoS in UDDI. QoS is a combination of several attributes (qualities or properties) of a service [19]. Table 1 describes the Web service QoS attributes [19, 22, 16] represented by this structure.

QoS Attribute	**Description**
Response time	The time period a service takes to complete its task
Latency	The time period taken to start servicing a service request
Throughput	The request processing rate a service supports
Scalability	The throughput increase rate in a given time period
Capacity	The number of concurrent requests a service allows
Availability	The time percentage a service is operating
Reliability	The time period for continuity of correct service and for transition to correct state
Accuracy	The service error rate over a time period
Robustness	The service resilience level to incorrect inputs and invocation sequences
Stability	The change rate of service interface and implementation
Security	Defines whether the service offers mechanisms of authentication, authorization, confidentiality, integrity, auditing and non-repudiation
Reliable messaging	Determines if the service offers mechanisms to guarantee reliable message delivery
Integrity	Defines whether the service supports transactional properties
Cost	The measure of the cost involved in using the service
Standards	Defines if the service is compliant with industry specific standards
Interoperability	Determines if the service is compliant with interoperability profiles defined by the Web Services Interoperability Organization

Table 1: Web service QoS attributes

3.3 *tModel*

The *qosPolicy* structure is based on the *tModel* concept [9]. A QoS *tModel* supports specification sharing. For instance, consortia or standardization bodies can create *tModels* that specify QoS attributes that services of specific industries may have. The *qosPolicy* structure refers to *tModels*, which are used to define QoS-related concepts, including attributes, units, relationships, dependencies and measurement techniques [19, 22, 15, 16].

QoS attributes other than those applicable to multiple domains (Table 1) can also be defined. General attributes may be redefined. For instance, the throughput may specify a function that describes how the processing rate varies with the load instead of the maximum request-processing rate. Moreover, different measure units can be used for a particular attribute. The employed measure unit has to be defined.

QoS attributes can be interconnected and correlations can be defined. These correlations may indicate trade-offs among the various QoS attrbutes. For instance, security may decrease performance. Furthermore, environmental conditions can influence QoS attributes. For instance, response time is a function of load intensity, which is related to the number of consumers.

QoS attributes can be measured objectively or subjectively, and automatically or with human intervention. Techniques used by QoS verification entities may be defined. These third-party entities are responsible for monitoring or certifying Web service QoS [8]. In [22, 15], metrics for general QoS attributes are presented.

3.4 API

UDDI defines APIs [9] that standardize communication within and between UDDI implementations. The APIs are grouped into sets. Extended UDDI includes extensions to UDDI Inquiry, Publication and Subscription API sets. Moreover, it uses UDDI Security API set during the execution of operations in other sets.

The new and extended API calls used to manipulate QoS data stored in UDDI registries using the *qosPolicy* structure include: *save_binding* (publishes binding template, including associated QoS policy), *find_binding* (discovers instances of desired service based on QoS policy), *get_bindingDetail* (selects binding template based on key), and *update_qosPolicy* (updates service instance QoS policy).

3.5 Policy Specification and Processing

To use extended UDDI facilities, it is necessary that providers specify Web service QoS using WS-Policy. Thus, consumers may select services considering their QoS demands also described using WS-Policy.

WS-Policy provides an XML model and syntax for expressing Web service properties as policies. A policy is a collection of alternatives and each policy alternative

is a collection of assertions. Policy assertions specify characteristics that are critical to service selection and usage, for instance, QoS characteristics [24].

Policies may be used during the different phases of the Web service life cycle [20]. During design and deployment time, service providers may define Web service policies describing how Web services should work. These provider policies may merge server-independent policies and policies that state the characteristics that services deployed on a particular application server are expected to support. During runtime, service consumers may define policies associated with partner services. These consumer policies state the properties that should be offered by required Web services.

The consumer and provider policy assertions may be intersected in order to compute the effective assertion set [3]. This assertion set drives the service selection. After effective assertion sets are computed, Web services should be monitored to check if the stated properties hold [4].

In the case of QoS policies, consumer policy assertions define QoS requirements and provider (or Web service) policy assertions describe Web service QoS attributes. A Web service policy is an extension of a Web service interface described in WSDL. QoS attributes contained in policy assertions have to be extracted and then can be stored in UDDI registries.

The interaction with UDDI is done through Brokers. A Broker implements QoS management operations. It uses extended UDDI API sets to retrieve data from the UDDI registry, and to publish and update information contained in UDDI. Brokers process provider QoS policies and map policy assertions into the *qosPolicy* structure. The policy processing is based on WS-policy operations [3] that allow policy normalization and decomposition. The normalization operation is the process of reducing policies to a standardized format. It simplifies the manipulation of policies. In order to decompose policies into assertions, they must be first normalized. Consumers provide QoS policies to Brokers that used them to select Web services.

4 An Architecture for Web Service-based BPMSs

This section describes a BPMS architecture that includes the proposed UDDI extension. Its main components are Brokers, Monitors and UDDI registries. A Broker is responsible for service discovery and matching using QoS attributes. The extended UDDI registry stores information about provided Web services including descriptive, technical, categorization and QoS information. The Monitor updates the registry with QoS information acquired during business process enactment. The components of the architecture interact through a SOAP engine.

An overview of the architecture is presented in Figure 4. The responsibilities of the architecture components are discussed below.

4.1 Registries and Organizations

UDDI registries offer a place where provider organizations publish their services and consumer organizations discover services. Thus organizations have a way to find potential trading partners in a global environment using a distributed service registry. The registries hold a broad range of service information allowing a search based on many capabilities and attributes such as QoS.

Providers may use Brokers to register required information about offered services in registries. Providers are responsible for service management including operations for service insertion and removal, and related information update.

Consumer organizations demand services to be executed as process activities within a business process. Business process models may contain internal activities as well as external activities. Activities defined as Web services may be discovered in UDDI registries through Brokers.

4.2 Broker Services

Broker Services facilitate service registry access [25, 13].

Brokers obtain QoS information from provider policies and register it into UDDI registries along with other information needed for the service publication and subsequent discovery. They also select Web services during business process enactment according to functional and non-functional requirements as specified by consumers.

The Broker is also a Web service. This enables the architecture deployment in restricted and open environments and the use of Brokers with different features, according to specific needs.

The use of Brokers raises security issues. For instance, it is important to verify whether the employed Broker is trustful [8].

4.3 Monitor Services

Monitor Services may be implemented as Web services to measure QoS of other Web services [16].

Monitors monitor QoS attributes during service execution considering QoS policies. Monitors act as an interceptor layer between consumers and Web services. There is a Monitor for each Web service. Providers may register services with trusted intermediaries [8]. These entities will be responsible for creating and managing Monitors. Security, transaction and context issues are important whenever intermediaries are involved, as observed in [8].

Monitors may update QoS information stored in registries after a service execution. In this case, the UDDI authorization mechanism must be used. Monitors interact with registries through brokers.

4.4 Architecture Component Interactions

A typical usage scenario (Figure 4) is described in this section considering an example in which a consumer organization employs a Web service of a provider organization in its business process.

- Step 1. The provider can offer an electronic service as a Web service. It requests service publication and provides the QoS policy of the Web service. The Broker registers the service in the UDDI registry and returns a confirmation.
- Step 2. The BPMS allows the use of a Web service as a business activity. At business process definition time, an activity represents a service and a QoS policy can be specified for the activity. At process runtime, when the activity corresponding to Web service is reached, the workflow engine requests service discovery. The Broker selects a service in the UDDI registry according to the specified QoS policy and returns a service address.
- Step 3. The service can be invoked through the returned access point. The Monitor monitors service execution and QoS parameters. The service results are returned to the consumer.
- Step 4. Finally, the Monitor updates QoS attributes of the Web service according to monitoring results.

5 Implementation and Evaluation

This section presents implementation aspects of the proposed architecture and discusses preliminary tests.

MySQL Version 5.0.16 was used to implement the UDDI database. The *QOS_POLICY* table stores Web service QoS attributes. It refers to the *BINDING_TEMPLATE* table containing the Web service instances. This table refers to the *TMODEL* table containing technical models for QoS attributes.

UDDI4J Version 2.0.4, an open-source Java class library, supported by HP, IBM and SAP, was used to implement the UDDI API. The extended Subscription API set has not been implemented.

The extended UDDI is a component in the BPMS architecture. This architecture was partially implemented using the Sun Java Development Kit Version 1.5.0_06, Apache Axis Version 1.3 (providing SOAP and WSDL support), Apache WS-Commons/Policy Version 0.90 (providing WS-Policy support) and Apache Tomcat Version 4.1.24 (for application server support).

In the current implementation, the Broker is co-located with the UDDI registry and the Monitor with the consumer. Initially, service discovery is based on availability,

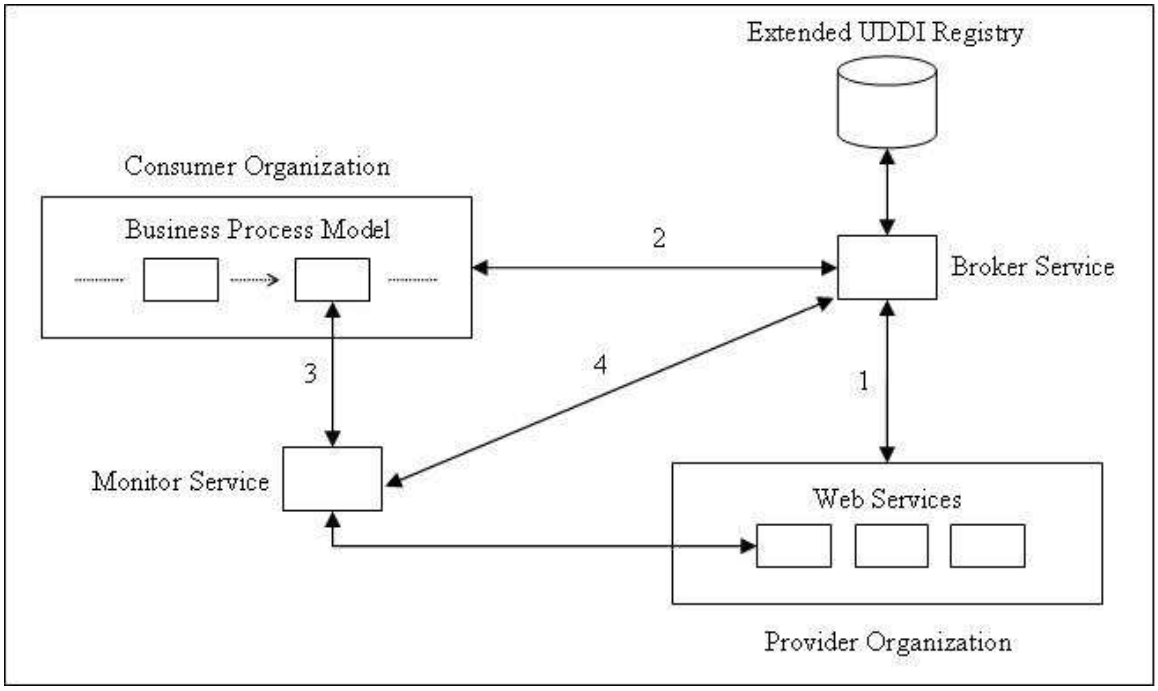

Figure 4: BPMS architecture based on the extended UDDI registry

cost and response time attributes. Service monitoring considers just the response time attribute. Support for additional QoS attributes will be included in near future.

The UDDI extension is compatible with the basic UDDI and both types of UDDI registries can coexist in the same environment. Moreover, it is possible to access services without using UDDI.

Some experiments were executed with simulations that involved consumers with different QoS policies (including availability, cost and response time constraints) searching for services in the extended UDDI registry. *tModels* were specified to define concepts related to the QoS attributes considered during the experiments. QoS policies were specified using WS-Policy. Web services together with their policies were stored in the extended UDDI registry. The extended API sets were used to advertise and discovery services, and different cases were tested to evaluate QoS update, including policy fulfillment and violation.

The goal of these experiments was to evaluate whether the UDDI extension enables consumers to select suitable providers with respect to QoS policies in addition to functional requirements. Preliminary results have shown that the UDDI extension can be included in a BPMS architecture to support QoS management without compromising performance.

6 Related Work

Industry organizations and academia prescribes that service descriptions should follow predefined industry categorizations in order to allow service discovery according to consumer requirements [12].

The inclusion of QoS in Web service standards has been pointed out in previous work in both academia [19] and industry [14]. In addition, Leymann et al [14] discuss the importance of QoS management in the context of BPMSs. Challenges in this area must tackle QoS models, verification, and QoS-enriched service publication and discovery.

Ran [22] and Ludwig [15] discuss QoS models for Web service. A QoS ontology is proposed by Maximilien and Singh [16].

Frameworks to support QoS verification are described in [22, 25, 4]. The first is based on certification. However, it lacks support for the dynamism of Web services. Singhera [25] deals with this issue including QoS monitoring in its framework. The framework proposed in [4] can check both functional and non-functional requirements. This framework relies on WS-Policy to express the monitoring policies.

The need of extending UDDI has generated efforts. ShaikhAli's approach [23] is based on the businessService structure, but potential QoS changes are not considered. Chen et al [7] use a server that receives QoS reports and stores them in a separate database.

In [29], an advanced search mechanism is used to increase the efficiency of e-business application development. Consumers can define criteria to filter search results returned from different UDDI registries. Examples of filters are price range, availability or high reputation. However, QoS policies are not used and empirical aspects are not considered in service selection.

In [26], services in UDDI may be linked to WSDL files and each of the WSDL files may be linked to Web Service Endpoint Language (WSEL) files with QoS details of the service, which are used in ranking the discovered services. WSEL is a format for the description of non-operational and behavioral characteristics of services, such as QoS properties, but at present no specification exists for this language.

Lee et al [13] consider historic information of Web service execution. A separate QoS server that complements UDDI is used to store this information. In [17], agencies along with UDDI support QoS information sharing. In [6], service selection is based on historical and contextual information.

Du et al [11] propose a UDDI architecture with a monitoring mechanism. However, only service name, key, version and availability are monitored and kept up-to-date.

Recently, thanks to several research and industrial efforts in QoS-enriched service publication and discovery, a substantial progress has been done. The solutions, however, are typically limited for some of the reasons discussed below:

- UDDI is complemented by other registries. This makes the incorporation of

the registries in the current Web service architecture harder. An integrated way to deal with functional and non-functional requirements may be offered by extending UDDI.

- The supported QoS management facilities do not consider the possibility of changes in QoS attributes. Thus, the update of published QoS descriptions is not allowed and a mechanism for verifying QoS is not employed.

- WS-Policy is not used. WS-Policy gives a flexible open-standard for expressing QoS for Web services. This standard matches with the fundamental requirements of Web services (simplicity and extensive support in industry [1]).

- Finally, the *tModel* is not used for the establishment of shared QoS concepts. This is important as it allows the description of QoS attributes in specific application domains. *tModel* has another advantage: it is part of the UDDI standard and is currently used for specifying a broad range of concepts related to Web services.

In this paper, the above issues are addressed by extending UDDI to include QoS descriptions. General QoS attributes are considered and the UDDI Technical Model concept is used to define QoS taxonomies. One of the main components of the proposed architecture is the monitor that updates QoS in UDDI registries according to information acquired during service execution. The main advantage of the Web service technology is the interoperability achieved through standardization. In this paper Web service standards are used and the UDDI standard is extended with the purpose of integrating Web service and BPMS technologies.

7 Conclusions

Web service technology constitutes a new paradigm for BPM, allowing business processes to include Web services. It is still maturing. UDDI is one of the basic components of the current Web service model. Currently, it is lacking mechanisms to manage Web service QoS.

In this paper, the UDDI standard was extended to be used in the context of Web service-based BPMS. The approach combines proposals to employ WS-Policy for QoS specification, QoS-enriched Web service publication and discovery, and QoS information update. Implementation aspects and evaluation of the proposed architecture were presented. Preliminary results demonstrated that the QoS management supported by the extended UDDI helps service discovery in conformity with functional and non-functional requirements.

The main contributions of this paper are: a UDDI extension to provide Web service QoS management, the use of WS-Policy to specify QoS policies, the use of

tModels to define QoS related-concepts, and an architecture to integrate the extended UDDI into Web service-based BPMSs.

Future work involves enhancing the capabilities of the proposed architecture to handle other QoS attributes and its integration with a Workflow Management System. Another direction is the extension of UDDI with context-aware facilities to enable the on-line delivery of services that adapt to the consumer context.

References

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web services: Concepts, Architectures and Applications*. Springer-Verlag, 2004.

[2] Gustavo Alonso and Fabio Casati. Web services and service-oriented architectures. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, page 1147, Washington, DC, USA, 2005. IEEE Computer Society.

[3] Siddharth Bajaj, Don Box, Dave Chappell, Francisco Curbera, Glen Daniels, Phillip Hallam-Baker, Maryann Hondo, Chris Kaler, Dave Langworthy, Anthony Nadalin, Nataraj Nagaratnam, Hemma Prafullchandra, Claus von Riegen, Daniel Roth, Jeffrey Schlimmer, Chris Sharp, John Shewchuk, Asir Vedamuthu, Umit Yalcynalp, and David Orchard. *Web Services Policy Framework (WS-Policy), Version 1.2, Specification*. BEA Systems Inc., International Business Machines Corporation, Microsoft Corporation, Inc., SAP AG, Sonic Software, and VeriSign Inc., March 09, 2006. http://www-128.ibm.com/developerworks/library/specification/ws-polfram/, accessed on 05/2006.

[4] Luciano Baresi, Sam Guinea, and Pierluigi Plebani. WS-Policy for service monitoring. In Christoph Bussler and Ming-Chien Shan, editors, *TES '05: 6th VLDB International Workshop on Technologies for E-Services*, volume 3811 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2006.

[5] Fabio Casati and Ming-Chien Shan. Process automation as the foundation for e-business. In *VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 688–691, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[6] Dipanjan Chakraborty, Suraj Kumar Jaiswal, Archan Misray, and Amit A. Nanavati. Middleware architecture for evaluation and selection of 3rd-party Web services for service providers. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services*, pages 647–654, Washington, DC, USA, 2005. IEEE Computer Society.

[7] Zhou Chen, Chia Liang-Tien, Bilhanan Silverajan, and Lee Bu-Sung. UX - an architecture providing QoS-aware and federated support for UDDI. In *ICWS '03: Proceedings of the International Conference on Web Services*, pages 171–176. CSREA Press, 2003.

[8] Mike Clark, Peter Fletcher, Jeffrey J. Hanson, Romin Irani, Mark Waterhouse, and Jorgen Thelin. *Web Services Business Strategies and Architectures*. A-Press, 2003.

[9] L. Clement, A. Hately, C. von Riegen, and T. Rogers. *UDDI, Version 3.0.2, UDDI Spec Technical Committee Draft*. OASIS, October 19, 2004. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, accessed on 05/2006.

[10] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity search for Web services. In *VLDB '04: Proceedings of the 30th International Conference on Very Large Data Bases*, pages 372–383, San Francisco, CA, USA, 2004. Morgan Kaufmann Publishers Inc.

[11] Zongxia Du, Jinpeng Huai, and Yunhao Liu. Ad-UDDI: An active and distributed service registry. In Christoph Bussler and Ming-Chien Shan, editors, *TES '05: 6th VLDB International Workshop on Technologies for E-Services*, volume 3811 of *Lecture Notes in Computer Science*, pages 58–71. Springer, 2006.

[12] Jianchun Fan and Subbarao Kambhampati. A snapshot of public Web services. *SIGMOD Record*, 34(1):24–32, 2005.

[13] Eunjoo Lee, Woosung Jung, Wookjin Lee, YoungJoo Park, Byungjeong Lee, Heechern Kim, and Chisu Wu. A framework to support QoS-aware usage of Web services. In *ICWE '05: Proceedings of the 5th International Conference on Web Engineering*, pages 318–327. Springer, 2005.

[14] F. Leymann, D. Roller, and M.-T. Schmidt. Web services and business process management. *IBM Systems Journal, New Developments in Web Services and Electronic Commerce*, 41(2):198–211, 2002.

[15] H. Ludwig. Web services QoS: external SLAs and internal policies or: how do we deliver what we promise? In *WISE '04: Proceedings of the 4th International Conference on Web Information Systems Engineering Workshops*, pages 115–120. Springer, 2004.

[16] E. Michael Maximilien and Munindar P. Singh. A framework and ontology for dynamic Web services selection. *IEEE Internet Computing*, 8(5):84–93, 2004.

[17] E. Michael Maximilien and Munindar P. Singh. Self-adjusting trust and selection for Web services. In *ICAC '05: Proceedings of the 2nd International Conference on Automatic Computing*, pages 385–386, Washington, DC, USA, 2005. IEEE Computer Society.

[18] Brahim Medjahed, Boualem Benatallah, Athman Bouguettaya, Anne H. H. Ngu, and Ahmed K. Elmagarmid. Business-to-business interactions: issues and enabling technologies. *VLDB Journal*, 12(1):59–85, 2003.

[19] Daniel A. Menascé. QoS issues in Web services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[20] Nirmal K. Mukhi and Pierluigi Plebani. Supporting policy-driven behaviors in Web services: experiences and issues. In *ICSOC '04: Proceedings of the 2nd International Conference on Service Oriented Computing*, pages 322–328, New York, NY, USA, 2004. ACM Press.

[21] Mike P. Papazoglou and D. Georgakopoulos. Service-oriented computing - guest editorial. *Communications of the ACM*, 46(10):24–28, 2003.

[22] Shuping Ran. A model for Web services discovery with QoS. *SIGecom Exchange*, 4(1):1–10, 2003.

[23] Ali ShaikhAli, Omer F. Rana, Rashid Al-Ali, and David W. Walker. UDDIe: An extended registry for Web services. In *SAINT-W '03: Proceedings of the 2003 Symposium on Applications and the Internet Workshops*, pages 85–89, Washington, DC, USA, 2003. IEEE Computer Society.

[24] Vishal Sikka. Data and metadata management in service-oriented architectures: some open challenges. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 849–850, New York, NY, USA, 2005. ACM Press.

[25] Zafar U. Singhera. Extended Web services framework to meet non-functional requirements. In *SAINT-W '04: Proceedings of the 2004 Symposium on Applications and the Internet-Workshops*, pages 334–340, Washington, DC, USA, 2004. IEEE Computer Society.

[26] K. Sivashanmugam, J. Miller, A. Sheth, and K. Verma. Framework for semantic Web process composition. *International Journal of Electronic Commerce*, 9(2):71–106, 2005.

[27] Wil M. P. van der Aalst. Process-oriented architectures for electronic commerce and interorganizational workflow. *Information Systems*, 24(9):639–671, 1999.

[28] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske. Business process management: A survey. In *BPM '03: Proceedings of the 1st International Conference on Business Process Management*, pages 1–12. Springer, 2003.

[29] Liang-Jie Zhang, Tian Chao, Henry Chang, and Jen-Yao Chung. XML-based advanced UDDI search mechanism for B2B integration. *Electronic Commerce Research*, 3(1-2):25–42, 2003.