Performance Evaluation for Scientific Workflow Interoperability

Ahmed Alqaoud, Computer Science Department, Shaqra University

Abstract— **Scientific workflow is a special type of workflow for scientists to formalize and structure complex e-Science applications. Many workflow systems have been released to solve problems in special domains. In large collaborative projects, it is often necessary to recognize the heterogeneous workflow systems already in use by various partners and any potential collaboration between these systems requires workflow interoperability. Publish/Subscribe Scientific Workflow Interoperability Framework (PS-SWIF) approach was proposed by author to achieve workflow interoperability among workflow systems. This paper evaluates the PS-SWIF approach and its system to achieve workflow interoperability using Web Services with asynchronous notification messages represented by WS-Eventing standard. In this paper, the author presents quantitative measurements that show that PS-SWIF can scale in a workflow heterogeneous distributed environment by assessing the scalability of the system in terms of notification message loads using large number of machines and different sizes of datasets the system can manage.**

Index Terms— **publish/subscribe, scientific workflow, web services, workflow interoperability.**

I. Introduction

There are various workflow systems to resolve problems in special domains, such as gravitational-wave physics, geophysics, bioinformatics and astronomy. In each of these domains, a variety of tools and functions are available to scientists. In large collaborative projects, it is often necessary to recognize the heterogeneous workflow systems already in use by various partners and any potential collaboration between these systems requires workflow interoperability.

Workflow interoperability was officially addressed for the first time in 1996 by the Workflow Management Coalition (WfMC) [1], with the WfMC defining Workflow interoperability as: '*the ability of two or more workflow engines to communicate and interoperate in order to coordinate and execute workflow process instances across those engines*'. The WfMC has published different standards and specifications [1,2,3] to achieve workflow interoperability at different levels and using various models. Since that time there has been little research in this area.

Over the last ten years, a variety of workflow systems have been released and a diversity of e-Science projects have made scientific workflow interoperability an important subject for researchers. Many workshops and meetings [4,5,6,7] have been organized by distributed computing committees to discuss, from different perspectives, how interoperability can be achieved among scientific Workflow Systems. Workflow interoperability can be classified at different levels according to the workflow lifecycle presented by Deelman [8].

A general approach to achieving interoperability among workflow systems, based on a WS-based notification messaging system, was proposed by author. This approach presents a Publish/Subscribe Scientific Workflow Interoperability Framework (PS-SWIF) and for validation, it is implemented in multiple workflow systems to provide run-time interoperability. The Publish/Subscribe paradigm provides a loosely-coupled communication pattern for large scale distributed computing and the resulting asynchronous messaging exchange between workflow systems promises scalability and flexibility for distributed applications.

Communication using an asynchronous messaging paradigm is characterized by a decoupling of participants in both time and space [9]. The PS-SWIF system is based on Web Services that enable scientists to use a Publish/Subscribe mechanism to publish a topic, and enables different workflow systems to subscribe to this topic and receive notification messages when an event is executed in the first workflow. The second workflow system can further process results and send to further systems, if applicable, in the similar way.

This paper evaluates the PS-SWIF approach and its system to achieve workflow interoperability using Web Services with asynchronous notification messages represented by WS-Eventing standard. It evaluates the PS-SWIF system and approach in a number of ways. Several experiments are conducted to determine scalability of the system when applied with different numbers of machines and a large volume of data that transfers between different

F. A. Ahmed Alqaoud is with the Shaqra University, P.O. Box 33 Shaqra 11961 SA (phone: 966-1-6224481; fax: 966-1-6224439; e-mail: alqaoud@su.edu.sa).

978-1-4799-0540-9/13/$31.00 ©2013 IEEE

scientific workflow systems. The PS-SWIF system is shown to be scalable and reliable with different numbers of machines and large sizes of messages.

In the following *Section II: Related work* introduces other work in this area, and highlights the strengths and weaknesses of these approaches. *Section III: PS-SWIF architecture and design* presents the PS-SWIF approach and describes how WS-Eventing is used to achieve workflow interoperability. *Section IV: Performance Evaluation for Scientific Workflow Interoperability* evaluates the PS-SWIF approach and its system to achieve workflow interoperability using Web Services with asynchronous notification messages represented by WS-Eventing standard, and *Section V: Conclusions* presents a summary of the performance evaluation for scientific workflow interoperability.

II. RELATED WORK

To date, only limited or *ad-hoc* solutions have been attempted to achieve scientific workflow interoperability between e-Science workflow systems. In this section the author presents the most popular approaches that have tried to achieve interoperability within Scientific Workflow Systems.

A. P-GRADE/GEMLCA

This approach is based on integration of GEMLCA [10] with P-GRADE [11] to achieve workflow interoperability [12]. GEMLCA (Grid Execution Management for Legacy Code Architecture) is produced by Westminster University as an application repository for deploying legacy code applications as Grid services. The P-GRADE Portal (Parallel Grid Run-time and Application Development Environment) is a workflow-oriented grid portal which supports applications at all stages of grid workflow development and manages how Web-based Grid portals can be executed on top of Globus middleware [13]. To achieve workflow interoperability in the GEMLCA/P-GRADE approach, it is required that a *home* workflow must first integrate with GEMLCA. The home workflow can be any workflow, such as Triana [14], Taverna [15] or Kepler [16], while other workflow systems can execute as nodes.

B. VLE-WFBus

VLE-WFBus [17] is a workflow bus based solution developed by Dutch Virtual Laboratory for the e-Science (VL-e) project [18] to achieve workflow interoperability. The VLE-WFBus combines different scientific workflow systems as federated components on one workflow system using a software bus. This workflow is called a workflow bus and any workflow wrapped to a workflow bus is called a sub-workflow. The VLE-WFBus wraps four popular workflow systems, namely Triana, Taverna, Kepler and VLAM-G (Virtual Laboratory Amsterdam for Grid) workflow systems [19] as the sub-workflow.

C. SIMDAT

SIMDAT [20] Data grids for process and product development using numerical simulation and knowledge discovery, started in 2004. The SIMDAT is funded by the European Commission under the Information Society Technologies Programme (IST). There are numerous publications on the SIMDAT project and the one most related to this paper is Design Document for Workflow Interoperability and Management [21, 22]. SIMDAT provides two models to achieve workflow interoperability: Web Service model and Grid Service model.

D. Kepler/Pegasus Integration

Pegasus [23] is a workflow that provides mapping and planning for resources in distributed systems at an abstract level. In Pegasus a workflow is described in resource-independent ways and Pegasus finds appropriate resource to execute them. Kepler/Pegasus integration [24] provides an integration of Kepler workflow with Pegasus that aims to allow Kepler users to construct a workflow in resource-independent ways and benefits of the advantage provided by Pegasus.

E. SHIWA Project

SHIWA [25] **SH**aring **I**nteroperable **W**orkflows for large-scale scientific simulations on **A**vailable DCIs (Distributed Computing Infrastructures) is a project submitted to the EU; the 7th Research Framework Programme (FP7). The primary goal of the project is to design Services that provide workflow interoperability between five leading scientific workflow systems: ASKALON [26], MOTEUR [27], P-GRADE, Pegasus and Triana, using coarse-grained and fine-grained interoperability approaches.

The other approaches discussed in this section (GEMLCA/P-GRADE, VLE-WFBus, SIMDAT, Kepler/Pegasus Integration, and SHIWA Project) are limited to specific types of workflow systems, whereas the PS-SWIF approach presented in this paper is general and can be applied to different workflow systems.

In general, the other approaches discussed in this section require the installation of an environment exposing an API on a user's machine, which requires an expert or developer to set up the environment and apply the configuration for the system. In contrast, the PS-SWIF approach presented by the author is simple and based on standardized messaging Web Services standards and as long as a workflow environment already provides access to Web Services, it does not require any further installation or configuration on a user's machine or modification of the workflow engine.

The other approaches also require more modifications in the constructed workflow in order to repeat the experiment and support reusability. Within the PS-SWIF approach the reusability of the same experiment or a similar experiment with different data input and parameters can be achieved without major modifications to the system.

III. THE PS-SWIF ARCHITECTURE AND DESIGN

In the PS-SWIF approach [28], the Event Source responsibility is delegated to three Web Services: Publish Topic Web Service, Source Web Service and Publish Information Web Service. The Publish Topic Web Service is created as the WS-Eventing specification [29] does not explain how a topic is created in the Event Source. The Publish Topic Web Service generates the Source Web Services automatically. The Source Web Service is created to receive a request message from Event Sink and create a response message. The Publish Information Web Service is created to deliver notification messages to Event Sink. Within the context of a workflow system, the Event Source represents the Workflow System Producer that create topics, receive request messages and sends the notification message to other workflow systems.

In the PS-SWIF approach, the Event Sink responsibility is also delegated to three Web Services: Sink Web Service, Subscriber Web Service and Subscription Manager Web Service. The Sink Web Service could be a predefined Sink Web Service or a standard Web Service. The predefined Sink Web Service is created to support workflow systems that do not have the ability of deploying an instance of workflow as a Web Service, such as the Taverna and Kepler workflows. The predefined Sink Web Service methods are invoked by interested workflows to allow them to receive notification messages. If the workflow products support deployment of instance workflow as a Web Service, such as the Triana workflow, then this Web Service represents the Sink Web Service and receives notification messages instead of using the predefined Sink Web Services.

The Subscriber Web Service in the PS-SWIF is used to allow workflow system A to create a subscription request to workflow system B. The subscription request in the PS-SIWF approach is similar to the subscription request defined in the WS-Eventing section, except that two delivery modes are applied; with push mode (asynchronous) and pull mode (synchronous) to deliver the notification message to the event Sink Web Service. The push model is applied when one uses a standard Web Service as a Sink Web Service and the pull mode is applied when one uses the predefined Web Service to act as a Sink Web Service.

The Subscription Manager Web Service is used to manage the subscription created by Subscriber Web Services. Within the context of a workflow system, the Event Sink represents the Workflow system consumer that subscribes to another workflow system and receives notification messages.

The Internal Subscription Manager entity fulfils a mediation layer between the Event Sink, Event Source and other required entities in the PS-SWIF API, such as the Topic XML file and Subscription Database. When the notification message is sent by Publish Information Web Service, this entity checks the subscription list with the Subscription Database and then delivers the notification message to those Sink Web Services that made a subscription to this event.

The Subscription Database is used to store the subscription information when the Subscriber Web Service sends a request to the Source Web Service. This information includes; Subscription ID; a unique value for every subscription; Source Web Service address, Sink Web Service address and expiry date.

IV. PERFORMANCE EVALUATION FOR SCIENTIFIC WORKFLOW INTEROPERABILITY

This section evaluates the PS-SWIF system. An experiment was conducted to evaluate the performance of the PS-SWIF system with various numbers of machines and using different ranges of data size. A total of 30 tests were undertaken: for each set of machines and sets of data sizes, three tests were undertaken. Measurements of the average value for each machine with a specific data size were taken.

During such experiments, results need to be archived in a log file for each machine. The format of these file must include:

- Test ID;
- A unique identifier for each test performed;
- Total number of machines used in the test;
- Message size sent by a workflow producer to workflow subscriber;
- Duration (sec), of delivery time to transfer the notification message to the workflow subscriber.

Log files and data were then moved to the main machine for analysis. Results are gathered and presented in a meaningful, human readable form. Time is synchronized on each machine by time server "ntp0.cf.ac.uk" using network time protocol (ntp) to guarantee the notification times, between notification sender and notification receiver, are measurable in a standard way.

A. Test-bed

The test-bed for the experiments includes 30 Linux-based machines. The first machine M(S) with a 1.80 GHz Intel(R) Pentium(R) M processor, 1.5 GB of memory, Fedora 3 as Operating System, and Java version 1.6.0_07. The M(S) machine represents the host machine for the PS-SWIF Server, using a WSPeer framework [30], and provides deployment for all the PS-SWIF Web Services. The M(S) machine also hosts the PS-SWIF application using Apache Tomcat Version 6.0.10.

The remaining 29 machines (M1 to M29) have a 2.8 GHz Intel(R) Pentium(R) 4 CPU processor and 1.0 GB of memory, Fedora 7 as operating system, and Java version 1.6.0.14. All machines were connected through an Ethernet local-area-network with 100 Mbps.

B. Experiment Setup

A NTS (Network File System) mounted *Home* directory was created that can be accessed from each machine in the University laboratory. The content of the *Home* directory is

described in Table I. The test is based on the remote execution of processes using SSH, and, as this is an automated process, SSH setup is configured to skip the password prompt for each laboratory machine using the *ssh-keygen* command to generate private/public key pair. The public key is copied onto remote machines. After the SSH configuration, the remote machines can be accessed without password prompt, and the remote machines can access the centralized *Home* directory where the data and workflow are installed.

File or Directory	Description
Data folder	Store data files with different size
lib	Present a library for java classes
Workflows	Store Taverna and Kepler workflows
createsubscriptions.sh	A script file to create a subscription
generatefile.sh	A script file to create a file with different size
machines.txt	Store the Linux lab machines name
processlogs.sh	A script file to generate logs files
runmachinetest.sh	A script file used to run the Kepler workflow on lab machines
executeworkflow.sh	A script file used to run the Taverna workflow
runtest.sh	A script file used to run the experiment

Table I: Home Directory Description

C. Create Topic and Subscriptions

The experiment uses Taverna Workflow and Kepler Workflow as the workflow publisher and workflow subscriber, respectively. A *TestSuite* topic is published manually using the PS-SWIF application. A new Source Web Service is automatically generated and available on M(S) machine. Subscriptions are made through the *createsubscriptions.sh* script file. This file subscribes all machine names specified in the *machines.txt* file with the *TestSuite* topic. The machine's name represents the Kepler workflow and the *TestSuite* topic represents the Taverna workflow. Once subscriptions are successfully made, subscription detail is shown on the PS-SWIF application.

D. Taverna Workflow

The Taverna workflow is constructed using a key component which involves the invocation of the Publish Information Web Service. The *SendNotification* operation of the Publish Information Web Service is used to send notification messages to any subscribed workflow. The *sendNotification* operation takes two parameters and *TestSuite* is specified for the first parameter representing the target topic, with the second parameter representing the message to be sent to all subscribers to this topic. The message is read from a file specified when the experiment is later run using *runme.sh* from the command line.

E. Kepler Workflow

The Kepler workflow is constructed using only one Web Service instance, which invokes the Sink Web Service. The *receiveNotification* operation of the Sink Web Service receives a message from the Taverna Workflow. Two parameters are specified for this operation. The first parameter represents the Source Web Service the message comes from, and *TestSuite* is specified. The second parameter, the workflow subscriber, specifies when the subscription is made, and, in this case, machine name will be reserved for the parameter. The received massage will be saving in an external file.

F. Experiment Execution

Every time a test is executed, a unique ID is assigned for this test in the *teseid.txt* file. The test is executed using the *runtest.sh* script file that takes two arguments:

runtes.sh datafile numberofmachines

The *datafile* represents the data size that will be passed to the workflow consumer as a notification message. The *numberofmachines* defines how many machines must be used on the test; if specified the first *n* machines defined in the *machines.txt* file will be used in the test, if not specified all machines will be used. The *runtest.sh* executes all Kepler workflows specified by machines and executes the Taverna workflow that sends notification messages to Kepler workflows.

After any number of tests with any number of machines a *CSV* (Comma-Separated Value) file is generated with the integrated information from all the machines using the *processlogs.sh* script file. The output date is stored in a log file that can be opened with Microsoft Excel or an OpenOffice Spreadsheet for further analysis.

G. Experiment Analysis

Three sets of runs are conducted; each set representing a specific number of machines representing workflow subscribers, Kepler in this case. Limited to 29 machines available at the University laboratory at the time of the experiment, the three sets are divided into 10 machines, 20 machines and 29 machines. Each set consisted of three groups of three runs each for observation and analysis purposes. The three groups contained 100 Kb, 1 Mb and 10 Mb which represent a message size to be sent to the workflow subscribers. Other experiments are conducted with different message sizes, such as 0 Kb and 25 Mb, to find the overhead time and the limitation of the system respectively. Figures 1, 2 and 3 show performance results for three sets of experiments.

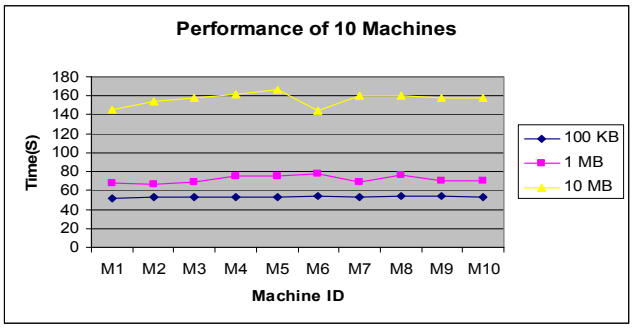

Figure 1: Performance of 10 Machines

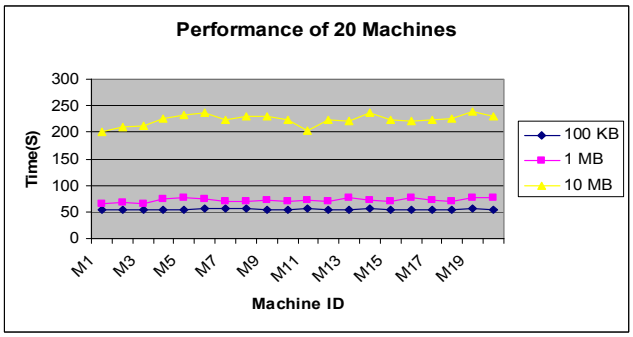

Figure 2: Performance of 20 Machines

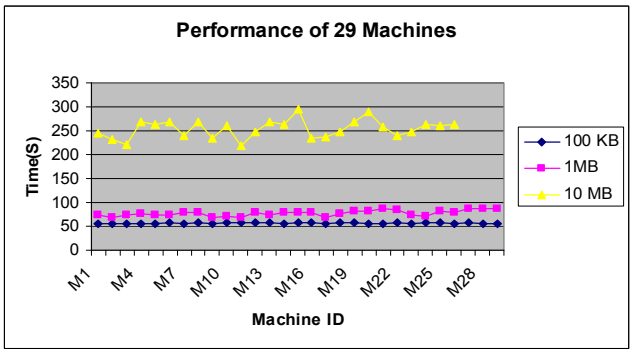

Figure 3: Performance of 29 Machines

Experimental results in Table II show the total average delivery time for 10 machines, 20 machines and 29 machines.

	100 Kb	1 Mb	10 Mb
10 Machines	53	72	157
20 Machines	55	72	224
29 Machines	57	77	254

Table II: Average Delivery Times

The total average delivery time (obtained from Table II) for various numbers of machines is calculated with 100 Kb as a baseline, with results in Figure 4.

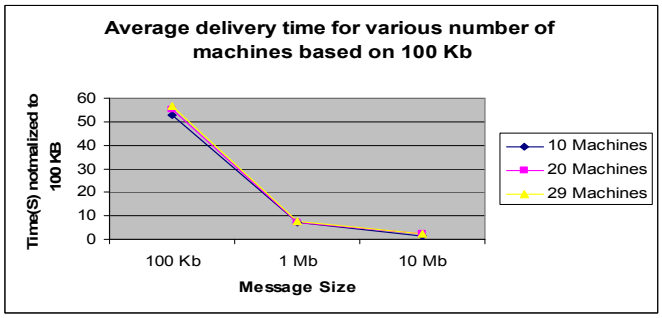

Figure 4: Average Delivery Time

All three set of machines; 10, 20, 29 machines have similar behaviour with different sets of data size. This is expected, since the message is not delivered to the machines in sequential order but is delivered to all machines in parallel. Figure 5 shows interesting results for delivery time compared to message size. For example, for 29 machines, the time taken to deliver 100 Kb when the original message size is 100 Kb is 57s and the time taken to delivery the same volume of data, (100 Kb), if the original message size of 10 Mb is 2.5s. The delivery time for a specific volume of data decreases when a larger message size is used.

Figure 5 shows the average data transferred for the different numbers of machines. To get the throughput in Kb/s, these values of 100, 1,000 and 10,000 are divided by the values in the columns of 100 Kb, 1 Mb and 10 Mb in Table II. Figure 5 shows the volumes transferred increases when a large message size is used. For example, the transfer rate for 29 machines is 1.75 Kb/s when 100 Kb is used as message size, whereas the transfer rate for the same number of machines is 39 Kb/s.

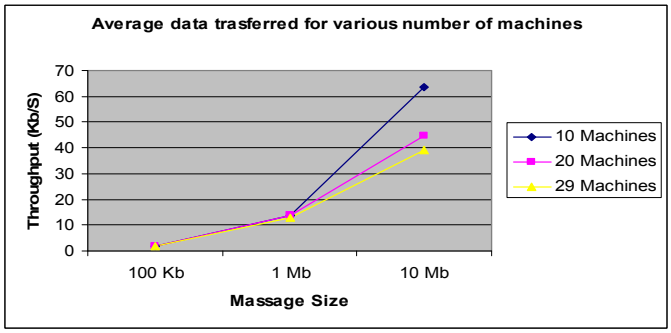

Figure 5: Average Data Transfer

Another experiment was conducted to find the fixed overhead time taken to deliver a message to workflow subscribers. The best way to do this is to send a message with a 0 Kb size. To measure this overhead, an empty file is created and a notification message is read from this file. This message is sent to the three different sets of machines 10, 20 and 29. The overhead average results are 52, 52 and 55 for 10, 20 and 29 machines respectively.

A further analysis subtracted the overhead times from the values presented in Table II, with new values displayed in Table III.

	100 Kb	1 Mb	10 Mb
10 Machines	1	20	105
20 Machines	3	20	172
29 Machines	2	22	199

Table III: Average Delivery Time without Overhead Time

The expected result should be if a message with 100 Kb takes 1 second to be delivered, then a message with 1 Mb and 10 Mb should take 10 seconds and 100 seconds respectively.

In Table III the delivery time for 100 Kb ranges from 1s to 3s, for 1 Mb from 20s to 22s and for 10 Mb ranges from 105 to 199. The delivery times for 100 Kb shows a short period of time which indicates any extra second will make a significant change with this value taken as the base for further calculation of results on higher measurable units. For example if one has two results for 100 Kb which show delivery times of 1s and 2s, as in this case, the delivery time for a message size of 10 Mb would be 100s and 200s. The variations in time with a message size of 10 Mb is significant, whereas the variation in the time with a message size of 100 Kb is quite small. A reasonable explanation for variation values in Table III for 100 Kb is that the delivery time is recorded in second as the measurable unit. At the analysis stage it was recognized that if the delivery time is recorded in milliseconds, one would get more accurate results. The results in Table III for messages 1 Mb and 10 Mb show more reliable consistent values.

To further evaluate the PS-SWIF system, another experiment is conducted to find the system limitation in terms of the maximum load of message size with maximum number of machines the system can handle before failure; essentially to find the system scalability. System limitation is defined when a message is not delivered successfully or takes a very long time to deliver. As shown in Figure 3 the system cannot deliver a message size of 10 Mb to more than 26 machines. Moreover, the system manages to deliver a message size of 20 Mb to only 5 machines in 204s. With a message size of 25 Mb, the system was not able to deliver the message to any machines. The reason for these failures was the memory size allocated to the PS-SWIF server that hosts the service. The maximum memory allocated to this server was 1,500 Mb as the total memory available to the machine running this experiment is 1.5 GB. Further investigating of this failure by debugging the PS-SWIF code showed that the problem is caused by WSPeer and/or Axis and not the PS-SWIF code. The current implementation of WSPeer based on the old version of Apache's Axis (1.2) which does not support the delivery of a large message. This can be solved by looking for ways to improve the handling of large messages in Axis and/or WSPeer such as SOAP MTOM (Message Transmission Optimization Mechanism).

V. CONCLUSION

The PS-SWIF system is shown to be scalable and reliable with different numbers of machines and large sizes of messages. The PS-SWIF system was not able to deliver a message to more than 26 machines with 10 Mb or any message with a size of 25 Mb. This failure is not caused by the PS-SWIF system itself but by the current implementation of WSPeer with Axis that does not support the delivery of a large message.

ACKNOWLEDGMENT

Author thanks Dr Ian Taylor and Dr Andrew Jones form School of Computer Science of Cardiff University, for their expert guidance and support throughout this research.

REFERENCES

[1] WfMC, 1996, Workflow Management Coalition Workflow Standard - Interoperability Abstract Specification.

[2] WfMC, 2000, Workflow Management Coalition Workflow Standard - Interoperability Internet e-m MIME Binding.

[3] WfMC,2001, *Workflow Management Coalition Workflow Standard - Interoperability Wf-XML Binding*.

[4] Harrison, Andrew, October, 2007, Workflow sharing and Interoperability.

[5] Klingenstein, K and Gannon, D, October, 2007, Improving Interoperability, Sustainability and Platform Convergence in Scientific And Scholarly Workflow, University of Colorado and Indiana University.

[6] Taylor, Ian, February 2008, Workflow Management Research Group - WFM-RG.

[7] Toth, Adrian, May 2007, Levels of the Grid Workflow Interoperability.

[8] Livny, Ewa Deelman and Miron, The Pegasus Approach to Building a Workflow Management System.

[9] Son, H and Li, X,September 2007, *PARMI: A Publish/Subscribe Based Asynchronous RMI Framework for Cluster Computing*, in *High Performance Computing and Communications*, Springer Berlin / Heidelberg. p. 19-29.

[10] Delaitre, T, Kiss, T, and Goyeneche, A,2005, *GEMLCA: Running legacy code applications as grid services.* Journal of Grid Computing. **3**(1): p. 75-90.

[11] Nemeth, C, Dozsa, G, Lovas, R, et al.,2004, *The p-grade grid portal*, in *Computational Science and Its Applications – ICCSA* Springer: Berlin / Heidelberg. p. 10-19.

[12] Kukla, T, Kiss, T, Terstyanszky, G, et al.,2008, *A general and scalable solution for heterogeneous workflow invocation and nesting*. in *The 3rd Workshop*

on Workflows in Support of Large-scale Science (WORKS08). Austin, TX: IEEE.

[13] Kertesz, A., Sipos, G., and Kacsuk, P.,2007*, Brokering multi-grid workflows in the P-GRADE portal*, in *Euro-Par 2006: Parallel Processing* Springer: Berlin / Heidelberg. p. 138-149.

[14] Cardiff, University, 1998*, The Triana Project*.[cited; Available from: http://www.trianacode.org/].

[15] Oinn, T, Addis, M, and Ferris, J, *Taverna: a tool for the composition and enactment of bioinformatics workflows*, in *Bioinformatics*. 2004, Oxford Univ Press.

[16] Foundation, National Science, 2002, The Kepler Project.

[17] Zhao, Z, Booms, S, Belloum, A, et al.,2006, *Vle-wfbus: a scientific workflow bus for multi e-science domains*. in *Second IEEE International Conference on e-Science and Grid Computing*: IEEE Computer Society.

[18] The Virtual laboratory for e-science. 2004,[cited 2008; Available from: http://www.vl-e.nl/].

[19] Afsarmanesha, H, Bellemana, R, and Bellouma, A, *VLAM-G: A Grid-based virtual laboratory*. 2002, IOS Press. p. 173-181.

[20] The SIMDAT Projec*t*. 2004,[cited; Available from: http://www.simdat.org/].

[21] Ghanem, M and Azam, N, 2006, Consolidated Report on Workflow Interopreability and Business Processes, Information Society Technologies.

[22] Ghanem, M and Azam, N, 2007, Report on Grid infrastructure interoperability challenges, Information Society Technologies.

[23] Deelman, E., Blythe, J., Gil, Y., et al., *Pegasus: Mapping scientific workflows onto the grid*. 2004, Springer. p. 11-20.

[24] Mandal, N, Deelman, E, and Mehta, G.2007, *Integrating existing scientific workflow systems: the Kepler/Pegasus example*. in *The 2nd Workshop on Workflows in Support of Large-Scale Science*. Monterey, California, USA: ACM.

[25] Kacsuk, P, 2011, SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs (SHIWA), *EUROPEAN COMMISSION, 7TH RESEARCH FRAMEWORK PROGRAMME (FP7).*

[26] Fahringer, T., Prodan, R., Duan, R., et al.,2005, *ASKALON: A grid application development and computing environment*. in *6th International Workshop on Grid Computing*. New York,: IEEE Computer Society Press.

[27] Moteur Workflow Enactor.[cited 2009; Available from: http://www.aci-agir.org/].

[28] A.Alqaoud, I.Taylor, A.Jones,2010, Scientific Workflow Interoperability Framework. International Journal of Business Process Integration and Management. (Scientific Workflows).

[29] D. Box et al. Web services eventing (ws-eventing), 2004. Available at: URL http://www.w3.org/Submission/WS-Eventing/.

[30] Harrison, A, 2006*, The WSPeer Project*.[cited 2006; Available from: http://www.wspeer.org/].