# Dynamic Workflow in a Grid Enabled Problem Solving Environment

Zhiming Zhao Adam Belloum Hakan Yakali Peter Sloot Bob Hertzberger
Informatics Institute, University of Amsterdam
Kruislaan 403, 1098SJ, Amsterdam, the Netherlands
{zhiming|adam|yakali|sloot|bob}@science.uva.nl

## Abstract

*In a Problem Solving Environment (PSE), a scientific workflow management system (SWMS) provides a meta environment for managing activities and data in scientific experiments, for prototyping experimental computing systems and for orchestrating the runtime system behaviour. A Grid infrastructure makes data and computing intensive experiments feasible in PSEs but also requires the management of workflow to support dynamics of the flow execution. A dynamic SWMS includes a human user in the runtime loop of a flow execution, and allows an engine to flexibly orchestrate a workflow according to the human decision and the runtime states of the environment. In this paper, we present our research in an ongoing project: Virtual Laboratory for e-Science (VL-e). An agent based solution is proposed to enhance an existing Grid enabled Problem Solving Environment framework called VLAM-G. The intelligence for problem solving strategies and for workflow orchestration is encapsulated as knowledge in two types of agents: study managers and scenario conductors.* [1]

**Key words:** Grid, Problem Solving Environment, Scientific Workflow, Multi Agent Systems, Petri net.

## 1. Introduction

In scientific research, a Problem Solving Environment (PSE) organises different software utilities, e.g., simulators, visualisation and data analysis tools, as a meta experimental environment and allows a scientist to assemble these utilities and customise their interaction for different purposes of experiments [10]. An important guise of such meta environment is a scientific workflow management system [21]. By explicitly modelling the dependencies between experiment processes, a scientific workflow management system (SWMS) orchestrates the runtime behaviour of involved resources according to a high level description. Data flow, control flow or Petri net based mechanisms have been used to model the scientific workflow [8, 17, 24, 34].

Grid environments couple heterogeneous resources, such as computing elements, storage devices and software components, and allow a group of trusted users (Virtual Organisations (VO)) to deploy the resources based on certain polices [12]. Utilising a Grid infrastructure, a Grid enabled PSE benefits VO wide resources and makes data and computing intensive experiments feasible. However, a number of reasons, which include 1) the increasing ambition of the user on benefiting the Grid computing capacity, 2) the discovery and utilisation of distributed software resources, and 3) the stability of the available Grid resources [30–32], also require a Grid enabled PSE to have additional features in its SWMS. One of these demanded features is to handle the *dynamic issues* in a workflow, such as the activities of the human users, the availability of resources, and adaptation of flow descriptions. A system that supports such enhanced workflow is called a *dynamic* SWMS.

Since the 1990s, scientific workflow and its support environments have become an important subject in the community of scientific computing; a large number of successful systems have been reported [3, 17, 20, 23, 25]. However, in most of these systems, the support for dynamic workflow has not been explicitly addressed or only in a limited extent. From the flow modelling point of view, data based dependencies are still the most popular mechanism to describe a workflow, which is straightforward for many data stream based experiments, but is not sufficient for modelling the interaction dynamics when human is in the loop. Apart from it, the interpretation mechanism of the engine assumes that all the dependencies between resources are predefined in a flow; the runtime adaptation of the flow description is not supported by most of the engines. When a PSE can only access limited computing power, little can be done about these shortcomings. Since the realisation of the data based interaction receives a much higher priority than the support for

[1]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).

dynamic workflow, in particular for data and computing intensive applications. In Grid enabled PSEs, facilitating the existing scientific workflow environment with the support for dynamic issues becomes an important research issue.

In this paper, we present our work on dynamic workflow. The research is carried out in the context of Virtual Laboratory for e-Science [28]. VLAM-G (Virtual Laboratory Amsterdam for Grid) environment [2], a Grid enabled PSE framework developed in a previous project[2], is currently used as the first prototype. The VLAM-G environment provides a user friendly interface for managing software components and for composing reusable experiment templates. But the modelling of the runtime experiments is based on data flow, which has limited concerns for the dynamic facts in the flow execution. This paper is organised as follows. First, we analyse the basic issues of dynamic workflow and review the related work. Next, we briefly describe the current state of the VLAM-G environment and propose an agent based flow control architecture. After that, we discuss the implementation issues.

## 2 Dynamic workflow and related work

In this section, we first analyse the runtime interaction in a SWMS and summarise the requirements for supporting a dynamic workflow, then we briefly review related work.

### 2.1 Runtime dynamics and dynamic workflow

In a PSE, a basic SWMS consists of four components: a workflow description, the software resources being utilised in a workflow, an engine orchestrating the system behaviour, and a middleware that realise the communication and binding between resources and between resources and the engine [3]. The interaction dynamics at run time can then be examined between every two components in the SWMS, as shown in Fig. 1. We can enumerate a number of desired supports for the dynamic issues.

*Flow composition and adaptation.* A flow can be composed with the support of automated discovery and assembling of available resources. At run time, the engine can adapt the content of a flow to meet the dynamic changes occurred in the deployed resources.

*Between a user and the flow engine.* At run time, a user is allowed to control the progress of the flow execution (VCR like operations: play, pause, stop, proceed and operations), to steer the execution of a flow, to interact with the other users with constraints defined in a flow, and to change or modify the flow description.

[2]Virtual Laboratory II.

[3]A workflow description contains not only experiment processes but also the meta data about the workflow context and the data interfaces. To simplify the discussion, we here omit optional components such as information management from the SWMS.

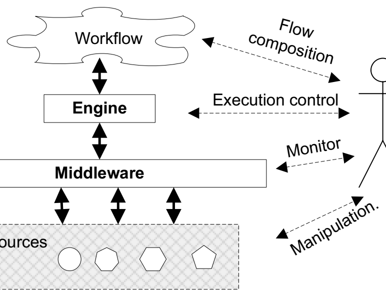

**Figure 1. Runtime interactions in a SWMS.**

*Between the flow engine, the resources and the middleware.* Desired supports include discovering software resources and schedule their execution according to the flow description, migrating computing tasks when it is necessary, e.g., due to the unavailability of computing elements in Grid, and generating temporal flow description due to the availability of software resources or computing infrastructure.

*Between a user and the resources.* A user is allowed to monitor the executing of software resources, and modify the behaviour of a component, e.g., tuning the control parameters, or interacting with an interactive visualisation component.

We can clearly see the main characteristics of these supports: *robustness*, *autonomy*, *flexibility* and *adaptability*. This leads to our opinion on the features of a dynamic SWMS:

1. **Support human-in-the-loop interaction**. At run time, a user is allowed not only to interact with the flow engine and the involved components, but also to adapt the description of a workflow.
2. **Automated resource discovery and mapping**. The mapping between the flow description and the underlying software resources needs to be transparent and automated. At run time, the engine is able to dynamically adapt the description of a workflow or to change the mapping between the workflow and the resources according to the states of the resources.
3. **Flexible flow execution**. The flow engine is able to adapt the high level execution strategy for orchestrating a workflow according to the characteristics of the problem domain or the requirements from the user.

### 2.2 Related work

The development of a dynamic SWMS is complex and highly interdisciplinary, not only the modelling of the interaction dependencies in a flow is difficult, but also the realisation of the runtime control between components is time consuming. A number of research lines can be distinguished from the related work of dynamic workflow.

The first line is about the issues of modelling a dynamic workflow. In the WORLDS environment, Bogia addressed the importance of local adaptation of particular workflow instances [5]. Jorgensen proposed to use *interaction* as the key framework to model flexible workflow [18]. A modelling language called *Workware* allows the inclusion of ambiguities in the models and also the dynamic interpretation of modelling elements.

The second one is related to the composition of flow. Dynamic flow composition has been addressed as an important support for utilising Grid services [7]. In [6], Bubak discussed an ontology based search and match mechanism for discovering and assembling Grid services.

The third one is about the execution strategies of workflow. Baggio [4] studied how the scheduling issues improves efficiency of the flow execution; he proposed a *guess and solve* technique to explicitly model the uncertain issues in the flow execution. Jin et al., [16] compared the round robin and load aware scheduling mechanisms; one of the conclusions is that the load-aware strategy is more suitable for the heterogeneous flow environment, and the round robin scheduling is better in the uniform structure. Load balance is another issue in the flow execution and management. Underlying middleware and the migration tools, such as Condor [1] and Dynamite [14], provide feasibility to dynamically migrate computing tasks.

The fourth line is about the human interaction in the flow execution. Interacting with a specific component in the flow, or controlling the progress of the flow execution using a sort of VCR actions have been included as inherent functionality of a number of flow engine e.g., Taverna [22] and VLAM-G [2]. It has been realised for almost a decade that computer supported collaborative working environments are an efficient mechanism to manage the interaction dynamics of organisation activities [11].

The research in this paper benefits the previous work and focuses on a generic solution to extend existing PSE to support dynamic workflow. The rest of the paper is organised as follows. We first give an overview of the research context: Virtual Laboratory for e-Science, and then propose an agent based architecture to provide dynamic workflow support.

## 3 Virtual Laboratory for e-Science and the VLAM-G environment

### 3.1 Background

Virtual Laboratory for e-Science (VL-e) is an e-Science research project in the Netherlands; it aims to realise a Grid enabled generic framework where scientists from different domains can share their knowledge and resources, and perform domain specific research. Currently, there are six domains are included: food informatics, medical diagnosis and imaging, bio-diversity, bio-informatics, high energy physics, and tele-science. One of the core ideas is to identify the common characteristics of scientific experiments in different domains and abstract the support for these common issues into a shared framework. VLAM-G, a Grid enabled PSE framework developed in the previous project of VL-e, is currently used as the first prototype of the shared framework. Fig. 2 shows the basic architecture.

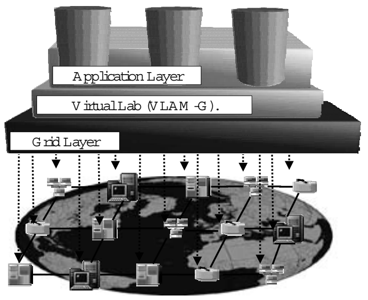

**Figure 2. The basic architecture of VL-e.**

The VLAM-G environment provides generic services for managing data and software resources, and for supporting scientific workflow. Using the VLAM-G environment, a user is allowed to perform his experiment location independently and to utilise Grid resources in his experiment transparently. Apart from it, the VLAM-G framework also integrates the information management services with the lifecycle of a scientific experiment [19].

In VLAM-G, the processes, *activities based on both regular lab instruments and on computing tasks* [4], of an experiment are explicitly modelled using three types of elements: *physical entities* which are the instruments to be used, *activities* to be performed by the scientists, and *data elements* which are the input/output of the activities. Using these building blocks, an experiment is modelled in three levels: abstract descriptions of the dependencies between experiment processes called *Process Flow Templates* (PFT)-, instantiations of a template called *Studies*-, and finally the actual flow of data analysis and computing activities in a *Study* called an *Experiment Topology*.

Using the GUI provided by the VLAM-G environment, a domain expert defines PFTs, and a scientist instantiates a PFT as a *Study*. The activity steps in a *Study* give the details of actual manual laboratory steps in the experiment or the computing tasks. A set of self-contained software entities, called *modules*, for performing computing tasks are

[4]Computing tasks include simulation, visualisation and data analysis.

described as an *Experiment Topology*. The VLAM-G environment provides a GUI for a scientist to describe a topology and to execute it via an engine (also called a Run Time System (RTS)).

At run time, a topology can be executed on the underlying Grid infrastructure via a resource manager in VLAM-G. A scientist can have two types of interactions with the experiment instance: tuning the parameters of a specific component via the VLAM-G interface, or manipulating a specific module when it is interact-able. A database infrastructure is employed to manage both the static and runtime information.

### 3.2 Shortcomings in the VLAM-G runtime environment

In the VL-e project, the development of the generic framework and the research of the domain specific applications are dependent but they have to be carried out in parallel. This is the main reason that the current VLAM-G framework starts from the most generic model to describe the constraints between software components: data dependencies. This model is straightforward for the data stream based experiments, but provides limited support for the dynamic issues in the flow control.

In the VL-e project, both the development team of the generic framework and the scientists from different domains realise the key role that a SWMS plays in an e-Science environment. The domain scientists in the VL-e community have described their requirements on the workflow support in a detailed wish list [29], which covers different levels of issues in the workflow support. Based on the wish list, we enumerate following shortcomings of the current VLAM-G environment:

1. Limited human interaction support. Human activity based interaction dependencies are not explicitly modelled in the current PFT model. A user can not steer the execution of an experiment at run time.

2. Limited adaptability. The execution control of the flow is not explicit; when a flow instance is submitted, the relation between modules (components) is fixed. The user can not modify the content of a topology when it is running.

3. Limited flexibility. A topology can be executed on Grid in a transparent way, however limited strategies are provided to adapt the execution according to the availability of the resources and the characteristics of the problem.

Because of these shortcomings, a scientist has to develop control intelligence inside the components when he wants to realise a complex scenario for his experiment, which hampers the reusability of both components and the experiment template. The key issue is that the current VLAM-G has limited capability for modelling complex workflows and for supporting the related runtime issues. An enhanced flow model and a flexible engine are needed. In the next section, an agent based solution is proposed.

## 4 An agent based solution: VL-e Workflow Conductor (VLWF-Conductor)

### 4.1 Basic idea

A solution is proposed to improve the current VLAM-G environment with the following functionalities:

1. Extending the data flow based model with capabilities of describing temporal interaction dependencies between a user and the system components.

2. Distinguishing the control for orchestrating activities of flow components and for high level execution strategies. These two levels of control are mixed in the current VLAM-G environment, which hampers the flexibility for flow execution.

3. Allowing a user not only to interact with the flow engine and the components and but also to adapt its flow content at run time.

Interaction dependencies can in principle be specified in a number of ways, e.g., activity diagrams, state charts and Petri Nets. Because of the well-established theoretical framework and more importantly the suitability for representing the common flow patterns [27], a Petri Net based approach is adopted as the mechanism to extend the current flow description.

Agent technologies provide a suitable approach to include control intelligence with the behaviour of a set of operations, therefore, we use them to encapsulate the control intelligence and to carry out the flow control.

A framework called VL-e Workflow Conductor (VLWF-Conductor) is proposed. An agent called *study manager* controls high level strategies of a scientific experiment. A number of agents called *scenario conductors* handle the interpretation of experiment scenarios of a workflow. The scenario conductors can also wrap different legacy flow engines through specialised interface. Flow description and the execution strategies are modelled as knowledge in the agents. The basic architecture of VLWF-Conductor is depicted in Fig. 3.

In the rest of this section, we will discuss its particular features.

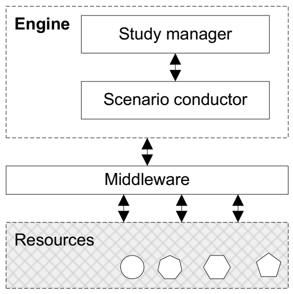

**Figure 3. The basic architecture of VLWF-Conductor.**

## 4.2 Agent definition

In our view, an agent is a software component which has an explicit model of the external world, a reasoning kernel to make decisions on the activity, and a number of sensors and effectors to interact with the external world. From a high level, an agent is an entity to encapsulate certain intelligence for behaviour control. At run time, agents can reside in different machines and can communicate via an agent framework.

## 4.3 Flow modelling

The concept of a Petri Net was originally developed by C. A. Petri in the 1960s. It has been a widely applied for modelling system behaviour, in particular concurrent activities. Place transition graphs are a sort of automata that can handle relations between conditions and the occurrence of events [9]. Basically, a place transition (PT) graph consists of three parts: a finite set of *places*, a finite set of *transitions*, and a finite set of relation links between places and transitions. In our earlier work, we implemented a PT graph based mechanism called *scenario net* [34] to model the interaction constraints between components in an interactive simulation system.

In a scenario net, transitions and places have unique names. Transitions are used to specify activities or nested scenario nets. When specifying an activity, a transition contains an action and a role name, in which the role is expected to perform the action at run time, and the role is also called the *responsible* role of the transition. When specifying a nested scenario net, a transition contains the name of a scenario net and a special action called *Do_Scenario*. Places and the links between places and their post sets are used to describe the conditions. A place is optionally associated with a set of expressions, named *place expressions*, which contain three subsets, for describing the initial conditions, control conditions and the state-modification rules of the place. Each link between a place and its post set is optionally associated a set of guard expressions.

A scenario net models the interactions using two types of dependencies between the roles' activities. The concurrency dependencies between them constitute the first type, which are represented as the relation links between places and transitions, and the tokens of the places. The second type of the dependencies is the specific conditions for each action, which are represented as the control expressions in the pre-set of the transition and in the links between the transition and its pre-set. The execution of a transition updates the state of the scenario net. The basic rules of the PT graph handle the concurrency dependencies between the activities by changing the marking of the net, and the execution rules for the place expressions update the control conditions for the activities.

In [33], we have demonstrated the application of scenario nets in modelling scenarios in interactive simulation systems. In VLWF-Conductor, scenario net is used as the basic mechanism to model the dynamic workflow.

## 4.4 Control intelligence encapsulation

In a scenario net, a sub scenario can be nested as a transition in a higher level scenario. The execution rules of a scenario role are encapsulated in a scenario conductor. The scheduling strategies for the flow execution are realised in the study manager. These scheduling strategies are related to the specific problem being studied, for instance, for a parameter studying problem, the study manager schedules multiple scenario conductors to execute a same scenario but with different parameters. At run time, a study manager can create multiple scenario conductors to perform the execution of different scenarios in parallel.

Another functionality of scenario conductor is to wrap legacy flow engine. One of the goals of the VLWF-Conductor is to reuse existing flow engine. Using VLWF-Conductor, a data flow based description is possible to included in the workflow, and be executed by a scenario conductor which interfaces with the existing VLAM-G engine.

## 4.5 Human in the loop flow execution

Human in the loop flow control is realised at different levels. In addition to interacting with a specific component, VLWF-Conductor also allows a human user:

1. to interfere with the study manager to make decisions on choosing scheduling strategies.
2. to interact with the scenario conductor to steer the execution of a scenario net of a flow. In a scenario net, a place can be associated with a number of user opinions; a user can choose one of these opinions via a user interface.

3. to collaborative work with the other users under the coordination of a scenario conductor.

In this section, we briefly discussed the design considerations of VLWF-Conductor. In the next section, we will put the VLWF-Conductor in the context of VL-e project and discuss the integration between it and the VLAM-G environment.

## 5 Implementation state

VLWF-Conductor will be implemented based on the existing framework of the VLAM-G; it will serve the high level domain specific applications as the generic support for managing runtime experiments and studies (see Fig. 2). VLWF-Conductor will be integrated with the VLAM-G framework at three levels: composition, orchestration, and resource management. In this section, we will discuss the integration issues and figure out the relation between VLWF-Conductor and the other ongoing research in VL-e.

The implementation of VLWF-Conductor is still in its initial stage. Jade, a FIPA compliant agent framework, is currently used as the agent environment. The reasoning procedures for choosing execution strategies and interpreting scenario net, and the maintenance of the world model are realised as agent activities, which are invoked periodically by the agent core. The reasoning module is realised using Prolog. Jade 3.3 [15] and SWI Prolog 5.4.7 [26] are currently used. The implementation is based on Java.

The communication between agents is handled by the Jade runtime environment; and the messages between scenario conductor and the computing resources are handled by the original Grid middleware of VLAM-G environment.

A workflow is described using a XML schema. A user friendly tool is under development. At run time, the location of the workflow is passed from an experiment manager to a scenario conductor. The scenario conductor parses the workflow as Prolog terms, and then applies its interpretation knowledge to execute it. Using the Jade framework, a scenario conductor can be instantiated in a local or remote machine.

## 6 Discussion and conclusions

In this paper, we reported our work on dynamic workflow in the VL-e project. We first discussed the interaction dynamics in a SWMS and the design requirements for supporting dynamic workflow. And then we analysed the shortcomings of the current VLAM-G environment and proposed an agent based solution.

In addition to the basic management services for workflow, a dynamic SWMS also has to implement the support for the dynamic issues. However, when a PSE can only access limited computing power, the support for dynamic workflow is often considered as a sort of *nice to have* features compared to the solutions to data storage and processing. Apart from it, a uniformed description of Grid resources and flow is required when realising automatic flow composition. It is unlikely to have a single solution to dynamic workflow in the near future.

From the work, we can at least conclude follows:

1. In Grid enabled PSEs, dynamic workflow merges as a desired support for managing experiment activities and for orchestrating the system behaviour.

2. The rich semantics of Petri nets can describe the interaction constraints not only from the perspective of data dependencies, as often used in *scientific workflow* systems, e.g., SciRun, Sculf, and GridAnt [3, 17, 23], but also from the concurrency relations between activities. It is a suitable paradigm for modelling dynamic workflow.

3. Agent technologies are a suitable solution to realise the control intelligence for flow control. Jade provide a uniformed framework for agent communication. The Java based implementation makes the integration with Grid middleware easy.

## 7 Future work

The development of VLWF-Conductor is still ongoing. It will be validated by applying test cases from different domains. One of the applications we are currently working on is the parametric modelling. The basic activities of computing and result comparison are described as a workflow. The user is allowed to make decision to optimise the parameter configuration.

A lesson learned from VLAM-G project is that scientists will not choose a novel architecture simply because it looks beautiful unless it can work with the existing ones and provide exciting new features [13]. An important future work is to study the mapping schemes between existing architectures and VLWF-Conductor so that legacy architectures can get benefit of the layered integration from VLWF-Conductor in an easy and efficient way.

## References

[1] Process migration. *ACM Comput. Surv.*, 32(3):241–299, 2000.

[2] H. Afsarmanesh, R. Belleman, A. Belloum, A. Benabdelkader, J. van den Brand, G. Eijkel, A. Frenkel, C. Garita, D. Groep, R. Heeren, Z. Hendrikse, L. Hertzberger, J. Kaandorp, E. Kaletas, V. Korkhov, C. de Laat, P. Sloot, D. Vasunin, A. Visser, and H. Yakali. VLAM-G: A Grid-based

Virtual Laboratory. *Scientific Programming: Special Issue on Grid Computing*, 10(2):173–181, 2002.

[3] K. Amin and G. von Laszewski. GridAnt: a Grid workflow system. In *Manual at http://www-unix.globus.org/cog/projects/gridant/*, February 2003.

[4] G. Baggio, J. Wainer, and C. Ellis. Applying scheduling techniques to minimize the number of late jobs in workflow systems. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 1396–1403, New York, NY, USA, 2004. ACM Press.

[5] D. P. Bogia and S. M. Kaplan. Flexibility and control for dynamic workflows in the worlds environment. In *COCS '95: Proceedings of conference on Organizational computing systems*, pages 148–159, New York, NY, USA, 1995. ACM Press.

[6] M. Bubak, T. Gubala, M. Kapalka, M. Malawski, and K. Rycerz. Grid service registry for workflow composition framework. In *Proceedings of International Conference on Computational Science, LNCS 3038*, pages 34–41. Springer, June 2004.

[7] D. Caragea and T. Syeda-Mahmood. Semantic api matching for automatic service composition. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 436–437, New York, NY, USA, 2004. ACM Press.

[8] J. G. Chin, L. R. Leung, K. Schuchardt, and D. Gracio. New paradigms in problem solving environments for scientific computing. In *Proceedings of the international conference of Intelligent User Interface*, San Francisco, 2002.

[9] L. Czaja. Place/transition Petri net evolutions: recording ways, analysis and synthesis. *Fundam. Inf.*, 51(1):43–58, 2002.

[10] J. R. R. Efstratios Gallopoulos, Elias Houstis. Computer as thinker doer: Problem-solving environments for computational science. *IEEE Computational Science and Engineering*, 2:13–23, 1994.

[11] C. S. Ellis. Many groupware products have recently been a workflow architecture to support dynamic change. *SIGOIS Bull.*, 15(2):23–27, 1994.

[12] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann; 2nd edition, 2003.

[13] L. O. Hertzberger. Introduction of VLAM-G and VL-E. In *Internal seminar*, 2004.

[14] K. A. Iskra, R. G. Belleman, G. D. van Albada, J. Santoso, P. M. A. Sloot, H. E. Bal, H. J. W. Spoelder, and M. Bubak. The Polder computing environment, a system for interactive distributed simulation. *Concurrency and Computation: Practice and Experience(Special Issue on Grid Computing Environments)*, 14(13–15):1313–1335, 2002.

[15] Jade. Jade agent development framework. In *http://jade.tilab.com/*, 2005.

[16] L. jie Jin, F. Casati, M. Sayal, and M.-C. Shan. Load balancing in distributed workflow management system. In *SAC '01: Proceedings of the 2001 ACM symposium on Applied computing*, pages 522–530, New York, NY, USA, 2001. ACM Press.

[17] C. Johnson, S. Parker, and D. Weinstein. Large-scale computational science applications using the SCIRun problem solving environment. In *Proceedings of Supercomputer*, 2000.

[18] H. D. Jorgensen. Interaction as a framework for flexible workflow modelling. In *GROUP '01: Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, pages 32–41. ACM Press, 2001.

[19] E. C. Karletas. *Scientific information management in collaborative experimentation environments)*. PhD thesis, Universiteit van Amsterdam, Amsterdam, NL, (Promoter, Prof. Dr. L. O. Hertzberger), 2004.

[20] M. Lorch and D. G. Kafura. Symphony - a java-based composition and manipulation framework for computational grids. In *CCGRID*, pages 136–143, 2002.

[21] R. McClatchey and G. Vossen. Workshop on workflow management in scientific and engineering applicationsreport. *SIGMOD Rec.*, 26(4):49–53, 1997.

[22] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Greenwood, C. Goble, A. Wipat, P. Li, and T. Carver. Delivering web service coordination capability to users. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 438–439, New York, NY, USA, 2004. ACM Press.

[23] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics Journal.*, online, June 16, 2004.

[24] M. Peleg, I. Yeh, and R. Altman. Modelling Biological Processes using Workflow and Petri Net Models. *Bioinformatics*, 18(6):825–837, 2002.

[25] M. Romberg. The unicore grid infrastructure. *Scientific Programming*, 10(2):149–157, 2002.

[26] Swi-Prolog. The homepage of swi-prolog. In *http://www.swi-prolog.org/*, 2005.

[27] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(3):5–51, 2003.

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

[29] VLAM-G team. VLAM-G and workflow support wish list. In *http://www.vl-e.nl/*, 2005.

[30] G. von Laszewski and I. Foster. Grid infrastructure to support science portals for large scale instruments. In *Proceedings of the Workshop Distributed Computing on the Web (DCW)*. University of Rostock, Germany, June 1999.

[31] R. Wolski. Experiences with predicting resource performance on-line in computational grid settings. *SIGMETRICS Perform. Eval. Rev.*, 30(4):41–49, 2003.

[32] K. Yang, X. Guo, A. Galis, B. Yang, and D. Liu. Towards efficient resource on-demand in grid computing. *SIGOPS Oper. Syst. Rev.*, 37(2):37–43, 2003.

[33] Z. Zhao. *An agent based architecture for constructing interactive simulation systems*. PhD thesis, University van Amsterdam, Amsterdam, The Netherlands, (Promoter: Prof. Dr. P. M. A. Sloot), 2004.

[34] Z. Zhao, D. van Albada, and P. Sloot. Agent based flow control for hla components. *Simulation transaction, Special Issue: Agent directed simulation. (to appear)*, 2005.