ICT 269978

Integrated Project of the 7th Framework Programme

COOPERATION, THEME 3
Information & Communication Technologies
ICT-2009.5.3, Virtual Physiological Human

VPH-Share

Work Package: WP6

User Access Systems

Deliverable: D6.1

Workflow Composition/Management: State-of-the-Art

Version: 2.2

Date: 31-May-2011

This page is intentionally blank

DOCUMENT INFORMATION

IST Project Num	FP7 – ICT - 269978	**Acronym**	VPH-Share
Full title	Virtual Physiological Human: Sharing for Healthcare – A Research Environment		
Project URL	http://www.vph-share.eu		
EU Project officer	Joël Bacquet		

Work package	**Number**	6	**Title**	User Access System
Deliverable	**Number**	6.1	**Title**	Workflow composition/management: state-of-the-art

Date of delivery	**Contractual**	31-May-2011	**Actual**	31-May-2011
Status	Version 2.2		Final ☒	
Nature	Prototype ☐ Report ☒ Dissemination ☐ Other ☐			
Dissemination Level	Public (PU) ☒ Consortium(CO) ☐	Restricted to other Programme Participants (PP) ☐ Restricted to specified group (RE)☐		

Authors (Partner)	UPF, Cyfronet, SCS			
Responsible Author	Luigi Carotenuto		**Email**	Luigi.carotenuto@upf.edu
	Partner	UPF	**Phone**	+34-93542-2255

Abstract (for dissemination)	In this document we provide an overview of the state-of-the-art scientific workflow management systems, in order to identify the existing technologies as well as the open issues related to workflow composition and execution. A review of the main available tools is performed in order to highlight the principal features of each of them and to position these features into a more general description of the functionalities required for fulfilling the life cycle of scientific workflows.
Keywords	Scientific workflow composition/management, workflow orchestration, workflow enactment, data provenance

Version Log			
Issue Date	**Version**	**Author**	**Change**
10-May-2011	1.0	L. Carotenuto	First draft
25-May-2011	2.0	L. Carotenuto	Including comments from Marek Kasztelnik, Debora Testi, Norman J. Powell
28-May-2011	2.1	PMO	Minor changes for release
31-May-2011	2.2	PMO	Final Proof Read

This page is intentionally blank

1 EXECUTIVE SUMMARY

The continuing quest for ever-larger simulation facilities has led to recent significant growth in the use of workflows within the scientific community whilst, in business, workflow application is already common practice and has led to the development of *de facto* standard languages (WS-BPEL and BPEL [Ref. 1]) for workflow composition and interoperability. The object of this document is to provide an overview of state-of-the-art scientific workflow management systems (SWMS), to identify existing technologies and to examine their open issues.

A review of the main systems currently available was performed and the principal features of each highlighted, to identify the functionality required for workflow life cycle management. Though a certain level of consensus has recently been reached by the business community on workflow composition language, the heterogeneous nature of scientific workflows and their data-oriented paradigm seems to have delayed such moves in the scientific domain. Consequently each SWMS has its weaknesses, often associated with its role in meeting the particular needs of the scientific sub-domain (bioinformatics, astronomy, gravitational wave science, meteorology, neuroscience, etc.), for which it has been developed.

The selection of a workflow system must therefore include a number of considerations:

- It must take into account any domain-related limitations.
- It must ensure that adequate interoperability is possible.
- The degree to which the system offers compatibility with semantic composition and data provenance techniques must be assessed.
- In VPH-Share, scientific workflows will be composed, accessed and executed through a centralised web-based master interface, and an early conclusion is that interaction will be facilitated by an intuitive graphical builder.
- In addition, the need to execute scientific workflows on distributed environments and to share large varieties of data-oriented services, requires an efficient enactment engine, in order to orchestrate complex and distributed sequences of tasks and data.

Given these needs, the Triana [Ref. 2] and Taverna [Ref. 3] systems seem to provide an adequate solution, since each includes an attractive user interface for workflow-building and dataflow-management; additionally, both systems include large numbers of web-services suitable for incorporation in new workflows. Finally, in each case the authors are currently working on self-contained workflow archives, so as to share workflows in a portal environment, so possibly meeting the need for a VPH-Share master interface.

2 INTRODUCTION

According to the Workflow Management Coalition (WfMC), workflows can be defined as "the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" [Ref. 4]. This definition perfectly applies to workflows for the business community. More generally a workflow can be described as a sequence of operations or tasks needed to manage a business process or a computational activity [Ref. 5]. The latter definition can also be applied to scientific workflows, which are meant to decompose complex scientific experiments into a series of repetitive computational steps that could be run on supercomputers or distributed on a cloud system.

As a consequence of this growing need, a great number of Scientific Workflow Management Systems (SWMS) have been developed in recent years, trying to address the needs of specific scientific communities. The heterogeneity and rapidly varying nature of scientific workflows has meant that, in contrast to what has happened in the business community, these systems have not yet converged, neither to a common definition of a workflow composition language nor to a common execution strategy for workflow enactment engines.

Different reviews of the current available SWMS have been proposed in recent years [Ref. 6 - Ref. 7], though, due to the dynamic nature of scientific workflows domain, new systems appear on the scene very frequently. This means that a classification of systems will hardly provide an exhaustive and up-to-date overview of currently available SWMS and therefore a complete classification is beyond the scope of the current document. We will therefore limit our attention to the identification and the analysis of the main phases of a scientific workflow life-cycle, in order to underline the issues and the more advanced solutions proposed by an exemplar subset of SWMS.

In this document we will briefly describe the differences between business and scientific workflows (section 3), in order to understand which lessons can be learned from the business community, and enhance some specific peculiarities of scientific workflows, calling for specific solutions. In section 4, we will discuss the differences between data flow and control flow systems, while in section 5 we will describe the life-cycle of scientific workflows, trying to underline the issues and proposed solutions for each phase. In section 6, we will give an overview of the main available SWMSs, by describing the features of each, and with reference to the different life-cycle phases. Finally, in section 7, we will discuss general recommendations for the VPH-Share project on the most appropriate SWMs that could be used for building, executing and distributing workflows through the VPH-Share Master Interface.

3 BUSINESS VS SCIENTIFIC WORKFLOWS

Business workflows fulfil the need for automating business processes in order to make them faster and reduce the errors while performing tedious and repetitive steps. On the other side scientific workflows express the need to execute and parallelise large and complex

experiments on distributed systems. Furthermore it is very frequent that particular tasks of a scientific workflow need the supervision of a scientist and may need to be cancelled or repeated according to the specific partial results.

Barga *et al.* [Ref. 7] analyse the main commonalities and differences between business and scientific workflows. For example, both need technologies for defining ACID (Atomicity, Consistency, Isolation and Durability) transactions and long-running workflows and both could take advantage of web-service technologies (already used in BPEL – Business Process Execution Language, that represents a standard *de facto* for business workflow composition [Ref. 1]) for distributing tasks in heterogeneous networks.

However, scientific workflows have typically a more dynamic and varying nature and they may involve voluminous data transactions Furthermore the need for storing and sharing the workflow in common repositories where it can be reused for composing new workflows is an exclusive need of the scientific domain, like the ability to run, interrupt, revise and resume a workflow during its execution, to modify the task of an experiment while the experiment proceed [Ref. 7]. The latter, together with the presence in the workflow of tasks that could create new data and/or dynamically modify the workflow itself raises the need for technologies for semantic provenance, as will be discussed in section 5.4.

4 DATA VS FLOW CONTROL

Probably the most relevant classification of workflow systems in terms of a workflow execution model is the distinction between control flow and data flow models. While control flow models are based on a transfer of control from the preceding task to the following one, the data flow models are based on flow of data between workflow activities [Ref. 5]. While the first model is commonly used by business workflows, the second better expresses the dynamics in a typical scientific workflow.

There are also hybrid workflow representations, resulting from a combination of both models. These representations are typically able to switch from one model to the other according to certain conditions, though they are normally biased toward one of the two models [Ref. 5].

5 WORKFLOW LIFE-CYCLE

As in Deelman *et al.* [Ref. 8] the life-cycle of scientific workflows can be described by the following phases:

1. Composition phase: in which the workflow is composed, typically in an abstract form and according to a determined composition language;

2. Mapping phase: in which the abstract workflow representation is actually mapped to the underlying resources;

3. Execution phase: in which the mapped workflow is enacted on the underlying resources by

a specialised enacting engine;

4. Metadata and Provenance phase: in which the metadata and provenance information are recorded during the entire workflow lifecycle.

In the following we will analyse all the phases and the main issues for each of them, with a particular attention to the different approaches and solution proposed by the different SWMS.

5.1 WORKFLOW COMPOSITION

Workflow composition represents an important phase of the workflow management life cycle. During this phase the workflows are described, according to a particular representation. Workflow composition methods can basically be divided into two main categories: textual and graphical compositions.

Textual systems are widely deployed in the business community where the workflows are usually defined by software developers who customise them according to the end-user /client needs. In this context, the BPEL (Business Process Execution Language)[1] is the *de facto* standard for Web-service-based workflows and a great number of implementations have been developed from the major software vendors active in the sector.

On the other hand, within the scientific community, where workflows are usually designed by the same scientists who will use them and who could have little or no expertise in software programming, graphical representations have been more widely employed. This usually calls for a double representation of the workflow within the same SWMS: an internal representation based on a particular workflow language and an external one, typical a graphical representation.

The most typical graphical representations used by the different SWMS are basically all variants of the directed graph, both acyclic (DAG) and cyclic (DCG), though other representations are also used (as UML [Ref. 9], Petri-Net [Ref. 10], VDL [Ref. 11]). The representation with graphs is usually internally mapped to an XML based representation that varies according to the single SWMS.

Though graphical rendering is a powerful tool to simplify workflow composition, it could become a bottleneck when designing large and complex workflows, requiring a large number of steps and also “for/while” cycles. Consequently most graphical tools include methods for defining nested or sub-workflows hierarchies, and ways of expressing cycles and conditional control primitives.

[1] http://en.wikipedia.org/wiki/Business_Process_Execution_Language

5.1.1 SEMANTIC COMPOSITION

Scientific workflows may be composed of a large number of tasks that could be chosen from a list of available services. In a web service scenario, the number of these services on the Internet could grow very rapidly thus impeding a manual browsing over all the available options. This scenario calls for the need of semantic services, containing formal descriptions that can enable better, semi-automatic discovery of the services themselves, thus enhancing the composition, monitoring and interoperability of scientific workflows [Ref. 12].

Some SWMS have started considering rich semantic descriptions of workflows services and workflow templates, enable significant improvements in automation and assistance for workflow composition and in general for managing and automating complex scientific processes [Ref. 8].

5.1.2 TASK BASED VS SERVICE BASED WORKFLOWS

Workflow systems can also be classified as task-based or service-based [Ref. 13]. Task-based systems focus on the definition of computing tasks to be executed, thus concentrating on the mapping of resources and definition of execution methods, while delegating the higher-level composition to external tools. In contrast, service-based systems focus their attention on the composition phase, defining a series of services (usually web-services) seen as black boxes by the composer, which only knows their invocation API.

5.2 RESOURCE MAPPING

Resource mapping consists of binding the abstract representation of the workflow defined during the composition phase to available resources. Some SWMS use a simple manual mapping strategy for this resource association, where the user has to specify, during the composition phase, which resource will be used for each task and eventually must specify a list of alternative resources that will be used according to a predefined failure recovery strategy. On the other side other SWMS define methods for discovering available resources when the workflow is actually instantiated. These methods use either internal schedulers or external brokers to bind tasks dynamically to resources at runtime [Ref. 8].

5.3 WORKFLOW EXECUTION

Workflow enactment is performed by execution engines, which have to coordinate the workflow execution according to the control-flow constructs (sequence or tasks, conditional statements, loop cycles, parallel execution, etc.). The execution engine must also deal with the data-flow dependencies [Ref. 14].

The current progress of a workflow invocation is usually shown by the SWMS in a dedicated control panel, which can permit the user to browse the intermediate and final results.

Another important aspect of the workflow execution phase is fault-tolerance. Within the scientific domain, in which workflows must be executed in distributed environments, failure can occur in any step of the workflow execution and for different reasons: unavailability of services, overloading of resources, network error, etc. Modern SWMS should be able to handle these failures and react to them accordingly. The adopted strategies could vary from a simple notification of the user (and a consequent request for manual intervention), to more sophisticated task-level techniques, including retry, alternate resources, checkpoint/restart and replication [Ref. 6]. The retry action simply tries to execute the task again, until a timeout occurs; the alternate resource submits the task to an alternate resource that was specified at the moment of the workflow composition; the checkpoint/restart tries to move the failed task transparently to other resources; the replication runs the same task simultaneously on different infrastructures.

The execution phase should also take into account the possibility of adaptively modifying the workflow on-the-fly, responding to data produced in real time by sensors or other instruments [Ref. 8]. To a further extent, this means that a workflow could be able to modify itself during execution, by means of some sort of demon service, which could inject code in response to data creation. Of course this eventuality opens exciting possibilities but also implies the consideration of new security and integrity issues for the workflow management systems.

5.4 SEMANTIC PROVENANCE

Each data object created during the execution of the workflow, together with the exact sequence of processes involved, should be recorded by the workflow system. In scientific workflows systems, this aspect is crucial for an effective reproducibility of the experiment that the workflow represents.

The majority of the examined SWMS offer capture systems that are very similar in terms of the captured information. The real difference is more on the way this information is presented to the user. Some systems use internal structures, while others rely on external, generic services [Ref. 8].

Furthermore, apart from the data provenance, since the mapping operation (or even the adaptive execution) could produce substantial modifications to the workflow that was originally designed by the user, a provenance mechanism should also be implemented for the systems that allow this kind of modification.

6 WORKFLOW SYSTEMS

Zhao *et al.* [Ref. 15] and Deelman *et al.* [Ref. 8] present a comprehensive, though not exhaustive, classification of the major SWMS available. In this section, following a similar approach, we will try to summarise the main characteristics of the most relevant systems, with attention to the workflow life cycle phases described above. We will limit our analysis to those systems providing an open source licensing model, since it constitutes an important prerequisite for reusing the system in the context of the VPH-Share project.

6.1 CONDOR/DAGMAN

Condor[2] is a resource management system (RMS) developed at the University of Wisconsin-Madison. The goal of the Condor Project is to develop, implement, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributed computing resources. In this sense Condor focuses on providing reliable access to distributed computing resources over long periods of time, instead of high-performance computing for short periods of time [Ref. 16].

Workflow Composition:

Condor can be classified as a task-based system, since its focus is centred on the mapping of services to available resources. The actual workflow composition is delegated to Direct Acyclic Graph Manager (DAGMan[3]) that permits the graphical representation of the dependencies between workflow steps (called jobs). In DAGMan each job is represented as a multi-parent/multi-children node in a graph, while edges among nodes represent their dependencies. The Condor/DAGMan system is distributed under the General Public License (GPL).

Workflow Execution:

For the execution phase, DAGMan nodes cannot run until their parents have completed, and cycles between jobs that are "mutual descendent" are forbidden, to avoid deadlocks.

The execution of jobs is controlled by the Condor scheduler, which is also able to handle job failures. The simplest failure tolerance mechanisms implemented in Condor permit retrying the job for a specified number of times after failure. Upon ultimate failure Condor notifies DAGMan, which provides the error reporting system and is responsible for generating a rescue diagram that could be resubmitted by the user once the error has been corrected.

[2] http://www.cs.wisc.edu/condor
[3] http://www.cs.wisc.edu/condor/dagman

6.2 PEGASUS

Pegasus[4] is a workflow manger developed by the University of Southern California. It is a task-based system, performing a mapping from an abstract workflow to a set of available resources on a Grid. The result is an executable workflow [Ref. 17]. Pegasus is distributed under the Globus Toolkit Open License (GTOL)[5].

Workflow Composition:

The composition of the abstract workflow is performed by an external tool or can be provided in the DAX (DAG XML description) language. Thus, an abstract workflow represents components and their dependencies in the form of a DAG, while the computation is expressed in terms of logical files and logical application components.

Resource mapping and optimisation:

Pegasus can access a metadata catalogue service (MCS), which stores metadata information about logical data items (e.g. files). The user can dynamically add metadata attributes to the MCS and define user-defined attributes. A logical file uniquely identifies the content of a file, without giving any information of any physical instances of that file. Thus, MCS is typically used in combination with the Globus Replica Location Service (RLS), which registers and locates physical instances of a given logical file.

Pegasus also uses optimisation strategies aimed at reducing the abstract workflow before the mapping phase actually starts. The reduction algorithm works by removing any antecedents of spurious jobs that have no descendants, thus reducing the complexity of the executable workflow.

During mapping, resource selection can be performed randomly or by a performance prediction algorithm. In this case, Pegasus can use an external tool to perform performance analysis of parallel and distributed applications.

Data Provenance:

Data provenance in Pegasus is provided by upper layer workflow compositions tools. An interesting solution is proposed by the Pegasus/Wings system, where Wings [Ref. 18] acts as the workflow composition tool for generating abstract workflows to be mapped by Pegasus.

The Pegasus/Wings framework enables detailed semantic metadata to be made available before execution, and rich optimisation trails for the executed workflow [Ref. 19].

Furthermore, to register modifications to the workflow itself that may occur during the mapping process, Pegasus is also currently studying the possibility of including provenance tracking to record workflow changes during mapping.

[4] http://pegasus.isi.edu
[5] http://www.globus.org/toolkit/legal/4.0/license-v3.html

6.3 TRIANA

Triana[6] is an open source problem-solving environment developed at Cardiff University that combines an intuitive visual interface with powerful data analysis tools. Triana was originally developed as a data analysis environment for a gravitational wave detection project [Ref. 2, Ref. 20]. Triana is distributed under the Apache Software License[7].

Workflow Composition:

Triana provides a visual programming interface that permits visual workflow composition, through a series of powerful editing capabilities, tools and GUI builders. In Triana the basic workflow element, called a 'unit', can be chosen by a set of already available units or can be created by the user. The units can be dragged and connected in the workspace, to design the workflow. Since Triana comes from the gravitational wave field, it provides an extensive series of tools for this domain; recently it has also been used for workflow composition in other fields such as image processing and audio analysis.

Triana is a data-flow system, where each dependency between units is a data dependency and the workflow sequence flows from data provider to data consumer units. As such, it doesn't provide any control construct at all, thus simplifying the workflow execution. Conditional constructs can be performed by the use of specific units that output data on different branches, upon specific conditions. Similarly loops are performed by circular connections among units that could be broken with the introduction of a conditional unit inside the loop.

Internally, Triana represents workflows by the use of a DCG format of its own, but it is also able to import and export workflow representations in other formats (DAGMan, Virtual Data Language).

Resource mapping:

Triana can interface to a variety of execution environments, by the use of different technologies according to the type of workflows to be executed (service-based or task-based). For service-based workflows, the user can either specify information about the service to be invoked or map part of the workflow to distributed services, via internal scripts. For task-based workflows, Triana permits the definition of parts of the workflow that require intensive computational resources. For these tasks, the resources will be selected at runtime by interfacing to external broker-services.

Workflow Execution:

Task execution in Triana is provided by the GridLab Grid Application Toolkit[8], which can make use of different execution engines (GRMS, GRAM, Condor) to actually submit the job. For service level execution, Triana makes use of Grid Application Prototype (GAP), for

[6] http://www.trianacode.org
[7] http://en.wikipedia.org/wiki/Apache_License
[8] http://www.gridlab.org

binding to different type of services (Web, WS-RF, P2P).

Data Provenance:

Triana records data provenance according to its own internal format. The recorded information includes the sequence of executed units, the input parameters and data. Furthermore Triana can easily be integrated by external provenance services.

6.4 TAVERNA

Taverna[9] is an open source and domain-independent Workflow Management System, developed by the myGrid Team.

Taverna was originally developed for composition and execution of bioinformatics workflows on Grid systems [Ref. 3]. It can be classified as a service-based workflow system and one of its key values is the availability of a large number of services (over 3500) mainly concentrated in the bioinformatics domain. Taverna is distributed under the GNU Lesser General Public License[10].

Workflow Composition:

Taverna offers an intuitive GUI that makes use of semantic service annotations and employs semantically-enabled service functions and reasoning techniques to infer service annotations [Ref. 21]. Workflow developers can easily add new services to the Taverna service palette. Pre-existing workflows can be loaded as services into a new workflow, thus enabling the definition of sub-workflows. The graphical representation of the current workflow can be customised by the user who can choose to hide parts of the workflow representation to give a high level view. This kind of visualisation is performed by means of the differentiation between domain services, performing scientific functions and shim services, created during workflow design, specifically to connect the inputs and outputs of closely related services in order to achieve interoperation between domain services. Shim services can be hidden in order to give a high level overview.

For internal workflow representation, Taverna uses an XML-based DAG format called Simple Conceptual Unified Flow Language (SCUFL). The Taverna workflow model is based on data flow but allows both data and flow control, and is able to switch from one model to the other according to particular conditions.

Resource mapping:

In Taverna, resource mapping is performed manually by the user during the composition phase. The user can specify a list of alternatives resources that will be used whenever a resource fails. The current version of Taverna (2.x) also includes late service binding

[9] http://www.mygrid.org.uk
[10] http://www.gnu.org/licenses/lgpl.html

capabilities [Ref. 22].

Execution:

The first versions of Taverna used FreeFluo[11] as a workflow enactment engine. FreeFluo is a workflow orchestration tool for web services, originally developed by myGrid project. In the current Taverna version (2.x), Freefluo is so heavily customised that it is no longer recognisable as the original project.

At the execution level, Taverna allows multithread parallelisation: the user can specify the number of threads for parallel execution and speed up iterations.

The current progress of a workflow can be visualised in Taverna by means of an intuitive status panel, which allows the user to browse the intermediate and final results, and store them locally or remotely.

Fault tolerance is obtained by setting the fault configuration for each step of the workflow. If a step fails, it is repeated a number of times specified by the user before skipping to the next alternative step defined at composition time.

Data provenance:

Taverna version 2.x allows data provenance recording in a native format, and provenance traces can be exported as Janus-compliant Resource Description Framework (RDF)[12] graphs, and as Open Provenance Model (OPM)[13] graphs.

6.5 ASKALON

Askalon [Ref. 14] is a SWMS developed by the University of Innsbruck, Austria. The goal of Askalon[14] is to simplify the development and optimisation of applications that can harness the power of Grid computing. The ASKALON project crafts a novel environment based on new innovative tools, services, and methodologies to make Grid application development and optimisation for real applications an everyday practice. Askalon is distributed under the Global Toolkit Public License.

Workflow Composition:

Askalon represents workflows with an XML-based language called AGWL (Abstract Grid Workflow Language) [Ref. 23]. It provides a rich set of constructs to express sequence, parallelism, choice, and iteration workflow structures. In addition, programmers can specify high-level constraints and properties defined over functional and non-functional parameters for tasks and their dependencies, which can be useful for a runtime system to optimise the

[11] http://freefluo.sourceforge.net
[12] http://en.wikipedia.org/wiki/Resource_Description_Framework
[13] http://tw.rpi.edu/portal/Open_Provenance_Model
[14] http://www.askalon.org

workflow execution.

Askalon also offers to the end user the possibility of composing workflows through a graphical modelling service based on the UML standard, by combining Activity Diagram modelling a hierarchical fashion [Ref. 14].

Resource mapping/Workflow execution:

Askalon is a task-based system, focused on workflow mapping onto resources. Askalon's resource manager (GridARM based on the Globus tools[15]) enables resource discovery, advanced reservation and virtual organisation-wide authorisation along with a dynamic registration framework for activity types and activity deployments [Ref. 8].

The Askalon scheduler mixes dynamic and static optimisation strategies. Static scheduling consists of mapping the whole workflow, optimising the resources based on a variety of metrics (execution time, efficiency, economical cost, other Quality of Service – QoS - parameters) [Ref. 14]. Dynamic scheduling also takes into account dynamic events such as machine crashes and network load, that are estimated while the task is executed.

Askalon includes a fault tolerance execution engine that supports reliable workflow execution in the presence of resource failures, through check pointing and migration techniques.

6.6 KARAJAN

Karajan [Ref. 24 - Ref. 25] is developed by Argonne National Laboratory. Karajan is part of Java CoG Kit[16], which has a modular design and provides mechanisms for fast application development and easy integration of Grid middleware. Karajan is distributed under the Globus Toolkit Public License (GPTL).

Workflow Composition:

Karajan can be defined as a hybrid system where control flow and data flow may be switched according to particular conditions. The switch is handled by particular "future" components, representing data dependencies that will be produced in the future. Such components can therefore block the control flow with a data flow dependency [Ref. 8].

Karajan represents workflows in an XML-based workflow language. It supports sequences, parallelism, choices and loops for structuring the workflow. The user can define new workflow elements by simply specifying name, parameters and descriptions for the new element [Ref. 26].

Resource mapping:

Karajan supports dynamic binding of tasks to resources. When defining a workflow, the user

[15] http://www.globus.org
[16] http://www.cogkit.org

can specify the tasks at an abstract or concrete level, where a single workflow can be composed of a mix of tasks at different levels of abstraction. The concrete tasks are executed directly by an engine, whereas abstract tasks are sent at runtime to a scheduler for mapping onto a resource. Karajan supports pluggable schedulers and provides a simple built-in implementation. Karajan also supports user- defined task clustering, where the cluster is sent to a single resource for execution.

Workflow Execution:

Karajan supports hierarchical workflows based on DAGs with control structures and parallel constructs; it makes use of underlying Grid tools for the actual job submission. Workflows can be visualised and tracked by an engine and modified at runtime through interaction with a workflow repository or scheduler for dynamic allocation of resources to tasks.

Fault tolerance strategy in Karajan supports the concept of check-pointing, enabling the user to store intermediate states for later roll back when a failure occurs.

6.7 KEPLER

Kepler[17] is developed through the cross-project Kepler collaboration, which is led by a team consisting of several of the key institutions that originated the project: UC Davis, UC Santa Barbara, and UC San Diego. Kepler is distributed under the UC Berkeley License.

Workflow Composition:

Kepler [Ref. 27] workflow composer uses Vergil GUI from Ptolemy II[18] for workflow composition. The workflow system is modelled as a set of independent components called actors that can communicate through defined interfaces. Complex sub-workflows can be included as composite actors. Each actor has well-defined input and output ports as well as parameters.

Kepler is a hybrid system regarding the data vs control flow classification. The switch between the two strategies is performed by specific objects called directors that can interact with the actors and specify the type of their dependencies, according to the specifications of the user. This approach allows different execution models without the need to change the components of the workflows [Ref. 8].

Resource Mapping:

In Kepler, resource mapping can be performed manually by the user during the workflow composition phase.

Workflow Execution:

[17] https://kepler-project.org
[18] https://kepler-project.org/developers

Kepler allows adaptive execution, thus the workflow can modify itself during execution. Furthermore workflows can be exchanged between actors as if they were data, and executed by the receiving actor.

Fault tolerance methods include the invocation of alternative services upon failure and the production of partial results even when the entire workflow fails. Kepler failure tolerance capabilities can also be extended with external tools that can be triggered by exception-catching actors.

6.8 ICENI

The Imperial College e-Science Network Infrastructure (ICENI[19]) is developed at the London e-Science Centre. The goal of the ICENI project is to provide high-level abstractions for e-science (scientific computing) which will allow users to construct and define their own applications through a graphical composition tool integrated with distributed component repositories and to deliver this environment across a range of platforms and devices. ICENI is distributed under the ICENI Open Source Code Licence[20].

Workflow Composition:

Starting from a semantic description of each component of the workflow, in terms of meaning, control flow and implementation, ICENI [Ref. 28-Ref. 29] is able to build an abstract workflow, according to a specific internal non DAG representation. Both a spatial and temporal description of the workflow can be defined by the user, permitting the description of relations and interactions between components together with temporal dependencies.

ICENI represents the workflow with a language that is similar to the YAWL (Yet Another Workflow Language) [Ref. 30], including all basic workflow structures such as sequence, parallelism, choice and iteration.

Resource mapping and Workflow Execution:

The mapping of abstract components to available resources is performed by the scheduler service [Ref. 31], according to several algorithms, including random choices in a subset of best results, simulated annealing and other algorithms derived from game theory. The optimisation of the workflow in terms of execution time and computational cost is performed on a global basis, taking into account all the components of the application. ICENI can also make use of third-party scheduling algorithms.

During workflow execution the performance of each resource, together with meta-data about the resource, can be measured and recorded in a performance repository system [Ref. 32] in order to be able to schedule future runs of the application.

[19] http://www.lesc.ic.ac.uk/iceni
[20] http://www.lesc.ic.ac.uk/iceni/licence.jsp

7 CONCLUSIONS

Workflow construction and execution will help the exploitation of the VPH-Share infostructure for the benefit of the VPH community. Workflows will be accessed and built through the VPH-Share Master Interface, which will also permit their enacting by mapping each task onto the underlined atomic services distributed on the infostructure, and will schedule their execution.

In recent years several SWMS have been developed, in order to fulfil the needs of different scientific communities (bioinformatics, astronomy, gravitational wave science, meteorology, neurosciences, etc.) calling for easy-to-use systems to compose and run experiments in distributed environments. Now the need for a rigorous formalisation of the scientific workflow life cycle is emerging, in order to identify the common characteristics that a SWMS should have to satisfy the requirements of the entire scientific community.

From the analysis performed in this document, it emerges that each SWMS has its peculiar strengths and weaknesses, since it expresses the particular needs of the scientific sub-domain for which it has been originally developed. Therefore, at this stage, any decision about the choice of a specific workflow system should take into account the particular area for which the system has been conceived, with a special attention to its level of interoperability and the degree of compatibility with semantic composition and data provenance techniques, which are topics of particular interest for the VPH-Share community.

Since in the VPH-Share project scientific workflows will be composed, accessed and executed through a centralised web-based master interface, the SWMS to be used in the context of this project should provide an intuitive graphical builder. Furthermore, the need to execute the scientific workflows on distributed environments, and to share a large variety of data oriented services, requires an efficient enactment engine, in order to orchestrate complex and distributed sequences of tasks and data.

In this document, we have reviewed the available Scientific Workflow Management Systems (SWMS), in order to provide recommendations for the VPH-Share project on state-of-the-art technologies for workflow composition/management systems that could be reused in the project and identify the research aspects that deserve further investigation. According to the identified needs, both the Triana and Taverna systems seem to represent the most adequate solutions, since they both include attractive user interfaces for workflow building and data flow management. Moreover both projects already include a large number of available web-services that can be reused by the scientific community for building new workflows. In addition, they are currently working on self-contained workflow archives in order to be able to share workflows in a portal environment, which could fit the need of the VPH-Share master interface.

REFERENCES

Ref. 1 OASIS. Web Services Business Process Execution Language Version 2.0. 2002

Ref. 2 Churches D, Gombas G, Harrison A, Maassen J, Robinson C, Shields M, Taylor I, Wang I. Programming Scientific and Distributed Workflow with Triana Services Concurrency and Computation: Practice and Experience (Special Issue: Workflow in Grid Systems). 2006; 18(10):1021–37

Ref. 3 Oinn T, Addis M, Ferris J, Marvin D, Senger M, Greenwood M, Carver T, Glover K, Pocock M R, Wipat A, Li P. Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. Bioinformatics. 2004; 20(17):3045–54

Ref. 4 Hollingsworth D. Workflow Management Coalition: the Workflow Reference Model. Technical report, Workflow Management Coalition. 1995

Ref. 5 Shields M. Control- Versus Data-Driven Workflows. Workflows for E-science: Scientific Workflows for Grids I.J. Taylor, et al., Editors, Springer-Verlag. 2007; 167-73

Ref. 6 Yu J, Buyya R. A taxonomy of workflow management systems for grid computing. Journal of Grid Computing – GRID. 2005; 3(3-4):171-200

Ref. 7 Barga R, Gannon D. Scientific versus Business Workflows. Workflows for E-science: Scientific Workflows for Grids, I.J. Taylor, et al., Editors, Springer-Verlag. 2007;9-16

Ref. 8 Deelman E, Gannon D, Shields M, Taylor I. Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems. 2009; 25:528–40

Ref. 9 Rumbaugh J, Jacobson I, Booch G. Unified Modeling Language Reference Manual. Addison-Wesley. 2004

Ref. 10 Petri C A. Kommunikation mit Automaten" PhD thesis, Institut fur Instrumentelle Mathematik, Bonn. 1962

Ref. 11 Zhao Y, Wilde M, Foster I. Virtual Data Language: A Typed Workflow Notation for Diversely Structured Scientific Data. Workflows for E-science: Scientific Workflows for Grids, I.J. Taylor, et al., Editors, Springer-Verlag. 2007; 258-75

Ref. 12 Gil Y. Workflow Composition: Semantic Representations for Flexible Automation. Workflows for E-science: Scientific Workflows for Grids, I.J. Taylor, et al., Editors, Springer-Verlag. 2007; 244-57

Ref. 13 Glatard T, Sipos G, Montagnat J, Farkas Z, Kacsuk P. Workflow-Level Parametric Study Support by MOTEUR and the P-GRADE Portal. Workflows for E-science: Scientific Workflows for Grids, I.J. Taylor, et al., Editors, Springer-Verlag. 2007, 279-299

Ref. 14 Fahringer T, Prodan R, Duan R, Hofer J, Nadeem F, Nerieri F, Podlipnig S, Qin J, Siddiqui M, Truong H, Villazon A, Wieczorekm M. ASKALON: A Development and Grid Computing Environment for Scientific Workflows. Workflows for E-science: Scientific Workflows for Grids, I.J. Taylor, et al., Editors, Springer-Verlag. 2007; 279-99

Ref. 15 Zhao Z., van Hooft P, van Oudenaarde P, Terpstra F, Belloum A, Korkhov V, Sloot P, HertzbergerIncluding L The state of the art scientific workflow management systems in an e-Science environment. Proceedings of 1st IEEE International Conference on e-Science and Grid Computing. 2005 Dec 5-8; Los Alamitos, CA, US. e-Science 2005, IEEE CS Press

Ref. 16 Couvares P, Kosar T, Roy A, Weber J, Wenger K. Workflow Management in Condor. Workflows for E-science: Scientific Workflows for Grids, I.J. Taylor, et al., Editors, Springer-Verlag. 2007; 357-75

Ref. 17 Deelman E, Singh G, Su M H, Blythe J, Gil Y, Kesselman C, Mehta G, Vahi K, Berriman G B, Good J, Laity A, Jacob J C, Katz D. Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems. Scientific Programming Journal. 2005; 13(3):219–37

Ref. 18 Gil Y, Ratnakar V, Deelman E, Mehta G, Kim J. Wings for Pegasus: Creating large-scale scientific applications using semantic representations of computational workflows. Proceedings of the 19th Annual Conference on Innovative Applications of Artificial Intelligence, IAAI. 2007 July 22-26; Vancouver, British Columbia, Canada

Ref. 19 Jihie K, Deelman E, Yolanda G, Gaurang M, Varun R. Provenance trails in the Wings-Pegasus system. Concurrency and Computation: Practice & Experience - The First Provenance Challenge. 2008; 20(5): 587-97

Ref. 20 Taylor I, Wang I, Shields M, Majithia S. Distributed Computing with Triana on the Grid. Concurrency and Computation: Practice and Experience. 2005;17(9):1197–214

Ref. 21 Belhajjame K, Embury S M, Paton N W, Stevens R, Goble C A. Automatic annotation of Web services based on workflow definitions. ACM Trans. Web, 2008; 2(2):1–34

Ref. 22 Missier P, Soiland-Reyes S, Owen S, Tan W, Nenadic A, Dunlop I, Williams A, Oinn T, Goble C. Taverna, Reloaded. Lecture Notes in Computer Science. 2010; 6187/2010: 471-81

Ref. 23 Fahringer T, Qin J, Hainzer S. Specification of Grid Workflow Applications with AGWL: An Abstract Grid Workflow Language. International Symposium on Cluster Computing and the Grid (CCGRID). 2005 May 9-12; Cardiff, UK; 2:676–685. IEEE Computer Society Press, New York; 2005

Ref. 24 Laszewski von G. Java CoG Kit Workflow Concepts for Scientific Experiments. Technical Report, Argonne National Laboratory, Argonne, IL, USA. 2005

Ref. 25 Laszewski von G, Hategan M. Java CoG Kit Karajan/GridAnt Workflow Guide Technical Report, Argonne National Laboratory, Argonne, IL, USA. 2005

Ref. 26 Laszewski von G, Hategan M, Kodeboyina D. Java CoG Kit Workflow. Workflows for E-science: Scientific Workflows for Grids, I.J. Taylor, et al., Editors, Springer-Verlag. 2007; 340-56

Ref. 27 Altintas I, Berkley C, Jaeger E, Jones M, Ludascher B, Mock S. Kepler: an Extensible System for Design and Execution of Scientific Workflows. 16th International Conference on Scientific and Statistical Database Management (SSDBM). 2004 June 21-23, Santorini Island, Greece; 423–424

Ref. 28 McGough S, Young L, Afzal A, Newhouse S, Darlington J. Workflow Enactment in ICENI. UK e-Science All Hands Meeting. 2004 Sept, Nottingham, UK. IOP Publishing Ltd, Bristol, UK; 894-900

Ref. 29 McGough S, Young L, Afzal A, Newhouse S, Darlington J. Performance Architecture within ICENI. UK e-Science All Hands Meeting. 2004 Sept, Nottingham, UK. IOP Publishing Ltd, Bristol, UK; 906-11

Ref. 30 Van der Aalst W M P, Ter Hofstede A H M. YAWL: Yet Another Workflow Language. Technical Report, Queensland University of Technology, Brisbane. 2002

Ref. 31 Young L, McGough S, Newhouse S, Darlington J. Scheduling Architecture and Algorithms within the ICENI Grid Middleware. UK e-Science All Hands Meeting. 2003 Sept, Nottingham, UK. IOP Publishing Ltd, Bristol, UK; 5-12

Ref. 32 Mayer A, McGough S, Furmento N, Lee W, Newhouse S, Darlington J. ICENI Dataflow and Workflow: Composition and Scheduling in Space and Time. UK e-Science All Hands Meeting. 2003 Sept, Nottingham, UK. IOP Publishing Ltd, Bristol, UK; 627-634

LIST OF KEY WORDS/ABBREVIATIONS

WfMC	Workflow Management Coalition
ACID	Atomicity, Consistency, Isolation and Durability (the abbreviation is used to describe a particular type of transactions)
API	Application programming interface
QoS	Quality of Service
SWMS	Scientific Workflow Management systems
RDF	Resource Description Framework
OPM	Open Provenance Model
QoS	Quality Of Service
UML	Unified Model Language
VDL	Virtual Data Language