2004 IEEE INTERNATIONAL CONFERENCE ON WEB SERVICES (ICWS 2004)

Theme: Convergence of Web Services, Grid Computing, e-Business and Autonomic Computing

SAN DIEGO, CALIFORNIA, USA
JULY 6 - 9, 2004

Sponsored by
The IEEE Technical Community for Services Computing (TCSC)
# Table of Contents

**IEEE International Conference on Web Services — ICWS 2004**

Message from the General Chair ................................................................. xv
Message from the Program Chairs ............................................................. xvi
Conference Officers .................................................................................... xvii
Program Committee ................................................................................... xviii
Steering Committee ................................................................................... xx
IEEE TCSC Executive Committee .............................................................. xxi
External Reviewers ..................................................................................... xxii

**Keynote Speakers**

Keynote 1: Convergence of Web Services, Grid Services and Business Processes ......................................................... xxiii
  *D. Ferguson*

Keynote 2: Extending the Innovation Ecosystem ........................................ xxiv
  *S. Nunes*

Keynote 3: Going Public with Software-as-a-Service ............................... xxv
  *E. Feig*

Keynote 4: Business Services Networks .................................................. xxvi
  *J. Tenenbaum*

Keynote 5: Managing the Intelligent Enterprise ..................................... xxvii
  *U. Duyal*

**Research Sessions**

**Research Session 1: Web Services in Organizations**

  *H. Cattuero-Monge and M. Rodríguez-Martínez*

Web Services as Information Systems Innovation: A Theoretical Framework for Web Service Technology Adoption ...................................................... 11  
  *M. Haines*

WS-I Basic Profile: A Practitioner's View .............................................. 17  
  *K. Kumar, A. Das, and S. Pudmanabhuni*
Short Paper Session 3: Access Control and Failure Handling in Web Services

Access Control for Semantic Web Services..........................770
S. Agarwal and B. Sprick

Authorization Service for Web Services and its Implementation........774
S. Indrakanti, V. Varadharajan, and M. Hitchens

JULIET: A Distributed Fault Tolerant Load Balancer for .NET Web Services...778
R. Murty

Team-Based Agents for Proactive Failure Handling in Dynamic Composition of Web Services........782
X. Fan, K. Umaapathy, J. Yen, and S. Purao

Short Paper Session 4: Web Services Applications

Combining Query and Preference — An Approach to Fully Automatize Dynamic Service Binding........788
M. Klein and B. König-Ries

Anonymizing Web Services through a Club Mechanism with Economic Incentives........792
M. Jenamani, L. Lilien, and B. Bhargava

A Meta-Modeling Approach to Web Services..........................796
F. Cao, B. Bryant, W. Zhao, C. Burt, K. Rajeev, A. Olson, and M. Auguston

A Dynamic Data Structure for Geospatial Web Services Integration.........800
L. Chen and N. Jing

Poster Sessions

Poster Session 1: Quality of Service and Service Composition

Web Services Container Reference Architecture (WSCRA)................806
A. Dhesiaseelan and V. Rayanathan

WS-QDL Containing Static, Dynamic, and Statistical Factors of Web Services Quality........808
S. Yoon, D. Kim, and S. Han

J. Durham, W. Torrez, and J. Manley

An Approach for Parallel Execution of Web Services..........................812
A. Kut and D. Birant

A Web Service Composition and Deployment Framework for Scientific Workflows........814
I. Altintas, E. Jaeger, K. Lin, B. Ludaescher, and A. Memon
WS-QDL containing static, dynamic, and statistical factors of Web services quality

SeokHyun Yoon, DongJoon Kim, SangYong Han
Department of Computer Science and Engineering, Chung-Ang University,
221 Huksuk-Dong, Dongjak-Ku, Seoul, 156-756, Korea
{lazecool, djkim}@archi.cse.cau.ac.kr, hansy@cau.ac.kr

Abstract

Web Services following distributed object computing technology like DCOM, CORBA provides remote procedure call mechanism based on XML-based open standard such as SOAP, WSDL, and UDDI, and it is spotlighted as means of integration and collaboration at e-business. Especially, UDDI is the Web Services Registry enabling to register and search Web Services, that takes charge of providing infrastructure for Web Services. However, the existing UDDI has a few problems that searching process is very simple and it cannot provide information of Web Services quality and quality-based retrieval. Therefore, this study suggests improved UDDI model that evaluates the Web Services quality and use this information for searching.

1. Introduction

Web Services is the technology based on the standards of SOAP, WSDL and UDDI that provides RPC mechanism in distributed environment called the web. In particular, UDDI is the Web Services registry which enables registration and retrieval of Web Services, and it functions as a huge UBR(Universal Business Registry) enabling businesses to store and retrieve their business data as well as information on the fundamental Web Services. But the UDDI information retrieval method too is simple and cannot evaluate the web services quality in retrieval process. The present study proposes an improved UDDI model that provides a web services quality evaluating retrieval method.

2. Web Services Quality Evaluation Factor

The Web services quality factors can be largely divided into quality factors that are determined by the operation policies of the service providers such as the charge or the priority order, and the basic quality factors that are not affected by the provider’s operation policies such as network availability. Present study will not deal with the former group of factors and only take the latter group of basic quality factors into consideration. The Web service standard giving account of the service charges or the operation policies is still in the immature state. This study does not deal with the policy factors on the service operation since UDDI 2.0, which is largely used at present, does not contain such factors although UDDI 3.0 does [1].

The factors generally considered as Web services evaluation factors are execution price, execution duration, reputation, reliability, availability, etc [2][3]. Since evaluation factors related to the provider’s policies are excluded from this study, execution price will not be considered here. Instead, other items were added as Web services evaluation factors as shown below in Table 1.

The fundamental factors for Web service quality evaluation dealt with in this study can be largely divided into static, dynamic and statistical factors. Static factors do not change as long as no changes occur within the service since they are dependent to the service in concern. Meanwhile, dynamic factors represent quality information that changes according to certain situations such as network traffic. Statistical factors are evaluated based on the statistical data of the service.

<table>
<thead>
<tr>
<th>Table 1: Web Services Quality Evaluation Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamic Factors</strong></td>
</tr>
<tr>
<td>Service Availability</td>
</tr>
<tr>
<td>Network Availability</td>
</tr>
<tr>
<td>Execution Duration</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

- **Dynamic Factors**
  - Service Availability
  - Network Availability
  - Execution Duration
- **Statistical Factors**
  - Service Reliability
Table 1: Static Factors of Web Services Quality

<table>
<thead>
<tr>
<th>Factor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Reliability</td>
<td>How stable was the service network?</td>
</tr>
<tr>
<td>Execution Reliability</td>
<td>How frequently is the reply sent back within a standard period of time?</td>
</tr>
<tr>
<td>Reputation</td>
<td>How good is the reputation of the service compared with other services of the same type.</td>
</tr>
<tr>
<td>Security</td>
<td>Does the service abide by security factors such as WS-Security?</td>
</tr>
<tr>
<td>Regulatory</td>
<td>Which standard does the service abide by?</td>
</tr>
</tbody>
</table>

3. Web Services Quality Description Language

At present there exists no language to describe Web services quality. Only simple descriptions are supported in the ontology-based DAML-S [4]. The present study proposes WS-QDL (Web Services Quality Description Language) based on the Web services quality evaluation factors. WS-QDL, the newly suggested language for describing Web services quality, uses the namespace "http://ec.cse.cau.ac.kr/WS-QDL/" and contains static, dynamic, and statistical information.

4. An Advanced UDDI Model for Retrieval based on Web Services Quality

This section forms the WS-QDL containing all three static, dynamic, and statistical factors of Web services quality as shown in Figure 1 and suggests a new Web services quality based UDDI retrieval structure that can be used in Web services retrieval in the UDDI.

![Figure 1. Web Services Quality based UDDI Retrieval Model](image)

First, the consumer agent forms a Web services quality based inquiry based on the consumer's inquiry request and makes a request. Based on the Web services quality based inquiry, the retrieval agent retrieves Web services from the former UDDI information and obtains static and statistical information on the retrieved Web services from a separate database. Then it tests the retrieved Web services on real time to obtain dynamic information and finally produces the WS-QDL. It is used to rank the results according to the consumer's service quality based inquiry and return the result to the consumer. This process of retrieval is executed through collaboration between agents by applying the multi-agent method. When the retrieval results are sent back to the consumer agent, the consumer agent shows the appropriately ranked information to the retrieval requestor (consumer) and receives feedback. Then the consumer can retrieve the best Web services by testing the execution duration of the Web services optionally, which are ranked according to the former quality information. The consumer agent records the information of consumer's feedback and applies it in the statistical database.

5. Conclusion and Future Studies

The UDDI is a Web services registry for registering and retrieving Web services. It plays the role of the Web services market infra. However, being more a registry than a repository, the quality of stored Web services is not fully reliable. Due to this, the study suggests that the Web services quality factors that is represented by WS-QDL and proposes a new UDDI model that is capable of Web services quality based retrieval by using the represented quality information.

For future studies, the UDDI model proposed in this study will be implemented and tested for verification. Furthermore, a DAML-S-based Web services quality description language can be developed in order to include the service provider's service operation principles. Factors in consideration and as a result extend the proposed UDDI model into a transaction framework that supports Web services quality based commerce. The factors mentioned here were excluded in the present study.

6. References

Net-Centric Human-Robotics Operations
Integration of Business/Military Process Management, Grid Computing, and Robotics

Jayson T. Durham and William C. Torrez
Space and Naval Warfare Systems Center (SSC-SC)
San Diego, CA 92152-5000, USA
jtdurham@spawar.navy.mil, william.torrez@navy.mil

James E. Manley
MITRE Corporation
San Diego, CA 92152 USA
jman@mitre.org

Abstract—An effort is underway to more fully integrate Business/Military Process Management (BPM) and Grid Computing with distributed Robotics and Automation capabilities. This work is a logical extension of BPM whereby process and workflow management are extended to include the configuration and reengineering of both human and software agents that collectively synchronize and coordinate both human and robotics driven activities.


I. INTRODUCTION

Information systems and their operational application play a major role in today’s daily activities, spanning from military operations across the globe [11-14] to almost all business operations throughout the world-wide business community. Within this larger context of military and business operations, robotics continues to play an ever increasing role in improving the overall quality of products and services, while continuously improving productivity. As the Internet and related global networking capabilities become more pervasive, cost-effective distributed computing resources will continue to fuel this emergence of hybrid human-robotics operations that support such improvements in organizational operations.

A number of emerging capabilities, e.g., process/workflow management [1-8], web services [9-10], network centric operations [11-14], grid/distributed computing [15-19], robotics and automation [20-31], are providing an unprecedented infrastructure for connecting otherwise isolated computing resources and commercial activities. Ironically, human-centered and robotics-centered activities are not being developed, managed, and utilized as integrated and interoperable resources. For network-centric operations, where the operational activities are a seamlessly highly-synchronized networked configuration of both human and robotics platforms, the execution of such human-robotic operational processes requires this much-needed level of interoperability and compatible infrastructure for supporting a combination of both human-intensive and robotics-intensive operations.

In addition to the need to align and standardize the existing resources and components of today’s ad-hoc net-centric human-robotics operations, new developments are necessary. The ability to dynamically compose higher levels of autonomy from the configuration of “lower levels” of autonomy is a critical high-payoff feature and benefit of human-robotics operations, but no such operational capability exists and is widely available today. The effort described herein concentrates and focuses on developing this ability to dynamically compose and manage the resources necessary for human-robotics operations. This combined human-robotics operations is called Composable Human-Robotics Operations (CmHRO).

II. CONTRIBUTIONS

A. Infrastructure Support

To enable this new level of integration and emergent behavior, computational challenges are being addressed. In particular, the global computational infrastructure (e.g., Internet, Grid, etc.) must provide real-time fault-tolerant computing services for time-sensitive operations. Such services are not yet fully operational and Commercial-Off-The-Shelf (COTS) capabilities do not exist. Work is currently underway to develop and standardize such much needed capabilities, while also developing strategies for working around such issues.

B. Tools, Techniques, and Procedures

The current CmHRO effort is working to create a synthesis and federation of tools and techniques, combining contributions of various disciplines. The current success of highly automated distributed operations is based on the fundamental work done in otherwise diverse disciplines. Robotics and automation, web services, grid computing, computational intelligence, networking, WWW-Internet, wireless communications, process management, workflow management, distributed databases and distributed object standards form a mix of technologies and standards for the development of Composable Human-Robotics Operations (CmHRO) and associated applications. By bringing these otherwise separate human-intensive and robotics-intensive activities together, more broadly and easily accessible resources will be available for quickly configuring more advanced and cost-effective operations that seamlessly compose human-robotics operations.
III. CONCLUSION

Composeable Human-Robotics Operations integrates human-centered Business/Military Process Management (BPM) operations with grid computing and dynamically composeable robotics capabilities. This work is a logical extension of BPM that extends process and workflow management to include both human and software agents that ultimately manage operations for the purpose of synchronizing and coordinating both organic and electromechanical-based capabilities.

ACKNOWLEDGMENT

The authors wish to thank our other team members (Jennifer Padgett, Fred McDonald, Lois Yu, and Weber Lin) of the SSC-SD/MITRE Advanced Military/Business Process Management Concepts and Exploitation Team (AMCET) for their hard work, dedication, and commitment to military process modeling. This work was supported in part by internal SSC-SD initiative.

REFERENCES


An Approach for Parallel Execution of Web Services

Alp Kut, Derya Birant
Department of Computer Engineering, Dokuz Eylul University 35100 Izmir, Turkey
alp@cs.deu.edu.tr, derya@cs.deu.edu.tr

Abstract
This paper presents a model which combines the processing power of parallel computation with the ease of web service usage. In this model, parallel programming environment can be embedded in a visual environment. Parallelization of Web Services is provided by using multi threading technology with dataset parameters. This work also provides parallel usage of computers located in different places via a wide area network such as internet.

1. Introduction
The problems such as image processing, matrix manipulations, sorting large lists problems generally require too long processing time when executing in sequentially. For this reason, they need parallel processing solutions because of decreasing processing time in practice.

The next generation of Internet applications is yielding to web services. A web service is a service that can be published, located, and invoked across the web. After adding a web reference to a project, any methods provided by the indicated web service can be used within the application. [1][2][4]

This study proposes parallel programming by using web services. Web services are parallel in their nature and it can be embedded in the visual language and interactive environment for parallel and distributed programming. [3] There are many reasons as to why web services will become popular in the modern parallel and distributed systems:

- It is possible for web services to be written in any programming language. Web services are supported on all types of clients (Java, phones, Microsoft .NET)
- More than one web services can be executed at the same time on different machines.
- The concepts behind Web Services are very easy to be understood.

- The system becomes easily scalable.

2. Material Method
The model contains three different environments, the first one consists of client side(s), the second one consists of application server and the third part consists of a set of computer which each of them has a web service for parallel processing. When the application site gets a request from the client, it carries out the incoming requests by distributing the work over web service processes. Input data are transferred partially or completely to the web services that process the data and return partial results to the application after they have finished processing. Each service process independently carries out standard functions: connection establishment, reception of a request, processing data, answering the request, connection closing. (Figure 1)

![Figure 1: The model's execution diagram](image-url)
A process can create one or more threads to execute a portion of the program code associated with the process. In order to provide parallel execution, a new thread is firstly created and started for each web service. Then each thread executes indicated function of the web service with the argument list.

Normally execution of web services runs in sequentially, although they are located in different machines. However using multi-thread technology solves this problem and gives the possibility of parallel execution.

3. Implementing a Case Study

Our work includes a case study’s screen shots for explaining how these parallel web services work. Figure 2 shows initial descriptions for two thread structures for parallel execution of two different web services. This description may be used for more than two parallel web services with adding new description lines for new threads. After description of multi-thread environment, the second step contains necessary calls for execution.

```vbscript
Dim Tasks As New WebServiceClass
Dim Thread1 As New System.Threading.Thread(AddressOf Tasks.Task1)
Dim Thread2 As New System.Threading.Thread(AddressOf Tasks.Task2)

Tasks.StrArg = "Some Arg" ' Set a field that is used as an argument
Thread1.Start() ' Start the new thread.
Thread2.Start() ' Start the new thread.
```

Figure 2. Description of multi-thread environment

Multi-thread environment also includes starting points of each web service call. Figure 3 indicates the basic structures of each multi-task procedure for activating web service of count 2. The number of web services can easily be increased just adding new task modules like in figure.

```vbscript
Class WebServiceClass
    Friend StrArg As String
    Friend Result As Boolean

    Sub Task1()
        Dim DS1 As New DataSet
        -- Read partial part of data from database DS1
        Dim a = New WebReference1.Service1
        Dim br = a.QuickSort(DS1)
        PetVal = True ' Set a return value in the return argument.
        End Sub

    Sub Task2()
        Dim DS2 As New DataSet
        -- Read partial part of data from database DS2
        Dim a = New WebReference2.Service1
        Dim br = a.QuickSort(DS2)
        PetVal = True ' Set a return value in the return argument.
        End Sub
End Class
```

Figure 3. Usage example of multi-thread application

This example work is developed by Microsoft’s .NET 2003 version Visual Basic language. [5] In our case study Quick-Sort Algorithm is used as a web service for each computer.

4. Conclusion and Future Works

This work shows a model for parallel usage of web services. For handling this case threads are used. Using virtual environment with parallel web services, wider usage of parallel programming can be provided without knowledge of parallel programming tools. And this tool also provides usage of algorithms in different programming languages.

Our future work will include automatic configuration of remote computers for providing more efficient usage of parallel web services execution. For handling these studies public interfaces and bindings of web services in XML format will be reconfigured for automation.

References


A Web Service Composition and Deployment Framework for Scientific Workflows

Ilkay Altintas  Efrat Jaeger  Kai Lin  Bertram Ludaescher  Ashraf Memon
San Diego Supercomputer Center, University of California, San Diego
{altintas, efrat, klin, ludaesch, amemon}@sdsc.edu

Abstract
This poster presents the web services framework in the Kepler scientific workflow system and illustrates them with a real-world example.

1. Introduction
Kepler [1] is a system for the design and execution of scientific workflows. It is built on top of the Ptolemy II system, a modeling and design tool for assembling concurrent components by means of various models of computation [2].

In a variety of disciplines (e.g., geology, chemistry, biology, ecology) scientists need adaptable interfaces and tools for accessing scientific data and executing complex analyses on the retrieved data. Such analyses can be modeled as scientific workflows. While traditional business workflows are oriented towards document processing, task management, and control-flow, scientific workflows typically are data-intensive and/or computationally expensive, dataflow-oriented, and often involve data transformations, analysis, and simulations. Kepler is unique in that it seamlessly combines high-level workflow design with execution and runtime interaction, access to local and remote data, and local and remote service invocation. Other unique features are inherited from the underlying Ptolemy II system, e.g., the ability to combine different models of computations in a clean way.

2. Web Service Components in Kepler
Kepler’s web service components allow scientists to utilize computational web service resources and web service accessible data sources in a distributed scientific workflow. This functionality can be achieved by entering a single WSDL location or by specifying a web service repository (e.g. a UDDI repository or a web page with links to WSDL descriptions). For the latter, a search term can be specified so that only the matching web services are “harvested” and plugged into the system. Grid extensions to Kepler are described in [3].

2.1. Web Service Actor
Computational units in Kepler are called actors, which are reusable components that communicate with each other via input and output ports. We have implemented a generic web service actor that is used as a web service client in distributed workflows. Using this actor, any application that can be deployed as a web service, can be used as a Kepler actor.

![Figure 1. Example instantiation of a web service actor](image)

The WebService actor, as indicated in Figure 1, provides the user with a simple plug-in mechanism to execute any WSDL-defined web service. The user can instantiate the generic web service actor by providing the WSDL URL and choosing the desired web service operation. The actor then automatically specializes itself and adds ports with the inputs and outputs as described by the WSDL. The so instantiated actor acts as a proxy for the web service being executed and links to the other actors through its ports.

2.2. Web Service Harvester
Kepler provides a web service harvester capability for importing web services from a repository. This feature was developed for conveniently plugging in a whole set of (possibly related) services. The web service descriptions
to be imported can be harvested from a simple web page or retrieved from a UDDI repository. Once imported, the web services are saved as actors. These actors can then be reused in different scientific workflows.

3. Deployment of Workflows as Web Services

The Kepler environment allows users to view and execute workflows on a web page using applets. This enables using the Kepler engine with no installation requirements. In order to use workflows execution results within other applications or execute compute intensive tasks on higher performance machines, Kepler also comes with a mechanism, to deploy workflows as web services.

Several obstacles are encountered while attempting to deploy a workflow as a web service. First, most of Kepler’s output means use a separate display window. Second, no user interaction is available using web services.

We overcome these obstacles using three different deployment models. The first model assumes no user interaction or separate display window. The deployment of a workflow requires deployment of a small execution engine with the workflow. The second model assumes no user input. We provide Kepler with two execution modes: local and remote. In a remote mode all outputs are streamed into a file instead of a window channel. The mode is set either by the director or by each actor individually. The third model allows user input. A special interaction actor is added before each user interaction actor. This actor, when reached, returns the control to the client to provide the necessary input along with a session id pointing to the execution thread (the resume point).

4. Scientific Workflow Example

The current web service components of the Kepler system have been used in various scientific domains, including molecular biology, geosciences, and ecology.

One example is the “Geological Map Information Integration Workflow” depicted in Figure 2. This workflow was designed by a geologist to integrate State Geologic Maps using rock and geologic age ontologies. This model demonstrates the use of distributed processes within a workflow. See [4] for details of this workflow.

The application workflows show how to employ Kepler’s web service components to compose distributed scientific workflows. Since web services are often not designed to fit, data transformations between the outputs of previous steps and inputs of subsequent steps are usually required. For this purpose, specialized data transformation actors (e.g. XSLT, XQuery) have been implemented. User interaction and workflow output are performed via a browser actor.

5. Future Plans

Based on our experiences with workflows from various application domains, new features are being added to Kepler. Currently, the web service actor operates only for basic XML Schema datatypes. Research and planned development activities include extensions to the Ptolemy II type systems to allow for semantic types, optimization of web service quality, fault tolerance and fail over, and last not least recent Grid developments such as WSRF.

Acknowledgements. Kepler includes contributors from SEEK [5], SDM Center-SPA [6], Ptolemy II [2] and Geon [7]. Work supported by NSF ITRs 022567 (SEEK), 0225673 (GEON), DOE DE-FC02-01ER25486 (SciDAC/SDM), and DARPA F33615-00-C-1703 (Ptolemy).

6. References