UvA-DARE (Digital Academic Repository)

Scientific workflow design: theoretical and practical issues
Terpstra, F.

Link to publication

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Chapter 5

Workflow Systems Analysis

5.1 Introduction

E-science environments, which enable scientists to achieve the e-science goals of massive computing, massive data handling, virtual organizations and shared resources consist of a basic framework of different layers of middleware. These different layers of middleware manage grid resources, computing tasks, data and information. On top of these there is a scientific workflow management system (SWMS), which gives the scientist support in executing experiments using the resources made available by the different middleware. A SWMS is crucial for the introduction of e-Science to scientists in specific application domains [80, 52]. There are many reasons for employing a SWMS:

- Facilitating scheduling and management of computing tasks
- Automating flow control between computing tasks, thereby reducing the administrative work for scientists
- Through the use of meta data, allow interfacing with distributed resources and services for integration and customization within specific application scenarios.

The most important one however is to allow a scientist to interact with an experiment in an e-Science environment without having to focus on the networking, (parallel) computing and data management details. A SWMS should allow a scientist to focus on the high level domain specific aspects of the experiments being performed. In this chapter an overview is presented of the elements that make up a SWMS, and how they relate to sharing software resources. This is followed by a look at the state of the art in workflow systems for E-science, which includes discussion on which state of the art system provides the most suitable context for shared software resources. Finally important future directions in research are given for areas
of SWMSs which need further development in order to achieve the e-Science goals.

5.2 Scientific Workflow Management Systems

There are many workflow management systems in existence today, both in business and scientific fields. Business workflow systems differ from scientific systems in that they focus primarily on the business processes. Scientific systems on the other hand focus mainly on massive data involved in e-Science[93]. As the research in this thesis concerns scientific workflow management systems the following description is specific to these systems, although many elements might also be found in a business oriented system.

5.2.1 Workflow lifecycle

A workflow being performed within a SWMS has a lifecycle consisting of three distinct phases[82]:

- Design
- Execution
- Analysis

Each of these phases imposes a different set of requirements on the SWMS. Within the design phase the workflow is defined at a high level by an application scientist. The order of all of the steps involved, the way these steps are connected and their parameters for a particular experiment are defined in this phase. Most SWMSs offer a graphical programming environment to keep this task accessible to scientist who are not experts at programming languages. Workflows can either be defined as abstract, without explicit reference to the underlying resources used, or concrete where these resources are defined by the user. Some SWMSs allow for hierarchical composition where one workflow-step is actually an interface to an underlying sub-workflow. This is done to keep a clear overview in complex workflows. Also included in the design phase is resource development. This involves programmers who make software components or other resources such as scientific instruments available for use in a workflow. This usually entails developing the resource as a web or grid service according to an API which is supported by the SWMS. The result is that a resource is able to communicate, according to defined standards, with other resources and that meta data is defined which defines how a resource can be used.

The execution phase handles the enactment of a workflow. This involves mapping high level workflow steps to appropriate low level grid resources and orchestrating the runtime behavior as defined in the composition phase.
Analysis phase involves actions after a workflow has finished its execution such as abstracting a template from a successful experiment for future use or changing resource meta data based on experiment results.

From the above description of the workflow lifecycle it should be clear that a SWMS involves different layers of middleware. In order to support these three phases three functional components can be distinguished within a SWMS. These are the workflow model, workflow engine and user support. Figure 5.1 gives a schematic overview of the middleware and the place of the three functional components. We will now proceed to describe each of the three functional components in detail.

![Functional components of a Scientific Workflow Management System](image)

**Figure 5.1: Functional components of a Scientific Workflow Management System.**

### 5.2.2 Workflow model

The workflow model is an essential component for separating the application logic from resource functionality, allowing a scientist to model a workflow from a high level of abstraction. This is achieved by defining a standard to describe application scenarios. We will start by viewing this from high level concepts and work down towards actual languages used. Two main types of workflow models can be distinguished: data flow and control flow oriented. In data flow based models the order of execution is determined by data arriving at the next step in the workflow. This means that in the composition phase workflow steps are most logically arranged in a temporal view: each workflow step can only start executing if the previous step has started to produce data. In control flow based models the execution is determined by control statements such conditional statements (if then else) or loops (while, for). This means that in the composition phase a workflow is most intuitively edited in a process logic description view, allowing for easy
explicit definition of parallelism and more complex workflows in general. For the actual description of the processes different methods can be employed. Directed Acyclical Graphs (DAG) are the common process definition for Data flow based models [13, 3], while control flow based models are usually based on Directed Graphs [31, 32, 68] or in some cases on Petri Nets [70]. This process definition is captured in some kind of formal language, almost always XML based. Some SWMSs actually allow a user to directly define a workflow in this language [68]. Most SWMSs automatically derive the XML based workflow definition from a graphical programming environment. In some cases the workflow model not only describes the application logic but also some parts of resource functionality for instance the preferred scheduling strategy can be defined or a specific resource can be coupled to a workflow step.

5.2.3 Workflow engine

The workflow engine is responsible for the execution of workflows. It maps workflows onto computing resources, generates concrete computing tasks, schedules flow execution, controls runtime behavior and ensures quality of service. These tasks will be more elaborately explained in the following four topics: enactment and planning, scheduling, orchestration and service quality.

Workflow enactment also sometimes known as planning takes a workflow description and maps unto underlying resources. For this task it needs intelligence which takes the workflow semantics and the availability of resources into account when performing this mapping. The complexity of dealing with workflow semantics can vary between different SWMSs: some have workflow descriptions at multiple levels, an abstract one for the end user to compose and a concrete one to execute, others let the end user compose a concrete workflow. In cases where an abstract workflow, without any references to specific resources, is presented to the engine a lot more intelligence is needed to generate a concrete workflow. To reduce this complexity human assistance can be used to move from abstract to concrete workflows. The other important function that has to be performed during enactment is the discovery of resources. The complexity of this task can vary according to the homogeneity of the e-Science environment. This task can be a complex proposition when dealing with virtual organizations consisting of multiple partners, each having different restrictions associated with gaining access to their resources.

Scheduling a workflow presents another challenge to the workflow engine. Whether the SWMS is a centralized or a decentralized system plays an important role in this system. In a centralized system with one scheduler for the entire e-Science environment scheduling is simpler to realize than for a system in which the workflow engine has to deal with multiple sched-
5.2. SCIENTIFIC WORKFLOW MANAGEMENT SYSTEMS

ules. This last scenario can however be unavoidable in a heterogeneous environment. Thus apart from offering a scheduler with scheduling strategy a workflow engine can also offer additional features such as reserving cycles on resources not directly under its control, or offering dynamic scheduling that changes the workflows schedule while it is executing.

The orchestration of a workflow is the process of controlling the runtime behavior of resources as defined in the workflow description. This can be done through a centralized coordinator or conductor, a practice common in control flow oriented SWMSs, or it can be done implicitly through the dependencies and information flow between computing tasks, which is an approach taken by many data flow oriented SWMSs.

Finally, service quality deals with an issue already hinted at in scheduling: fault tolerance. There are many cases imaginable where the workflow has to deal with dynamic execution. When a resource suddenly becomes unavailable or the execution of a software component unexpectedly fails, the workflow engine has to handle the situation. For instance: through dynamic (re)scheduling of a workflow, but also by offering fault tolerance features such as rollback and checkpointing. Moving up from fault tolerance, a workflow engine should allow the dynamic composition of workflows, changing workflow components or data while the workflow is executing. Although this last item has been discussed in the context of some SWMSs [104, 130] no usable implementation exists at this time.

5.2.4 User support

At all phases in the workflow lifecycle a SWMS user can expect different forms of user support. The support can have different aims. Therefore we have divided this support into three categories:

- Passive support, giving greater understanding of what is involved in an experiment.
- Interactive support, enabling easy manipulation of the processes involved in an experiment.
- Automated support, completely shielding the user from certain processes allowing him to focus on other things.

What follows are details on the user support available at each phase of the workflow lifecycle and the demands this support places on the workflow model and workflow engine. The design phase has been split into workflow and resource design.

Workflow design

Workflow design is the phase in which an end user scientist will spend most of his time. Most of the user support will therefore be offered here. First of
all an end user should have a clear idea of what the SWMS and the resources he plans to use are capable of. This is achieved through passive user support consisting of clear documentation for the composition environment as well as the workflow steps from which the workflow is composed. During the composition the SWMS can offer interactive support, pointing out errors in the validity of a workflow through type and protocol checking. For instance: when one step outputs integers and it is connected to a step which expects doubles this is clearly wrong. This can be pointed out to the user either by refusing to make the connection in the first place or by giving some form of warning. This can be extended by the use of more sophisticated meta data, for instance when a step that outputs temperature in Fahrenheit is connected to one which expects Celsius as input. This can also be pointed out to the composer of the workflow. A second form of interactive support which is closely related, is semantic search, where a user can search for resources based on the meta data associated with it. For instance: when looking for a next step in a workflow, semantic search can support the user by offering only those workflow steps which can connect with the current workflow step. Finally parts of the composition process can be automated. In SWMSs with a multi-layer model where there is a high level abstract workflow description and a lower level concrete description, the transformation from abstract to concrete can be performed automatically. This process as it is implemented in systems such as ICENI[72] is meant for situations where there are multiple actual instances of one computational component running in different locations. The abstract description is without reference to a particular instance or location, the automatic concrete composition finds the most suitable running instance and uses that for its concrete workflow.

Resource design

The part of the design phase, in which resources are made available for use in a SWMS is usually not performed by the scientist end-user who composes a workflow. A resource developer requires a different type of support since it is assumed he has a lot more programming expertise. Passive support at this phase comes in the form of a well documented API which allows the developer to let a resource communicate with the different layers of middleware associated with an SWMS. Interactive support comes in the form of a rapid prototyping environment in which individual resources can be tested and debugged without executing a complete workflow or using the composition environment. Finally automatic support can be offered in defining meta data associated with a particular resource. Statistics on run time behavior can be automatically gathered and added to the meta data for instance.
5.2. SCIENTIFIC WORKFLOW MANAGEMENT SYSTEMS

Execution

During the execution phase the emphasis of user support is mainly on interaction between user and workflow execution. In this phase the user is assumed to be a scientist end-user just as in workflow design. Providing information on the progress of execution or monitoring is a form of passive support that can be offered. Interactive support allows the user to interact with the execution. This support comes in different forms. First the user can be allowed to manipulate the content of the data flow at runtime. Also the user can steer the execution by choosing the direction a workflow should take at certain points during execution. VCR like controls can allow a user to pause, resume or stop execution. A user can interact with certain steps in the workflow for instance by manipulating parameters, associated with a particular step, at runtime. Finally in some cases a user can be allowed to directly manipulate underlying middleware, for instance changing scheduling strategies. Automated support can also take place with scheduling strategies, in the adaptive workflow execution. For reasons such as a resource suddenly becoming unavailable or the unexpected failure of a software component, the workflow needs to be dynamically rescheduled. Alternatively when rescheduling is impossible execution should be halted in a user-friendly manner. For instance allowing a restart of execution from intermediate results.

Analysis

The analysis phase, from the workflow lifecycle point of view, deals mainly with the reuse of workflow results. Passive support in this case should offer a clear presentation of the results of workflow execution, allowing the scientist to judge the success of an experiment. Interactive support comes in the form of allowing the partial or complete replay of workflow execution. Classifying the success of an experiment according to predetermined criteria can be done automatically supporting the user when he has to perform a certain workflow many times. Finally for SWMSs with multi-layer models abstract templates of successful concrete workflows can automatically be generated.

User support can place particular demands on the other two functional SWMS components. In figure 5.2 these demands are made explicit. While the demands on the workflow model are not particularly great, some forms of support that only apply to multi layer models, are quite demanding on the engine, especially for the execution phase. In the next section we give a brief overview on how SWMSs are implemented in practice.
<table>
<thead>
<tr>
<th>FLOW COMPOSITION</th>
<th>USER SUPPORT</th>
<th>APPLICABILITY IN WORKFLOW MODELS</th>
<th>APPLICABILITY IN WORKFLOW ENGINES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passive</td>
<td>Provide documentation and description definition, e.g. XSquill (Taverna), MoML (Kepler)</td>
<td>Applicable to all models</td>
<td>Applicable to all engines</td>
</tr>
</tbody>
</table>
| Interactive      | • Validate workflow through type & protocol checking, e.g. SPA, Kepler  
|                  | • "Semantic" ontology-based search, e.g. Taverna (Peer) | | Requires that engine has enactment service and be able to communicate with knowledge based middleware |
| Automation       | (partial) automatic composition of abstract workflow, e.g. Triona | Multilayer model, e.g. Pegasus | |
| DEVELOPMENT      | Passive      | API for resource development → present in most systems | Applicable to all models | Applicable to all engines |
| Interactive      | Debugging or rapid prototyping, e.g. Kepler | | |
| Automation       | Automatic generation of components, e.g. IECENI for metadatas | | |
| EXECUTION         | Passive      | Monitoring execution → most systems | Connection to middleware execution monitor | |
| Interactive      | • Manipulate flow content  
|                  | • Steering  
|                  | • VCR control  
|                  | • Interact with components  
|                  | • Interact with the e-science middleware | Applicable to all models | • Capability to execute partial workflows, interrupts, handling intermediate results, interface for interaction  
| Automation       | Adaptive workflow execution | | • Capability to stop and restart execution  
|                  | | | • Provide user interface to component parameters  
| POST PROCESSING | Passive      | Workflow template abstraction → Distinction between abstract and concrete experiments needed | Applicable to all engines | |
| Interactive      | • Replay experiment → all methods  
|                  | • Reproduce experiment | No influence → all models  
|                  | | Deterministic models | |
| Automation       | • Automatic template abstraction  
|                  | • Classification | Applicable to multilayer model  
|                  | | Applicable to all models | |

Figure 5.2: User support at different stages in a workflow lifecycle and its influence on the workflow model and engine
5.3 State of the art

The overview of the state of the art given in this section is based on a survey carried out within the VL-e project [41] and included the following SWMSs: Askalon[62], GEODISE[130], GridAnt[33], GridBus[13], ICENI[68], Kepler[31], Pegasus[58], SPA[32], Taverna[104], Triana[96]. The aim of this section is not to give an exhaustive overview of all SWMSs but rather to show which direction research in the field is moving in. For more detailed information see the previously mentioned survey [41] or one of the other surveys that have been done in this area [22, 134, 43, 45]. For each functional component we will give examples of how existing SWMSs deal with the issues mentioned in previous sections.

SWMSs seem evenly split in their choice of workflow model with about half choosing a control flow based approach while the other half is based on data flow. This may seem surprising because in the beginning of this chapter it was mentioned that most SWMSs are focused on massive data involved in e-science and that the data flow oriented model should be the logical choice. While the choice of model is evenly split, most SWMSs represent the workflow to the user in a temporal view thereby emphasizing data flow, regardless of the actual model being used underneath. Most SWMS offer abstract workflow composition and therefore a distinction between abstract and concrete workflows, but only a few offer the possibility to automatically transform this abstract description into a concrete one, the best example being Pegasus.

For workflow engines there is a trend to extend existing engines instead of building from scratch: for instance both SPA and Kepler are build upon Ptolemy[79], while Pegasus is build on top of DAGMan[3]. Semantic based search is currently provided for GEODISE and Taverna. The use of meta data in general is an area in which much research is taking place but most of this research has not yet led to mature implementations. Scheduling is an area where big differences exists: ICENI for instance offers a sophisticated launching service which takes care of executing and scheduling jobs on the grid, while in Kepler it is up to the user to provide extensions to the SWMS for scheduling and grid access. Orchestration of runtime behavior can happen in two ways: first there is the centralized approach where all orchestrating messages originate from a central component in the engine, e.g. in Taverna and Kepler, secondly there is the decentralized approach where orchestration is handled implicitly by the workflow components themselves. Service quality is an area where much improvement is possible. Although some form of dynamic scheduling is possible in most SWMSs, for the handling of fault tolerance most rely completely on underlying middleware layers. Thus offering fault tolerance support tailored to workflow execution is lacking. Dynamic execution features, allowing for the dynamic composition of workflows, are being researched in the context of the GEODISE SWMS,
however a mature implementations are not available at this time. Taverna offers user controlled fault tolerance schemes at the SWMS level.

There is quite a lot of diversity in the types of user support offered by different SWMSs. This can in part be explained by the different origins of the systems. Systems build upon the Ptolemy engine inherit a very sophisticated user interface originally developed for signal processing. While systems that have their origins in high performance computing research, such as ICENI have only relatively recently started to develop their front end. Passive user support is well developed for all systems. They all offer documentation for the systems, their APIs and in most also monitoring of execution and its results is implemented. More could be done in offering passive support for individual workflow components, for instance offering links to online documentation concerning a component used during workflow composition could improve an end users understanding of what is possible. Interactive assistance is where many SWMSs differ, though all offer some form of workflow validation. For instance SPA does type and protocol checking on workflows, as well as pinging resources used in the workflow to see whether they are alive. Semantic search is offered for far fewer systems: for instance GEODISE and Taverna. Interactive support during execution is an area where much improvement is still possible as can be deduced from the lack of support for dynamic workflow execution in workflow engines. Interaction with components is often limited to manipulating parameters. A VCR like control to stop start or pause execution is common though.

5.4 Towards a shared software resource

Scientific workflow management systems offer solutions to problems facing scientist when doing experiments in an e-Science environment. Because of the different origins of different SWMSs there is diversity in the support offered by each of them. What they have in common is that their support is mostly focused on the electronic part of e-science, hiding the complexities of grid computing from the normal scientist. They do also present an opportunity to offer more support for the science part of e-Science. Especially the increasing trend for using meta data, an approach known under several names such as the semantic grid\(^1\) and cognitive grids\(^2\), should offer opportunities to support end user scientists in performing experiments which contain components from scientific fields not their own. The use of meta data can enable the creation of a scientifically sound experiment without the active participation of experts from all fields involved in that particu-

\(^1\)http://www.semanticgrid.org originated from UK e-Science programme, but now has wider support

\(^2\)http://www.isi.edu/ikcao/cognitive-grids University of Southern California Information Sciences Institute
lar experiment. Instead experts make their knowledge explicit in a generic form. In [81] William E. Johnston identifies three categories of capability which semantic solutions will have to meet.

- Check the validity of structures created by a user and assist in correcting errors.
- Automatically build simple composite operations from primitive operations on the basis of their semantics.
- Provide high-level constraints to facilitate correct interaction between complex models from different disciplines.

These three capabilities seem to be ordered by their likeliness of being implemented first. The first capability is already met by several systems, while research such as [84] points the way to how the second capability can be realized. Johnston acknowledges that though the first two capabilities are within the scope of current technology, the third one at this time is not. Sharing software resources fits within this trend of increased use of semantics in workflow composition. Nevertheless the steps presented by Johnston focus mainly on automation.

As it was shown in chapter 3 that automatic workflow composition is not possible in the general case, a shared software component should aim to aid the scientist in integrating it in a workflow, rather than just providing information aimed at automated composition. The first steps in this direction have already been taken: a prototype of an interactive assistant which provides semantics based advice to the user during composition has been described in [51]. This approach differs from the shared software resource in that it aims at composition within one domain. Also it provides assistance at each individual step, but does not provide a methodological overview. It does reinforce the view that facilitating the exchange of components between different domains, is feasible with current technology, through the use of an interactive semantic approach instead of the far more complicated automated approach.