Workflows in Computation Grids

Dominik Kuropka[1], Gottfried Vossen[2], Mathias Weske[1]

[1] *Hasso Plattner Institute at the University of Potsdam,*
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
[2] *European Research Center for Information Systems at the University of Münster,*
Leonardo-Campus 3, 48149 Münster, Germany
{dominik.kuropka | mathias.weske}@hpi.uni-potsdam.de,
vossen@wi.uni-muenster.de

Abstract

This paper shows that flexibility of workflow executions, i.e., the ability to deviate from a given model on the fly and during execution, is crucial for a wide applicability of workflow management technology in computing grids. Since flexibility concepts for workflows have been elaborated extensively in conventional business-oriented domains, we show how to reuse and adapt them for workflow management in computation grids.

1. Introduction and Motivation

Workflow management generally aims at modeling and controlling the execution of complex application processes in a variety of domains, including the traditional business domain, governmental applications, the wide field of electronic teaching and learning, or more recently the natural sciences. Workflow models are representations of application processes to be used by workflow management systems for controlling their automated execution. Workflow languages are used to specify workflow models. Since workflow modeling aims at mapping relevant information about application processes into workflow models, workflow languages need to have constructs for a variety of perspectives. For the area of scientific workflows, an important aspect is flexibility, which will be argued for in this paper.

As a motivating example, consider an application such as SETI@home, the prototypical application for the analysis of space data. In this project, two aspects are important: One is that a grid is needed for providing the adequate computing power needed to analyze the huge amount of data that are at hand here. The other is that such an analysis typically follows an exploratory line of work, the individual steps of which are all known in advance, yet the exact order of their execution evolves as a particular analysis progresses. Similar remarks apply to the Kepler system for scientific workflows [1], which is being used in such diverse scenarios as promoter identification in genomics, invasive species prediction in ecology, or functional raster image analysis.

We show in this paper that workflows in computing grids cannot do without an appropriate support for flexibility, i.e., the possibility to modify workflow instances while they are being executed, as has originally been pointed out some time ago in [2, 3]. To this end, the remainder of this paper is organized as follows: In Section 2 we briefly recall the most basic workflow concepts that are relevant to our discussion. Section 3 explains our view of workflows on the grid, while Section 4 establishes a number of requirements from scientific applications that play a key role in contemporary computation grid applications. Section 5 concludes the paper.

2. Workflow Concepts

In this section, we briefly introduce core workflow concepts and then elaborate on various workflow aspects. In essence, workflows are executable models of processes, and as such can distinguish a *schema* from an *instance*. The former is a time-invariant description of the automatable portion of a process, while the latter is a particular workflow instantiated from a schema. The conceptual specification of a workflow using a workflow model is the most fundamental aspect in workflow management, since it allows the analysis and tailoring of processes without touching code. To this end, workflow management can be regarded as model-driven development with a strong success record.

The development process of workflow applications in general goes through various phases, vastly similar

in the various application areas. The first phase deals with *information gathering*, relevant for the application process under investigation. In this phase, empirical studies such as informal interviews and available documentation are used. The activities of this phase centers around the application, and technical issues are not considered yet. Information gathering in grid workflows is typically done by domain experts, for instance scientists that use the grid for computations. In these typical grid environments, the setting up of a scientific experiment that should be enacted using grid workflow technology is a core activity in the information gathering phase.

The second phase involves *process modeling*, in which the information previously gathered is used to specify process models, i.e., deal with the processes carried out in a given application domain. The main purpose is to provide a general and easy-to-read notation, which enables information system experts and domain experts to validate and optimize the resulting process models, an activity often termed process *reengineering*. The result is a process model that is often complemented by a data model describing the underlying objects on which the process components are based and which are consumed or produced by the various activities; finally, it may also comprise an organizational model specifying the organizational structure of the underlying business as well as resources that are available at the various departmental levels for carrying out activities.

The purpose of the subsequent *workflow modeling* phase is to enhance the process model with information needed for the controlled execution of workflows by a workflow management system or a workflow engine. In this phase workflow languages are used. Typically, different languages are used for process modeling and for workflow modeling. Hence, process models have to be translated into the constructs of a workflow language. Besides that, execution information is added to the model; indeed, workflow modeling abstracts from irrelevant information and adds relevant information, mainly of technical nature. For instance, in workflow models application programs used to perform workflow activities are specified, including their execution environment. The result of the workflow modeling phase is a workflow model, which is used by a workflow management system for controlling the execution of the workflow.

To provide modularity in workflow modeling and for being able to refer to the different dimensions of workflow modeling explicitly, several workflow *perspectives* are discussed next. The *functional* perspective covers the functional decomposition of activities as present in application processes, i.e., it specifies which activities have to be executed within a workflow. To deal with the high complexity of application processes, the concept of nesting is used to describe the functional perspective of workflows, and is typically applied across multiple levels. In particular, workflows are partitioned into complex and atomic workflows, where complex workflows are composed of a number of (complex or atomic) sub-workflows. As a result, workflows typically have a tree structure, such that the root node of the tree represents the top-level (complex) workflow, the inner nodes represent other complex workflows, and the leaf nodes represent atomic activities. Synonyms for complex workflow include process, complex activity, block; atomic workflows are also called (atomic) activities or steps.

In general, workflows consist of a set of interrelated activities. Hence, the execution of a complex workflow has to take into account interrelationships of the complex workflow's sub-workflows; these issues are covered in the *behavioral* perspective. It specifies under which conditions the sub-workflows of a given complex workflow are executed during workflow executions. Important components of this perspective are control flow constraints, which represent the control structure of activities of the application process; a typical example are ordering constraints. Other forms of interrelationships are start as well as termination conditions. For each sub-workflow, a start condition specifies the precondition of its execution. Hence, an activity is started during a particular workflow instance only if the start condition of that activity is evaluated to true.

An important perspective of workflow languages, as mentioned, is the modeling of workflow relevant application data. In graph-based approaches, the *informational* perspective includes data flow between workflow activities. In particular, each activity is assigned a well-defined set of input and a set of output parameters. The transfer of data between workflow activities is generally known as data flow. By providing graphic language constructs to represent data flow between activities (such as the state information in a Petri net), the informational perspective can be visualized and used to validate and optimize application processes. Note that there are many technical issues to solve, e.g., different data formats of a given data flow, which may require the use of filters to allow seamless integration of different tools using different data formats. To this end, it is desirable that data as specified in a data flow is strongly typed.

Workflows are often executed in complex organizational and technical environments, and a major goal

of workflow management is enhancing the efficiency of application processes by assigning work to given resources, i.e., persons, hardware or software systems as specified by workflow models. To reach this goal, a workflow management system has to be provided with information on the organizational as well as on the technical environment in which the workflows will be executed. In general, atomic workflows can be either automatic or manual. Manual atomic workflows are executed by persons who may use application programs to do so; automatic atomic workflows are executed by software systems without human involvement. Since a strict assignment of workflow activities to persons is not feasible in most cases, the *role* concept is often used. A role is a predicate on the structure of an organization in which the workflow is executed. When an activity is about to start, the system uses predefined role information to select one or more persons which are permitted, competent and available to perform the requested activity. The process of selecting one or more persons to perform a workflow activity is known as *role resolution*.

The integration of existing tools and application programs into workflow applications is an important feature of workflow management systems, especially in scientific applications. The information required to this end is specified in the *operational* perspective, which covers mainly technical issues, like the invocation environment of application programs (including host and directory information of the executable program), the definition of the input and output parameters of the application program and their mapping to input and output parameters of workflow activities. As described above, persons are selected by role resolution to perform workflow activities. When a person chooses to perform an activity then the defined application program is started, and the input data specified in the workflow model is transferred to that application program. When the person completes that activity, the output data generated by that activity is collected in the output parameters of the activity to be transferred by the workflow management system to the next workflow activity, as specified in the respective workflow model.

The need to enhance the flexibility of workflow applications originates from different application areas discovered already in the late 90s. A recent survey is provided by [4]. Starting from applications in non-traditional domains like the natural sciences or hospital environments, flexibility also became an issue in business applications and is these days a well-understood requirement. Providing flexibility to workflow applications is based on the observation that during workflow modeling often not all perspectives of the application process can be specified completely. Indeed, there may be unforeseen situations during workflow executions which require flexible and dynamic reactions by the user or administrator of the system. Hence, additional features to model workflows and additional functionality to support flexibility are now needed. The success of present-day workflow management systems to a large extent depends on the ability and way workflow model changes or changes to the organizational or technical environment are supported in a user-friendly or application-specific way. There are different forms of flexibility, ranging from the change of role information and application program information to the change in the functional and behavioral perspectives of workflows. Adding an activity to a complex workflow while the workflow executes corresponds to a dynamic change in the functional perspective; changing the control structure of sub-workflows of a given workflow (e.g., parallel execution of workflow activities, originally defined to be executed sequentially) corresponds to the change in the behavioral perspective. Providing user intervention operations to allow users to skip, stop or repeat sub-workflows is another form of flexibility in the behavioral perspective. A change of role information and of application program information changes the organizational and operational perspectives, respectively. We remark that flexibility has to be supported by the workflow language and also by the workflow management system. For instance, workflow languages should allow to specify which activities can be skipped or repeated, and how data issues due to deleting workflow activities which would generate required data are solved. Flexibility is of even higher importance when workflow instances are executed on a grid of resources, which will be discussed next.

3. Workflows in the Grid

Grids are an emerging architecture for the execution of resource intensive programs by integrating large-scale, distributed and heterogeneous computing or storage resources [5]. For this reason the grid concept is of high interest to scientific communities with high requirements on computing or storage resources like high-energy physics, geophysics or astronomy. Moreover, scientific applications tend to combine grid computing with *service orientation* [6, 1], i.e., the components making up a scientific workflow are mapped to services provided by stations in a grid. Like in non-grid domains in the past, the demand for a controllable and traceable execution of workflows within a grid environment is emerging due to the rising

complexity of grid applications. Instead of reinventing the wheel, it is beneficial to reuse existing concepts wherever possible. For this reason it makes sense to verify whether the workflow concepts described in the previous section can be successfully reused in or adapted to the context of the grid.

In the grid literature often several differences between grid workflows and 'conventional' workflows are enumerated, among them long lasting workflow executions, heterogeneous resources, multiple administrative domains, dynamic resource availability and utilization, and large data flow. [7] Naturally the question arises what 'conventional' workflows are. If we assume that conventional workflows are business workflows, then most differences disappear. For example, long lasting workflow executions are a frequent case in business workflows, since several different human actors have to process non-digital documents in some cases or make negotiations with other business partners. Indeed, the processing of insured events in insurance industry can sometimes take weeks or even months if several expert reports have to be obtained. In business workflows also heterogeneous resources are used to process tasks. Different users may work on different systems and different applications, and if we do not focus on computational resources only, we see that acting humans are even much more heterogeneous than computer systems. They have different capabilities and backgrounds and their availability is almost uncertain due to the possibility of holidays and disease. So a truly dynamic resource utilization is already implemented in most business workflow management systems. In supply chain scenarios, for example, different administrative domains are usually involved in an inter-organizational workflow. From this point of view it is clear that grid workflows at this high conceptual level are much less different to business workflows than it appears on the first look. However, at least one difference still persists: in most business scenarios the amount of data transferred is noticeably smaller than in grid scenarios. Therefore a grid-specific solution is definitely needed to cope large data objects. Next we will show how grid workflows can be represented by conventional workflow concepts.

As already mentioned, there are at least two types of workflows supported by conventional workflow management systems: complex and atomic workflows. Complex workflows consist of several sub-workflows and a workflow schema defining the functional order of the workflow execution. Atomic workflows are "black boxes" for the workflow management system and their inner assembly is not known to it. Up to this stage, conventional workflows match very well the situation of grid workflows: Indeed, in the grid several programs which can be treated as atomic workflows typically have to be executed in a given functional order and can hence be collectively seen as a complex workflow.

In conventional workflows two types of atomic workflows exist: automatic and manual ones. Automatic workflows represent simple activities which can be done without user interaction, like for example data format conversions. In conventional scenarios they are usually short-running and not resource-intensive. Thus, no explicit scheduling or sophisticated handling is applied to this type of workflow in most conventional workflow engines. While automatic workflows play an assistant role in conventional workflow scenarios, manual workflows are the important issue. As a consequence, most workflow systems support fine granular modeling of non-functional requirements for manual workflows. These requirements are usually represented by a role concept. Since the executing resources of manual tasks are humans, a list of roles is assigned for every person. These roles define the capabilities of a person, such as 'speaks English', 'has expert knowledge in civil code' or 'is allowed to approve contracts up to 10.000 €'. For each manual workflow a set of requirements regarding the roles of the executing person can be defined. During runtime, these requirements are used for role resolution to find an adequate resource (person) to process the workflow by hand.

Since automatic workflows are playing a supporting role in conventional workflow scenarios, a direct transfer of their simple handling toward a grid scenario will not be successful. In grid scenarios automatic workflows play a central role, so they need to be modeled in more detail. One promising approach is to extend role concepts used for manual workflows also to automatic workflows. This extension means that additionally to humans, machines will be treated as resources too. Consequently, roles have to be assigned now to machines. By this way it is possible to define non-functional and technical properties of the different machines (resources) in the grid. Clearly, new roles have to be invented, such as 'operating system is Linux', '10 Giga byte of memory available' or 'has an i386 architecture'. As a side-effect the role-resolution mechanism will automatically become equivalent to what is called binding in a grid. In other words, role resolution amounts to a form of scheduling at a global level, but conventional workflow systems also support scheduling at a local level by work lists. For each human—depending on his or her roles—all possible tasks are presented (usually with additional information

like priority or scheduled deadline) in a dynamic and shared work list. Dynamic means that the work list is continuously updated (usually in short intervals via push principle) and shared means that one task can be on the work list of several humans at the same time. If one human starts a task, it automatically disappears from the work list of the others. This ensures a dynamic and as late as possible binding of resources which has positive effects on the reliability of the workflow execution. Since we have already extended the workflow model by treating machines equal to humans, we should consequently also provide each machine with a shared and dynamic work list. With this work list each machine in the grid has the possibility to make local optimizations and to pick the activity which will be executed next.

The execution of programs is supported by conventional workflow engines, since for each activity usually a specific program or application mask has to be started on the user's computer. So these concepts can be widely adopted to the grid scenario, but some modifications are necessary. First, programs which have to be performed in the grid usually have to be uploaded in advance onto the target machine and second, it usually makes sense not to transfer all Giga or Tera bytes of data between two programs through the workflow engine. Since most stand-alone workflow engines are adaptable regarding the technical aspects of the invocation of programs, uploading of programs prior to execution and usage of distributed data storage can be implemented by reusing grid standard protocols like GridFTP [8] prior to triggering the execution on a machine.

Figure 1 shows a model how workflow management systems can be linked with the Grid. Grid resources register their capabilities in the grid information service which is accessed by workflow management systems to carry out role resolution. Workflow management systems may belong to different organizations and are responsible for task scheduling and monitoring on a global scale regarding their workflow or organizational context. Resources do local scheduling on the basis of work lists they get from the workflow management systems. Thus it is possible that one resource holds several work lists form different workflow management systems which have to be merged and which are competing on the resource. To be able to make 'good' local scheduling decisions it is important that tasks which are provided on the work list include additional scheduling hints like priority information or deadlines.

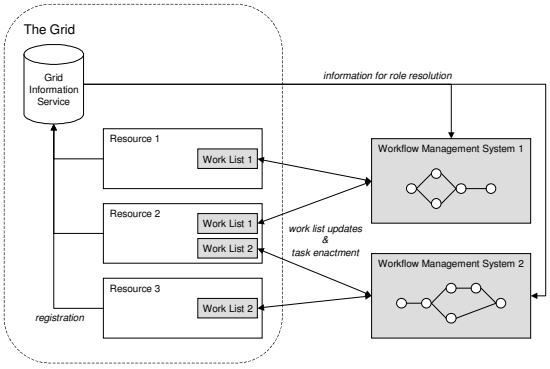

Figure 1: Workflow Management and the Grid.

4. Scientific Applications Requirements

This section discusses the requirements of scientific applications with respect to workflow technology in general and grid workflows in particular. Two types of scientific applications can be identified (i) high-performance applications with a fixed structure and (ii) experiment process management and data lineage. In high-performance applications (e.g., biotechnology or particle physics), the goal is to make large-scale globally distributed computing resources available for solving specific well-defined problems, which even have a piece of software that solves the problem. In this respect, grid workflow is coined today, where large similarities to traditional job scheduling can be identified. As discussed in Section 3, however, the grid community may take advantage of the concepts, languages and tools provided by workflow technology.

For the second type of scientific application, additional requirements are imposed on adequate workflow support: Due to the inherently dynamic nature of scientific progress, workflow support should provide additional flexibility and adaptability. In an early stage of workflow research, there were a number of projects on scientific workflow management, including WASA (Workflow support for scientific applications) [9] or Opera. In these projects, flexibility has been identified as a major design issue. Flexibility is required in order to deviate from a pre-defined workflow model in order to adapt to modifications of the original execution plan due to new scientific results obtained in earlier stages of the workflow or even in other, related workflows. This type of flexibility is not addressed in current workflow grid efforts. We argue that missing flexibility rendered workflow efforts unusable in many cases even for business scenarios, and overlooking flexibility requirements in grid workflows might render them unusable. Indeed, a careful study of scientific workflow applications such as the NSF SEEK project [1] show

that flexibility can even be achieved by suitable service compositions which are able to exchange individual services against equivalent ones on the fly.

A second aspect is distributed execution logging. By providing detailed information on the activities executed in a given scientific workflow, the data set under investigation can be traced back to the original data set. In addition, the particular tools which were used to perform given workflow tasks are also logged, so that for each data set a respective workflow system can return the activities, data sets and implementations used to come up with a particular result. This type of information is called data lineage, and workflow support in grid environments should address this issue.

5. Conclusions

In this paper we discuss that workflows in grid environments in general and workflows in scientific grids in particular should take advantage of the workflow concepts, languages, and tools that have been developed by the workflow community in the past ten years, as described in Section 3. Scientific applications impose new requirements to workflow support, namely flexibility and distributed logging. None of these are sufficiently addressed in current grid workflow proposals. In order to be useful for experiment management and data lineage, these types of requirements need to be supported adequately.

There are a number of additional related activities that receive much attention recently, including service level agreement management in the context of Web and Grid Services, both of which might also have further implications on the design and implementation of grid workflow solutions.

References

1. Ludäscher, B., I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, Y. Zhao: Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience*, Special Issue on Scientific Workflows, 2005.
2. Meidanis, J., Vossen, G., Weske, M.: Using Workflow Management in DNA Sequencing (Extended Abstract); Proceedings of the 1st CoopIS, 1996, Brussels, IEEE Computer Society Press, 114-123.
3. Wainer, J., Weske, M., Vossen, G., Bauzer Medeiros, C.: Scientific Workflow Systems; Proc. NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-Art and Future Directions, May 1996, Athens, Georgia, 1-5.
4. Rinderle, S.; Reichert, M.; Dadam, P.: Correctness Criteria for Dynamic Changes in Workflow Systems - A Survey. In: Weske, M., van der Aalst, W.M.P., and Verbeek, H.M.W. (Editors): Data and Knowledge Engineering, Special Issue on Business Process Management. Elsevier, 2004.
5. Foster, Ian; Kesselman, Carl: The Grid: Blueprint for a New Computing Infrastructure. 2nd Edition, Elsevier, 2004.
6. Huhns, M.N., Singh, M.P.: Service-Oriented Computing: Key Concepts and Principles; IEEE Internet Computing Jan./Feb. 2005, pp. 75 – 81.
7. Yu, Jia; Buyya, Rajkumar: A Taxonomy of Workflow Management Systems for Grid Computing. GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, 2005.
8. Allcock, W. (editor): GridFTP: Protocol Extensions to FTP for the Grid. Argonne National Laboratory. GFD-R-P.020 Proposed Recommendation. http://www.ggf.org/documents/GWD-R/GFD-R.020.pdf
9. Vossen, G., Weske, M.: The WASA2 Object-Oriented Workflow Management System; Proc. ACM SIGMOD International Conference on Management of Data 1999, Philadelphia, PA, 587-589