Distributed execution of aggregated multi domain workflows using an agent framework

Zhiming Zhao   Adam Belloum   Cees de Laat   Pieter Adriaans   Bob Hertzberger
Informatics Institute, University of Amsterdam
Kruislaan 403, 1098SJ, Amsterdam, the Netherlands
{zhiming|adam|delaat|pietera|bob}@science.uva.nl

Abstract

In e-Science, meaningful experiment processes and workflow engines emerge as important scientific resources. A complex experiment often involves services and processes developed in different scientific domains. Aggregating different workflows into one meta workflow avoids unnecessary rewriting of experiment processes and thus improves the reuse efficiency. Remote workflow engines explore the computing power of distributed environment. However, the diversity of workflow description and execution models makes the integration between engines difficult. An agent framework uses ontology based communication language and makes the integration to semantic information of resources seamless, it is thus suitable for coupling distributed engines. In this paper, we present our work in the context of Dutch Virtual Laboratory for e-Science (VL-e) project. A semantic registry for describing workflow engines is implemented, and mobile agents are used to manage distributed workflow coordination.

1 Introduction

In scientific research, workflow systems are emerging as an important mechanism to automate the management and integration of experiment processes. Using a workflow system, a scientist can prototype an experiment by assembling available software tools and services, and execute them via an engine. During the past decade, workflow systems have been utilized in different domains and a number of systems have been developed, e.g., Kepler [1], Taverna [2], Pegasus [3] and VLAMG [4].

The development of a scientific workflow system is heavily driven by domain specific applications; different design requirements make the models for describing and executing workflow processes diverse. Such a diversity hampers the interoperability between different workflow systems. This issue is getting increasingly important when interdisciplinary research is used as a key approach to explore new scientific findings.

Standardizing the description and execution models of workflow processes is recognized as an important step to improve interoperability between workflow systems; industrial standards, e.g., Web Services, and Business Process Execution Language (BPEL) are used to investigate such a possibility [5]. However, the difference between scientific domains makes the standardization of experiment process modeling difficult. Developing a workflow system with consideration of interoperability with other systems is another effort. In the system link initiative [6], developers from several workflow projects discussed how to allow their systems to invoke services developed by other projects. Wrapping the engine of one system as a composite component in another system is an often used approach. Since the systems developed in academic projects are evolving rapidly and lack of stability, in our early work, we argued that aggregating different workflows using a software bus is feasible to realize workflow integration. In [7, 8], we proposed a workflow bus architecture, which can wrap legacy workflow engines using autonomous agents, and couple them in one high level workflow via a runtime infrastructure.

Distributed execution is highly demanded for large scale workflows. Not only the distributed nature of software components and services used in a workflow makes centralized execution infeasible, but also installing engines for different sub workflows into one environment is practically difficult. In this paper, we continue our work on the workflow bus framework and discuss how distributed execution support is provided. The research is carried out in the context of Dutch Virtual Laboratory for e-Science (VL-e) project [9]. The paper is organized as follows. We first give a brief overview on involved issues in workflow aggregation and distributed execution. And then we review our early work on workflow bus architecture, after that we propose an ontology based repository and discuss how mobile agents are used to coordinate distributed workflow execution. Some experimental results will also be presented.

2 Workflow aggregation and distributed execution

VL-e is an e-Science project driven by different application domains; it aims to realize a Grid enabled generic framework via which scientists from different domains can share their knowledge and resources, and perform domain specific research. Effectively reusing existing workflow systems and aggregating needed systems as a meta environment is viewed as a main research mission. In this section, we discuss our early work on a workflow bus architecture and analyze the requirements for supporting distributed execution of aggregated workflows.

2.1 Distributed workflow execution

At different levels of abstraction, e.g., concrete computing tasks or high level experiment processes, the workflow of an experiment is described differently; mapping high level abstraction onto low level concrete processes, and eventually onto executable software components or services are basic way to execute a workflow [10]. There are a number of reasons for supporting distributed execution of a workflow: exploring the computing power over network, parallel workflow execution, and improve fault tolerance. Basically, the distributed execution of a workflow can be distinguished from three aspects:

1. *The execution of workflow processes.* When workflow processes use resources or services at different locations, computing tasks performed by remote components are by nature distributed.
2. *The integration of processes.* Data flow renders the basic dependencies between processes in a scientific workflow; routing the data flow via a centralized engine often results overhead for transferring data between processes and the engine. Decoupling the control information from the scientific data between processes is a basic way, in which the integration between data can be decentralized, e.g., via a storage service.
3. *Workflow coordination.* In addition to data integration, orchestrating the runtime behavior of components is important coordination intelligence for workflow execution. Such intelligence is typically realized inside the workflow engine, and it is centralized in many of the current systems. In distributed execution, a workflow is simultaneously collaboratively controlled via multiple engines, e.g., instances of same or different workflow engines.

Table 1 shows the distributed execution support in some of current workflow systems.

	Process level	**Data integration level**	**Coordination intelligence**
Kepler	Special *actors* interface remote resources.	In early versions, data integration is via centralised engine. Now, tools such as Griddles are being integrated for distributed data integration.	A hierarchical control mechanism is realised, which is based the Ptolemy *director* and *composite actor* architecture. But the support for distributed execution is limited.
Taverna	Web services compliant processes. Service registry is used.	SOAP based data passing. *Styx* based files transfer is used for distributed file accessing.	Centralized. A workflow is enacted and executed via engine, which is currently shipped with the workbench.
DAGMan	Concrete computing tasks.	Tasks are assumed to be relatively independently. Files are the basic data passing mechanism.	Centralized. A DAGMan engine schedules computing tasks to distributed environment. So sophisticated flow control.
VLAMG	Distributed resources are wrapped as VLAMG modules.	A CORBA based port library handles distributed data integration. File based access is also supported.	Centralized. The latest version of VLAMG uses GT4 service to submit workflow description.

Figure 1. Distributed workflow execution in some workflow systems.

From the tale 1, we can see, most systems support distribution execution of workflow components, however the coordination for components is mainly centralized. In Kepler, execution intelligence is explicitly decoupled from workflow description, and encapsulated as different *directors*. However, distributed directors are still in early development. The centralized control intelligence gives limited support for distributed execution of entire workflow, e.g., sweeping parameters using a workflow. This is an important motivation for us to propose a software bus architecture, see detailed discussion in [8].

2.2 A workflow bus architecture

The basic idea of a *workflow bus* is to wrap a number of popular and relative mature legacy SWMSs as federated components, and to loosely couple them as one meta workflow system using a software bus. In the context of *workflow bus*, we call a workflow being executed by a wrapped SWMS as a *sub-workflow*, the workflow being coordinated by the *workflow bus* as a *high level workflow* or a *workflow* in short. The execution of a *workflow* is called a *study*, and the execution of a sub-workflow is called a *sub-study* or a *scenario*. As a runtime infrastructure, the workflow bus has to provide basic services for interpreting and scheduling meta workflows, for orchestrating plugged legacy workflow engines, for passing and distributing data between workflow engines, and for supporting user interaction with workflows. From the system level point of view, the partial functionality from different systems are then aggregated and complemented as one meta system.

2.3 Requirements

To couple cross domains workflows via a workflow bus, a number of issues have to be taken into account:

1. *An integration platform* which couples distributed sub workflows. Currently, many workflow systems have decoupled engines which can be executed without going through the GUI component; however the invocation interface for the engine varies: as a local command line e.g., Kepler and Taverna, as set of API, e.g, Triana, or web service, e.g., VLAMG. A proper platform which can hide low level integration details between engines is preferable.

2. *A registry* which records information of installed workflow engines. In a meta workflow, sub workflows are *composite* processes which need to be executed by proper legacy engines. The installation of these engines is often too heavy to be shipped at runtime as part of computing jobs. Instead, using pre-installed engines over e-Science environment is a practical solution.

3. *An ontology* which serves as the basis for workflow integration and information provenance. When coupling workflows developed using different systems, synchronizing the ontology of different systems is essential for integrating data passed between them and for providing data provenance.

In the next section, we will discuss how we solve these problems in the current workflow bus prototype.

3 Solutions in VLE-WFBus

In [8, 11], we discussed how agent technologies are used to prototype the workflow bus: VLE-WFBus. JADE (Java Agent DEvelopment Framework), a FIPA compliant framework [12], is used to prototype a number of agents:

1. *Scenario managers* for wrapping legacy workflow engines; scenario managers provide interface to exchange information via the agent framework. Depends on the API of the legacy workflow engines, a scenario manager can control the legacy engine via different interfaces, e.g., Web Service, Socket, or command line.

2. *Study manager*, an agent for interpreting meta workflow, assigning tasks to *scenario managers* and coordinating *scenario managers* at runtime.

3. *User interface*, the composition and monitoring of workflow are realized as user interface which is decoupled from Study and scenario managers, but can communicate with them via the agent framework.

In the framework, a registry which has information of location of workflow engines is utilized, as shown in Fig. 2. Before we discuss how distributed execution is supported, we first give an overview on how the workflow bus works.

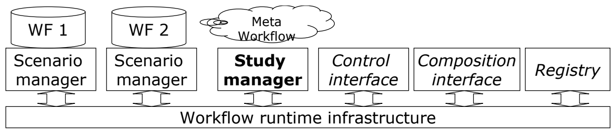

Figure 2. The runtime infrastructure of workflow bus.

3.1 A basic runtime scenario

At runtime, a study manager receives meta workflow from a GUI agent or other composition services; it interprets the workflow content and starts proper scenario agents for different sub workflows. Before executing the sub workflow, a scenario manager first checks registry and locates suitable installation of the workflow engine. The scenario manger decides whether it should migrate to the node where the engine is installed. A scenario manager uses the information from registry to invoke the workflow engine and execute the sub workflow. Fig. 3 shows the basic sequence diagram.

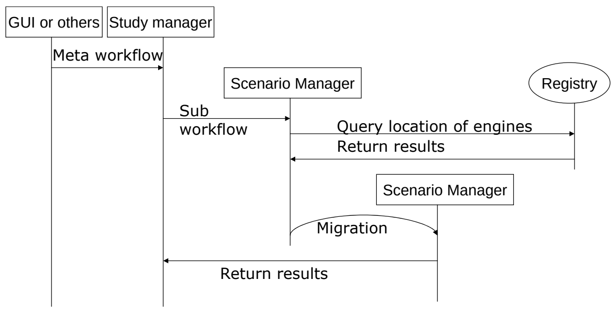

Figure 3. The sequence diagram of a simple runtime scenario.

The communication between agents are handled by the Message Transport Services (MTS) provided by the Jade agent platform; Agent Communication Language (ACL) models agent conversation using basic parameters, e.g., sender, receiver, content, ontology, encoding and reply [13]. Jade explicitly models the semantics of the message contents being communicated between agents as ontology, which differs from the earlier syntactic based interface description languages, e.g., IDL or WSDL in CORBA and Web services. A seamless interface between Ontology and the runtime java class is implemented in Jade for handling schemas of the communication messages. An important advantage is that such standardized ontology language also promotes the feasibility for the semantic based integration with other e-Science services, e.g., data storage and semantic discovery. A workflow bus ontology is developed.

3.2 A workflow bus ontology

Based on the several workflow related taxonomy and ontology [14, 15], we proposed an workflow bus ontology to model the concepts of workflow bus and the interaction between workflow bus agents. A meta workflow is modeled as *sub-workflows* and dependencies, called *relation links* between sub-workflows. A meta workflow and sub workflow is a sub class of generic concept *workflow*. A workflow has a content which can be accessed via an *access point*, and has property *in language* to indicate how it is described. A workflow has input, output and configuration parameters; the input, output and configuration parameters are sub classes of generic concept data. A data concept also has property *access point*. We did not try to model a generic ontology which can cover concepts used in different workflow systems. So far we consider them as black box, which can be can be accessed via the access point of its content. In future, we plan to link the specific ontology defined from that specific system.

The interactions between *scenario managers* and *study manager* are modeled as a set of basic agent actions, e.g., *assign workflow*, *register in workflow bus*, *update execution states* and *migration* are defined. These actions definitions will be mapped as content of messages exchanged between agents. Fig. 4 shows the basic definition of the ontology.

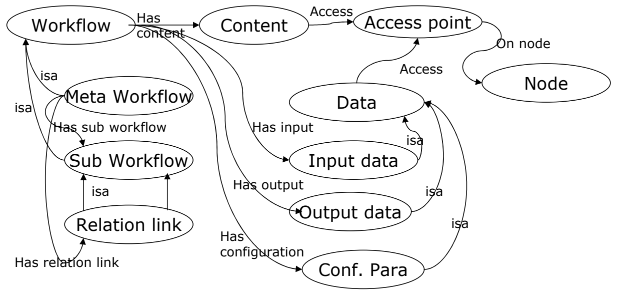

Figure 4. Partial ontology defined in workflow bus.

Using the ontology, a *study manager* can send sub workflows to different *scenario managers*.

3.3 Workflow engine discovery and registry

In the ontology, the workflow engine properties and the invocation interfaces are described. Several concepts are defined: *software package*, *workflow engine*, *invocation interface*, *access point*, *data*, and *parameter*. A *software package* has several properties: *access, a name, invocation method*, and *dependence*. The *workflow engine* concept inherits the software package properties and adds several new ones: *supported workflow descriptions*. *Invocation interface* has two sub class so far: *command line interface* and *web service interface*. Fig. 5 shows the related concepts.

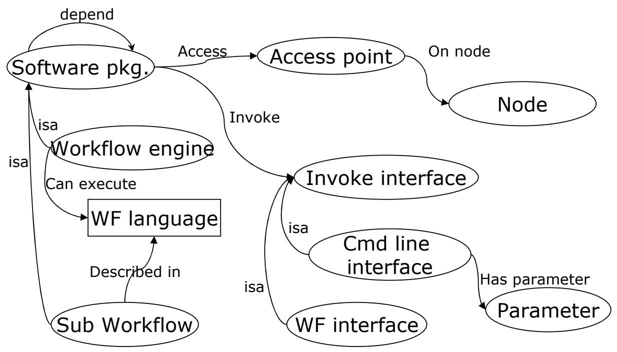

Figure 5. Part of the workflow engine related concepts.

This ontology is used to annotate the registry information and for agents to discover engines. Currently, a sesame engine [16] is used. A mysql data base is used as the backend repository. Since the ontology is also used to generate the communication interface for agents, thus the ontology for study and scenario agents is synchronized with the registry.

3.4 Agent based distributed execution

Using the registry information, a scenario manager determines the access point of the engine. Initially, the scenario manager is started at the same machine where the study manager starts. A scenario manager checks if the engines is executed by command line or web services. Agent can decide whether it needs to migrate. The decision is made based on several issues, e.g., the load of the machine where it is initialized and the location of the engine with which it will interact.

In JADE, migration is implemented as services of agent framework. Basically, it contains three steps: save states, do migration and restore states. The Agent Management Service (AMS) provides services for requesting migration. Via the AMS service, a scenario manager can also know if there is an available agent container on the machine.

3.5 Current prototype and experimental results

Currently, a registry is prototyped based on sesame 1.x. The workflow bus ontology is described using OWL[1]. The JADE 3.4 is used as basic framework. Several scenario agents are currently developed, which include Kepler, Tavernal and Triana.

In the implementation, a scenario agent is customized by extending the property of a *generic scenario* agent for interpreting invoking interfaces and generating proper call for remote engine. To keep the basic service in workflow generic and open; the description and execution model of meta workflow is decoupled from the code of study manager. The study manager can recognize new model of the meta workflow description by adding new *plug in* components.

Using the current prototype, we investigated the performance characteristics of agents and the registry. We measured the average time cost for a scenario agent to query engine information from the registry. Meta workflows which have 1, 2, 4, 8 and 16 identical sub workflows are made as test. The registry service was deployed in a node in local experimental network with a Pentium 4, 2.8 GHz CPU, 1.5 GB physical memory, and the scenario agents are executed in the super computer of the Dutch ASCI research school (DASII) [17]. Both sides use Linux systems; the connection between DASII and the local node is 100M bits. We investigate how the query time changes when the number of agents increases. In the measurement, the study manager creates different numbers of scenario managers, and the average time for all scenario managers are shown in Fig. 6. The error bars are standard deviations.

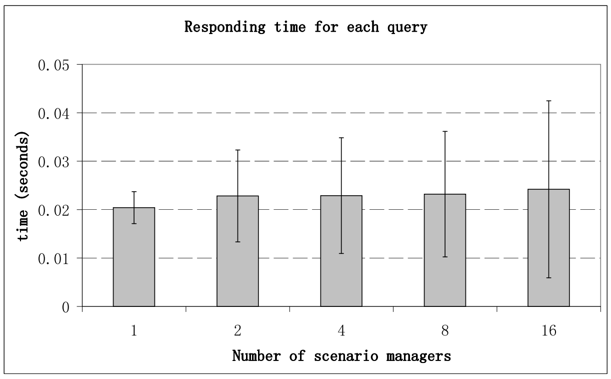

Figure 6. Query respond time for registry.

From the measurement, we can see, the average time cost for query gets high deviations when the simultaneous query increases, however the average time costs remain close. So far, the size of the test repository only contains 5 workflow engine instances (see the ontology); when the number of triples in the registry increases, it might also influence the query time. Nevertheless, from the experiment we can see the average overhead for each query is acceptable.

[1]The ontology is available at http://staff.science.uva.nl/∼zhiming/Ontology/Workflowbus.owl.

4 Use cases in VL-e

We are currently investigating the applicability of the workflow bus in several use cases; one of them is the medical imaging domain as proposed by Olabarriaga et al. [18] in the VL-e Medical Imaging and Diagnosis subprogram. The goal is to integrate a number of different workflows to support the development, evaluation and deployment of medical image analysis (MIA) applications along their complete lifecycle:

1. during the *development* of new MIA applications, an interactive prototyping environment is adopted for a small set of images. The DElft Visualization and Image processing Development Environment (DeVIDE [19]) will be used in this phase;
2. during *evaluation*, the method is applied repetitively to large amounts of images on a Grid environment. Nimrod [20] and VLAMG will be used in this phase; and
3. for *deployment* of the MIA application in the clinical routine, it must be inserted in the workflow and the information systems of the radiology department of a hospital. In this phase, the Distributed Workflow Management System for automated medical image analysis and logistics (DWMS [21]) will be used.

In addition to these workflows, services for storing and accessing large scale medical images, e.g., using Storage Resource Broker (SRB), will also be important components to be integrated with the workflow for supporting research activities in the Amsterdam Medical Center (AMC). The four candidate engines for supporting workflows during the lifecycle of a MIA application have different requirements on execution and user interactions: DeVIDE is more user interaction oriented, VLAMG and Nimrod are good at handling massive Grid computing tasks, and DMWS is adequate for managing MIA tasks at the hospital level. Integration of these systems will enable facilitated development and maintenance of MIA applications by reducing the current overhead for preparing the application to run in the different environments.

Since these engines have different workflow description models, considering them as black box with the workflow bus approach will promote a clean coupling paradigm as illustrated in Fig.7. The engine of different workflow engines are wrapped using scenario managers: a DeVIDE scenario manager, a VLAMG scenario manager, a Nimrod scenario manager, and a hospital workflow engine scenario manager.

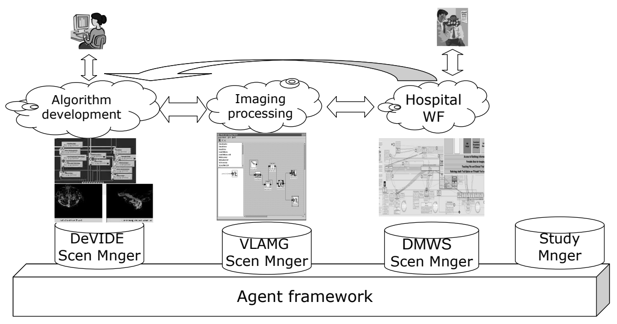

Figure 7. Workflow bus for integration of systems in the medical use case.

5 Discussion

In this section we will discuss the related work of the workflow bus from three aspects: agent technologies, semantic enhancement of registry and distributed workflow execution.

Agent based approaches, e.g., modeling and engineering or implementing part of the control intelligence of a system as agents, have shown attractive possibilities for realizing scientific workflow management and e-Science framework [22]. Compare to deliberative agents, the Jade agent framework does not focus on sophisticate symbolic reasoning on agent behavior, instead it provides a cross platform Java based distributed environment which extends normal distributed OO based middleware by adding features for scheduling activities and for managing agents. However, from the discussion, we also see limitations of the Jade framework. The simple model of Jade agents makes the agent development easy; but it also requires effort for including intelligence for complex control. Jade agents have limit support for intelligent control at low level; the ACL based messaging mechanism is more suitable for coordinating system level behavior. But if compared to the other available platforms, standardized architecture and community effort still make it as one of an optimal choice for our implementations. Using agents in Grid environment also has to face a number of challenging issues. Migrating mobile agents between Grid nodes which have different management policies and security concerns is the first issue. Second, integrate agent framework to the underlying e-Science services, e.g., SRB, and serve high level sub workflows in a transparent way. These issues will be part of the investigation in our future work.

In several workflow systems, e.g., in Kepler and Taverna, ontology has been used as basis to organize workflow components and to enhance the query for selecting components. The ontology in such system mainly focuses on experiment processes in specific scientific model instead of the engine properties, and thus it is not directly applicable for discovering engines in workflow bus. Grimoires, which is now part of the OMII release [23], extends web discovery standards Universal Description Discovery and Integration (UDDI) with semantic annotations. The UDDI standard was mainly designed for web services. Currently, many of the workflow systems are on the way to have their engines as web services. The achievements in the related work will bring fruitful results for the workflow bus.

An important application of distributed workflow execution is to schedule different instances of a workflow over grid to optimize parameters, which is similar to the systems e.g., Nimrod [24]. However, the workflow bus focuses on the workflow level scheduling.

6 Conclusions

In this paper, we discussed the distributed execution of aggregated workflows. The research is carried in the context of Dutch VL-e project. A workflow bus based architecture is presented. We discussed how to use agents to orchestrate workflow distributed execution, and ontology to enhance the query of registry. The implementation of the workflow bus is still on going; issues, such as semantic level integration of workflows, are still under study. However, from the discussion, we can at least draw the following conclusions.

1. By decomposing and encapsulating complex control intelligence, a muti agent framework provides a flexible platform for integrating distributed workflow processes.

2. The FIPA ACL provides a flexible way to describe semantics of the distributed components and to integrate relevant e-Science services e.g., ontology reasoning.

3. Semantic based registry is essential for realizing distributed execution of aggregated workflows. And sesame is a suitable tool to realize the registry.

7 Future work

We will continue our work in a number of directions. Integrating current registry with standardized description services, e.g., UDDI, will be an important issue. Another future work is to investigate use mobile agent to coordinate workflow execution in the scale of Grid environment.

Acknowledgment

This work was carried out in the context of the Virtual Laboratory for e-Science project (www.vl-e.nl). Part of this project is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ).

References

[1] B. Ludascher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience, Special Issue on Scientific Workflows*, page to appear, 2005.

[2] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics Journal.*, online, June 16, 2004.

[3] Yolanda Gil, Ewa Deelman, Jim Blythe, Carl Kesselman, and Hongsuda Tangmunarunkit. Artificial intelligence and grids: Workflow planning and beyond. *IEEE Intelligent Systems*, 19(1):26–33, 2004.

[4] H. Afsarmanesh, R.G. Belleman, A.S.Z. Belloum, A. Benabdelkader, J.F.J. van den Brand, and et al. VLAM-G: A Grid-based Virtual Laboratory. *Scientific Programming: Special Issue on Grid Computing*, 10(2):173–181, 2002.

[5] Kuo-Ming Chao, Muhammad Younas, Nathan Griffiths, Irfan Awan, Rachid Anane, and C-F Tsai. Analysis of grid service composition with bpel4ws. In *AINA '04: Proceedings of the 18th International Conference on Advanced Information Networking and Applications Volume 2*, page 284, Washington, DC, USA, 2004. IEEE Computer Society.

[6] Link up project. http://www.mygrid.org.uk/linkup/. 2006.

[7] Zhiming Zhao, Adam Belloum, Adianto Wibisono, Frank Terpstra, Piter T. de Boer, Peter Sloot, and Bob Hertzberger. Scientific workflow management: between generality and applicability. In *Proceedings of the International Workshop on Grid and Peer-to-Peer based Workflows in conjunction with the 5th International Conference on Quality Software*, pages 357–364, Melbourne, Australia, September 19-21 2005. IEEE Computer Society Press.

[8] Zhiming Zhao, Suresh Booms, Adam Belloum, Cees de Laat, and Bob Hertzberger. Vle-wfbus: a scientific workflow bus for multi e-science domains. In *Proceedings of the 2nd IEEE International conference on e-Science and Grid computing*, pages 11–19, Amsterdam, the Netherlands, December 4- December 6 2006. IEEE Computer Society Press.

[9] VL-e. Virtual laboratory for e-science. In *http://www.vl-e.nl/*, 2005.

[10] Ewa Deelman, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Sonal Patil, Mei-Hui Su, Karan Vahi, and Miron Livny. Pegasus: Mapping scientific workflows onto the grid. In *European Across Grids Conference*, pages 11–20, 2004.

[11] Zhiming Zhao, Adam Belloum, Cees de Laat, and Bob Hertzberger. Using jade agent framework to prototype an e-science workflow bus. In *Agent Based Grid Computing in the Proceedings of 7th IEEE International Symposium on Cluster Computing and the Grid*, page to appear, Brazil, May 14- May 17 2007. IEEE Computer Society Press.

[12] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. JADE: a FIPA2000 compliant agent development environment. In *Proceedings of the fifth international conference on Autonomous agents*, pages 216–217. ACM Press, 2001.

[13] The Foundation for Intelligent Physical Agents. Homepage of FIPA. In *http://www.fipa.org/*, 2004.

[14] Jia Yu and Rajkumar Buyya. A taxonomy of scientific workflow systems for grid computing. *Special Issue on Scientific Workflows, SIGMOD Record, ACM Press*, 34(3):to appear, September, 2005.

[15] C. Pahl and M. Casey. Ontology support for web service processes. In *Proceedings of the 9th European software engineering conference held jointly with 10th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 208–216. ACM Press, 2003.

[16] Aduna. Open rdf homepage. In *http://www.openrdf.org/*, 2007.

[17] (DAS-2). In *The Distributed ASCI Supercomputer 2, Homepage: http://www.cs.vu.nl/das2/*, 2002.

[18] S. Olabarriaga, J.G. Snel, C.P. Botha, and R.G. Belleman. Integrated support for medical image analysis methods: from development to clinical applications. *accepted to IEEE Trans. Information Technology in Biomedicine.*

[19] C.P. Botha. DeVIDE - The Delft Visualization and Image Processing Developmen t Environment. Technical report, TU Delft, May 2005. http://cpbotha.net/DeVIDE.

[20] D. Abramson, R. Sosic, J. Giddy, and B. Hall. Nimrod: A tool for performing parametised simulations using distrib uted workstations. In *The 4th IEEE Symposium on High Performance Distributed Computin g*, 1995.

[21] J. G. Snel, S. D. Olabarriaga, J. Alkemade, H. G. van Andel, A. J. Nederveen, C. B. Majoie, G. J. den Heeten, and M. van Straten a nd R. G. Belleman. A distributed workflow management system for automated medical image analysis and logistics. In *19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*, 2006.

[22] Ian Foster, Nicholas R. Jennings, and Carl Kesselman. Brain meets brawn: Why grid and agents need each other. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 8–15. IEEE Computer Society, 2004.

[23] Open Middle Infrastructure Institute UK. Omii server. In *http://www.omii.ac.uk/*, 2007.

[24] Tom Peachey, David Abramson, Andrew Lewis, Donny Kurniawan, and Rhys Jones. Optimization using nimrod/o and its application to robust mechanical design. In *PPAM*, pages 730–737, 2003.