VLE-WFBus: a scientific workflow bus for multi e-Science domains

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

Abstract

In e-Science, a Grid environment enables data and computing intensive tasks and provides a new supporting infrastructure for scientific experiments. Scientific workflow management systems (SWMS) hide the integration details among Grid resources and allow scientists to prototype an experimental computing system at a high level of abstraction. However, the development of an effective SWMS requires profound knowledge on both application domains and the network programming, and is often time consuming and domain specific. Integrating mature implementations of domain specific SWMS improves reusability of workflow resources and promotes a generic framework for different e-Science domains. In this paper, we discuss different options to derive a generic workflow management system from domain specific implementations, and propose a workflow bus based solution, called VLE-WFBus. Legacy SWMSs are wrapped as federated components and are loosely coupled as one workflow system via a runtime infrastructure. An agent based prototype is presented; the integration among different workflow management systems has been demonstrated.

Keywords: Grid, e-Science, Scientific Workflow.

1 Introduction

Grid environments greatly enhance scientific experiments; not only data and computing intensive experiment processes become feasible, but also collaborations involves large scale resources and people are enabled. Promoted by Grid infrastructure, a new paradigm for scientific research: e-Science [1] emerges. An important service in e-Science is a Scientific Workflow Management System (SWMS) which manages experiment processes, automates the routines, and hides the integration details [2–4].

During the past decade, a number of SWMSs have been developed. DAGMan included in CondorG can schedule concrete computing tasks according to a DAG (directed acrylic graph) based flow description [5]. On top of DAG, Pegasus realises a planning layer for mapping abstract workflow processes onto concrete computing tasks [6]. Kepler inherits a visual programming interface and a set of computing models from a problem solving environment called Ptolemy and supports rapid prototyping of computing processes in a scientific experiment [7, 8]. Aiming at integrating service based resources, Taverna implements a web service composition language called Scufl and a graphical user workbench [9]. In our early paper [3, 10, 11], we distinguished four important functional components from a SWMS: 1) a model for describing workflow processes and their dependencies, 2) an engine for executing workflows, 3) a set of components (resources) for assembling a workflow, and 4) different levels of user support in a workflow such as for composition, runtime control and data provenance.

A generic SWMS improves the reuse of workflow components and promotes the knowledge sharing between different domains, moreover it reduces the learning cost for different systems. However, the diversity in the application domains and target utilisation scenarios in e-Science result different guises of SWMSs which are often domain specific. For instance, Taverna is a product of myGrid for bio-science domains; Pegasus was originally developed for data intensive applications in high energy physics domain. Some systems claim to be generic; however, they have been only applied in specific applications, such as Kepler and Triana.

The differences in the architecture of workflow components, e.g., web services (in Taverna) and actor/director architecture (in Kepler), and in the models of workflow logic, e.g., MoML (Kepler), Scufl (Taverna), task graph (Triana), hamper the generality of these systems for different workflows. In this paper, we discuss different options to derive a generic workflow system from specific SWMS implementations and propose a workflow bus paradigm for an e-Science environment. The research is carried out in the context of Dutch Virtual Laboratory for e-Science (VL-e) project [12].

This paper is organised as follows. First, we introduce the context of the research and then discuss the requirements on generic workflow support. After that we propose a workflow bus approach and present an agent based prototype.

2 Generic workflow support for e-Science applications

VL-e is an e-Science project driven by applications; it aims to realise a Grid enabled generic framework via which scientists from different domains[1] can share their knowledge and resources, and perform domain specific research. One of the core ideas is to identify the common characteristics of scientific experiments in different domains and to abstract the support for these common issues into a generic framework. In this section, we review the lessons learned from VL-e applications and discuss the requirements for generic workflow services.

2.1 Research context and missions

On top of the Grid infrastructure, the generic VL-e framework proposes two levels of abstraction: the generic components in each specific domain are distinguished as an *application layer*, and then the generic components in the application level which can be used in different domains are included in a *Virtual Lab layer*. Scientific workflow management is one of the important services in the VL-e generic framework. By effectively reusing existing workflow systems, the generic VL-e framework aims to provide a coherent environment for supporting domain specific experiments, and for sharing information and transferring knowledge among these experiments. A two phase- approach is being adopted. First, a number of existing workflow systems will be recommended to VL-e application scientist according to their specific use cases; and then, based on the lessons learned from these systems a generic VL-e wide workflow system will be implemented. In the next section, we will first take a look at VL-e applications and requirements on workflow support for e-Science applications.

2.2 VL-e applications and requirements on workflow support

Life science is an important field in the VL-e project; it currently has three sub programs(SPs) in bio-informatics, bio-medical and bio-diversity respectively. The bio-informatics SP focuses on semantic integration among different data sources. One of the typical use cases is to integrate genome distribution (chromosome locations) of histone and transcription factor and to unravel the genetic background of special phenomena, e.g., diseases. Supporting large scale data integration and enabling interactively data annotation and discovery are highly demanded in the workflows in this SP [13]. The bio-medical SP is interested in automating the routines in analyzing complex medical images. Scheduling massive tasks and steering the computation are a main concern in the experiments. Moreover, storing and accessing large volume and distributed medical data with different levels of security and authorization control are also important issues [14]. In the bio-diversity SP, tuning models for different eco- phenomena and using the models for forecasting is a main experiment scenario. Farming the massive computing tasks and optimizing the parameter space are the important requirements [15].

In addition to life science, the VL-e project also has other domains; however, these VL-e application domains share similar requirements on workflow support. After reviewing 13 existing workflow systems, we learned no single system can cover all the features demanded by the VL-e application SPs: *visual interface, knowledge infrastructure for semantic based operations, job farming, parameter sweeping* and *human in the loop computing*. We thus recommended VL-e application scientists four complementary systems which together provide most of the features: Taverna, Triana, Kepler and VLAM-G (a system developed in an early project of VL-e). A detailed requirement analysis can be found in [12].

2.3 Lessons learned

To tackle specific application use cases, a number of small project teams are established. Starting with legacy code from early projects, the first exercise of these project teams is to incorporate the legacy code as part of a workflow using a recommended SWMS. Web services are promoted as the standard technology to wrap and integrate workflow process. It is assumed that such standardized interface can make the integration independent from the engine of the workflow system.

Shortly after these projects start, several application SPs complain about problems on exporting web services they developed for one SWMS to another when they want to try a new system to get mission functionality. The mechanisms for handling different data types are a common problem, e.g., serializing string array type between web services via a SWMS. Different models on workflow logic are another problem. Because of the difficulties in importing workflows between different systems, including functionality from different SWMSs to serve one workflow is highly demanded.

In the next section, we will discuss different options to derive a workflow system from legacy SWMS implementa-

[1] Currently six applications are included in VL-e: food informatics, medical diagnosis and imaging, bio-diversity, bio-informatics, high energy physics, and tele-science.

tions.

2.4 Optional approaches to generic workflow support

In principle, one can have three options to derive a generic workflow system from specific legacy SWMS implementations, as shown in Fig. 1.

1. An *abstract* approach, in which mature and generic functionality from different SWMSs are abstracted and implemented in one system.

2. An *extend* approach starts from a system which has most close functionality to an ideal generic system. It extends the system with the missing functionality which is available in other systems.

3. The third option is an *aggregate* approach; it aggregates legacy SWMSs via a software infrastructure and allows the functionality from the legacy systems to be utilized as one coherent system.

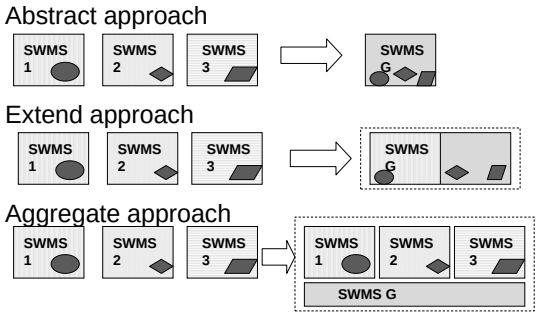

Figure 1. Optional approaches to derive a generic workflow system.

Each approach has advantages but also exhibits certain disadvantages. Ideally, the *abstract* approach can realise a *perfect* system for different domains. However, it requires the set of the systems being abstracted cover all the generic functionality demanded by different domains. This situation is unlikely to be true in the forecast-able future, since most of these systems are developed by academic community and application driven. The *extend* approach allows the generic system to be developed incrementally. However, it will also make the incorporation depend on the legacy system; the development of the extension has a high risk to be unstable if the system being chosen is not mature or in rapidly evolving. The third approach considers the legacy systems as black boxes and directly couples them into one meta-workflow environment, which can maximize the reuse of these existing systems. However, this approach requires the legacy systems provide flexible engine level API for the integration.

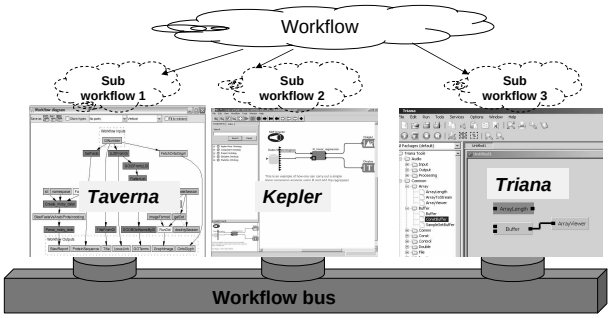

Figure 2. A bus paradigm for different workflow systems.

A number of reasons make us finally choose the aggregate approach. First, as products of academic projects, most of the existing SWMSs are in their evolution and unstable and their system functionality is often frequently updated; utilizing them as at function level can benefit the effort from the developer communities of these systems. Second, what we learned from the VL-e application use cases is that none of the existing workflow systems provide all the functionality demanded by application domains. Aggregating different SWMSs allows an application benefit the functionality from different systems as a whole. By comparing these approaches, we can see the aggregation approach in practice is the most feasible one.

A workflow bus architecture is proposed in the next section.

3 A workflow bus paradigm

The basic idea of a *workflow bus* is to wrap a number of popular and relative mature legacy SWMSs as federated components, and to loosely couple them as one meta workflow system using a software bus. Figure 2 shows a conceptual diagram, in which Taverna, Kepler and Triana are coupled.

In the context of *workflow bus*, we call a workflow being executed by a wrapped SWMS as a *sub-workflow*, the workflow being coordinated by the *workflow bus* as a *high level workflow* or a *workflow* in short. The execution of a *workflow* is called a *study*, and the execution of a sub-workflow is called a *sub-study* or a *scenario*. In the rest of the section, we will discuss the function of the workflow bus from the perspectives of workflow composition, runtime control and user support.

3.1 Multi level workflow modelling

In a workflow bus, different levels of workflow processes in the lifecycle of a scientific experiment are modelled. *Sub-workflows* are encapsulated as processes in a high level workflow, which hides the diversities in the models in sub-workflows. The *workflow bus* models the data and control dependencies among sub-workflows using data objects; here, the objects can be any types of data entities, e.g., input/output files, a set of parameters, or control configurations. The *workflow bus* provides mechanism to distribute and update data objects. In additional to the workflows in scientific experiments, the workflow bus also takes care of the processes involved in handling authentication, authorization and accounting related issues. The execution of this type of processes are handled as temporal workflows which are often invisible at the user level, however it is important when the workflow bus couples different levels of e-Science resources which cross different organizations.

3.2 Workflow execution

At runtime, federated SWMSs execute sub workflows, and the workflow bus coordinates runtime behaviour of these SWMSs in a high level workflow using a suitable computing model. A workflow bus employs two types of components to execute a workflow. *Scenario managers* wrap legacy SWMSs and plug them to the workflow bus. A *study manager* orchestrates a number of *scenario managers* and executes a high level workflow. To coordinate sub-workflows, a study manager implements different computing models, since the execution semantics of sub-workflows might be different, e.g., driven by data, time or events, for instance when a workflow involves discrete simulation. The *study manager* realises intelligence to adapt the workflow computing model to meet different requirements for sub workflows.

3.3 Different levels of user support

A workflow bus supports scientists at different phases of a scientific experiment: *composition, runtime control* and *post processing*. Tools inherited from legacy SWMSs will be used to compose sub-workflows, and the workflow bus also provides an environment, preferably with GUI, for assembling sub-workflows as a workflow. A long term goal will support automatically discovering different types of resources, e.g., meaningful workflows from different SWMSs, and making intelligent planning on assembling and workflow composition. At runtime, human in the loop computing is an important support for scientists to explore problem space in an experiment. In the scope of a sub-workflow, the interaction support provided by the legacy SWMS is inherited; in the scope of high level workflow, workflow bus also provides interface for steering the execution of sub-workflows. The provenance of a scientific workflow is emerging as a crucial mechanism for post analyzing experiment results. The provenance of sub-workflow relies on the inherent functionality of different legacy SWMSs; not all of these systems support the provenance. Depends on the legacy SWMS, the scenario manager can obtain execution state from the engines; based on these states, the workflow bus will conduct certain type of provenance of the high level workflow.

3.4 Usage in e-Science applications

A workflow bus can be utilised in e-Science applications in a number of cases. For instance, when a workflow requires resources or functionality from different SWMSs, a workflow bus can be employed to couple the SWMSs that can provide needed functionality and resources to serve the workflow. Another example, when sweeping or optimising workflow configurations, a workflow bus can execute a sub-workflow in multiple instances with different input conditions (data). Finally, SWMS specific resources can be in principle shared by another SWMS through a workflow bus when a system specific resource is invoked via the engine of the SWMS.

4 Prototype and experimental results

In this section, we discuss design considerations for a workflow bus and present an agent based prototype.

4.1 Design considerations

A suitable middleware for wrapping and integrating SWMSs and a flexible model for assembling and executing sub-workflows are essential to realize a workflow bus.

In principle, any middleware which can couple distributed components can be an option for developing workflow bus, e.g., web services, object-oriented middleware, multi-agent framework, or even an existing workflow system. Basically, the middleware has to handle transmissions of different types of the data and messages, such as workflow descriptions, data objects and execution states, among scenario managers. Object oriented middleware, e.g., CORBA [16] and HLA [17], only provides services for distributing objects or messages, and does not directly support the inclusion of semantic information from different domains. A workflow system integrates processes with specialized component architecture; such component architecture is normally based on data ports based interface, and gives little freedom to realise complex interactions between scenario managers. Web services interface is emerging as a

standardized interface to couple distributed resources; however, the SOAP based communication protocol of these predeployed services currently does not directly include semantic support. Multi agent framework advances OO middleware and provides feasibility for enabling semantic information between distributed components, such as in the Agent Communication Language in FIPA agents [18].

Most of the existing SWMSs consider data flow based execution as the default computing model. An extensive study on computing models between components has been done in the field of architecture simulation, such as in Ptolemy [19]. In Ptolemy, a set of execution models, e.g., Static Data Flow, Continuous Time and Process Net are implemented. These execution models are not originally proposed for Grid workflows, however it rich semantics give freedom to customize the execution model for a high level workflow. Part of the models have already been tried in Kepler.

A workflow bus is prototyped using JADE [18] which is a FIPA compliant agent framework and Ptolemy.

4.2 VLE-WFBus: a workflow bus prototype

VL-e Workflow Bus (VLE-WFBus) implements a *study manager*, a number of *scenario managers*, and a *user interface*. The *study manager* and *scenario managers* are realized as JADE Agents. The *user interface* is implemented using Vergil [19] a GUI provided by Ptolemy. To benefit the rich set of computing models provided by Ptolemy, the assembling language in Ptolemy called MoML is adopted for high level workflows in the prototype. Fig. 3 shows the basic architecture of the prototype.

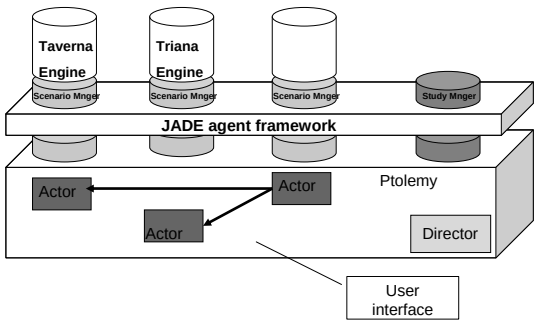

Figure 3. The basic architecture for VLE-WFBus.

In the user interface, *scenario managers* are represented as *actors* and the study manager is viewed as a *director*. In the current implementation of Ptolemy, the data tokens between actors are handled by the local director in the node where Vergil is launched. To decouple the data handling from the user interface, we added a separate agent layer to coordinate the execution between scenario managers and the plugged workflow engines.

4.3 Experiments

The prototype is based on PtolemyII 5.01 and JADE 3.3, and *scenario managers* for Taverna and Triana are tested. A number of experiments of VLE-WFBus have been performed on the Dutch DASII environment [20]. A simple workflow which contains two parts: one in Taverna from a microarray analysis workflow, and one in Triana for imaging processing. The images produced by the Taverna workflow will be passed to Triana workflow.

Using the test case, we studied performance characteristics of the prototype. Sending messages between scenario managers is important for study manager and scenario managers to coordinate the workflow execution. Most of the messages are small, e.g., for execution control, but it can also be large when the message contains a description of a sub-workflow. We measured the delay for sending different size messages between a study manager and a scenario manager, which are executed on two separated nodes with a fast Ethernet connection. The operating system on the nodes is Redhat Linux, and the data transmission between agents is reliable. The message starts from 4 bytes and the size doubles after each time step; in total 50 measurements have been made. Fig. 4 shows the results. We can see the latency between two agents is less 0.01 seconds and the delay increases nearly linearly after the size is larger than 256K.

Between the same nodes, we also measured the message delay using SOAP (AXIS 1.3). Comparing to SOAP, we can see study/scenario managers have a much better latency and transmission delay; moreover, SOAP is difficult to handle large size messages because of its large consumption on memory.

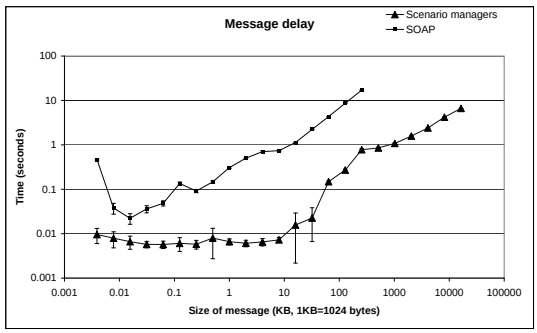

Figure 4. Message delay between scenario managers. The error bars show standard deviations.

We measured the overhead for executing a workflow us-

ing a scenario manager. At runtime, a scenario manager sends the sub-workflows to federated SWMS engines for execution; if the sub workflows do not require user interaction, no GUI will be shown. We measured the time cost for both Taverna and Triana sub-workflows in two situations: using a scenario manager to control the workflow engines, or using the GUI user environment from Taverna workbench and Triana environment. Fig. 5 the time cost for an imaging processing workflow in Triana; we can see a 10% to 20% improvement is gained. Because a scenario manager does not require the large memory consumption which is needed by the legacy GUI environment, a workflow bus is able to improve the performance of a workflow.

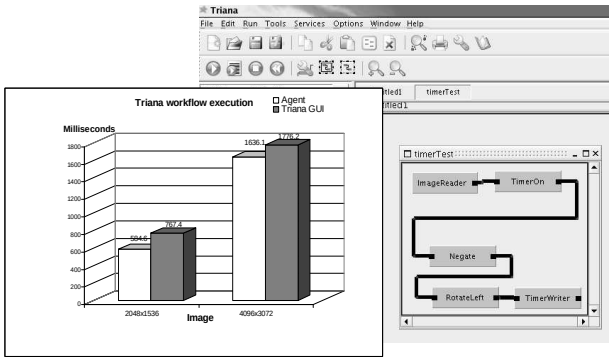

Figure 5. Overhead of the execution workflow using scenario managers.

Finally, we also looked at the scalability of the scenario managers. We executed the test workflow in different configurations: one study manager with 2, 4, 8 and 16 scenario managers. In the experiment, each scenario manager is executed on a separate node, and only Triana scenario managers are used. The time cost for the study manager to coordinate all scenario mangers finish the execution is measured. In the experiment, we did the measurement when all the agents have been loaded, which means the overhead for loading a scenario manager is not included. Fig. 6 shows the average of 10 measurements. We can see within the deviation range, the time costs are comparable; the prototype of VLE-WFBus is scalable.

5 Discussion

In e-Science, the development of workflow management systems faces two challenging issues. On one hand, the domain specific experiments from different applications require customized solutions in workflows for particular problems; on the other hand, to enable knowledge transfer and information sharing between different domains, a generic workflow solution is also demanded. An integration solu-

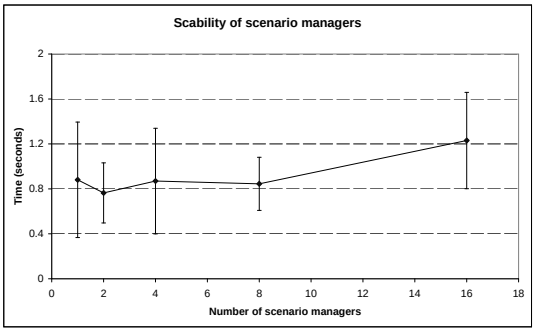

Figure 6. Scalability of scenario managers. The error bars show standard deviations.

tion to existing SWMSs is essential to reuse the different workflow resources and services and to realise a generic framework. In this section, we briefly review related work, and discuss the shortcoming of our approach.

5.1 Related work

Integrating different SWMS implementations and providing interoperability among the systems have been realised as an important mechanism to share the latest results among different research efforts in the e-Science community. "Link up project" [21] was one of such initiatives. In the link up project, researchers of different workflow systems, such as Pegasus, Taverna and Triana establish a forum to share their development experiences and try to realise interoperability among their implementations. In this context, mainly the resource level interoperability between SWMSs is studied, namely allowing one workflow system to invoke the components which were originally developed for another workflow system. Kepler has provided interface to systems such Nimroad [22] and web services components. Compared to the work in this paper, we take a different vision on the interoperability issue: engine level coupling. We consider a legacy workflow system as black box, and target at including them in one meta workflow.

The Ptolemy framework has also been applied in other SWMSs, e.g., Kepler. Kepler inherits four computing models from Ptolemy and implements a set of scientific computing actors. One of the reasons that we add a separate agent layer for remote execution is because currently neither Ptolemy nor Kepler support distributed execution of the workflow. In Kepler, remote Grid jobs are handled through special actors, the communication between actors is still local. We are also aware of the effort in the Kepler community to advance the current model with distributed execution.

In e-Science, agent technologies have been highlighted by researchers from both Grid computing and AI communities [23]. Agent technologies are being utilised to develop

specific SWMS in a number of projects [24–26]. In our work, we intend to benefit the agent oriented engineering and multi agent framework, and explore the semantic support for integrating distributed resources and data.

5.2 Shortcoming of the method

The design of workflow bus also has constraints. Preferably, a SWMS with a decoupled engine and provides necessary API interface for execution control can be wrapped and plugged in the workflow bus. The diverse API provided by different SWMSs requires customised design of scenario manager. Changes in the API may also influence the interface of scenario managers.

6 Conclusions

A main mission of workflow researchers in the VL-e project is to effectively reuse existing SWMSs and provide an integration solution to including them in one coherent e-Science environment. In this paper, we first analyzed the requirement for the generic workflow support for VL-e applications, and then proposed a workflow bus paradigm for different e-Science applications. We presented an agent based prototype and discussed some experimental results.

The implementation of VLE-WFBus is still in its early stage; issues, such semantic level integration of workflows, are still under-studying. However, from the experiments we have performed, we can at least draw the following conclusions.

1. Workflow bus is a feasible approach to integrate workflow systems and to realise a generic framework for different e-Science application domains.

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

3. Ptolemy distinguishes different computing models for system component and provides flexible user interface, which can be a good starting points for developing Grid based workflow systems.

7 Future work

In the theme of workflow bus, we are going to first obtain insights in suitable execution models for different types of Grid workflows. Handling data intensive applications and scheduling distributed computing tasks via workflow bus is another important issue. A number of scenario managers will be prototyped for different SWMSs; application use cases in the VL-e project will be used to validate the implementation. Finally, we will also study the data provenance support in workflow bus.

Acknowledgment

This work was carried out in the context of the Virtual Laboratory for e-Science project (www.vl-e.nl). Part of this project is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ).The authors of this paper would like to thank all the members in the VL-e SP2.5.

References

[1] Tony Hey and Anne E. Trefethen. The UK e-science core programme and the grid. *Future Generation Computer System*, 18(8):1017–1031, 2002.

[2] B. Ludascher, K. Lin, S. Bowers, E. Jaeger-Frank, B. Brodaric, and C. Baru. Managing scientific data: From data integration to scientific workflows. *GSA Today, Special Issue on Geoinformatics*, 2005.

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

[4] I. Altintas, S. Bhagwanani, D. Buttler, S. Chandra, Z. Cheng, M. Coleman, T. Critchlow, A. Gupta, W. Han, B. Ludcher L. Liu, R. Moore andA. Shoshani C. Pu, and M. Vouk. A modeling and execution environment for distributed scientific workflows. In *SSDBM*, 2003.

[5] DAGMan. Directed acyclic graph manager. In *http://www.cs.wisc.edu/condor/dagman/*, 2005.

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

[7] Ilkay Altintas, Chad Berkley, Efrat Jaeger, Matthew Jones, Bertram Ludäscher, and Steve Mock. Kepler: An extensible system for design and execution of scientific workflows. In *SSDBM*, pages 423–424, 2004.

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

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

[10] Zhiming Zhao, Adam Belloum, Hakan Yakali, Peter Sloot, and Bob Hertzberger. Dynamic workflow in a grid enabled problem solving environment. In *Proceedings of the 5th International Conference on Computer and Information Technology (CIT2005)*, pages 339–345, Shanghai, China, September 2005. IEEE Computer Society Press.

[11] Zhiming Zhao, Adam Belloum, Peter Sloot, and Bob Hertzberger. Agent technology and scientific workflow management in an e-science environment. In *Proceedings of the 17th IEEE International conference on Tools with Artificial Intelligence (ICTAI05)*, pages 19–23, Hongkong, China, November 14th-16th 2005. IEEE Computer Society Press.

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

[13] Marco Roos Lennart Post, M. Scott Marshall. The histone code case: A semantic web approach to data integration. submitted, 2006.

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

[15] Judy Shamoun-Baranes, Henk Sierdsema, Emiel van Loon, Hans van Gasteren, Willem Bouten, and Floris Sluiter. Linking horizontal and vertical models to predict 3d + time distributions of bird densities. In *INTERNATIONAL BIRD STRIKE COMMITTEE*, 2005.

[16] S. Vinoski. CORBA: Integrating diverse applications within distributed heterogeneous environments. *IEEE Communications Magazine*, February, 1997.

[17] Defence Modelling and Simulation Office (DMSO). High Level Architecture (HLA). In *https://www.dmso.mil/public/transition/hla/*, 2003.

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

[19] Ptolemy Hompage. http://ptolemy.berkeley.edu/. 2006.

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

[21] Link up project. http://www.mygrid.org.uk/linkup/. In *Submitted*, 2006.

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

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

[24] Liangzhao Zeng, Anne Ngu, Boualem Benatallah, and Milton O'Dell. An agent-based approach for supporting cross-enterprise workflows. In *ADC '01: Proceedings of the 12th Australasian conference on Database technologies*, pages 123–130. IEEE Computer Society, 2001.

[25] Junwei Cao, Stephen A. Jarvis, Subhash Saini, and Graham R. Nudd. Gridflow: Workflow management for grid computing. In *IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 198–205, May 2003.

[26] M. Brian Blake. Agent-based communication for distributed workflow management using jini technologies. *International Journal on Artificial Intelligence Tools*, 12(1):81–99, January 2003.