Including the state of the art scientific workflow management systems in an e-Science environment

Zhiming Zhao*  Paul van Hooft  Bas van Oudenaarde  Frank Terpstra  Adam Belloum
Vladimir Korkhov  Peter Sloot  Bob Hertzberger
Informatics Institute, University of Amsterdam
Kruislaan 403, 1098 SJ, Amsterdam, The Netherlands


Abstract

In e-Science environments, the service for scientific workflows facilitates the management of experiment data and activities, the prototyping of computing systems and the orchestration of the runtime system behaviour. Reusing the successful and stable design of Scientific Workflow Management Systems (SWMS) can not only improve the efficiency for developing advanced high-level application specific functionality, but also reduce cost and risks for utilising an e-Science infrastructure in a new problem domain. However, most of the existing workflow management systems are driven by the domain specific applications; the applicability to different domains is limited. Investigating the internal links between the characteristics of domain specific applications and the requirements on the development of workflow management systems is essential to realise a common e-Science framework for scientists from different domains to share their knowledge and to conduct domain specific scientific research. In this paper, we give an abstract model of SWMSs and survey the state of the art in existing systems, and based on that we discuss the challenges in developing an effective workflow management system. The research is conducted in an ongoing project: Virtual Laboratory for e-Science (VL-e).

Key words: Grid, e-Science, Scientific Workflow.

* Contact author: Zhiming Zhao (zhiming@science.uva.nl). 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).

1 Introduction

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 policies [19]. Such environments enable collaborations involving large numbers of people and large scale resources, and promote the emergence of a new paradigm for scientific research: e-Science [26]. Organising Grid services and software components as a meta experimental environment, an e-Science environment allows a scientist to deploy remote resources in his domain specific research at an abstract level. Different layers of middlewares, e.g., for managing Grid resources [1, 35], computing tasks [29], data [41], and information [11, 23], form the basic framework for realising an e-Science environment. Automating the experiment routines in scientific research and managing them through a workflow management system are being recognised a crucial support for introducing an e-Science environment to application scientists from different domains [2, 13, 14, 28].

In the past decade, scientific workflows and their support environments have attracted great interest in the context of e-Science; Scientific Workflow Management Systems (SWMS) have been applied in an e-Science environment for different purposes. An original motivation for using scientific workflows is to facilitate the scheduling and the management of computing tasks, e.g., in [29, 34]: a SWMS can automate the flow control among the computing tasks and thus simplifies the administration work of scientists. Another motivation is from the runtime perspective: a SWMS provides meta data to interface distributed resources and services and allows scientists to customise the integration among them for specific application scenarios, e.g., in [33]. The third motivation is from the user interaction perspective. Directly planning and conducting scientific experiments on e-Science infrastructure requires a scientist to have profound knowledge on computer networks and Grid programming. A SWMS hides the underly-
ing details of the Grid infrastructure and allows a scientist to focus on the high level domain specific aspects of the experiments [13, 32]. The diversity among application domains and the concomitant differences in design requirements result in a collection of guises of SWMSs.

Reusing the successful and stable results of different SWMSs can not only improve the efficiency for realising an e-Science environment and developing advanced high-level application specific functionality, but also reduce cost and risks for utilising an e-Science infrastructure in a new problem domain. The Virtual Laboratory for e-Science (VL-e) project is a Dutch e-Science project which aims to realise a generic framework, where scientists from different domains can share their knowledge and resources, and perform domain specific research. The goal of this paper is to propose solutions to improve the quality of the current VL-e environment by reviewing the latest state of the art of several SWMSs and comparing them with the current workflow services in the VL-e environment.

This paper is organised as follows. First, we give a big picture of a SWMS and briefly describe the research missions in the context of the Dutch VL-e project. After that we review some existing SWMSs and discuss the challenging issues in developing effective SWMSs.

2 Scientific workflow in an e-Science framework

A major difference between the workflow systems in business and scientific fields is that business workflows focus on the control flow between processes, while scientific workflows have special requirements on the computation and the data passed between processes [30]. Moreover, human activities are important in scientific experiments, since not all the processes are based on computing tasks.

2.1 A big picture

We consider the lifecycle of a scientific workflow to exist in four phases: composing workflow, developing necessary resources\(^1\), executing workflow, and data post processing. In a lifecycle, different types of users are involved: (i) a domain scientist composes a workflow, (ii) resource developers build necessary resources for the workflow when these resources are not available in the repositories, (iii) the domain scientist executes the workflow using the SWMS services, and (iv) finally he performs necessary data analysis and processing. A SWMS provides users necessary support at each phase.

It is thus clear that a SWMS crosses different layers of underlying e-Science services, e.g., for accessing distributed data, for scheduling and monitoring computing tasks, for managing static and runtime experiment information and the meta data and knowledge such as the associated Ontology of them. Fig. 1 shows a schematic picture of a SWMS in the context of an e-Science environment. A SWMS provides high level service for scientists to plan an experiment, but the execution of a workflow and the user support are realised at different levels of e-Science middleware. From the perspective of portability, we argue that an e-Science SWMS is necessarily on top of a generic Grid middleware.

![Figure 1. Functional components of a SWMS.](image)

In a SWMS, we distinguish three core functional components: a workflow model, an engine, and necessarily user support.

1. A suitable workflow model is the core of a SWMS. It forms the basis for composing workflows of scientific experiments and for verifying it before the execution.

2. A workflow engine executes a semantically correct workflow. It maps the workflow onto underlying Grid resources and orchestrates their runtime behaviour according to the flow description.

3. A SWMS necessarily provides user support at different levels, e.g., experiment planning, flow composition, and runtime control.

These functional components are often realised differently in cases of domain specific SWMSs. Before reviewing other SWMSs, we shall give an overview of the VL-e project and describe its challenging issues.

\(^1\)Here the resources refer to the elements being deployed in a workflow.
2.2 Research context: Virtual Laboratory for e-Science

The VL-e project currently includes six application domains: 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 e-Science framework; a conceptual vision is shown in Fig. 2.

![Figure 2. The conceptual vision of the VL-e project.](image)

The VL-e project uses VLAM-G, a SWMS for data intensive applications developed in a previous project, as the first prototype of the shared framework. The VLAM-G framework provides limited support for different application domains, e.g., in bio-medicine applications which require human interaction in the loop flow control [38]. Improving the workflow support in the VLAM-G environment and in particular including the results of the state of the art SWMSs in this community are an important research issue in the VL-e project.

The goals of this paper can thus be refined as follows:

1. **Reviewing the state of the art of existing SWMSs and investigating the relations between system developments and application characteristics** is essential to develop a generic workflow framework for different application domains. We will pick a number of well-known SWMSs and analyse their mechanisms for modelling and executing workflow, and for supporting user interactions at different levels.

2. **Comparing the related work with the VL-e environment and proposing solutions for its improvement.** Instead of re-implementing the same functionality that has been developed in a domain specific SWMS, reusing the functionality by providing an integration solution is then essential to realise an effective e-Science environment.

3 System development and the state of the art

There are a number of SWMS survey reports available. The report in [37] describes the basic information about a large number of existing SWMSs, but it only provides limited analysis and comparisons. In [40], Yu et al., give a taxonomy of different components in SWMSs and survey a number of existing systems; however neither have comparisons on workflow models and implementation details been discussed, nor have the dependencies on the application characteristic. SWMSs have also been reviewed from a number of specific perspectives, e.g., data retrieval in [8] and user interaction and interface in [7].

Taking one step further, we want to not only survey state of the art of the workflow model, engine, and user support in the existing systems, but also investigate their inside links, and the characteristics of the application domains and other issues. From the perspectives of domain applications and the features implemented, we choose and review 13 well-known systems: Askalon [18], GEODISE [39], DAG-Man [15], GridAnt [5], GridBus [16], Grid-Flow [22], Grid-Nexus [9], ICENI [20], Kepler [3], Pegasus [17], SPA [4], Taverna [33] and Triana [36].

3.1 General observations

We observe a number of things from the 13 systems listed above.

1. Scientific workflows are modelled differently, e.g., data streams between experiment instruments and analysis tools are modelled as a workflow in high energy physics applications [12], whereas human involved adaptations in predefined image processing are highlighted in medical imaging applications [24].

2. Even in one domain, scientific workflows are used differently, e.g., both Taverna and Pegasus are used in bio-informatics applications, but Taverna employs a visual interface to describe data flow based dependencies between web services, whereas Pegasus uses artificial intelligence technology to automatically generate a concrete workflow (based on Directed Acyclic Graphs (DAG)) from a description of an abstract workflow (using a Virtual Data Language) [21].

3. The interoperability between different SWMSs emerges as an important feature for extending one SWMS to support more complex applications, e.g., Kepler wraps the resources of other SWMSs like the engine of Nimrod [34] to improve its parameter sweeping support.

4. The execution of one workflow involves different levels of control issues in an e-Science environment, such
as resource claiming and security. In some systems, these issues are also managed using a workflow management system. Normally, these automatically triggered workflows are hidden from the application level users.

From these observations, we can see that the differences of these SWMSs originate not only from the development methodologies, but more importantly from the different characteristics of the application domains and the design goals. To obtain an insight on these issues, we review existing SWMSs from the perspectives of workflow model, execution and runtime control, and user support.

3.2 Workflow model

By explicitly modelling the processes and the constraints of resources in scientific experiments, a scientific workflow model is essential to realise the separation between the application logic and the functionality of resources, and to allow a user to describe the behaviour of underlying computing resources from the perspective of experiment processes. A workflow model also provides mechanisms for describing application scenarios, and for mapping the description onto computing resources. It is thus the basis for realising the control intelligence of a workflow engine, and for providing user support, e.g., composition and runtime control.

To model the runtime dynamics in scientific applications, one has to take different levels of issues into account, e.g., application logic, data, computing tasks, resources, and runtime quality issues. From these issues, workflow processes, and the dependencies or constraints between these processes are then distinguished. A workflow model provides not only a description of these issues, but also semantics for interpreting them. Driven by specific application requirements, workflow models are derived differently. In this section, we first survey the different models in existing systems, and then discuss how they are used in the context of SWMS and e-Science environments for scientific research.

1. A workflow model can be derived using different modelling mechanisms. A top-down approach decomposes application processes and abstracts the dependencies among them from the logics in applications, e.g., in Kepler the application processes are modelled as actors and the dependencies among them using data flow relations as in the real application scenarios. A bottom-up approach starts from the available resources and provides a rapid prototyping mechanism to utilise these resources in specific application domains, e.g., Taverna provides a web service based meta programming model.

2. The dependencies among workflow processes are important to capture the dynamics in scientific computing. Modelling the sequences among computing tasks as a Directed Acyclic Graph (DAG) is a straight way, e.g., in GridAnt, GridBus and DAGMan. Capturing data based dependencies is a popular mechanism, e.g., in SPA and Kepler. One of the problems in a data based model, however, is its limited capability to capture runtime control issues. ICENI and Askalon model the control flows in the application scenarios to overcome such problems. To include concurrent activities and take other dynamic issues into account, sophisticated models based on Petri Net, e.g., in Grid-Flow, are also used.

3. A SWMS necessarily provides a mechanism to describe workflows. Script based languages are a commonly used approach, e.g., Abstract Grid Workflow Language (AGWL) and CGWL (Concrete Grid Workflow Language) in Askalon, MoML in Kepler and XScufl in Taverna. XML emerges as the basis of these languages. Graphical notions are another important guise for workflow model, e.g. Askalon uses a UML activity diagram (Teuta), whereas both Kepler and Triana offer GUIs to compose data flow diagrams. Script-based workflows can also be visualised as diagrams, e.g. in the Workflow Construction Environment (WCE) in GEODISE.

4. The complexity of a model is managed in a number of ways. Including certain levels of details in the model is one. The model in ICENI includes strategies for resource brokering, whereas Pegasus merely schedules a fixed set of services over a dynamic set of computational resources, and Taverna offers possibilities to improve existing workflows by including newer, more sophisticated services. Constructing a model as multilayer is another approach, e.g., an abstract model in Pegasus includes only data and computing tasks; the mapping strategies and the generation of a concrete workflow are separated.

From the survey we can say that inclusion of dynamic issues in a workflow model has been highlighted in some systems. Using multi-layer model to manage the complexity emerges as an important method. In addition, Using XML based mechanisms to describe a natural model becomes a standard way to improve portability. In the next section we will discuss how the workflow models are executed.

3.3 Flow execution and control

A workflow engine is a machine for executing workflows using available e-Science resources. Based on a workflow model, an engine realises intelligences for interpreting workflow contents, mapping flow descriptions onto resources, generating concrete computing tasks, scheduling
the flow execution, and controlling the runtime behaviour. Decomposing the control intelligence and reusing existing implementations emerge as important development methods; e.g., extending a mature engine by adding additional layer for mapping flow descriptions onto other workflow models, e.g., Kepler is based on Ptolemy, and Pegasus is on top of DAGMan. In this section, we study the involved issues in implementing a workflow engine by surveying the 13 existing systems mentioned before. The survey is summarised from the perspectives of flow enactment, scheduling, orchestration, and quality of service.

1. **Enactment and planning.** To execute a workflow, an engine necessarily interprets the high level (abstract) description, and maps it onto underlying resources according to the semantics of the flow description and the availability of the resources; this preparation phase has different names in some systems, e.g., *enactment* in ICENI and *planning* in Pegasus. The intelligence required for the preparation is dependent on the level of details contained in the flow description. It can be simple, when the description is about concrete resources that are ready for executing, e.g., in Taverna. When the flow model is multi-layer, the mapping between a high level description and a lower-layer one requires semantic interpretation of both descriptions and the information on resources, e.g., requirements on resources and execution quality. A number of issues have to be taken into account. The first one is to efficiently discover proper resources via different e-Science middlewares. One of the solutions is to use a semantic based approach, e.g., GEODISE provides human-in-the-loop enactment with Ontology based information search; similar work is also being conducted in ICENI. Second, the dynamics in the resources, e.g., the accessibility and availability of the resources, are also an important issue. We will come back to this issue when we discuss the user support.

2. An engine also provides strategies for scheduling the execution of a concrete workflow. Baggio [6] 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., [27] 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.

3. **Orchestrating** the runtime behaviour of workflow components is another important issue for an engine. An engine maintains the execution state of flow and coordinates the behaviour of the components. Centralised coordinator is a common mechanism. For instance, in Kepler each workflow instance is orchestrated by a coordinator called *conductor*. Taverna is another example; the web service flow coordinated via messages between services and the engine. A different approach is to let the scheduler of the computing tasks to orchestrate the flow execution implicitly, e.g., DAGMan; the dependencies between computing tasks and the information flow among them are handled by the workflow components themselves.

4. **Service quality**, e.g., fault tolerance, adaptive computing and human in the loop interaction, are also considered as part of the engines functionality. In our early paper [42] we discussed dynamic issues of workflow execution. The DAGMan engine uses the Condor framework to provide fault tolerance for computing tasks. High level support, e.g., when the temporal fault of resources, is demanded; dynamic flow composition has been discussed in the framework of a number of systems, e.g., Tavern and GEODISE. However, from our survey, we can not see mature implementations yet. We will go back to the human related issues in the next section.

From the survey, we can see a number of technologies being highlighted in the implementation of a workflow engine, e.g., using artificial intelligence technology to facilitate the reasoning for workflow planning, applying knowledge engineering technology for semantic level resource discovery, and using information management systems to manage the runtime information. The choices on these technologies are tightly dependent on the workflow models, and the requirements on user support.

### 3.4 User support

User support is provided by a SWMS for each phase of a workflow lifecycle and at different levels of e-Science middleware (as shown in the Fig. 1). The support itself can differ according to the interaction mode, single user or collaborative, the type of user, e.g., domain scientist or resource developer, and the type of application, e.g., computing intensive or exploratory. Three types of support can be distinguished: (i) *automation*, keeping a task completely out of the users control, (ii) *interactive assistance*, the user controls a process or the system provides suggestions to the user, and (iii) *passive assistance*, offering documentation on a certain task. We will look at what support current SWMS’s offer.

1. Flow composition is currently supported by existing systems in three main guises. Firstly to construct a workflow using scripts in a programming environment,
e.g., in the MatLab environment of GEODISE, secondly to compose a workflow using a directly manipulated drag-and-drop GUI, e.g., in Taverna and Triana, and thirdly to automatically derive an executable workflow from an abstract description, e.g., in Pegasus. For composition, most systems allow a user to browse available tools and components, and to choose them for a specific workflow, e.g., semantic based searching, in GEODISE. The validity of a composed workflow can be checked, e.g., SPA checks for compatible data types and protocols in connections between workflow elements. It also pings services used in the workflow to check whether they are alive.

2. Most systems support the developing of workflow resources via a set of API. The toolkits for the development and debugging are normally outside the SWMS. When one SWMS works on top of another engine, the development of the resources for the high level engine can be partially supported by the toolkits of the underlying engine.

3. In [42] we have discussed the types of runtime user interaction with a scientific workflow: the flow, the engine, flow elements, and the runtime infrastructure. Interacting with the flow, i.e., modifying the content of a workflow at runtime, is not explicitly supported by any system; mostly the modification of the flow content results in the restart of execution for the entire flow. However, allowing users to make decisions, e.g., branch selection, in a pre-defined flow is supported by some, e.g., SPA. At the engine level, the interaction with the flow engine, e.g., via the VCR control, is provided in the Ptolemy based systems through Vergil. The interaction with flow elements, e.g., services in Taverna, is supported by almost all. The monitoring tools for workflow execution are mostly provided by the e-Science middleware, e.g., Condor-G is used in Pegasus to submit and monitor the computing tasks.

4. The post processing of a workflow is mainly for the analysis of computing data, preparing documents and making successful template for the next iteration of an experiment. Most of the SWMSs have database support, e.g., in Kepler and Triana; the runtime data can be captured by the system. The information is then stored in the information repository for the later processing.

We will conclude by highlighting areas which are most in need of improvement. Support for collaborative interactions, e.g., collaborative flow composition and flow steering, has not been addressed explicitly in most of the existing systems. Performing a computation within a workflow at a high level of abstraction calls for fault tolerance features at the SWMS level. At the moment most systems rely on the underlying grid middleware to provide fault tolerance, however in practice this means restarting the entire workflow when something goes wrong while better options could be provided at the SWMS level.

4. Findings from the existing systems

After surveying the implementation mechanisms of available SWMSs, we can see: 1) the mechanisms for developing workflow models, engines and the user support are dependent on each other, and 2) the development the SWMS functional components have certain links to the outside issues, e.g., application characteristics and resources. In this section, we obtain insights on these issues.

4.1 Between workflow model, engine and user support

In a SWMS, the model of workflows and the description of the model itself are crucial for the development of the workflow engine and the user support. Fig. Fig. 3 shows part of our investigation. From the Figure, we can see that all systems surveyed here provide passive and interactive user support in all stages of the SWMS cycle. The automation support provided in the workflow composition and execution stages by Pegasus and Triana, however, imposes extra requirements on the workflow engine. Both Taverna (through Feta) and GEODISE (through the Knowledge Advisor present in the Workflow Construction Environment) provide semantic ontology-based support during workflow generation. Since the user still has to make selections from a list the system comes up with, these are semi-automatic support features. Bubak et al., [10] recently described an effort towards a fully automatic brokerage service of the Grid resource management system through ontology-based resource description, discovery and correlation. True resource brokering is also provided by ICENI.

4.2 Application characteristics and choices on middleware

The development of a SWMS is related to a number of external issues.

First, the characteristics of the applications are essential to the development of a workflow model. In [31] the following characteristics are mentioned: data-intensive, compute-intensive, analysis-intensive, visualization intensive. For our analysis we have simplified this to just three main characteristics, data intensive, compute intensive, and user interaction intensive. To our opinion visualisation is a form of computation and analysis can occur in data-compute- and human interactive intensive forms. Within SWMSs features exist to deal with each of these application characteristics.
1. Data intensive applications demand support for dealing with large amounts of distributed data. Solutions come in the form of virtual data systems and automated abstract to concrete workflow transformations both of which shield the end user from dealing with the complexities of multiple distributed data sources.

2. Detailed composition of a workflow should be possible for computationally complex applications. Enabling features are a sophisticated workflow composition tool and an expressive workflow model. The first offers visual programming with user support, e.g., tools for finding the right resources and for debugging of workflows. The second makes sure the complexity can be properly expressed, e.g., by offering programming constructs such as loops, conditionality and parallelism, but also makes sure the complexity be managed through sub workflows.

3. In human interaction intensive applications, the scientist needs to control and interact with an experiment while it is running. This can be facilitated by allowing interfaces with running workflow components, allowing manipulation of the data flow at runtime, workflow components that are performed by hand and an engine with VCR controls to stop and start a workflow.

In figure 4 we order the systems we surveyed according to the amount of features they offer in support of the three above mentioned characteristics. This is not meant be seen as a ranking but rather as an overview of priorities in the field of SWMSs. From this figure we can see that there is relatively little support for human interaction. Furthermore while most systems offer support for compute intensive applications, two focus almost solely on the data intensive. To obtain the goal of a generic system the features of existing systems need to be combined in some way; the next section describes how we plan to do this within VL-e.

5 Summarising discussion

In this paper, we describe our vision on scientific workflows and SWMSs and survey the state of the art of the existing SWMSs.

As a rapidly developing field, both SWMS and e-Science environment are playing increasingly important roles in scientific research; reusing and sharing stable implementations of SWMSs is essential to develop an effective e-Science environment. In the survey, we use our abstract model to analyse the functional components of a SWMS, and investigate the internal links between SWMS and the characteristics of domain specific applications.

As we mentioned in the beginning, supporting different application domains using a common e-Science environment is the basic idea of the VL-e project. Up to now, it remains difficult for us to answer whether the scientific experiments from different application domains can be modelled using a single model; from the current work in the e-Science community, we can not see much effort in that direction. Extending a domain specific SWMS by integrat-
ing with other systems and therefore providing support for multi domains is being recognised as a more practical approach. In this section, we will discuss the challenges in the context of the VL-e project.

5.1 Challenges

In the VL-e project, the research on domain specific applications and the development of a generic e-Science framework are dependent, but they have to be carried out in parallel. The domain scientists need a flexible e-Science framework to pursue their investigation on the domain specific problems, but on the other hand, the developers of the e-Science framework expect the domain scientists to provide clear requirements on the framework. It is imperative for the e-Science framework developers to provide solutions in two scales: a short term one which aims at quick solutions for domain scientists to start their application specific research, a long term one which captures stable features from the short term solution and includes them in a stable VL-e generic e-Science framework.

The test bed of the VL-e project contains three parts: for rapid prototyping, for software certification, and for proof of concept (PoC). The PoC environment contains stable software and is for domains scientists to conduct their research. The rapid prototyping (RP) environment is for framework developers to develop middleware and to experiment new features. Only after the packages developed in the RP environment have been approved by a test in the certification environment, these packages can be migrated to the PoC. Since the lifecycle for transferring software from RP to PoC is about half a year, the short term solution tends to minimise the delay for scientists to use PoC.

5.2 Solutions

A short term solution. We thus propose a short term solution, which imports stable SWMS from the outside of the VL-e, and allows scientists from different application domains to soon conduct their domain specific research. From the survey we discussed above, solutions from the existing systems for bio-informatics, e.g., Taverna and bio-medicine, e.g., Kepler and Triana, are chosen as candidates. As the first step, the integration between the VLAM-G framework and the other SWMSs will be loose, where the knowledge sharing among these systems can be at offline resource level.

A long term solution. A long term solution is proposed to complement the short term one; it aims to encapsulate the advanced features of the systems which have been accepted in the short term solution, and to integrate them using a VL-e framework. It includes three aspects:

First, we will investigate the shortcomings of the existing systems by including the application scientists in the loop, and develop the features which are necessary for the application domain but have not been realised by the existing systems (used in the short term solution). This step will be iterative.

Second, we will include these implemented features in the VLAM-G environment to improve its functionality.

Finally, the VL-e framework realises the interoperability between the SWMSs that have been proved to be useful for the domain scientists, so that the users of the different SWMSs can share their knowledge.

6 Future work

In future, the tracking of the state of the art SWMS will be considered as a continuous task; reusing the latest results of the SWMS will always be a core idea for improving the VL-e environment. 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. The development of the VL-e environment will necessarily take the existing environment for domain scientists into account, and provide integration solutions for them so that the scientists can benefit the e-Science resources without revolutionary change on their working style.

Acknowledgement

The authors of this paper would like to thank all the colleagues who joined or helped the survey: Hakan Yakali, Vladimir Korkhov, Adianto Wibisono, Piter T. de Boer, Victor Guevara-Masis, Nader Mirzadeh, Sander Klous, Robert Bellemann, Silvia D. Olabarriaga, Scott Marshall, Marco Konijnenburg and others.

References


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


