Amadeus: A Holistic Service-oriented Environment for Grid Workflows *

Ivona Brandic, Sabri Pllana and Siegfried Benkner
Institute of Scientific Computing
University of Vienna
Nordbergstraße 15, 1090 Vienna, Austria
{brandic,pllana,sigi}@par.univie.ac.at

Abstract

In this paper we present Amadeus, which is a holistic service-oriented environment for QoS-aware Grid workflows. Amadeus considers user's requirements, in terms of QoS constraints, during workflow specification, planning, and execution. Within the Amadeus environment, workflows and the associated QoS constraints are specified at a high-level using an intuitive graphical notation. A set of QoS-aware service-oriented components is provided for workflow planning to support automatic constraint-based service negotiation and workflow optimization. For improving the efficiency of workflow planning, we introduce a QoS-aware workflow reduction technique. By following either a static or a dynamic planning strategy the workflow is executed using the selected services in the manner that the specified user's requirements are met. For each phase of the workflow lifecycle we experimentally evaluate the corresponding Amadeus components.

1 Introduction

Grid computing is considered as a promising solution for relevant problems in various domains such as life sciences, financial services, and high performance computing [5]. Commonly the user defines the process needed for the problem solution as a flow of activities, each capable of solving a part of the problem. This form of specification of activities that should be performed on the Grid is referred to as *Grid workflow*. Resources that perform these activities are not necessarily located in the user's vicinity rather they are geographically distributed. This may enable the use of unique resources such as expensive measurement instruments or powerful computer systems. While the potential benefit of Grid computing is evident, much remains to be done in order to make it a widely accepted technology by end users such as medical practitioners or financial managers. In our opinion, for the wide acceptance of Grid technology two aspects are particularly relevant: (1) the process of specification of Grid activities should be further streamlined; (2) the execution of Grid activities should meet user's requirements regarding *Quality of Service (QoS)*.

Existing work addresses various domains of Grid workflows [20], such as the development of planning strategies for the execution of large scale Grid workflows [6] or cost-based scheduling of scientific workflows [12]. However, there is a lack of a holistic environment for Grid workflows that supports QoS in all phases of the workflow lifecycle from specification to execution. Existing Grid workflow systems either support the whole workflow lifecycle but lack QoS support, or provide only partial QoS support for certain phases of the workflow lifecycle.

In this paper we present *Amadeus*, which is a novel service-oriented environment for QoS-aware Grid workflows. A distinguishing feature of *Amadeus* is the holistic approach to QoS support during all stages of the workflow lifecycle: (1) at specification time *Amadeus* provides an adequate tool-support for a high-level graphical specification of QoS-aware workflows, which allows the association of a comprehensive set of QoS constraints to any activity or to the whole workflow; (2) during the planning phase *Amadeus* provides a set of QoS-aware service-oriented components that support automatic constraint-based service negotiation and workflow optimization; (3) during the execution phase, using the information from the planning phase, *Amadeus* executes the workflow activities in the manner that the specified requirements in terms of QoS constraints are met. A prerequisite for the QoS-aware workflow execution are QoS-aware services that are able to offer QoS guarantees. A QoS-aware service enables clients to negotiate about its QoS properties. This kind of support is provided by the Vienna Grid Environment (VGE) [4]. VGE provides application level QoS support, for example with respect to execution time.

*The work described in this paper was supported by the Austrian Science Fund as part of Aurora Project under contract SFBF1102 and by the European Union's GEMSS Project under contract IST 2001-37153.

Proceedings of the Fifth International Conference on
Grid and Cooperative Computing Workshops (GCCW'06)
0-7695-2695-0/06 $20.00 © 2006 IEEE

The main contributions of this paper include: (1) description of a holistic service-oriented environment for Grid workflows and its experimental evaluation; (2) explanation of an approach for QoS-aware workflow reduction that simplifies the planning phase; (3) description of the workflow execution models for static and dynamic planning.

The rest of this paper is organized as follows: Section 2 introduces a set of basic workflow-related prerequisites and briefly describes the relationship among the main components of the *Amadeus* environment. Furthermore, it introduces a sample workflow that is used for illustration and evaluation of the main *Amadeus* components. Section 3 describes how the *Amadeus* environment supports all phases of the QoS-aware Grid workflow lifecycle. A set of experiments that demonstrates the feasibility of QoS-aware planning and execution of Grid workflows is presented in Section 4. We present the related work in Section 5. Section 6 concludes the paper and briefly describes future work.

2 An Overview of Amadeus Environment

In this section we describe the basic prerequisites for QoS-aware Grid workflows, give a brief description of the Amadeus architecture, and present a sample QoS-aware workflow.

2.1 Basic Prerequisites for QoS-aware Grid Workflows

A prerequisite for QoS-aware Grid workflows are the means to express the QoS requirements. Furthermore, the proper execution of QoS-aware workflows demands the availability of QoS enabled services for the execution of activities that need QoS guarantees. Within the *Amadeus* environment we use QoS-aware Grid Workflow Language (QoWL) to express the QoS requirements, and Vienna Grid Environment (VGE) services as QoS enabled services.

QoWL is an XML-based language that comprises a subset of Business Process Execution Language (BPEL) [2] and a set of QoS extensions for specification of the QoS requirements of Grid workflows. The BPEL subset used for QoWL include: *Process*, *Invoke*, *Copy*, *Sequence*, *Flow*, *Receive*, *Reply*, *Switch*, and *While* elements. BPEL elements are extended with the QoS extensions for the specification of abstract and concrete workflows [7]. A distinctive feature of QoWL language is the ability to account, besides performance (i.e. activity execution time) and economical (i.e. activity price) aspects of QoS, for user's preferences regarding the execution *location affinity* for activities with specific security and legal constraints [8].

The VGE [4] is a service-oriented infrastructure for the provision of native applications (e.g. HPC applications) as Grid Services. VGE supports a flexible QoS negotiation model where clients may negotiate dynamically QoS guarantees on execution time, price and other constraints with potential service providers. VGE services encapsulate native HPC applications and offer a set of common operations for job execution, job monitoring, data staging, error recovery, and application-level quality of service negotiation. The main operations for job execution include: *upload* that transfers the input data from the client to the service, *start* that begins the execution of the native application, *download* that transfers the output data from the service to the client and *push* that transfers data form source to destination. VGE services are exposed using WSDL and securely accessed via SOAP/WS-Security. Within the context of European Commission funded GEMSS project [10], VGE has been successfully used for the development of a testbed for six medical simulation and image reconstruction Grid services [13].

2.2 The Amadeus Architecture

In this section we briefly describe the relationship among the main components of the *Amadeus* environment. Section 3 gives a more detailed description of Amadeus components and exemplifies their usage with a sample workflow.

Figure 1(a) shows the architecture of the *Amadeus* environment. The main components include: (1) *visualization and specification component*; (2) *planning, negotiation and execution component* called QoS-aware Grid Workflow Engine (QWE); and (3) a set of *Grid resources*.

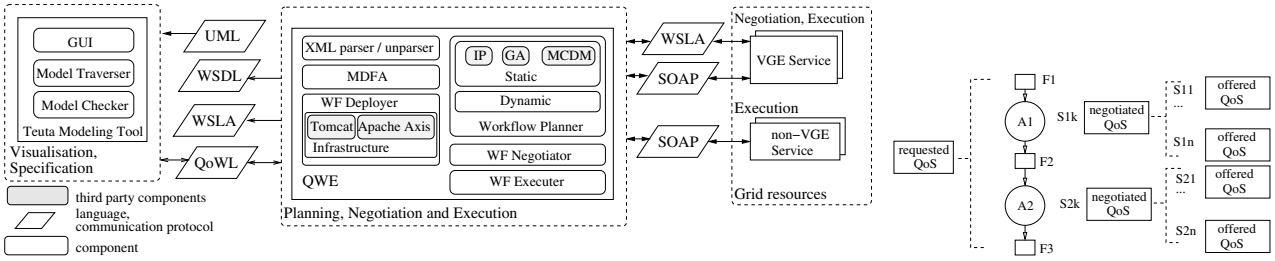
The specification and visualization component comprises Teuta [18], which is a tool for UML based Grid workflow modeling and visualization. A user may specify the workflow with Teuta by composing predefined workflow elements. For each workflow element different properties (such as execution time, price, location affinity) may be specified that indicate user's QoS requirements. After the validation of the specified workflow, Teuta generates the corresponding XML representation following the syntax of *QoWL* [7]. The QWE engine interprets the QoWL workflow, applies the selected planning strategy, negotiates with services, selects appropriate services and finally executes the workflow. For the activities annotated with QoS constraints, we use VGE services which are able to provide certain QoS guarantees. For other activities non-VGE services may be used.

2.3 A Sample QoS-aware Workflow

In this section we describe a workflow (WF) example that we use for illustration and evaluation of the main *Amadeus* components.

We consider three main phases of a workflow lifecycle: *specification*, *planning*, and *execution*. We describe how the

Proceedings of the Fifth International Conference on
Grid and Cooperative Computing Workshops (GCCW'06)
0-7695-2695-0/06 $20.00 © 2006 IEEE



(a) The Amadeus architecture

(b) Sample workflow

Figure 1. The Amadeus architecture and the sample workflow

corresponding *Amadeus* components support each phase of the workflow lifecycle in Section 3. In this paper, we use a sample workflow that is depicted in Figure 1(b) for the illustration and evaluation of *Amadeus*. Our workflow consists of two complex activities $A1$ and $A2$, each comprising a sequence of basic activities[1]. $F1$ represents the input file for activity $A1$, $F2$ is the output file of activity $A1$ and the input file of $A2$, and $F3$ is the output file of activity $A2$. The left hand side of Figure 1(b) depicts the *requested QoS*, which is specified during the workflow *specification* phase. The aim of the *planning* phase is to determine for each activity, from a pool of available services $\{S1, .., Sn\}$, a service with the *offered QoS* that best matches the *requested QoS*. The right hand side of Figure 1(b) depicts that for activity $A1$, the service $S1k$ offers the most suitable QoS.

3 Workflow Specification, Planning and Execution within the Amadeus Environment

In this section we explain how the the *Amadeus* environment supports all phases of a QoS-aware Grid workflow lifecycle. For each phase we describe the responsible *Amadeus* components and evaluate these components using the sample workflow introduced in Figure 2.3.

3.1 Specification and Visualization

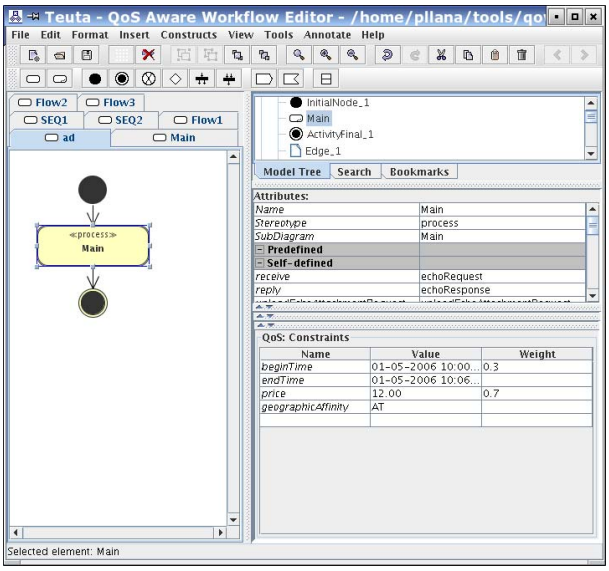
Within the *Amadeus* environment the user specifies workflows graphically using *Teuta*, which is a UML-based graphical editor. Teuta has been designed as a platform independent, configurable and extensible tool. Therefore, Teuta may be extended with new types of diagrams and modeling elements for various domains, such as high performance computing [18]. In the context of *Amadeus* we extended Teuta for the specification of QoS-aware Grid workflows [8].

Figure 2 illustrates the hierarchical specification process of a sample workflow, that we have introduced in Section 2.3, with Teuta. The workflow specification is based on our UML extension for the domain of Grid workflows [8]. The type of modeling elements of our UML extension is indicated with guillemets `<<type>>`. The semantics of these elements is based on QoWL. The user may define a workflow by combining the predefined UML modeling elements that are available in the Teuta tool bar. A set of properties (such as QoS constraints) may be associated to each modeling element by using the property panel that is located on the right-down corner of Teuta GUI (see Figure 2(a)).

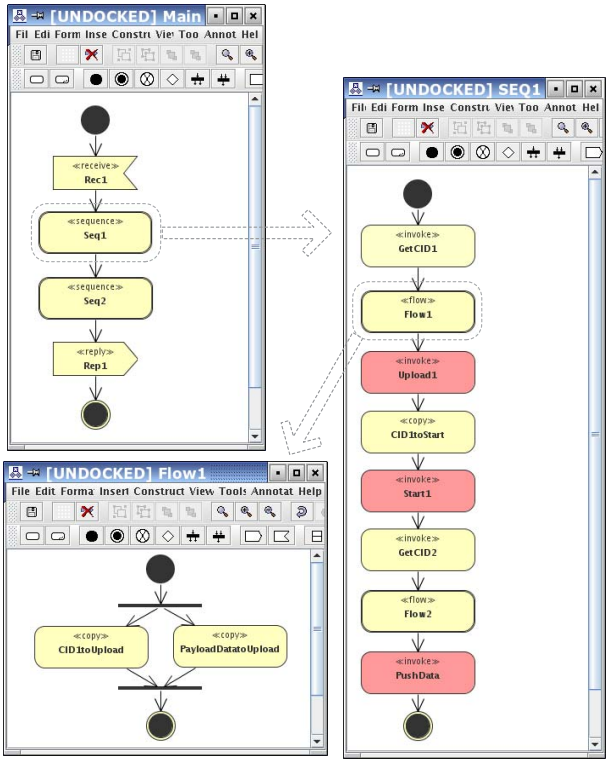
Figure 2(a) depicts the element *Main*, which is an instance of type `<<process>>`. Element *Main* represents the root of the workflow, which encapsulates the whole workflow. The QoS constraints of the `<<process>>` element are shown in the bottom-right corner of Figure 2(a). For instance, the user may define the earliest possible time of the workflow execution $beginTime = 01 - 05 - 2006\ 10 : 00$, and the latest possible time of the workflow completion $endTime = 01 - 05 - 2006\ 10 : 06$. The property $geographicAffinity = AT$ indicates that the selected services of the workflow should be located in Austria. The body of the *Main* element is depicted in the upper-left corner of Figure 2(b). The *Main* element is composed of a `<<receive>>`, `<<reply>>` and two `<<sequence>>` elements *Seq1* (corresponds to $A1$ of sample workflow) and *Seq2* (corresponds to $A2$ of sample workflow). The body of the element *Seq1* is shown on the right-hand side of Figure 2(b). The *Seq1* element contains several `<<invoke>>`, `<<copy>>` and `<<flow>>` elements. The *Upload1*, *Start1* and *PushData* elements are marked with a different color. This indicates compute intensive activities that should be considered for the QoS-aware workflow planning (see Section 3.2.2). The bottom-left corner of Figure 2(b) depicts the body of the complex activity *Flow1* where two `<<copy>>` elements are executed in parallel. *Seq2* is specified analogously.

[1] Basic activities are not shown for the sake of simplicity. The workflow details are shown in Figure 2, Section 3.1.

Proceedings of the Fifth International Conference on
Grid and Cooperative Computing Workshops (GCCW'06)
0-7695-2695-0/06 $20.00 © 2006 IEEE



(a) Association of QoS constraints



(b) Hierarchical workflow specification

Figure 2. Specification of a sample workflow

3.2 Planning

The sample abstract workflow, specified in Figure 2, is transformed into a concrete workflow using the *Workflow Planner* component. The aim of this component is to automate the selection of services in accordance with the *requested QoS*. *Workflow planning* comprises the following phases: (1) selection of the *workflow optimization strategy*, (2) *QoS-aware workflow reduction*, (3) *negotiation* and (4) the *workflow optimization*.

3.2.1 Workflow Optimization Strategy Selection

We distinguish between *static* and *dynamic* workflow planning strategies. The decision whether static or dynamic planning techniques should be used, depends on the *meta data* of the invoked services. If all *meta data* required for QoS prediction is statically known (e.g. the size of the input file), the static planning strategy can be selected. If the *meta data* is generated or changed during workflow execution, the dynamic planning approach has to be used. Static planning implies the generation of the concrete workflow before the execution of the first workflow activity. In the case of the dynamic planning approach the concrete parts of the workflow are created for the ready-to-start activities during the workflow execution. The *Meta Data Flow Analyzer (MDFA)* verifies whether the static planning approach is feasible. We distinguish two types of variables: (1) the *payload variables (PV)*, such as input data of the invoked services, and (2) the *meta data variables (MDV)*, that describe the service input data (for instance the size of the input file, matrix size, etc.). The *MDFA* checks whether any *MDV* appears as output of any *invoke*, *receive* or *copy* activity. In this case, only dynamic planning approach can be used.

3.2.2 QoS-aware Workflow Reduction

The planning for real-world workflows is a NP-hard problem. But, the task of workflow planning may be alleviated by reducing the complexity of the workflow. Commonly, Grid workflows are composed of several activities where some of them are time intensive, cost intensive or security relevant. Such activities determine the QoS of the overall workflow. Other kinds of activities (e.g. control tasks) that do not significantly affect the overall QoS may be neglected during the optimization phase of the workflow planning.

Figure 3(b) depicts the reduced workflow considering only QoS relevant activities $A1$ and $A4$. The process of elimination of activities without QoS constraints is called *QoS-aware workflow reduction*. In the simplest case workflow reduction is performed by eliminating all activities which do not have specified QoS constraints. As shown in Figure 3(b) the QoS model may be specified for: (1) a single activity (e.g. $A1$) which can be basic or complex, (2) the overall workflow (e.g. workflow comprising $A1$ and $A4$). The QoS model considers the selected optimization strategy and the aggregation function. Figure 3(c) shows the reduced workflow model of the sample workflow (see

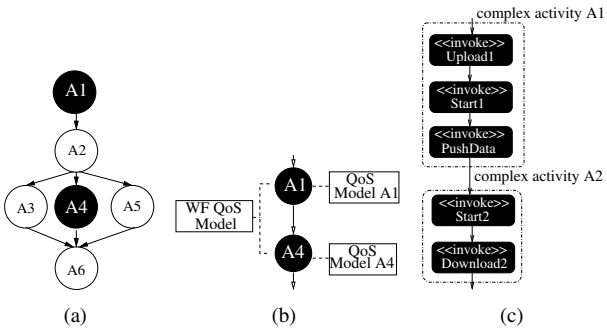


Figure 3. Workflow QoS: (a) QoS-based workflow reduction model, (b) reduced workflow used for WF planning (c) reduced form of the sample workflow.

Figure 2). The workflow considers the following activities: $Upload1$, $Start1$ and $PushData$ which belong to the complex activity $Seq1$ (i.e. $A1$); $Start2$ and $Download2$ activities which belong to the complex activity $Seq2$ (i.e. $A2$). The next section describes the negotiation process within the *Amadeus* environment.

3.2.3 QoS Negotiation

Based on the selected *optimization strategy* the *WF Negotiator* queries the specified registries, generates necessary *requested QoS* and receives *offered QoS* from services. Figure 4 depicts the negotiation process and participating components. The *WF Planner* starts the negotiation by initializing one or more instances of *WF Negotiator*. Each instance is responsible for the negotiation process of one activity. In case of static planning approach negotiation starts concurrently for all activities of the reduced QoS-aware model.

After the initialization of *WF Negotiator*s, each *WF Negotiator* supplies each candidate service with a *QoSRequest* (QoSReq) and a *RequestDescriptor* (ReqDesc). A *QoSReq* contains requested QoS, whereas a *ReqDesc* contains meta data about input data necessary for the evaluation of QoS constraints. As depicted in Figure 4, *QoSReq* is an optional input. If the requested QoS is not specified, the WF Negotiator supplies an empty *QoSReq* and the service responses with *offered QoS* that is not optimized for any QoS constraint (i.e. for time or price). If the *QoSReq* is specified, each service tries to meet the specified constraints. For instance if a low price is requested, a service may run the application on fewer nodes to meet the price constraint. After collecting all offers each *WF Negotiator* notifies the *WF Planner* about received offers. Thereafter, *WF Planner* selects appropriate services which fit into global or local constraints by applying the selected optimization strategy. The selected services are confirmed and the WSLA is generated between services and the QWE.

3.2.4 Optimization

For the *static approach* we use *lp_solve*, which is a mixed-integer linear programming (lp) package [15]. *lp_solve* is suitable for solution of global optimization problems. It can be applied to the planning of the whole workflow, or to the planning of complex activities. The optimization process with *lp_solve* involves the association of an *ActivityID* with each activity of the reduced workflow, and a *serviceID* with each candidate service of an activity. Each QoS-constraint (e.g. maximum time, maximum price) of an activity represents an *integer programming (IP) constraint*. The outcome of the optimization process with *lp_solve* is an array that contains the IDs of the selected services. The order of service IDs within this array corresponds to the execution order of activities of the reduced workflow. The interested reader may find a theoretical elaboration of the IP approach in [7]. Since most of our reduced workflows have the form of a simple straight-line code, IP is a more appropriate approach than complex heuristic techniques (e.g. genetic algorithms).

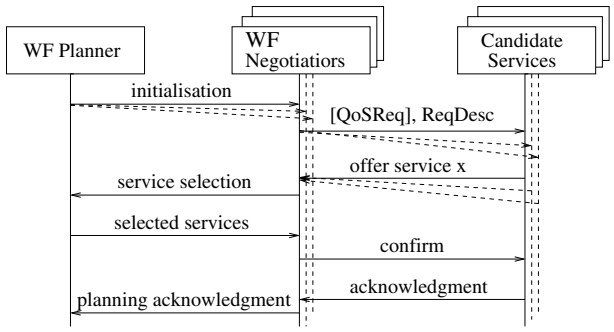


Figure 4. WF Negotiation process

For the *dynamic approach* we use the Multiple Criteria Decision Making (*MCDM*) approach where we select local optima for each activity separately. In case of dynamic approach planning, negotiation and execution processes are invoked in an iterative fashion (see Figure 5 in Section 3.3).

3.3 Execution

The outcome of the workflow planning phase is a concrete workflow which is ready for execution. In the case of static planning approach the workflow execution phase is started after the completion of workflow planning phase. But, in the case of dynamic planning the execution and planning phases are performed for each activity of the workflow in an alternate fashion. Figure 5 illustrates the relationship of workflow planning and execution phases.

Figure 5(a) depicts the relationship between *static workflow planning* and *workflow execution*. Static workflow

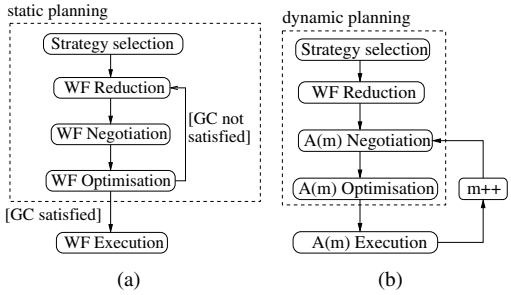


Figure 5. Workflow execution models: (a) with static planning, (b) with dynamic planning.

planning involves the following steps: *Strategy Selection*, *WF Reduction*, *WF Negotiation* and *WF Optimization*. As described in Section 3.2, for the static planning approach global constraints (GC), such as maximum price of the workflow, are specified for the whole workflow. If GCs can not be satisfied after the *WF Negotiation* and *WF Optimization*, the workflow is reduced by excluding the first activity of the workflow. The excluded activity is optimized separately from the rest of the workflow by searching for the candidate service that offers the best QoS. In the next iteration only the reduced workflow is considered. However, the user may annotate each activity with the expected average runtime in order to increase the chance to find a solution with fewer iterations. If the GCs are satisfied, the *WF execution* may start.

In case the execution of certain activities modifies the *meta data*, the dynamic workflow planning is applied (see Figure 5(b)). For each activity $A(m)$ the negotiation and optimization is performed individually before the execution. After the execution of activity $A(m)$, the next activity $A(m+1)$ is considered. This iterative process of dynamic planning is completed when all activities of the workflow are executed.

3.4 Workflow Deployment

The purpose of the *WF Deployer* is to publish the workflow as a Grid service. Based on the XML representation of concrete workflow the corresponding WSDL file is generated, which serves as input for the *WF Deployer*. The *WF Deployer* parses the WSDL file using *wsdl4j*, which is an IBM reference implementation of Java API's for WSDL. Thereafter, *WF Deployer* generates Java code for a Grid service that represents the workflow. Finally, *WF Deployer* deploys the generated Grid service using the Apache Axis and Tomcat [3] to the location that is specified by the user.

In the case of static workflow planning, the WSLA may be provided to the consumer of the workflow. WSLA includes the information on the negotiated QoS for the whole workflow. But, for the dynamic workflow planning WSLA is not provided.

4 Experimental Results

In this section we present experimental results for the sample workflow that we introduced in Section 2.3. The goal of the experiments is to demonstrate the feasibility of QoS-aware planning and execution. Furthermore, we report the time needed for the completion of various phases of the workflow lifecycle such as QoS negotiation, optimization, and workflow execution considering static and dynamic execution models. In our experiment, all files transferred between services and the engine have the size of 100 MB. In contrast to our previous work [7], the files are transferred using *SOAP attachments*, which enable the exchange of large amounts of data.

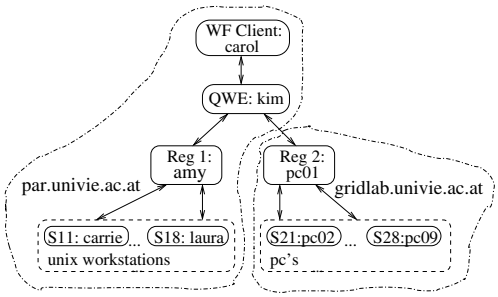


Figure 6. Experimentation platform

Figure 6 depicts the experimentation platform used for execution of the sample workflow. The *Amadeus* components (such as QWE, WF client) and the required Grid software infrastructure are deployed on the depicted experimentation platform (see Figure 6). The candidate services for the complex activity $A1$ of the sample workflow (see Figure 1(b)), namely $\{S1x : 1 \leq x \leq 8\}$, are deployed on eight Unix workstations. The registry $Reg1$ containing the information about $S1x$ services, the QWE and the WF client are deployed each on a different Unix workstation. The workflow client, QWE and the candidate services of activity A1 are deployed in the *par.univie.ac.at* domain. The candidate services for the complex activity $A2$ (see Figure 1(b)), namely $\{S2x : 1 \leq x \leq 8\}$, are deployed on eight PCs. Registry $Reg2$ is deployed on a separate PC. The registry and the candidate services of $A2$ are deployed in the *gridlab.univie.ac.at* domain.

Table 1 depicts the experimental results for the static planning approach. In this experiment, the solution (i.e. services that satisfy GCs) was found already in the first iteration of planning (i.e. *lp_solve* was called only once) due to the fact that we use a rather simple sample workflow. For

Proceedings of the Fifth International Conference on
Grid and Cooperative Computing Workshops (GCCW'06)
0-7695-2695-0/06 $20.00 © 2006 IEEE

real-world workflows more than one iteration of the planning phase may be needed to find a solution. The time to *upload* the file from client to the service $S1x$ is 27.676 seconds. Whereas, the time to *push* the file from service $S1x$ to $S2x$ is 18.086 seconds. This difference in data transfer time may be due to the fact that service $S1x$ is deployed on a machine with a shared file system with intensive read/write activity. The invocation of *Start1* operation starts a Java application which simulates invocation of a native application, which spends 2 minutes and 10.066 seconds for data processing. The total workflow completion time is defined as $T_{total} = T_{plan} + T_{exec}$. T_{plan} comprises the time needed for the negotiation with the candidate services and optimization with *lp_solve*. The execution time is defined as $T_{exec} = T_{red} + T_{rem}$. T_{red} comprises the time for the execution of the activities of the reduced workflow as depicted in Figure 3(c). T_{rem} indicates the time for the execution of the remained activities that are not included in the reduced workflow. As assumed T_{total} is dominated by T_{red} which is 98.16% of the overall workflow execution time (see the rightmost column of the Table 1).

WF Phase	Activity	T	T [%]
Planning	UML2QoWL transformation	0.095 s	0.03
	QoS negotiation $S1x$, $S2x$	3.319 s	1.01
	lp_solve optimization	0.553 s	0.17
	T_{plan}	3.967 s	1.21
Execution	<<invoke>> Upload1 ($S1x$)	27.676 s	8.41
	<<invoke>> Start1 ($S1x$)	2 m 10.066 s	39.53
	<<invoke>> PushData ($S1x$)	18.086 s	5.50
	<<invoke>> Start2 ($S2x$)	2 m 10.039 s	39.53
	<<invoke>> Download2 ($S2x$)	17.078 s	5.19
	T_{red}	5 m 22.950 s	98.16
	T_{rem}	2.088 s	0.63
	$T_{exec} = T_{red} + T_{rem}$	5 m 25.033 s	98.79
	$T_{total} = T_{plan} + T_{exec}$	5 m 29 s	100.00

Table 1. Static workflow planning and execution

Table 2 depicts the experimental results for the dynamic planning approach. Planning is performed for each activity individually before the execution. This involves multiple negotiations with candidate services, which results in an increase of total planning time for the complete workflow. For the static planning approach we obtained T_{plan} = 3.967 seconds (see Table 1), whereas for the dynamic planning approach the total planning time was T_{plan} = 7.26 seconds (see Table 2). Therefore, the total workflow completion time for the dynamic approach (T_{total} = 5 minutes and 34 seconds) is larger than in the static case (T_{total} = 5 minutes and 29 seconds). The workflow completion time T_{total} is dominated by T_{red}, which is 97.31% of the total workflow completion time (see the rightmost column of the Table 2). This demonstrates that the subset of activities that are comprised in the reduced workflow (see Figure 3(c)) determines the completion time of the whole workflow.

WT Phase	Activity	T	T [%]
Planning 1	UML2QoWL transformation	0.095 s	0.03
	QoS negotiation $S1x$	4.544 s	1.36
	T_{plan1}	4.639 s	1.39
Execution 1	<<invoke>> Upload1 ($S1x$)	28.567 s	8.55
	<<invoke>> Start1 ($S1x$)	2 m 10.081 s	38.95
	<<invoke>> PushData ($S1x$)	18.234 s	5.46
	T_{red1}	2 m 56.882 s	52.96
Planning 2	QoS negotiation $S2x$	2.621 s	0.78
	T_{plan2}	2.621 s	0.78
Execution 2	<<invoke>> Start2 ($S2x$)	2 m 10.094 s	38.95
	<<invoke>> Download2 ($S2x$)	18.031 s	5.40
	T_{red2}	2 m 28.125 s	44.35
	$T_{plan} = T_{plan1} + T_{plan2}$	7.260 s	2.17
	$T_{red} = T_{red1} + T_{red2}$	5 m 25.007 s	97.31
	T_{rem}	1.733 s	0.52
	$T_{exec} = T_{red} + T_{rem}$	5 m 26.740 s	97.83
	$T_{total} = T_{plan} + T_{exec}$	5 m 34 s	100.00

Table 2. Dynamic workflow planning and execution

5 Related Work

Most of existing work on workflow systems may be grouped in three categories: (1) systems that are tailored for *scientific workflows* based on Globus Grid infrastructure [6, 1]; (2) systems for *business workflows* that are based on Web Services related technologies (such as SOAP, WSDL and BPEL) [16, 11, 19]; and (3) systems for scientific workflows that use and further extend the standard Web Services technologies developed for business workflows [17, 14, 7, 9]. In comparison to business workflows, the distinguishing features of Grid workflows are the long execution time of compute intensive activities and the large data transfer between activities (typically the data exchange is performed via large files).

However, there is a lack of a holistic environment for Grid workflows that supports QoS in all phases of the workflow lifecycle from specification to execution. The Gridbus project is addressing the QoS-aware Grid workflows [12]. But, within the framework of the Gridbus project workflows are specified textually based on XML, which has been proved as a non-intuitive and error-prone approach. While

time and cost constraints are considered, there is no support for security and legal QoS constraints [8]. In contrast our *Amadeus* system provides QoS support during the whole workflow lifecycle from specification to execution considering a rich set of QoS constraints. Besides performance (i.e. activity execution time) and economical (i.e. activity price) aspects of QoS, *Amadeus* considers the user's preferences regarding execution *location affinity* for activities with specific security and legal constraints [8].

6 Conclusions and Future Work

In this paper we have presented a holistic service-oriented environment that supports the whole lifecycle of QoS-aware Grid workflows. For each phase of the workflow lifecycle we have used a sample workflow to illustrate and experimentally validate the usefulness of *Amadeus* environment. By applying our technique of QoS-aware workflow reduction we have obtained a reduced workflow suitable for optimization with linear programming. The reduced workflow comprises fewer activities with a simple control flow, which reduces the complexity of planning. We have observed that the time spent for optimization based on linear programming represents only a small percentage (about 0.17%) of the total workflow completion time. Furthermore, in our experiments we have observed that about 98% of workflow completion time is spent for execution of activities that are comprised in the reduced workflow. By optimizing the execution of activities of the reduced workflow the total workflow completion time may be improved.

In the future we plan to extend our approach with workflow adaptivity mechanisms.

References

[1] K. Amin, G. von Laszewski, M. Hategan, N. Zaluzec, Sh. Hampton, and A. Rossi. *GridAnt: A Client-Controllable Grid Workflow System*. 37th Hawaii International Conference on System Sciences. Big Island, HI, USA, 2004.

[2] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana: *"Business Process Execution Language for Web Services Version 1.1"*, 2003 http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[3] Apache Tomcat http://tomcat.apache.org

[4] S. Benkner, I. Brandic, G. Engelbrecht, R. Schmidt. *VGE - A Service-Oriented Grid Environment for On-Demand Supercomputing.* Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (Grid 2004), Pittsburgh, PA, USA, November 2004.

[5] F. Berman, G. Fox, A. J.G. Hey. *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley & Sons, 2003.

[6] J. Blythe, E. Deelman, Y. Gil. *Automatically Composed Workflows for Grid Environments*. IEEE Intelligent Systems 19(4): 16-23 2004.

[7] I. Brandic, S. Benkner, G. Engelbrecht, R. Schmidt. *QoS Support for Time-Critical Grid Workflow Applications.* Proceedings 1st IEEE International Conference on eScience and Grid Computing, Melbourne, Australia, December 2005.

[8] I. Brandic, S. Pllana, S. Benkner. *High-level Composition of QoS-aware Grid Workflows: An Approach that Considers Location Affinity.* Workshop on Workflows in Support of Large-Scale Science. In conjunction with HPDC-15, June 19th-23rd 2006, Paris, France, 2006..

[9] J. Chen and Y. Yang. *Multiple States based Temporal Consistency for Dynamic Verification of Fixed-time Constraints in Grid Workflow Systems*, Concurrency and Computation: Practice and Experience, 2006, 18, pp. 1-18.

[10] The GEMSS Project: Grid-Enabled Medical Simulation Services, EU IST Project, IST-2001-37153, http://www.gemss.de

[11] K. Gomadam, K. Verma, A. P. Sheth, J. A. Miller. *Demonstrating Dynamic Configuration and Execution of Web processes*. 3rd International Conference on Service Oriented Computing, Amsterdam, The Netherlands, 2005.

[12] The Gridbus Project. http://www.gridbus.org

[13] D. M. Jones, J. W. Fenner, G. Berti, F. Kruggel, R. A. Mehrem, W. Backfrieder, R. Moore, A. Geltmeier. *The GEMSS Grid: An evolving HPC Environment for Medical Applications*, HealthGrid 2004, Clermont-Ferrand, France, 2004.

[14] P. M. Kelly, P. D. Coddington, A. L. Wendelborn. *Distributed, Parallel Web Service Orchestration Using XSLT*. First International Conference on e-Science and Grid Computing Melbourne, Australia, December 2005.

[15] The lp_solve Project http://www.geocities.com/lpsolve 2005.

[16] J. Oberleitner, F. Rosenberg, S. Dustdar.*A Lightweight Model-driven Orchestration Engine for e-Services*. 6th VLDB Workshop on Technologies for E-Services, colocated with VLDB Trondheim, Norway, 2005.

[17] B. Plale, D. Gannon, D. A. Reed, S. J. Graves, K. Droegemeier, B. Wilhelmson, M. Ramamurthy. *Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD*. 5th International Conference on Computational Science, Atlanta, GA, USA, 2005.

[18] S. Pllana and T. Fahringer. *Performance Prophet: A Performance Modeling and Prediction Tool for Parallel and Distributed Programs*. The 2005 International Conference on Parallel Processing (ICPP 2005 Workshops), Oslo, Norway, June 2005.

[19] Y. Wang, M. Li, J. Cao, F. Tang, L. Chen, L. Cao: *An ECA-Rule-Based Workflow Management Approach for Web Services Composition*. 4th International Conference on Grid and Cooperative Computing, Beijing, China, 2005.

[20] J. Yu and R. Buyya. *A Taxonomy of Workflow Management Systems for Grid Computing*, Technical Report, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005. http://www.gridbus.org/reports/GridWorkflowTaxonomy.pdf