A QSQL-based Collaboration Framework to Support Automatic Service Composition and Workflow Execution

Kaijun Ren[1,2], Jinjun Chen[2], Nong Xiao[1], Weimin Zhang[1] ,Junqiang Song[1]

[1]College of Computer, National University of Defense Technology, Changsha, Hunan 410073，P.R. China
renkaijun@nudt.edu.cn
[2]Centre for Information Technology Research, Swinburne University of Technology, Melbourne 3122, Australia
jchen@swin.edu.au

Abstract

In high performance computing field such as climate, biology, we often need to integrate resources across distributed, heterogeneous, and autonomous systems to enable e-scientists to solve complex scientific problems in collaborative way. However, current resource (service) collaboration methods still suffer from either low efficiency for automatically building a composition plan because of the involved ontology reasoning and manual processing, or lacking of flexibility for resource's sharing to support the execution of such composition plans. In this paper, we present a QSQL-based collaboration framework to support automatic service discovery, composition and execution. Our proposed method has the following two distinguished characteristics. First, for a given query, abstract composition plans can be automatically created basing on QSQL without much ontology reasoning. Secondly, concrete service instances can be dynamically bound to abstract service composition plans at runtime by considering multiple non-functional factors. Totally, our proposed method will not only facilitate e-scientists quickly create composition plans from a large scale of service repository; but also make resource's sharing more flexible.

1. Introduction

In high performance computing field such as climate, biology, we often need to integrate resources across distributed, heterogeneous, and autonomous systems to enable e-scientists to solve complex scientific problems in collaborative way. However, current resource (service) collaboration methods still suffer from either low efficiency for automatically building a composition plan because of the involved ontology reasoning and manual processing, or lacking of the flexibility for resource's sharing on supporting the execution of such composition plans. For example, there exist many different research efforts aimed at automating service composition. Especially, semantic service compositions that take semantics of services into account to automatically solve the discovery and composition problem, have been an recent active research field [1, 2, 3, 4, 5]. However, despite the merits and the importance of semantic information contained by services, some drawbacks existing in most semantic-based composition methods have prevented them moving forward. One drawback is the low efficiency brought by the direct reasoning algorithm. For instance, the paper[6] provided a hybrid match method based on the direct reasoning for OWL-S[7] described services. The provided examples contain 582 services, 29 query requests, the average response time for each query is about 8 seconds when being simulated in the computer with 2.4G cpu, 1024M memory. Besides, involving a large number of manual processing can be regarded as another weakness. Finally, in high performance computing fields, service collaboration and sharing still face some other solid problems. For example, the traditional meteorological application programs are often bundled with special hardware resources or platform-dependent, which mean that these programs are only able to be executed in specialized grid nodes. As such, even if other grid nodes are free, the user-selected meteorological application service can only be responded and processed by those grid nodes where those special meteorological programs resided. As a result, grid resources cannot be shared and collaborated efficiently and flexibly.

For the above issues, in this paper, we present a QSQL-based collaboration framework to support automatic service composition and execution. With this framework, QSQL (Quick Service Query List) where the important reasoning relationships among ontology concepts and the published service information have been stored can make sure the quick query response during service discovery. Further, a QSQL- based

978-0-7695-3177-9/08 $25.00 © 2008 IEEE
DOI 10.1109/GPC.WORKSHOPS.2008.24

collaboration framework can facilitate e-scientists quickly and intelligently construct abstract service workflow. In addition, the detached technologies between grid service instance and abstract service function description in this framework can also make grid resource and service sharing more flexible. Finally, the experiment demonstrates that our proposed method is not only feasible, but more efficient and applicable.

The remainder of this paper is structured as follows. Section 2 presents the principle for QSQL-based collaboration framework. Section 3 presents the simulation and evaluation. Section 4 discusses the related work. The final section presents the conclusion and future work.

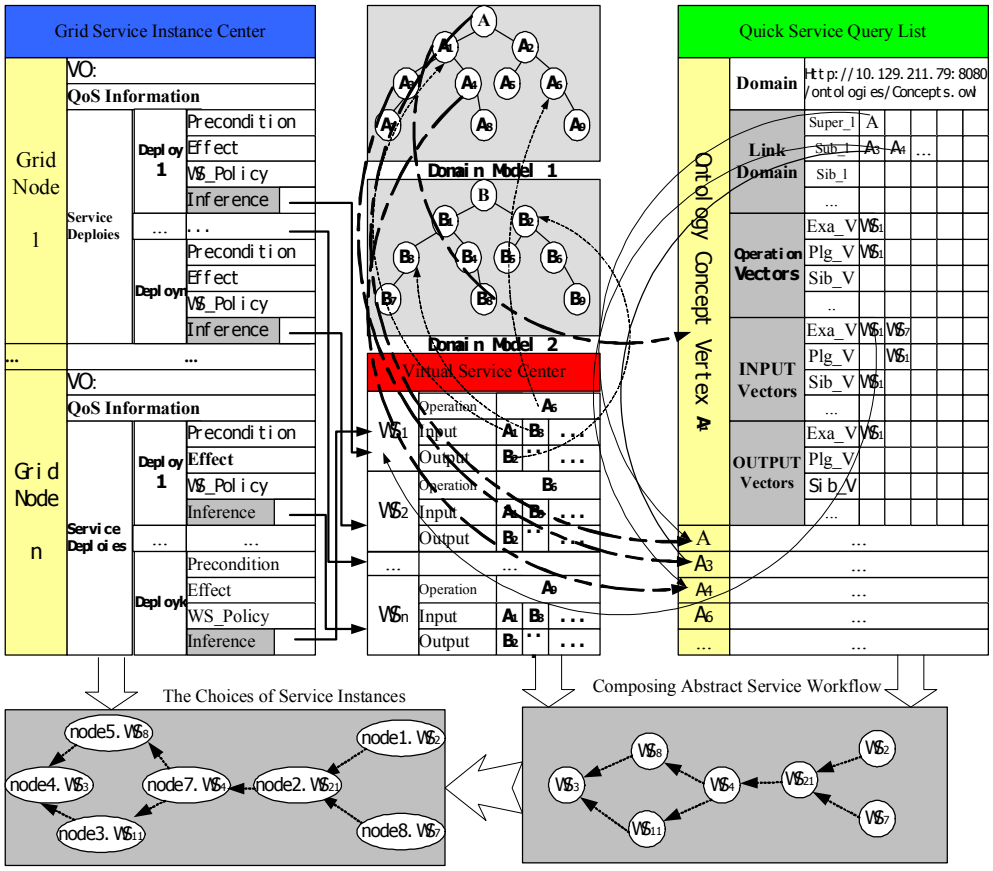

Figure 1. QSQL-based collaboration framework

2. QSQL-based Collaboration Framework

2.1. Framework Description

Figure 1 shows our proposed collaboration framework. With this framework, when service providers advertise their services, the produced wsdl documents will be recorded in a virtual service center. Especially, these wsdl documents will be departed into two parts. One part represents service functional description mainly including inputs/outputs and operations; and the other part represents concrete service instances which mainly includes non-function properties. Service functional description will be annotated by semantic information such as adding ontology concepts to their inputs/outputs parameters by semi-automatic methods[8] (as shown in the middle of Figure 1). Currently, many semantic tools and methods have been proposed to help annotate semantic information to services[5, 9, 10, 11] according to the semantic similarities[12, 13]. Further, these semantic-annotated services will be published to QSQL by service publication algorithm for forming a quick service index list. The upper right part of Figure 1 shows the dynamically built QSQL (we will discuss this in section 3). The upper left part of Figure 1 is the registration center of grid service instances. The main aim of this center is to make grid resource's collaboration more flexible. First, in grid environments, when service providers advertise their services, these services will probably be deployed in many grid nodes. Thus, the same service can be executed in multiple grid nodes. Therefore, we need an efficient method to

Table 1. The definition of input/output data vectors for ontology concept in QSQL

Notes: A_i :an ontology concept; $WS(I_i, O_i)$: an abstract service model; I_i : the collection of input parameters of WS_i; O_i :the collection of output parameters of WS_i; UID:unique Identification of WS_i	
A_i.Input_Exact_vector	If $\exists C_x \in I_i$, s.t. $A_i \xleftarrow{has\text{-}subclass} C_x$, or $C_x \xrightarrow{has\text{-}subclass} A_i$, then WS_i.UID $\in A_i$.Input_Exact_vector
A_i.Input_Plugin_vector	If $\exists C_x \in I_i$, s.t. $C_x \xrightarrow{has\text{-}superclass} A_i$, then WS_i.UID $\in A_i$.Input_Plugin_vector
A_i.Input_Sib_vector	If $\exists C_x \in I_i$, s.t. $A_i \xleftarrow{has\text{-}subclass} C_x$, then WS_i.UID $\in A_i$.Input_Sib_vector
A_i.Input_Grapar_vector	If $\exists C_x \in I_i$ s.t. $A_i \xrightarrow{has\text{-}grandparent} C_x$, then WS_i.UID $\in A_i$.Input_Grapar_vector
A_i.Input_Grachd_vector	If $\exists C_x \in I_i$, s.t. $A_i \xrightarrow{has\text{-}grandchild} C_x$, then WS_i.UID $\in A_i$.Input_Grachd_vector
A_i.Output_Exact_vector	If $\exists C_x \in O_i$, s.t. $A_i \xleftarrow{has\text{-}subclass} C_x$, or $A_i \xrightarrow{has\text{-}subclass} C_x$, then WS_i.UID $\in A_i$.Output_Exact_vector
A_i.Output_Plugin_vector	If $\exists C_x \in O_i$, s.t. $A_i \xrightarrow{has\text{-}superclass} C_x$, then WS_i.UID $\in A_i$.Output_Plugin_vector
A_i.Output_Sib_vector	If $\exists C_x \in O_i$, s.t. $A_i \xleftarrow{has\text{-}subclass} C_x$, then WS_i.UID $\in A_i$.Output_Sib_vector
A_i.Output_Grapar_vector	If $\exists C_x \in O_i$, s.t. $A_i \xrightarrow{has\text{-}grandparent} C_x$, then WS_i.UID $\in A_i$.Output_Grapar_vector
A_i.Output_Grachd_vector	If $\exists C_x \in O_i$, s.t. $A_i \xrightarrow{has\text{-}grandchild} C_x$, then WS_i.UID $\in A_i$.Output_Grachd_vector

decide which grid node will more suitably execute user's requirement. For this aim, in our methods, we developed and designed grid service instance center which will offer important information for dynamically binding concrete service instances to abstract service model by the scheduling algorithm. The basic working procedures are as follows. First, all gird nodes, where concrete service implementations of some abstract service models in virtual service center have been deployed in advance, should be registered in this center. Second, the execution conditions of service instances such as precondition and effects should be simultaneously advertised to this center. Third, some non-functional properties of services such as cost, contracts should also be bounded to this center with corresponding service instances. Finally, the dynamic information of grid nodes including the state of CPU/MEMORY, running processes, job queue needs to be updated periodically. Normally, users care more about whether their needs can be quickly met rather than which grid nodes will response to their requests. Therefore, ideal grid systems should be transparent for users, and our collaboration approach is exactly an embodiment of such requirements. First of all, as shown in the right-hand bottom of Figure 1, e-scientists or users can quickly and intelligently find a single service or more abstract services as a combination to form an abstract service workflow by our provided service discovery algorithm or composition algorithm from QSQL. Second, the resource scheduling and service instance selection algorithm will dynamically determine which grid nodes to execute the corresponding service instances by judging the global information such as QoS information, user-demanded constraints and other information. The left bottom of Figure 1 shows the ideas. Consequently, our QSQL-based solution results in an effective sharing and collaboration of grid resources and services. In the following sections, we will give a detail about the building of abstract service workflow plans and their execution.

2.2. Summary of QSQL

In order to overcome the low discovery efficiency brought by the traditional semantic service discovery algorithm based on direct reasoning, we have proposed a QSQL-based service discovery method in [14]. In brief, QSQL is an efficient service index list which was built dynamically when services were published. In QSQL, the semantic relationships between ontology concepts and published service models can be recorded in the special designed data structures basing on graph storage theory. Such data structures mainly include two parts. One part is the domain of link which is mainly used to avoid repeat reasoning when service models to be published possibly have the same mapped ontology concepts from those previously published service models. Another part is the domain of data which is primarily used to record service information in corresponding INPUT/OUTPUT data vectors such as Exact_vector 、 Plugin_vector 、 Sib_vector 、 Grapar_vector 、 Grachd_vector according to their corresponding semantic relationships. Table 1 gives the formal definition of all INPUT/OUTPUT vectors in the domain of data of ontology concepts.

2.3. Creating Abstract Workflow Plan from QSQL

QSQL can facilitate e-scientists quickly and intelligently construct abstract service workflow. The procedure can be done either in static way or dynamic way.

Static creation: It takes place during design-time, and it is provided for skilled users. Skilled users usually have good knowledge about how to compose service process to finish a complex task. In our QSQL-based framework, skilled user only needs to query some necessary ontology concepts from QSQL. Then, based on these returned concepts, they can construct an inputs/outputs flow. Finally, QSQL discovery engine

will find concrete abstract service models recorded in QSQL to fill out or replace the corresponding inputs/outputs flow according to the before and after the input-output relations. During the whole process, QSQL discovery engine does not need to do any reasoning.

Dynamic creation: It is primarily provided for ordinary users. Usually, it is difficult for these users to build up workflow service composition by manual way like static creation. What they need to do is to give the requirement description. Semantic translator handler will extract the key information such as output, input and type. According to the information, the dynamic discovery model cooperated with QSQL discovery engine module can build up the abstract process flow from QSQL by employing a backward chaining composition algorithm with less much reasoning. Therefore, the design of dynamic composition is transparent to users.

2.4. Instance Selection for Executing Abstract Workflow Plan

When the generated abstract workflow plan is executed in grid environments, the same abstract service model can probably be executed in multiple grid nodes. Therefore, we need an efficient scheduling algorithm to decide which grid node will be more suitable for executing such abstract service model. For this aim, in our methods, we developed and designed grid service instance center which will offer important information for the execution of concrete service instances by schedule algorithm. According to the afore-mentioned description in QSQL-based collaboration framework, for each service instance, the related information such as QoS information should be registered into the center. Generally, such registration information can be achieved from the following sources: service providers, user's feedback and active grid execution monitoring. Service providers may advertise their partial QoS information such as cost, security. The client side can provide user's feedback about their using experience of services such as reputation, response time. Active grid execution monitoring cooperates with grid component, which not only can detect the states of grid node in which the services deploy such as cpu_capacity, memory_capacity, availability, the number of running processes, but can also monitor the states of network such as connection bandwidth and network traffic factors. Our collaboration framework provides both the capturing and the updating mechanisms for QoS information.

In our information registration model, the registered information is divided into two categories, namely obtained information and computed information. Obtained registered information such as cpu_capcity, memory_capcity, scalability, availability, cost, network_bandwith, number_of_processes generally can be captured directly from grid nodes where service instances have deployed previously or from the provider side, or from the network which services depend on. Obtained QoS information needs to be updated whenever they change. Computed registration information is the information which needs to be computed based on the obtained basic registration information.

3. Simulation and Evaluation

Presently, we have built a meteorological grid prototype(http://grid.cma.gov.cn:8080/gridsphere/cmag), which are running across several province in china.

Based on this prototype, we have finished some simulation experiments to prove the performance brought by our collaboration framework. Considering the current lack of meteorological application services, the experiment produced 3000 abstract service models for testing by using the concepts of the six selected ontology domains[15] as the inputs and outputs of all service collections. In addition, we also produced 30 queries for each domain. In our experiment, we mainly finished the comparison and analysis of the performance between QSQL-based collaboration method and the traditional collaboration method based on direct reasoning during service composition.

During discovery of service composition, we produced 180 queries. For each query and each discovery method (QSQL, the traditional direct reasoning, keyword), the response time and the corresponding discovery results have been recorded.

Table 2. Average response time for each query

Discovery Methods	Average Response Time for Each Query
QSQL-based Collaboration	258(ms)
Direct Reasoning Collaboration	9014(ms)
Keyword-based Collaboration	273(ms)

Table 2 shows the average response time by using different discovery methods for 180 queries. As we see, the response time (258ms) by using QSQL-based collaboration method is the lowest. By comparison, the response time (9104ms) by using the traditional direct reasoning collaboration method is the highest. Because the large number of ontology reasoning was processed at service publication stage, the discovery algorithm in QSQL had a quick response to any query request

without any reasoning. Therefore, the response time by using QSQL-based collaboration method is similar to the time by the keyword-based collaboration method. Contrarily, the traditional direct reasoning collaboration method including a great deal of ontology reasoning during discovery period had a long response to process any query request, and the corresponding average response time is even 34 times more than the response time for QSQL-based collaboration method. Accordingly, the collaboration efficiency basing on QSQL has been improved greatly. Figure 2 shows the response time's distribution when the direct reasoning collaboration discovery algorithm continuously processed 20 queries.

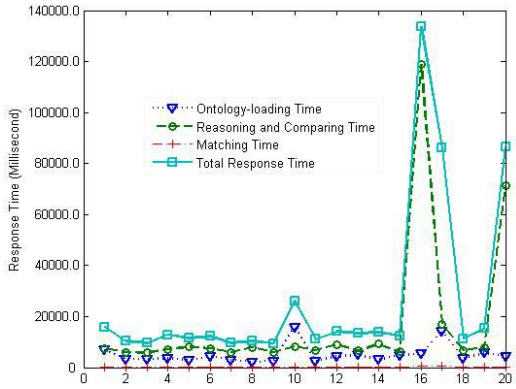

Figure 2. Response time of 20 query requests continuously processed by direct reasoning-based collaboration method

As shown in figure 2, when the direct reasoning collaboration algorithm continuously processed the queries, the response time appeared the increase. This is because when the discovery algorithm continuously processed the queries, the involved large number of ontology reasoning produced more instances, which means they needed more free memory space. However, the RAM memory was limited in our running environment, so the garbage collection engine in JVM had to work to release more memory which cause to more latency.

4. Related Work

Over the last several years there has been substantial progress in building grid applications by composing them from predefined components and web services. The myGrid project is one of Semantic Grid projects and aims to provide a problem-solving workbench for biologists. Taverna is part of the myGrid project[16], focused on building middleware to support data intensive experiments in molecular biology. Taverna has more than a thousand services that can be used as components in workflows. Triana provides an elegant and well tested composition tool and a large toolbox of ready-touse components[17]. For grid application, Triana uses a software layer, called the grid application prototype, to distribute subsystems of the workflow graph to remote grid resources for execution. Kepler takes is based on an actor oriented model that allows hierarchical modeling and dataflow semantics[18]. The Kepler tools support a well designed graphical composition interface that is very intuitive and easy to use. Chen et. al. [19]outline a knowledge-based framework which provides advice as the user constructs a workflow. The system allows the user to store workflows, hence facilitating reuse. Cardoso and Sheth [20] propose a framework which provides assistance to the user by recommending a service meeting the user's needs. This is done by matching the user-specified Service Template (ST) with the Service Object (SO). Vikas et.al.[21] describe a two-step methodology for end to end composition of web services by semantically annotating web service components, their service creation environment can then be used to generate potential workflows for achieving the desired functionality reusing existing web services. Shalil et.al.[22] propose a dynamic and adaptive mechanism for automating the construction of experiment workflows. Verma, K et.al [23] implemented a scalable infrastructure of Web service registries for semantic publication and discovery of services. It is implemented as a P2P network of UDDI registries. The authors in [24] gave an analysis and comparison for the present service composition platforms.

By comparison, our proposed QSQL-based collaboration framework is a little different from the above mentioned methods. QSQL can make a large number of ontology reasoning processed at service publication stage. Thus we can make sure the quick query response and high recall and precision rate without much reasoning during service collaboration discovery. Further, a QSQL-based collaboration framework not only can facilitate e-scientists quickly and intelligently discover services, interact with them and compose scientific workflows, but also can make grid resource and service's sharing more flexible. Therefore, our proposed method is not only feasible, but more efficient and applicable.

5. Conclusions and Future Work

In this paper, we have presented a new efficient QSQL-based collaboration framework to support automatic service discovery, composition and execution. Specially, with QSQL (Quick Service Query List), the large number of ontology reasoning is processed at service publication stage which enables

the quick query response in service discovery. Therefore, our proposed collaboration framework can not only offer the convenience for e-scientists to quickly and intelligently discover services, compose scientific workflows without much reasoning, but also enable grid resources and services to be shared more flexible. Our simulation experiment has demonstrated that our method's advantages over the traditional collaboration methods basing on direct reasoning.

In the future, we will focus on making the QSQL-based collaboration framework work in the prototype of Chinese Meteorological Application Grid.

6. Acknowledgement

We are very grateful for the foundation support by the National "973" Research Plan Foundation of China under Grant No. 2003CB317008 and by National Nature Science Foundation of China under Grant No. 60573135, 40505023 and 60736013.

7. References

[1]Brahim Medjahed, A.B., A Multilevel Composability Model for Semantic Web Services. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2005. **17**(7): p. 954-968.

[2]Keita Fujii, T.S., Semantics-Based Dynamic Service Composition. IEEE Journal on Selected Areas in Communications, 2005. **23**(12): p. 2361-2372.

[3]Ulrich Küster, B.K., Mirco Stern,Michael Klein. DIANE: An Integrated Approach to Automated Service Discovery, Matchmaking and Composition. in the 16th International World Wide Web Conference. May 2007. Banff, Alberta, Canada: ACM Press.

[4]Danny Gagne, M.S., Scott Bennett,Susan Powers. Using Data Semantics to Enable Automatic Composition of Web Services. in 2006 IEEE International Conference on Services Computing. September 2006. Chicago,USA: IEEE computer society.

[5]Katia Sycara, M.P., Julien Soudry,Naveen Srinivasan, Dynamic Discovery and Coordination of Agent-Based Semantic Web Services. IEEE Internet Computing, 2004. **8**(3): p. 66-73.

[6]Matthias Klusch, B.F., Mahboob Khalid,Katia Sycara. Automated Semantic Web Service Discovery with OWLS-MX. in the 5th International Joint Conference on Autonomous Agents and Multiagent Systems. 2006. Hakodate,Japan: ACM.

[7]David Martin, M.B., Jerry Hobbs,Etc. OWL-S: Semantic Markup for Web Services. in http://www.w3.org/Submission/OWL-S. 2004.

[8]Jacek Kopecky, T.V., Carine Bournez ,Joel Farrell SAWSDL: Semantic Annotations for WSDL and XML Schema. IEEE Internet Computing, 2007. **11**(6): p. 60-67.

[9]IBM Semantic Tools for Web Services. Available from: http://www.alphaworks.ibm.com/tech/wssem. Accessed on Jan 21, 2008.

[10]Radiant:WSDL-S/SAWSDL Annotation Tool. Available from: http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=1. Accessed on Jan 21, 2008.

[11]The OWL-S Editor. Available from: http://owlseditor.semwebcentral.org/download.shtml. Accessed on Jan 21, 2008.

[12]M. Andrea Rodrı´Guez, M.J.E., Determining Semantic Similarity among Entity Classes from Different Ontologies. IEEE Transaction on Knowledge and Data Engineering, 2003. **15**(2): p. 442-456.

[13]Jorge Cardoso, A.S., Semantic e-Workflow Composition. Journal of Intelligent Information Systems, 2003. **21**(3): p. 191-225.

[14]Kaijun Ren, J.S., Jinjun Chen, Nong Xiao,Cancan Liu. A Pre-reasoning based Method for Service Discovery and Service Instance Selection in Service Grid Environments. in The 2nd IEEE Asia-Pacific Service Computing Conference. 2007. Tsukuba Science City,Japan: IEEE.

[15]Selected domain Ontologies :people+pets.owl,travel.owl,countries.owl,consciousness 1.owl,koala.owl,generations.owl. Available from: http://protegewiki.stanford.edu/index.php/Protege_ontology_library. Accessed on Dec 21, 2007.

[16]T. Oinn, M.A., J. Ferris, D. Marvin,and Etc., Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics, 2004. **20**(17): p. 3045-3054.

[17]David Churches, G.G., Andrew Harrison, Jason Maassen, Craig Robinson, Matthew Shields, Ian Taylor, Ian Wang, Programming scientific and distributed workflow with Triana services. Concurrency and Computation: Practice and Experience, 2006. **18**(10): p. 1021 - 1037.

[18]Bertram Ludäscher, I.A., Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, Yang Zhao, Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience 2006. **18**(10): p. 1039 - 1065.

[19]G. Nadarajan, Y.-H.C.-B., Translating a typical business process modelling language to a Web Services Ontology through lightweight mapping. IET Software, 2007. **1**(1): p. 1-17.

[20]Jorge Cardoso, A.P.S., Semantic E-Workflow Composition. Journal of Intelligent Information Systems, 2003. **21**(3): p. 191-225.

[21]Vikas Agarwal, K.D., Neeran Karnik,and Etc. A Service Creation Environment Based on End to End Composition of Web Services. in www2005. 2005. Chiba, Japan.: ACM.

[22]Shalil Majithia, D.W.W., and Etc. Automating scientific experiments on the semantic grid. in ISWC2004. 2004.

[23]Rohit Aggarwal, K.V., John Miller, William Milnor. Constraint Driven Web Service Composition in METEOR-S. in 2004 IEEE International Conference on Services Computing. September 2004. Shanghai,China: IEEE computer society.

[24]Zakaria Maamar, D.B., Philippe Thiran, Chirine Ghedira,Schahram Dustdar, Subramanian Sattanathan, Towards a context-based multi-type policy approach for Web services composition. Data and Knowledge Engineering, 2007. **62**(2): p. 327-351.