

Workflow Management for Loosely Coupled Simulations

A. Hoheisel

Fraunhofer Institute for Computer Architecture and Software Technology (Fraunhofer FIRST), Berlin, Germany (andreas.hoheisel@first.fraunhofer.de)

Abstract: In the distributed computing domain, several approaches have been evolved that allow the user not only to execute single programs on single hardware resources but also to support workflow schemes that enable the composition and execution of complex applications on distributed resources. Within the Fraunhofer Resource Grid, we developed a Grid computing architecture for composing and executing loosely coupled simulations by means of an abstract dataflow and control flow description. We use a Petri-net-based workflow model that allows the graphical definition of arbitrary workflows with only few basic graph elements – just by connecting data and software components. The communication between the distributed simulation components is realized via file transfer in the actual implementation, but we plan to include tighter coupling schemes using Web Service technology. We validated our approach with applications from the environmental simulation domain, such as the Environmental Risk analysis and Management System ERAMAS.

Keywords: Grid Computing; Coupled Simulations; Workflow Management

1. INTRODUCTION

Currently, there is a high demand of building complex simulation systems that are composed of several distributed and coupled models. One reason for this trend is the increasing specialization of researchers and research organizations to a specific scientific topic on the one hand and the wish to combine the knowledge included in the models to an overall picture on the other hand. Environmental simulation projects often involve a variety of research groups from different organizations that are distributed all over the world. In the past, a common way to couple several models was to merge the models to one monolithic program and to use direct FORTRAN or C function or subroutine calls to realize the communication between the models. Nowadays, this approach is reaching its limits because of the complexity of the models and the difficulty to bundle the know-how of all the models in one working group. Furthermore, the implementations of the models require more and more computational power as well as data storage capacity, and the simulation models are often written in different programming languages for specific operating systems that run on special hardware platforms, such as Linux clusters or MS-Windows PCs.

One approach that promises to satisfy all these requirements is the new paradigm of distributed computing called *Grid computing*. The main vision of Grid computing is to realize a unified interface for arbitrary computational resources – including hardware, software and data – that everybody can use without having to care about the hardware infrastructure and the implementation details of the software components; just as easy as getting electricity through a standardized plug from the *electric power grid* [Foster and Kesselman, 2003]. While it is unsure if this overall vision will ever be accomplished, it can be said that the Grid computing hype is coming to maturity and that important parts of the Grid computing technology have reached production status. In contrast to *cluster computing*, Grid computing considers heterogeneous and non-reliable computing environments as well.

In the simulation community, the research and development of Grid computing techniques was first driven by the demand for computational power and data storage [NASA, 2004; Segal, 2000]; meanwhile nowadays organizational issues – such as building virtual organizations, enabling complex workflows and collaborative working, as

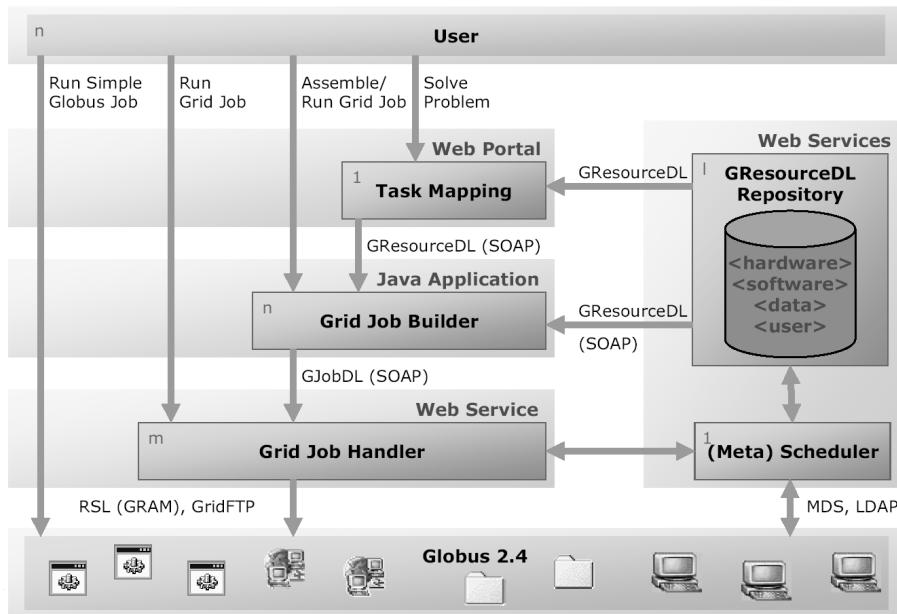


Figure 1. The layered Grid architecture of the Fraunhofer Resource Grid.

well as security, accounting, and billing aspects – are gaining more importance.

Several techniques have been established in the Grid community in order to define the workflow of coupled applications. A very promising approach is the usage of graphs for this purpose as they possess very intuitive ways of visualization that can be handled easily even by non-expert users. This article focuses on the workflow management system for loosely coupled simulation models being developed within the Fraunhofer Resource Grid (FhRG) [Fraunhofer, 2004]. In contrast to other workflow approaches which usually are based on directed acyclic graphs, the FhRG workflow is built on the more expressive formalism of Petri nets [Petri, 1962]. *Dynamic workflow graph refinement* is introduced as a technique to transform abstract workflow graphs into the concrete ones needed for execution and to automatically add fault tolerance to the workflows [Hoheisel and Der, 2003b].

The following section describes the architecture of the Fraunhofer Resource Grid. Section 3 focuses on the techniques we use to define the workflow of coupled simulation models and their data. The enactment of the workflow on a Grid computing environment is presented in section 4, whereas section 5 shortly lists some applications that are being ported to the Fraunhofer Resource Grid as case studies. I complete this paper with conclusions and future work in section 6.

2. FRAUNHOFER RESOURCE GRID

The Fraunhofer Resource Grid (FhRG) is a Grid initiative of five Fraunhofer institutes funded by the German federal ministry of education and research with the main objective to develop and to implement a stable and robust Grid infrastructure within the Fraunhofer-Gesellschaft, to integrate available resources, and to provide internal and external users with an easy-to-use interface for controlling distributed applications and services in the Grid environment [Fraunhofer, 2004].

The component environment supports loosely coupled software components where each software component represents an executable file that reads input files and writes output files. The execution of such a software component we call *atomic job*. The communication is realized via file transfer. Legacy code can be integrated easily using shell scripts to encapsulate the program. We plan to include *Web Service invocations* as atomic jobs in future releases of the FhRG framework in order to make it OGSA compatible [Foster et al., 2002]. Up to now, the workflow architecture does not support tight coupling schemes like CORBA [OMG, 2002], MPI [1997], or HLA, but tightly coupled applications can be included as a whole like an atomic job. Most of the software developed within the Fraunhofer Resource Grid will be made Open Source (GPL) under the label *eXeGrid* [2004].

Figure 1 depicts the Grid architecture of the Fraunhofer Resource Grid that is currently built on top of the Globus 2.4 toolkit [Globus, 2003]. The

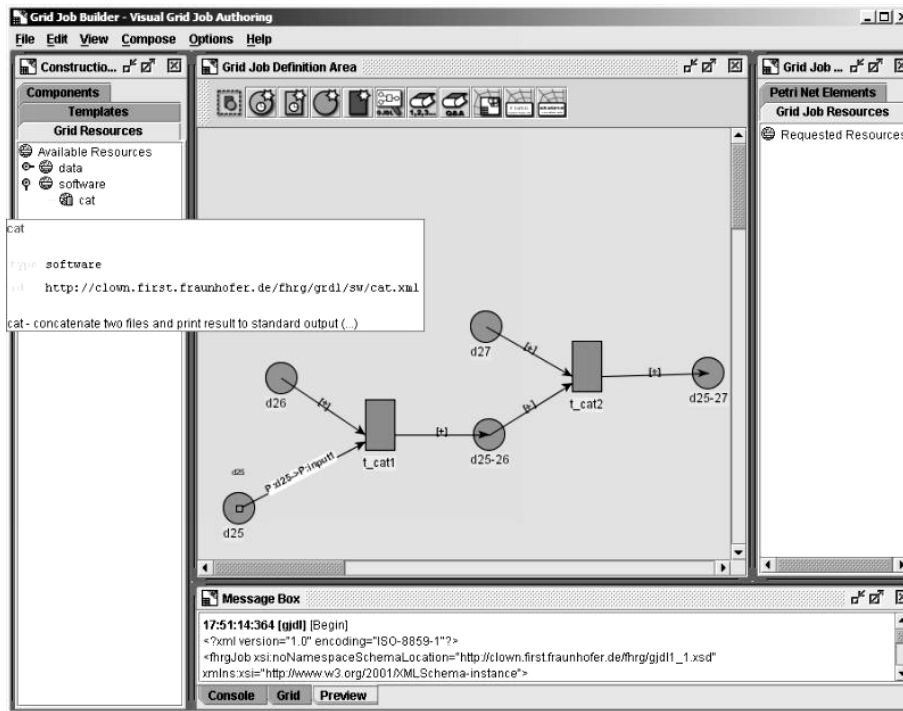


Figure 2. A screenshot of the Grid Job Builder, developed by Fraunhofer IGD, including a Grid resource browser (*left*), a composition panel for Petri-net-based workflows (*middle*), and a job inspector (*right*). The Grid Job Builder supports drag and drop to introduce new components to the Grid job workflow.

user has four alternatives to access Grid resources within this architecture:

1. The user can directly use the standard *Globus services* like GRAM or GridFTP in order to run simple Globus jobs (atomic jobs) on a specified Grid node.
2. If the user wants to run a predefined Grid job, e.g., a coupled simulation, he can use the *Grid Job Handler Web Service*. In this case, the user must provide an XML document that specifies the Grid job. The selection of suitable Grid resources is done during runtime based on current information.
3. The user may use the graphical *Grid Job Builder* to assemble and configure the resources to form a coupled Grid job.
4. If the user does not know which resources to use in order to solve his problem, he may invoke the *Task Mapping* of the web portal. There, the user navigates through a task tree in order to restrict the application area of the problem and to map it onto a suitable set of Grid resources.

3. COMPOSING COUPLED SIMULATIONS

We define the term *Grid job* as a Grid application that is composed of several Grid resources with a

specified workflow. A Grid job can, e.g., represent a complex simulation run or sequential data processing steps, and it may induce a variety of single tasks (atomic jobs) that are indivisible components of a Grid job. According to our definition, *Grid resources* are either abstract classes or concrete instances of software, hardware or data.

There are several possibilities to provide a workflow management that coordinates the execution of Grid jobs. The workflow is either defined inherently by the software components themselves, or by software agents that act on behalf of the software components, resulting in a self-organizing or hard-wired Grid job. Another alternative is to define the workflow on a meta level on top of the software components, providing a complete view of the workflow. To describe this kind of workflow it is very important to have suitable semantics.

In our approach, we use a Petri-net-based workflow model that allows the graphical definition of arbitrary workflows with only few basic graph elements – just by connecting data and software components. Figure 2 shows a screenshot of the Grid Job Builder, a Java application providing a graphical user interface for assembling such Grid jobs [Jung et al., 2004]. The output of the Grid Job Builder is a *Grid Job Definition Language (GJobDL)* document, which defines the Grid job.

This GJobDL document can be saved as a file or transmitted directly to the Grid Job Handler Web Service in order to enact the workflow. The GJobDL description of a Grid job contains the resource descriptions of the basic resources that are required to define the Grid job and the model of the Grid job workflow using the concept of Petri nets [Petri, 1962] as shown in Figure 3.

The idea to use Petri nets to control the workflow of complex applications has been borrowed from the Graphical Simulation Builder that is being developed by the Potsdam Institute for Climate Impact Research (C. Ionescu, pers. comm.). Marinescu [2002] describes a similar approach in his book about internet-based workflow management.

Petri nets belong to a special class of directed graphs. The type of Petri nets we introduce here corresponds to the concept of Petri nets with individual tokens (colored Petri net) and constant arc expressions which are composed of places, denoted by circles (○), transitions, denoted by boxes (□), arcs from places to transitions (○→□), arcs from transitions to places (□→○), individual and distinguishable objects that flow through the net as tokens (•), an initial marking that defines the objects which each place contains at the beginning, and an expression for every arc that denotes an individual object. A place p is called input place (output place) of transition t if an arc from p to t (from t to p) exists. A brief introduction to the theoretical aspects of colored Petri nets can be found, e.g., in Jensen [1994]. The standardization of the Petri net concept is currently in progress as an ISO 15909 committee draft [ISO, 1997]. Van der Aalst and Kumar [2000] give an overview of how to describe different workflow patterns using Petri nets. Petri nets are suitable to describe the sequential and parallel execution of tasks with or without synchronization; it is possible to define loops and the conditional execution of tasks.

We use Petri nets not only to *model*, but furthermore to *control* the workflow of Grid jobs. In most cases, the workflow within Grid jobs is equivalent to the dataflow, i.e., the decision when to execute a software component is taken by means of availability of the input data. Therefore, the tokens of the Petri net represent real data that is exchanged between the software components. In this case, we use Petri nets to model the interaction between software resources represented by software transitions, and data resources represented by data places. In some cases, however, the workflow is independent from the dataflow, and in addition to the data places and software transitions we have to introduce control places and control transitions. The corresponding tokens contain the exit status of the process (e.g., done, failed). Control transitions

evaluate logical conditions. For further details about this Petri net approach refer to Hoheisel and Der [2003a; 2003b].

We introduced a dedicated XML syntax – similar to the Petri Net Markup Language (PNML) developed by Weber and Kindler [2002] – in order to describe Petri nets within the GJobDL. The job description consists of the declaration of the places, transitions, and arcs that build the Petri net of the Grid job. Transitions and places may be linked to external or internal resource descriptions. Control transitions may possess conditions that are evaluated prior to the firing of activated transitions.

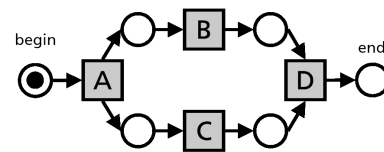


Figure 3. A Petri net representing the parallel execution of atomic jobs B and C and the sequential execution of A , B , D and A , C , D .

4. EXECUTING COUPLED SIMULATIONS

The Grid Job Handler is responsible for the enactment of the coupled simulation. Therefore, the Grid Job Handler parses the Grid job description, resolves the dependencies between the Grid resources, and searches for sets of hardware resources that fulfill the requirements of each software component. A meta scheduler (see Figure 1) selects the best-suited hardware resource of each set of matching hardware resources according to a given scheduling policy (fastest, cheapest, etc.). In the current implementation, the Grid Job Handler maps the resulting atomic jobs onto the Globus Resource Specification Language (RSL) [Globus, 2000] and submits them via GRAM to the corresponding Grid nodes. For the communication between the Grid Job Handler and the Globus Grid middleware, we use a patched version of the Java Commodity Grid Kit [von Laszewski et al., 2001]. The Grid Job Handler itself is deployed as a Web Service with possibilities to create, run and monitor Grid jobs remotely. The desktop version of the Grid Job Handler includes a graphical user interface (see Figure 4) and additional command line tools.

The refinement model of the Petri net theory allows substituting parts of a Petri net by new sub Petri nets. The Grid Job Handler takes advantage of this feature and supplements the workflow during runtime by introducing additional tasks that are necessary to complete the Grid job. The user is not

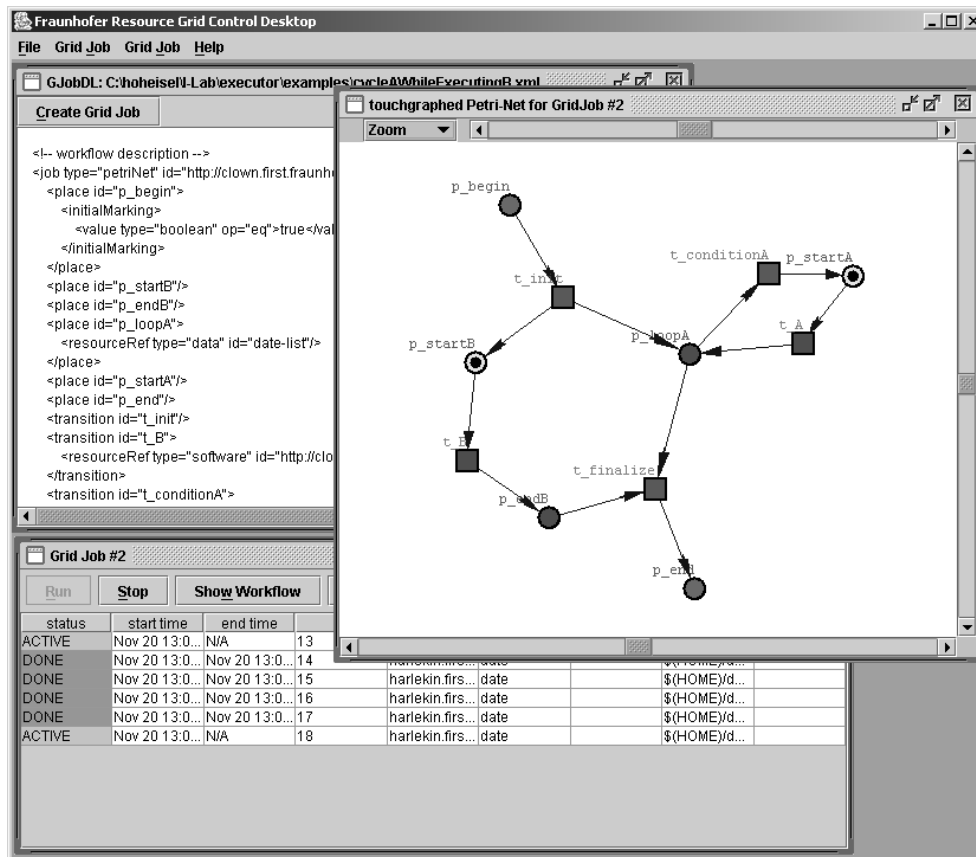


Figure 4. Screenshot of the graphical Grid Job Handler user interface. The upper left panel displays an excerpt of the GJobDL document. The right panel shows a graphical representation of the corresponding Grid job workflow. The lower left panel lists the atomic jobs that are induced by the Grid job with their actual status.

required to model every detail of the workflow – he just has to include the essential transitions and places that are related to the software components and the data he wants to include in his Grid job. Additional tasks that have to be invoked due to specific properties of the Grid infrastructure (e.g., network topology) are detected by the Grid Job Handler and considered by automatically introducing additional transitions and places before or during runtime of the Grid job. In the current version of the Grid Job Handler, data transfer tasks and software deployment tasks are automatically added to the workflow if they are missing in the initial Grid job definition provided by the user. A data transfer task may be introduced to transfer files that are not available on the remote computer via GridFTP. A software deployment task may be introduced to install software components on a remote computer using standard Globus protocols. Further Petri net refinements could concern authorization, accounting, billing and fault management tasks.

5. CASE STUDIES

An example Grid application that takes advantage of the described workflow framework is the *Environmental Risk Analysis and Management System (ERAMAS)*, developed by Fraunhofer FIRST in collaboration with the *Ingenieurbüro Beger für Umweltanalyse und Forschung* and the *Dresdner Grundwasser Consulting GmbH* [ERAMAS, 2004; Unger et al., 2003]. ERAMAS is a simulation-based analysis framework for risks caused by carcinogenic and chemically toxic substances that are released during accidents in industrial installations, the transport of dangerous goods or by terrorist attacks. It is designed to be employed for real-time emergency management as well as for preliminary studies concerning approval procedures and emergency plans.

Other applications that are recently ported to the Fraunhofer Resource Grid are MAGMASOFT [2004], LUMOS [2004], and EIQU.

6. CONCLUSIONS AND FUTURE WORK

Within the Fraunhofer Resource Grid, we have developed a workflow management architecture that is suitable for composing and executing loosely coupled simulation models on a Grid computing environment. The Petri-net-based workflow model seems to be a very promising approach that allows the definition of arbitrary workflows with only three different components: transitions, places and arcs. This enables the easy orchestration of complex workflows, including conditions and loops and regarding the dataflow as well as the control flow of coupled simulations. Within this framework, distributed applications can be defined independently from the hardware infrastructure, just by connecting simulation models and data.

Future work that we planned in the domain of workflow management for coupling simulation models considers the implementation of tight coupling schemes on basis of the Web Service standard. Another important issue that we did not cover so far with the framework presented in this paper is the abstraction from specific data formats. In the present implementation it is up to the user to care about data formats and to ensure that the software components, which interchange data within a Grid job, use compatible data formats. It is planned to make use of special meta data in order to achieve automatic data conversion.

7. REFERENCES

- van der Aalst, W., and Kumar, A., XML based schema definition for support of inter-organizational workflow, 2000.
- ERAMAS homepage: <http://www.erasmas.de/>, 2004.
- eXeGrid homepage: <http://www.exeGRID.net/>, 2004.
- Foster, I., and Kesselman, C. (eds.), *The Grid*, Morgan Kaufmann Publishers, Inc., 2003.
- Foster, I., Kesselman, C., Nick, J., and Tuecke, S., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.
- Fraunhofer Resource Grid homepage, <http://www.fhrg.fhg.de/>, 2004.
- Globus Project, *The Globus Resource Specification Language RSL v.1.0*, http://www-fp.globus.org/gram/rsl_spec1.html, 2000.
- Globus Project, *The Globus Toolkit 2.4*, <http://www.globus.org/gt2.4/download.html>, 2003.
- Hoheisel, A., and Der, U., An XML-based Framework for Loosely Coupled Applications on Grid Environments. ICCS 2003. *Lecture Notes in Computer Science*, 2657, 245–254, 2003a.
- Hoheisel, A., and Der, U., Dynamic Workflows for Grid Applications, *Proceedings of the Cracow Grid Workshop '03*, Cracow, Poland, 2003b.
- ISO 15909, High-level Petri Nets – Concepts, Definitions and Graphical Notation. Committee Draft ISO/IEC 15909, Version 3.4, 1997.
- Jensen, K., An Introduction to the Theoretical Aspects of Coloured Petri Nets, *Lecture Notes in Computer Science*, 803, 230–272, 1994.
- Jung, C., Einhoff, M., Noll, S., and Schiffer, N., Grid Job Builder – A Workflow Editor for Computing Grids, *IEEE Information Technology, Coding and Computing*, ITCC, 2004.
- von Laszewski, G., Foster, I., Gawor, J., and Lane, P., A Java Commodity Grid Kit, *Concurrency and Computation: Practice and Experience*, 13, 643–662, 2001.
- LUMOS project homepage, <http://www.projekt-lumos.de/>, 2004.
- MAGMASOFT homepage, <http://www.magmaSoft.com/>, 2004.
- Marinescu, D. C., *Internet-Based Workflow Management – Toward a Semantic Web*, Wiley, 2002.
- Message Passing Interface Forum, MPI-2: Extensions to the Message-Passing Interface, <http://www.mpi-forum.org/docs/>, 1997.
- NASA Information Power Grid. <http://www.ipg.nasa.gov/>, 2004.
- Object Management Group, *Common Object Request Broker Architecture: Core Specification*, 2002.
- Petri, C. A., *Kommunikation mit Automaten*, Ph.D. dissertation, Bonn, 1962.
- Segal, B., *Grid Computing: The European Data Project*, *IEEE Nuclear Science, Symposium and Medical Imaging Conference*. Lyon, France, 2000.
- Unger, S., Hoheisel, A., Beger, E., and Beims, U., ERAMAS – Analyse- und Managementsystem von schadstoffbedingten Umweltrisiken *Technische Überwachung*, 44 (4), 46–49, 2003.
- Weber, M., Kindler, E., *The Petri Net Markup Language*. In: *Petri Net Technology for Communication Based Systems*. *Lecture Notes in Computer Science*, Advances in Petri Nets, 2002.