World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:7, No:6, 2013

Scientific Workflow Interoperability Evaluation

Ahmed Alqaoud

***Abstract*—**There is wide range of scientific workflow systems today, each one designed to resolve problems at a specific level. 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 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. This experiment covers different types of communication models provided by Workflow Management Coalition (WfMC). These models are: Chained processes, Nested synchronous sub-processes, Event synchronous sub-processes, and Nested sub-processes (Polling/Deferred Synchronous). Also, this experiment shows the flexibility and simplicity of the PS-SWIF approach when applied to a variety of workflow systems (Triana, Taverna, Kepler) in local and remote environments.

***Keywords*—**Publish/subscribe, scientific workflow, web services, workflow interoperability.

I. Introduction

WORKFLOW systems have become attractive to scientific computing projects for their ability to describe experimental processes in a way that makes it easy to create, manage and execute over a distributed set of resources. There are various workflow systems developed 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 has been developed and are available to scientists. In some cases, scientists may need to invoke and use different tools from other systems which are not available on their workflow system to complete their experiments or improve performance results. To collaborate between these systems and tools, interoperability is essential. Within large collaborative projects [1]-[3] combinations of workflow systems are already in use. Workflow interoperability is a significant problem that can determine if collaboration between e-science projects, using heterogeneous workflow systems can be successfully conducted.

Workflow interoperability is receiving increasing attention from the distributed-computing community. Different standards and levels have been set to achieve interoperability among Scientific Workflow Management Systems (SWFMSs), for example WfMC and provisional research surveys have been conducted by the Open Grid Forum (OGF) [4]-[6]. A special workshop focussing on SWFMSs and improving interoperability has taken place in Baltimore [7]. Different originations and committees participated in this workshop, and a technical report and recommendations issued discuss workflow interoperability levels and provides different opportunities to achieve workflow interoperability. Workflow interoperability is classified on different levels according to the workflow lifecycle presented by Deelman [8]. These levels represent workflow design, workflow mapping and execution, and workflow and data provenance.

A general approach to achieving interoperability among workflow systems, based on a WS-based notification messaging system, was proposed by [9]. This approach presents a Publish/Subscribe Scientific Workflow Interoperability Framework (PS-SWIF) and for validation, it is implemented in multiple workflow systems to demonstrate run-time interoperability.

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. This experiment covers different types of communication models provided by WfMC. The experiment has proof that different workflow engines can use the PS-SWIF approach to qualitatively improve their capabilities by accessing different workflows from third party systems without internal modification. This result shows the PS-SWIF proof of concept facilitates a qualitative difference, which could form the basis for futures standardization of the approach in OGF or similar.

In the following, *Section II: Workflow Interoperability Standards*, workflow interoperability levels and standards provided by the workflow management coalition (WfMC) and OGF community are discussed; with the current publish/subscribe paradigms presented. *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: Scientific Workflow Interoperability Evaluation* 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 PS-SWIF approach.

II. Workflow Interoperability Standards

Workflow interoperability is receiving increasing attention from the distributed computing community. Different standards and levels have been set to achieve interoperability

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).

among workflow systems. The Workflow Management Coalition (WfMC) [10] defines various standards for workflow interoperability. One of these is the Workflow Standard-Interoperability Abstract Specification [11]. In this specification different strategies can be used to achieve workflow interoperability: (1) Direct Interaction: Workflow Systems use a common API to allow direct interaction; (2) Message Passing: Workflow Systems exchange information by sending packets of data messages through a communication network; (3) Bridging Strategy: Workflow Systems apply a bridging mechanism using a gateway technique to move data and tasks between systems via protocol converters; (4) Shared Data Store: The transfer of data and tasks between workflow Systems is achieved through a shared database.

Further, the Workflow Standard-Interoperability Abstract Specification classifies workflow interoperability to eight levels: (1) No Interoperability Level: No communication between workflow products at this level, and interoperability cannot be applied. (2) Coexistence Level: No standard approach to interoperability between workflow products. Workflow products share the same run time environment, such as operating system and network. No direct interaction between different workflow products. Interoperability can be achieved at this level when an application implements different parts of a complete process, using different workflow products. (3) Unique Gateways Level: Workflow products use a bridging mechanism to work together to route operations between engines. One possibility is to use a common Gateway API among workflow products. (4) Limited Common API Subset Levels: Workflow products can interoperate directly using a common standard API. A multiple API may be needed for a given workflow product to interoperate with other workflow products. (5) Complete Workflow API Level: A single standard API is shared by all workflow products to allow access to the entire range of potential functions. (6) Shared Definition Formats Level: Requires a shared format for process definitions implemented by workflow products. Each process supported on the workflow system must have a single definition by an organization and guarantee the behaviour of the process, regardless of the workflow system used. (7) Protocol Compatibility Level: Requires that all API client and server communication must be standardized. (8) Common Look and Feel Utilities Level: In addition to the earlier levels, this level requires that all workflow products maintain the same, or at least a similar, interface. This level may not be achieved for commercial and practical reasons [11]. WfMC defines a further specification called Workflow Standard Interoperability Internet e-mail MIME Binding [12] which provides a concrete definition for a message that uses Internet e-mail with MIME (Multipurpose Internet Mail Extension) encoding as transfer method between two workflow engines to achieve interoperability as defined in the Workflow Standard-Interoperability Abstract Specification. In addition, the WfMC released Workflow Standard Interoperability Wf-XML binding in 1997 [13]. The goal of this standard is to produce a specification based on an XML language used as basis for the functionality provided in the Workflow Standard-Interoperability Abstract Specification. Workflow interoperability has recently received much interest from the distributed computing community, as can be seen from a number of current workshop, for example, in the Open Grid Forum (OGF) [4]-[6], three levels for interoperability are identified: (1) Workflow embedding, allowing workflows to run within their own environment, but invoked from another; (2) Development of a meta language, allowing different proprietary languages to be mapped to a single standard one; and (3) Semantic annotation/description/classification, important when sharing information. In October 2007, a workshop focusing on scientific workflow and improving interoperability took place in Baltimore [7]. Different originations and committees participated in this workshop, and a technical report, with recommendations, discusses workflow interoperability levels and provides different opportunities to achieve workflow interoperability. These levels include workflow design, workflow mapping and execution, and workflow and data provenance. In this model, designing a workflow involves first creating a description of the workflow at abstract level. Abstract Workflow describes a selection of application components and defines their dependencies. The application components could be tasks, jobs, services or any executable units. Dependencies between these components defines the order in which components can be executed [7], and recommends the use of a common high level specification to describe what the workflow does to achieve the interoperability at workflow design level, regardless of the workflow language used. Using the common high level specification can lead to several advantages: (1) If a workflow system no longer exists, it is possible to re-render workflows that used the old workflow system to a different language; (2) Using such specification enhances the ability of existing workflows to be published, discovered and be more understandable; (3) The specification could be used as a standard metadata language or annotation language for describing workflows. Workflow execution refers to turning the components application in the abstract workflow into an executable state. Workflow execution interoperability is essential when an instance in a workflow system needs to invoke an instance in another system. In general, the data provenance refers to the ability to obtain the history of data products. In scientific workflow systems this not only includes reproducing the data, but also includes troubleshooting and optimizing efficiency [14]. For example, when data product X generated from a workflow system A is then used by another workflow system to generate new data Y. The significant advantages of workflow interoperability and data provenance here is when provenance record Y is used to trace back to original data A.

III. The PS-SWIF Architecture and Design

In the PS-SWIF approach [9], 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 [15] 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 [16] and Kepler [17] 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 [18] 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. Scientific Workflow Interoperability Evaluation

This section evaluates the PS-SWIF system to achieve workflow interoperability using Web Services with asynchronous notification messages represented by WS-Eventing standard. This experiment covers different types of communication models provided by WfMC. These models are: Chained processes, Nested synchronous sub-processes, Event synchronous sub-processes, and Nested sub-processes (Polling/Deferred Synchronous). Also, this experiment shows the flexibility and simplicity of the PS-SWIF approach when applied to a variety of workflow systems (Triana, Taverna, Kepler) in local and remote environments.

A. Experimental Hypotheses

This section presents and explains the experiment hypotheses:

1- The experiment involves three different workflow systems, namely Triana, Kepler, and Taverna that run in three different machines to show that the PS-SWIF approach can be applied to different workflow systems that run in remote environments. Moreover, the experiment also involves two different workflow systems, namely Triana and Kepler, to show that the PS-SWIF approach can be applied to different workflow systems that run in a local environment. Choosing the order of running these workflow systems is arbitrary and the experiment can be run in any order.

2- The Experiment uses the PS-SWIF application to manage the exchanging of data between different workflow systems. Four topics are created, namely *Test3M* for Triana workflow run on machine M1, *Test3M_Tavern* for Taverna workflow run on machine M2, *Test3M_Kepler* for Kepler workflow run on machine M1, and *Test3M_Triana* for Triana workflow run on machine M3. Six subscription requests are made: Tavern workflow on M2, Kepler workflow on M1 and Triana workflow on M3 are subscribed to *Test3M* topic which represents the Triana workflow on M1. The Kepler workflow on M1 subscribed to *Test3M_Taverna* topic which represents the Taverna workflow on M2. The Triana workflow on M3 subscribed to the *Test3M_Kepler* topic which represents the Kepler workflow on M1. The Taverna workflow on M2 subscribed to the *Test3M_Triana* topic which represents the Triana workflow on M3. The experiment shows the ability of the system to manage the data through using the PS-SWIF application. The exchange of data depends on these subscriptions and without these subscriptions their data cannot be exchanged between these workflow systems. Moreover, the PS-SWIF allows

users to unsubscribe or renew the subscription. These options are considered to be part of managing the data.

3- To prove the ability of the system to control communication between different workflow systems the experiment involves invoking the PS-SWIF Web Services 8 times:
   1. The *sendNotification* operation of the Publish Information Web Service is invoked on Triana workflow on M1 to send notification messages to Tavern workflow on M2, Kepler workflow on M1 and Triana workflow on M3.
   2. The *receiveNotification* operation the Sink Web Service receives is invoked on Taverna workflow on M2 to receive the notification message from Triana workflow on M1.
   3. The *sendNotification* operation of the Publish Information Web Service is invoked on Taverna workflow on M2 to send a notification message to Kepler workflow on M1.
   4. The *receiveNotification* operation the Sink Web Service receives is invoked on Taverna workflow on M2 to receive the notification message from Triana workflow on M3.
   5. The *sendNotification* operation of the Publish Information Web Service is invoked on Triana workflow on M3 to send a notification message to Taverna workflow on M2. The Triana workflow system support deploys a workflow as a web service, so the Triana workflow will receive the notification message once a subscription is made without the need to invoke the Receive Notification operation of the Sink Web Service.
   6. The *receiveNotification* operation the Sink Web Service receives is invoked on Kepler workflow on M1 to receive the notification message from Triana workflow on M1.
   7. The *receiveNotification* operation the Sink Web Service receives is invoked on Kepler workflow on M1 to receive the notification message from Taverna workflow on M2.
   8. The *sendNotification* operation of the Publish Information Web Service is invoked on Kepler workflow on M1 to send a notification message to Triana workflow on M3.

The control communication between these workflow systems is achieved through invoking the PS-SWIF Web Services at the appropriate stage. Moreover, the invoking of these web services is not arbitrary. They are invoked in order to satisfy and fitful the requirements to achieve the workflow interoperability models provided by WfMC.

B. Experiment Design

Fig. 1 shows the experiment scenario. Four workflow systems are used: two workflows (Triana and Kepler) are installed on M1, Taverna installed on M2, and Triana installed on M3.

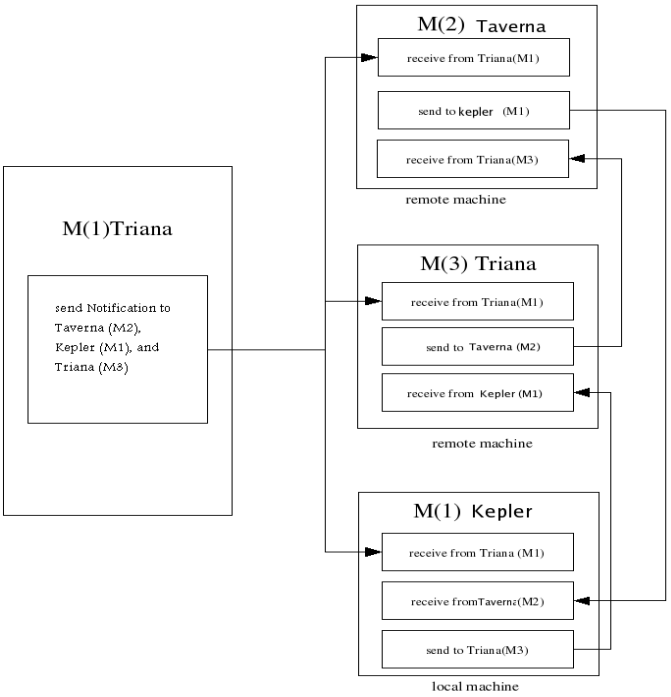

Fig. 1 Experiment Scenario

The scenario explained:

1. Triana workflow in M1 sends a message to all subscribed workflows, namely; Kepler workflow on M1, Taverna workflow on M2 and Triana workflow on M3.
2. The Taverna workflow on M2 receives a notification message from the Triana workflow on M1. The Taverna workflow does some processing with the message received and sends it to the Kepler workflow on M1. At a later stage, the Taverna workflow (M2) receives a message from the M3 Triana workflow.
3. The Triana workflow on M3 receives a notification message from Triana workflow on M1. The Triana workflow does some processing with the message received and sends it to Taverna workflow on M2. At a later stage, the Triana workflow on M3 receives a message from Kepler workflow on M1.
4. The Kepler workflow on M1 receives a notification message from Triana workflow on M1. The Kepler workflow does some processing with the received message. At a later stage the Kepler workflow on M1 receives a message from the Taverna workflow on M2 and also does more processing with the received message and then sends it to the Triana workflow on M3.

C. Test-Bed

The test-bed for the experiments includes three machines: the first machine M1 is the same machine M(S) used for the performance experiment section. The other two machines M2 and M3 have similar specifications with a 3.2 Ghz Intel(R) Pentium(R) processor and 1 GB of memory, Fedora 7 as operating system, and Java version 1.6.0.14. All machines were

connected through a private Ethernet network which was not shared by other users. M1 installed Triana and Kepler workflow systems, M2 installed a Taverna workflow system and M3 installed the Triana workflow system.

D. Triana Workflow (M1)

Fig. 2 shows the Triana workflow that runs on M1, as used to send a message to other workflows. Five tools are used to construct this workflow and Table I provides a description for each tool. The main tool in this workflow is the *sendNotification* tool which represents the operation of the Publish Information Web Service. The Publish Information Web Service is invoked in Triana using service tools to send message to any subscribed workflows.

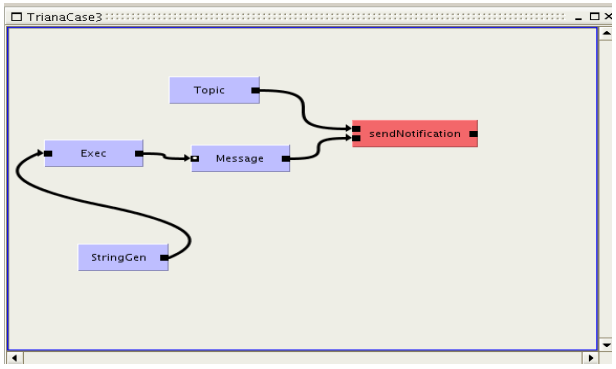

Fig. 2 Triana Workflow on M1

TABLE I
TRIANA UNITS DESCRIPTION ON M1

Triana Tool	Description
sendNotification	An operation of Publish Information Web Service used to send messages to subscribed workflows.
Topic	A tool used to specify the topic name for sendNotification tool
Message	A string value that should be sent by sendNotification tool
Exec	A tool used to execute storetime.sh scrip to store the time.
StringGen	A string unit required to execute the Exec unit.

E. Taverna Workflow (M2)

Fig. 3 shows the Taverna workflow on remote machine M2. There are 14 components used to construct this workflow and Table II gives a brief description for each component. The main components of this workflow are *receiveNotification*, *sendNotification*, and *receiveTrianaNotification* components. The *receiveNotification* is an operation of the Sink Web Service and used to receive a notification message from Triana Workflow on M1. The *sendNotification* is an operation of the Publish Information Web Service and used to send a message to the Kepler workflow on M1. The *receiveTrianaNotification* is an operation of the Sink Web Service to receive a notification message from the Triana workflow on M3. (The original name for this operation is receiveNotification but changed here to distinguish it from the previous operation used earlier in this workflow).

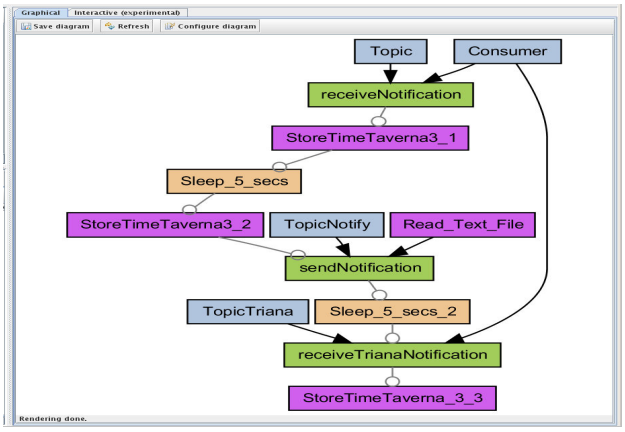

Fig. 3 Taverna Workflow on M2

F. Triana Workflow (M3)

Fig. 4 shows the Triana workflow that runs on remote machine M3. The Triana workflow is constructed from several tools and Table III presents descriptions for each tool. The primary tools in this workflow are *receiveNotification* to receive a notification message from Triana workflow on M1, *sendNotification* to send a message to the Taverna workflow on M2 and *receiveNotification1* to receive a notification message from the Kepler workflow on M1.

G. Kepler Workflow (M1)

Fig. 5 shows the Kepler workflow run on local machine M1. Various actors are used to build the Kepler workflow and Table IV gives a brief description for each actor. The primary actors in this workflow are Web Service Actor2 which represents the *receiveNotification* operation used to receive a notification message from the Triana workflow on M1, Web Service Actor represents another *receiveNotification* operation used to receive a notification message from the Taverna workflow on M2, and Web Service Actor3 which represents the *sendNotification* operation used to send a message to the Kepler workflow on M1.

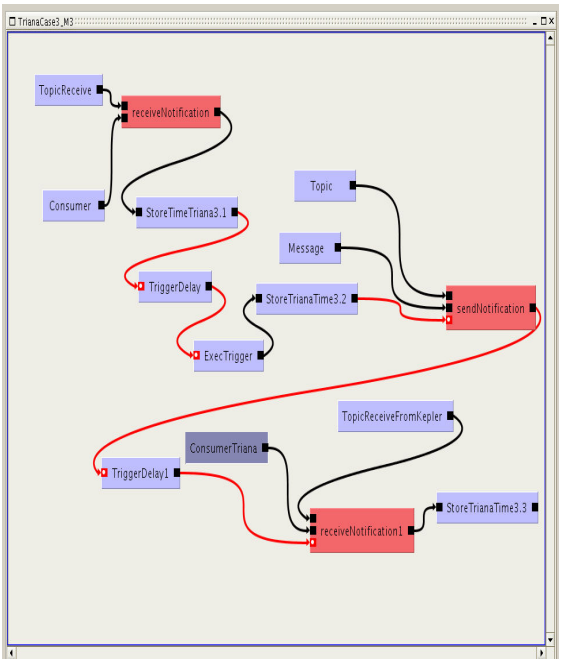

Fig. 4 Triana Workflow on M3

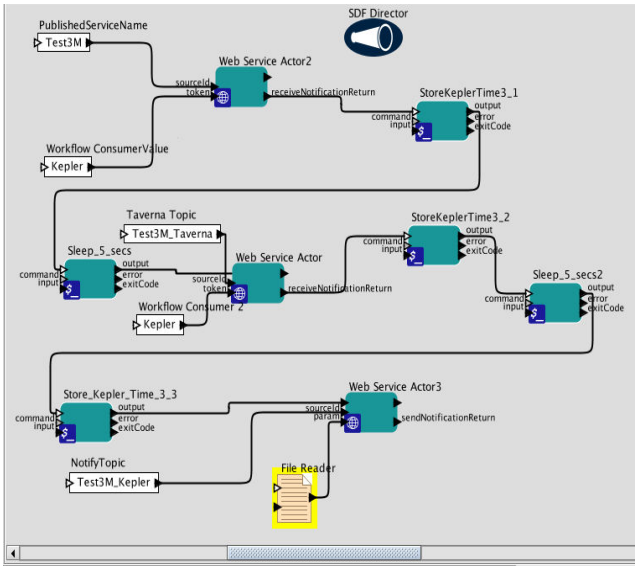

Fig. 5 Kepler Workflow on M1

TABLE II
TAVERNA COMPONENTS ON M2

Taverna Component	Description
Topic	A component used to specify the topic name for ***receiveNotification*** component
Consumer	A component to specify a consumer workflow
receiveNotification	An operation of Sink Web Service used to receive a notification message
StoreTimeTaverna3_1	Store time when the notification message is received
Sleep_5_secs	Used for a process, sleep 5 second thereafter
StoreTimeTaverna3_2	Store time when message is sent by ***sendNotification*** operation
TopicNotify	Component used to specify the topic name for ***sendNotification*** component
Read_Text_File	Component is used to read a text file
sendNotification	An operation of Publish Information Web Service used to send a notification message
TopicTriana	Component used to specify the topic name for ***receiveTrianaNotification*** component
Sleep_5_secs_2	Used for a process, sleep 5 seconds thereafter
receiveTrianaNotification	An operation of Sink Web Service that used to receive a notification message
StoreTimeTaverna_3_3	Store time when the message is received by ***receiveTrianaNotification*** components

TABLE III
TRIANA UNITS DESCRIPTION ON M3

Triana Unit	Description
TopicReceive	A unit used to specify the topic name for ***receiveNotification*** tool
receiveNotification	Operation of Sink Web Service used to receive notification message
Consumer	A unit to specify a consumer workflow
StoretimeTriana3_1	Store time when the message is received by ***receiveNotification*** unit
TriggerDelay	Used to do some process, sleep 5 second
ExecTrigger	String unit is used to execute the storetime.sh in the ***StoreTrianaTime3_2***
StoreTrianaTime3_2	Store time when the message is sent by ***sendNotification*** operation
Message	String value that should be sent by sendNotification tool
Topic	Unit used to specify the topic name for ***sendNotification*** tool
sendNotification	Operation of Publish Information Web Service used to send a notification message
TriggerDelay2	Used to do some process, sleep 5 second
ConsumerTriana	Unit to specify a consumer workflow
TopicReceiveFromKepler	Unit to specify the topic name for ***receiveNotification1*** tool
receviceNotification1	Operation of Sink Web Service used to receive notification message
StoreTrianaTime3_2	Store time when message received by ***receiveNotification1*** operation

TABLE IV
KEPLER ACTORS DESCRIPTION

Kepler Actor	Description
Test3M	A topic name that used to specify the topic for ***receiveNotification*** operation
Kepler	A consumer name that used to specify the consumer workflow
Web Service Actor2	An operation of Sink Web Service that used to receive a notification message
StoreKeplerTime3_1	Store time when the message is received by ***receiveNotification*** operation
Sleep_5_secs	This is used to do some process, sleep 5 second
Test3M_Taverna	A topic name that used to specify the topic for ***receiveNotification*** operation
kepler	A consumer name that used to specify the consumer workflow
Web Service Actor	An operation of Sink Web Service that used to receive a notification message
StoreKeplerTime3_2	Store time when the message is received by ***receiveNotification*** operation
Sleep_5_secs2	This is used to do some process, sleep 5 second
Store_Kepler_time_3_3	Store time when the message is sent by ***sendNotification*** operation
Test3M_Kepler	A topic name that used to specify the topic for ***sendNotification*** operation
File Reader	A message that should be send by ***sendNotification*** operation
Web Service Actor3	An operation of Publish Information Web Service that used to send a notification message

H.Experiment Process

The experiment can be run in any order, no matter which workflow runs first. If the message is sent by the workflow publisher and there is no one to receive it, the message will be held in a queue until pulled by a workflow subscriber. If the workflow subscriber executes first and there is no notification message at this time, the workflow subscriber keeps listening until the notification message arrives.

The only aspect affected by the execution order is the time calculated between the message sent tool of workflow publisher and message tool of workflow subscriber. Taverna workflow on M2 was executed first and then Triana workflow on M3 second and Kepler workflow on M1 third, and the Triana workflow on M1 last, because the Triana workflows on M1 was the initiator of the interactions between these workflow.

The following description explains how the workflow interoperability models provided by WfMC are achieved:

1. The chained processes model is achieved when the Triana workflow on M1 use Publish Information Service to send the notification message to the other workflows.
2. The Nested synchronous sub-process and Event synchronous sub-process models are achieved when the Triana workflow uses the Publish Information Web Service on M1 to send the message to the Taverna workflow on M2. The Taverna workflow receives it through the Sink Web Service and then simulates some processing of the receive message, using the sleep 5 second component, and sends to the Kepler workflow on M1.

The Nested synchronous sub-process and Event synchronous sub-process models assume that the notification message should be sent back to the first workflow that initiates the communication; which is the Triana workflow on M1 in this case. The PS-SWIF approach can handle this assumption easily but to avoid implementing each model in separate experiments, one experiment that covers all primary aspects of each model is used.

3. The Nested sub-process (Polling/Deferred Synchronous) model is achieved when the Triana workflow on M3 complete their processes except the *receiveNotification1* tool which waits until all other workflows 'Taverna' and 'Kepler' finish the entire workflow processes and send the notification message to the Triana workflow on M3 which explains why the Triana workflow on M3 is the last to finish execution.

I. Experiment Observation

To observe the experiment, the PS-SWIF application is used to monitor the published topic and the subscription. Fig. 6 shows that all topics are successfully published, and all the subscriptions are successfully made. The details of these subscriptions are also shown in the same figure. Triana workflow on M1, Taverna workflow on M2, Triana workflow on M3, and Kepler workflow on M1 successfully invoked the PS-SWIF Web Services and this is shown in Figs. 2-5. The experiment was successfully executed and the data was moved among these workflow systems.

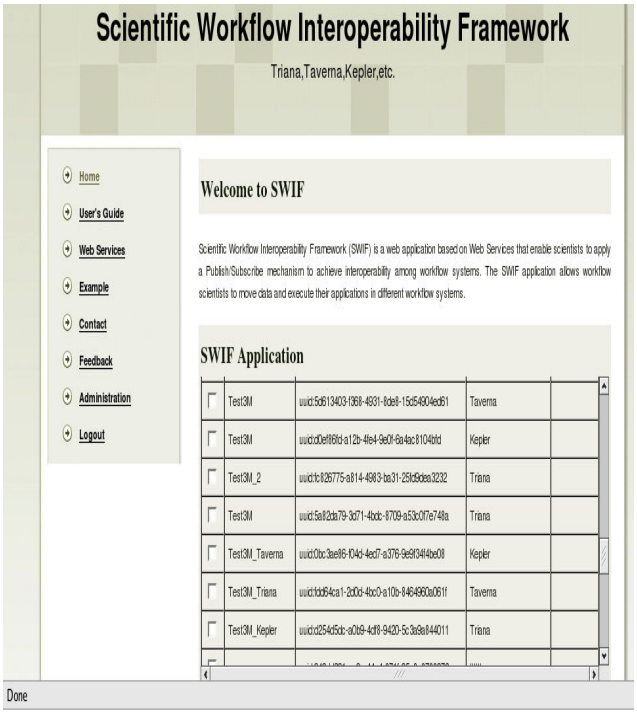

Fig. 6 Experiment Observation

J. Experiment Achievements

1. The experiment showed how the Web Services with asynchronous notification messages can be invoked and deployed by different workflow systems, namely Triana, Taverna, and Kepler, to move and mange data between these workflow systems without modification to those systems.
2. The experiment proved that different types of communications between workflow systems can be achieved by satisfying the requirements of workflow interoperability models provided by WfMC.
3. The experiment proved the flexibility and simplicity of the PS-SWIF approach when applied to a variety of workflow systems (Triana, Taverna, Kepler) in local and remote environments.
4. This experiment provides a sophisticated example of how the system can handle different models of interoperability using different types of workflow systems. Moreover, other experiments that cover the following scenario have been conducted to prove that all possibilities of communication between different workflow systems (Tirana, Kepler, and Taverna) can occur:
   a) Tirana, Taverna, and Kepler
   b) Triana, Kepler, and Taverna
   c) Taverna, Triana, and Kepler
   d) Taverna, Kepler, and Triana
   e) Kepler, Taverna, and Triana
   f) Kepler, Triana, and Taverna

V. Conclusion

In this paper, we have discussed workflow interoperability for scientific applications, describing the levels identified and the strategies identified which can achieve interoperability among different SWFMSs. A novel approach to workflow interoperability is introduced in this paper, based on a WS-based notification messaging system that uses a mechanism for decoupling and enabling asynchronous messaging to achieve workflow interoperability. The PS-SWIF application is presented based on a set of web services that follows WS Eventing specifications. This application enables scientists to run their experiments among SWFMSs that execute remotely.

These experiments evaluate the PS-SWIF approach and its system to achieve workflow interoperability. The PS-SWIF approach is easier for scientists and provides interoperability among a wide range of SWFMSs.

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] The National Science Foundation. Linked environments for atmospheric discovery (lead), 2003. Available at: URL http://www.renci.org/focus-areas/project-archive/lead.

[2] Scott Koranda. Ligo inspiral analysis workflow, 2007. Available at: URL https://spaces.internet2.edu/display/scischworkflow/Home.

[3] Philip Maechling. Scec earthquake wave propagation and source validation workflow, 2007. Available at: URL https://spaces.internet2.edu/display/scischworkflow/Home.

[4] Adrian Toth. Levels of the grid workflow interoperability. Open Grid Forum OGF20, May 2007.

[5] Andrew Harrison. Workflow sharing and interoperability. GridNet2 Report - Open Grid Forum OGF21, October 2007.

[6] Ian Taylor. Workflow management research group - wfm-rg. GridNet2 Report - Open Grid Forum OGF22, February 2008.

[7] K Klingenstein and D Gannon. Improving interoperability, sustainability and platform convergence in scientific and scholarly workflow. Technical report, University of Colorado and Indiana University, 2007.

[8] Ewa Deelman and Miron Livny. The pegasus approach to building a workflow management system.

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

[10] D. Hollingsworth. Workflow management coalition: The workflow reference model. Document TC00-1003, Workflow Management Coalition, Jan, 1995.

[11] Workflow Management Coalition members. Workflow Management Coalition Workflow Standard - Interoperability Abstract Specification. The Workflow Management Coalition, 1996.

[12] Workflow Management Coalition members. Workflow Management Coalition Workflow Standard - Interoperability Internet e-m MIME Binding. The Workflow Management Coalition, 2000.

[13] Workflow Management Coalition members. Workflow Management Coalition Workflow Standard - Interoperability Wf XML Binding. The Workflow Management Coalition, 2001.

[14] Susan B. Davidson, Sarah Cohen Boulakia, and Anat Eyal et al. Provenance in scientific workflow systems. IEEE Data Eng. Bull., 30(4):44–50, 2007.

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

[16] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M.R. Pocock, A. Wipat, et al. Taverna: a tool for the composition and enactment of bioinformatics workflows, 2004

[17] National Science Foundation. The Kepler Project. 2002. Available at: URL https://kepler-project.org/ .

[18] Cardiff University. The Triana Project. Available at: URL http://www.trianacode.org.