High-level Composition of QoS-aware Grid Workflows: An Approach that Considers Location Affinity *

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

At a high level of abstraction Grid applications are commonly specified based on the workflow paradigm. We consider that for the wide acceptance of Grid technology it is relevant that the user has the possibility to express requirements on Quality of Service (QoS) at workflow specification time. However, most of the existing workflow languages lack constructs for QoS specification, or provide only limited QoS support. In this paper we present an approach for high level workflow specification that considers a comprehensive set of QoS requirements. Besides performance related QoS, it includes economical, legal and security aspects. For instance, for security or legal reasons the user may express the location affinity regarding Grid resources on which certain workflow tasks may be executed. Our QoS-aware workflow system provides support for the whole workflow life cycle from specification to execution. Workflow is specified graphically, in an intuitive manner, based on a standard visual modeling language. We illustrate our approach with a real-world workflow for maxillo facial surgery simulation.

1 Introduction

The emergence of Grid technology is strongly affecting the way in which the information processing tasks are performed. Grid users may specify the tasks that should be performed at several levels of abstraction that directly reflect their role within an organization. At the top level of abstraction the user specifies the tasks that should be performed on the Grid as *workflow*. This approach has the advantage that the user can map the problem from his/her domain of interest to a workflow in a straightforward manner. As a consequence, the user does not have to be an expert of Grid technology in order to specify the job that should be performed. Therefore, the workflow paradigm is considered to be very relevant for the wide acceptance of Grid technology.

Currently a significant research effort is invested in the development of workflow languages for Grid environments [25]. However, most of the existing workflows are specified in textual form, or based on a self-defined graphical notation. Moreover, there is a lack of adequate tool support for workflow specification, and workflow specification tools are usually not well integrated with the Grid environment. Furthermore, many workflow languages provide no language constructs for QoS specification, or provide only limited QoS support (for instance, only for performance and economical related QoS). We believe that while performance aspect is of paramount importance for time critical applications, the wide acceptance of Grid technology strongly depends on security and legal aspects. We have experienced that many potential users from industry hesitate to use Grid technology even if the performance and economical benefits are clear because of security and legal concerns.

There is a large number of application domains for Grid workflows, such as life sciences (e.g. medical simulation services) and engineering (e.g. vehicle development support), that demand a guarantee that workflow activities are performed within the specified *time*, *cost*, *security* and *legal* constraints. Therefore, we are developing an XML based language for QoS-aware Grid workflows (QoWL), by extending Business Process Execution Language (BPEL) [1] with language constructs for specification of QoS constraints [6]. A distinctive feature of QoWL language is the ability to account for user's preferences regarding the execution *location affinity* for activities with specific security and legal constraints. The concept of *location affinity*, introduced in this paper, conforms to the idea of *Virtual*

*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.

Organizations [11]. In order to streamline the process of workflow specification, we have extended our graphical editor Teuta [20] for QoWL. The user specifies with Teuta the workflow graphically by using the Unified Modeling Language (UML) [17]. From this representation Teuta automatically generates the corresponding QoWL representation, which serves as input to QoS-aware Grid Workflow Engine (QWE). QWE performs necessary steps for the QoS-aware workflow negotiation and execution [7]. A prerequisite for the QoS-aware workflow execution are QoS-aware services able to give QoS guarantees. A QoS-aware service enables clients to inquire about its QoS properties. This kind of support is provided by Vienna Grid Environment (VGE) services [4]. VGE has been utilized within the European Commission funded GEMSS project which developed a testbed for six medical simulation and image reconstruction Grid services [13]. We evaluate our approach by specifying a QoS-aware Grid workflow for maxillo facial surgery simulation.

The main contributions of this paper include: (1) development of the language support for specification of security and legal QoS constraints; (2) definition of a UML based Domain Specific Language (DSL) for QoS-aware Grid workflows; and (3) extension of Teuta for QoWL, and integration of Teuta with QWE.

The rest of this paper is organized as follows. Section 2 describes our approach for graphical specification of QoS-aware Grid workflows. Our implementation is outlined in Section 3. Section 4 demonstrates the application of our approach by modeling a maxillo facial surgery simulation workflow. We compare and contrast the work presented in this paper with the related work in Section 5. Section 6 presents our conclusions and describes the future work.

2 High-level Specification of QoS-aware Workflows

In this section we first briefly describe the QoWL language, which is an XML based language for QoS-aware Grid workflows. Thereafter, we present a UML-based graphical representation of QoWL.

2.1 Quality of Service Aware Grid Workflow Language (QoWL)

QoWL comprises a subset of Business Process Execution Language (BPEL) [1] and a set of QoS extensions that is used for specification of the QoS requirements of Grid workflows. The elements of our BPEL subset include: *Process*, *Invoke*, *Copy*, *Sequence*, *Flow*, *Receive*, *Reply*, *Switch*, and *While*. BPEL elements are extended by the QoS extensions necessary for the specification of QoS before the QoS negotiation and for the expression of QoS after the negotiation.

Figure 1. The structure of a QoWL element

Figure 1 depicts the structure of a QoWL element. A QoWL element is defined as an extended BPEL element with a set of QoS constraints. The attributes of the `qowl-element`, such as `name` and `portType`, are used as defined in the BPEL specification [1]. The `<qos-constraints>` element specifies the QoS of a specific workflow element. The attribute `reqDescVar` defines the variable which specifies the meta data. Based on the specified meta data (e.g. image resolution) a QoS-aware service can predict the execution time of the service for a specific request. Each `<qos-constraints>` element may contain several `<qos-constraint>` elements. Each `<qos-constraint>` element specifies a QoS constraint as a tuple $(name, value, weight)$.

Commonly used QoS constraints of QoWL for Grid workflows include: (1) `beginTime`, which describes the earliest possible begin time of the activity execution; (2) `endTime`, which represents the latest possible finish time of the activity execution; and (3) `price`, which specifies the maximum price for the activity execution (see Figure 1). Additionally, the user may express preferences regarding the location of Grid resources where an activity should be executed, by specifying the Grid site, organizational, or geographical affinity. For instance, the QoS constraint `geographicAffinity` with value `myCountryID` specifies that the activity should be executed on Grid resources that are geographically located within the specified country.

The interested reader may find a more detailed description of QoWL in [6, 7].

2.1.1 Specification of Location Affinity with QoWL

Most of the existing related work focuses on performance (i.e. activity execution time) and economical (i.e. activity price) aspects of QoS. We believe that while performance is of paramount importance for time critical applications, the

wide acceptance of Grid technology strongly depends on security and legal aspects. We have experienced that many potential users from industry hesitate to use Grid technology even if the performance and economical benefits are clear because of security and legal concerns [14]. Therefore, we consider that it would be useful if the user has the possibility to restrict the location of Grid resources on which certain activities may be executed. For instance, for security or legal reasons the user may specify that an activity should be executed only on Grid resources that belong to user's organization.

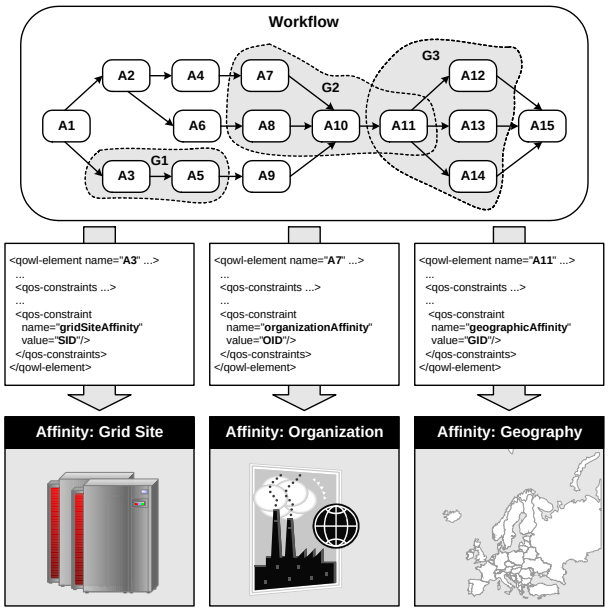


Figure 2. Specification of location affinity with QoWL

Figure 2 depicts how location affinity can be expressed with QoWL. The user may specify that a certain workflow activity should be executed on a specific *Grid site*, on the Grid resources of a specific *organization*, or on the Grid resources of a specific *geographical region*.

Commonly *Grid site affinity* is not specified by the user, but the Grid environment automatically maps workflow activities to Grid resources based on the availability and performance of resources. The goal is to minimize workflow execution time. But, in the case that the user has the information (related to security or law) which can not be automatically obtained by the Grid environment, then he can manually map the activity to a specific Grid site. Grid site preference is specified by using the `qos-constraint` named `gridSiteAffinity`. Moreover, the concept of *Grid site affinity* may be used for performance optimization. Figure 2 shows that activities `A3` and `A5` encompassed by group `G1` should be on the same Grid site. The reason could be the large data transfer between the activities `A3` and `A5` or some security reasons. The QoWL code of `A3` depicts the specification of the affinity on the language level (see Figure 2).

Organization affinity indicates the preference of the user regarding the location of activity execution on Grid resources that belong to a specific organization. These resources can be geographically distributed. The user's preferences may be based on established trust relationships with other companies. For instance, a vehicle producing company may wish to execute certain critical activities on a subset of the Grid in order to ensure that any relevant information is not visible for competitors. Please note that from information such as the duration of simulation it may be deduced about the development stage of the vehicle. Organization preference is specified by using the `qos-constraint` named `organizationAffinity`. Figure 2 depicts that activities `A7`, `A8`, `A10` and `A11` encompassed by group `G2` should be executed on resources that belong to the same organization.

Geographical affinity indicates the preference of the user regarding the location of activity execution on Grid resources that belong to a specific geographic region. Examples of geographical region include: country, state, or set of states. For instance several countries which have the same legal conditions for the electronic medical data processing may be eligible for execution of certain activities. Geographical preference is specified by using the `qos-constraint` named `geographicAffinity`. Figure 2 shows that activities `A11`, `A12`, `A13` and `A14` encompassed by group `G3` should be executed on resources that belong to the same geographic region.

The agreement on *time* and *cost* constraints requires negotiation process with the candidate services as described in [6]. The *security* and *legal* related QoS agreements are met based on the extracted information of the service's XML descriptors, without negotiation process. The specified location affinity may be integrated with the available security infrastructure, such as *Web Services Policy* [22]. The security and legal related QoS supports the concept of *Virtual Organization*.

In the following section we describe the UML-based modeling of QoWL elements.

2.2 The Definition of a UML-based DSL for QoS-aware Workflows

UML 2.0 specification [17] provides a large set of modeling elements and diagrams for modeling various types of software and hardware systems. UML has a modular nature, with the *diagram type* being the unit of modularity. From the available 13 UML diagram types, we use only *activity diagrams* for modeling Grid workflows. UML activity

diagrams are suitable for *flow modeling* of various types of software or hardware systems. Hierarchical capabilities of the UML activity diagram support modeling of systems at arbitrary levels of detail and complexity. For instance, it is possible to group a set of activities with the corresponding flow into a higher-level activity with a well defined input and output.

In order to enable the modeling of different types of systems, the UML modeling elements are specified in an abstract manner without conceptual connection with a particular domain. However, too generic semantics of UML modeling elements may present an obstacle for using UML in a specific domain. For this reason, the UML specification defines the mechanisms for specializing semantics of modeling elements for a particular domain. We have defined a *Domain Specific Language (DSL)* for QoS-aware Grid workflows by using the UML extension mechanisms. The UML may be extended by defining new modeling elements, *stereotypes*, based on existing elements, *base classes* (i.e. metaclasses). A stereotype is defined as a subclass of an existing UML metaclass, with the associated *tagged values* (i.e. metaattributes). Stereotypes are notated by the stereotype name enclosed in guillemets <<StereotypeName>>, or by a specific graphic icon.

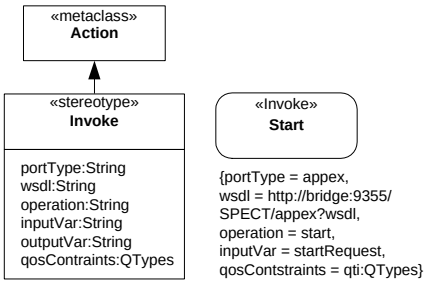
The benefits of definition of a DSL for the domain of QoS-aware Grid workflows include: (1) the user is exposed to only domain-relevant UML modeling elements, (2) the language concepts have domain-specific interpretation, and (3) models may be enriched with information that is used by tools for automatic model transformation (for instance to XML) or model processing (for instance for the purpose of QoS negotiation).

For each element of our XML-based language QoWL we have defined an element of UML-based DSL. Figure 3 depicts an instance of the procedure for defining elements of our DSL for the domain of QoS-aware Grid workflows. The XML representation of QoWL element $Invoke$ is depicted in Figure 3(a). The DSL modeling element $Invoke$ is defined by stereotyping the base class $Action$ (see Figure 3(b)).The $Invoke$ activity is used for the invocation of external services. The tagged values $portType$, $operation$, $inputVar$ and $outputVar$ may be used for specification of the information that is needed for service invocation. The graphical notation of stereotype $Invoke$ is illustrated with an example in Figure 3(c).

Figure 4 depicts the structure of type $QTypes$, whose instances are used to specify the tagged value $qosContraints$. $QTypes$ contains the $reqDescVar$ attribute, which specifies the meta data (e.g. file size, image resolution, number of iterations) that may be used for performance prediction of the service. Additionally, $QTypes$ comprises zero or more entities of type $QType$ that are used for description of specific QoS constraints (e.g. required ex-

```
<invoke name="start" portType="appex"
   wsdl="http://bridge:9355/SPECT/appex?wsdl"
   operation="start" inputVar="startRequest">
  <qos-constraints reqDescVar="startReqDesc">
    <qos-constraint name="beginTime" weight="0.3"
      value="18-08-2005 12:00:00,0 MET"/>
    <qos-constraint name="endTime" weight="0.2"
      value="18-08-2005 14:00:00,0 MET" />
    <qos-constraint name="price" weight="0.5"
      value="20.00" />
    <qos-constraint name="geographicAffinity"
      value="GID" />
  </qos-constraints>
</invoke>
```

(a) QoWL



(b) Definition (c) Usage

Figure 3. Stereotype *Invoke*

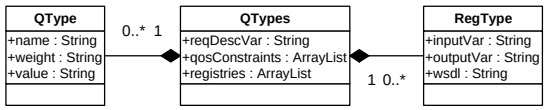


Figure 4. The structure of $QTypes$

ecution time). Moreover, $QTypes$ comprises zero or more entities of type $RegType$ that may be used for the specification of registries where the potential services can be found.

The rest of the elements of our UML-based DSL are defined in an analogous manner. Figure 5 depicts the complete list of modeling elements of our DSL for QoS-aware Grid workflows. The first column shows the names of newly defined UML modeling elements (such as $Process$). The second column shows the UML elements that serve as base classes for customization (for instance, $Activity$). Tagged values are shown in the third column. The fourth column provides the description of the DSL elements.

2.3 Graphical Representation of QoWL Elements with the UML-based DSL

Basic elements of QoWL are not further decomposed into other elements. QoWL elements of this kind are: *Invoke*, *Copy*, *Receive*, and *Reply*. Graphical representation of the element $Invoke$ is depicted in Figure 3. Other basic elements of QoWL are represented with the UML-based DSL in an analogous manner.

Complex elements of QoWL may comprise basic and complex elements. QoWL elements of this kind are: $Sequence$, $Flow$, $Switch$, $While$, and $Process$. The

Stereotype	Base Class	Tags	Description
Process «Process»	Activity	qosConstraints:QType, variables:VType	Indicates that Activity represents a workflow process
Invoke «Invoke»	Action	qosConstraints:QType, portType:String, operation:String, inputVar:String, outputVar:String, wsdl:String	Indicates that Action represents the operation invocation of an external Grid Service
Copy «Copy»	Action	from:String to:String	Indicates that Action represents the data value assignment
Sequence «Sequence»	SequenceNode	qosConstraints:QType	Indicates that SequenceNode represents a series of actions which are executed sequentially
Flow «Flow»	StructuredActivityNode	qosConstraints:QType	Indicates that SequenceNode represents a set of actions which may be executed concurrently
Receive «Receive»	AcceptEventAction	portType:String, operation:String, variable:String wsdl:String	Indicates that AcceptEventAction represents a blocking message receive
Reply «Reply»	SendSignalAction	portType:String, operation:String, variable:String wsdl:String	Indicates that SendSignalAction represents the reply message to a message that was received through a «Receive»
Switch «Switch»	DecisionNode	qosConstraints:QType	Indicates that DecisionNode represents the conditional execution
While «While»	LoopNode	condition:Boolean	Indicates that LoopNode represents a while loop. The loop body is executed until the condition is violated.

Figure 5. Elements of the UML-based DSL for QoS-aware workflows

modeling of QoWL complex elements with the UML-based DSL is described in the following.

2.3.1 The Sequence Element

The *Sequence* element specifies that a set of activities should be executed sequentially in the predefined order. Usually, the data dependency determines the execution order of activities within the sequence. In absence of data dependency, the reason for the sequential execution of a set of activities may be the avoidance of parallelization overhead.

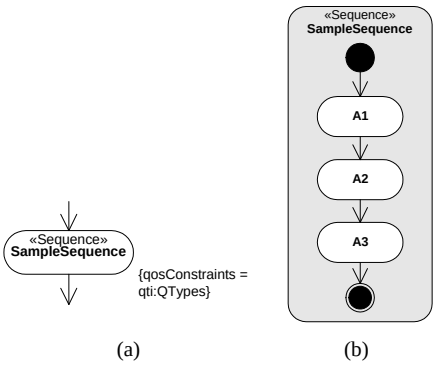


Figure 6. QoWL element *Sequence*

Figure 6(a) depicts an instance of *Sequence* element. The *qti:QTypes* attribute defines the QoS constraints. Figure 6(b) shows the comprised activities within the *Sequence* element.

QoS information: The user may specify the QoS constraints for the element *Sequence*. The execution of the comprised activities should comply with the QoS constraints of *Sequence* as follows,

- $\sum_{i=1}^{n} time(A_i)$ should not exceed the specified time for *Sequence*,
- $\sum_{i=1}^{n} price(A_i)$ should not exceed the specified price for *Sequence*,
- the *location affinity* is inherited by all the comprised activities,

where n is the number of comprised activities within the *Sequence*, and A_i is the i^{th} activity in the *Sequence*.

2.3.2 The Flow Element

The *Flow* element specifies that a set of activities should be executed concurrently.

Figure 7(a) depicts an instance of the *Flow* element. The *qti:QTypes* attribute defines the QoS constraints. Figure 7(b) shows the comprised activities, A_1 and A_2, within the *Flow* element that should be executed concurrently.

QoS information: The user may specify the QoS constraints for the element *Flow*. The execution of the comprised activities should comply with the QoS constraints of *Flow* as follows,

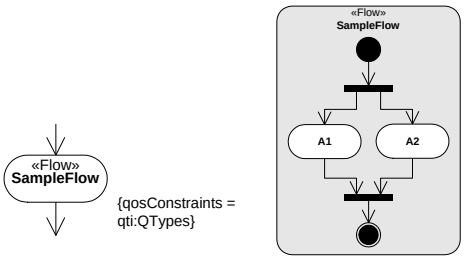


Figure 7. QoWL element *Flow*

- $Max\{time(A_i)|i = 1, .., n\}$ should not exceed the specified time for *Flow*,
- $\sum_{i=1}^{n} price(A_i)$ should not exceed the specified price for *Flow*,
- the *location affinity* is inherited by all the comprised activities,

where n is the number of comprised activities within the *Flow*, and A_i is the i^{th} activity of the *Flow*.

2.3.3 The Switch Element

The *Switch* element specifies that one of the alternate execution paths is selected based on a condition. Condition is specified as a Boolean expression.

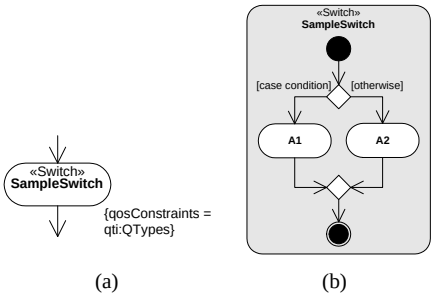


Figure 8. QoWL element *Switch*

Figure 8(a) depicts an instance of *Switch* element. The *qti:QTypes* attribute defines the QoS constraints. Figure 8(b) shows the comprised activities, A_1 and A_2, within the *Flow* element. If the Boolean expression *case condition* evaluates to *true* then activity A_1 is executed, otherwise A_2.

QoS information: The user may specify the QoS constraints for the element *Switch*. The specified QoS constraints have to be satisfied for each possible execution path. The execution of the comprised activities should comply with the QoS constraints of *Switch* as follows,

- $Max\{time(B_i)|i = 1, .., k\}$ should not exceed the specified time for *Switch*,
- $Max\{price(B_i)|i = 1, .., k\}$ should not exceed the specified price for *Switch*,
- the *location affinity* is inherited by all the comprised activities,

where k is the number of comprised execution paths within the *Switch* element, and B_i is the i^{th} branch of the *Switch*.

2.3.4 The While Element

The *While* element specifies the iterative execution of the comprised activities as long as the specified boolean expression evaluates to true.

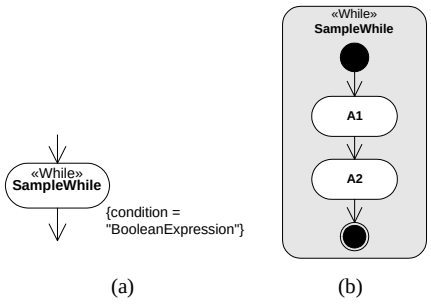


Figure 9. QoWL element *While*

Figure 9(a) depicts an instance of *while* element. The *condition* attribute specifies the Boolean expression which is evaluated before each iteration. Figure 9(b) shows the comprised activities, A_1 and A_2, within the *While* element.

QoS information: Because it is difficult to determine the number of iterations in advance, then the QoS constraints for the *While* element are not specified. However, it is possible to specify the QoS constraints for particular elements within the *while* loop.

2.3.5 The Process Element

The *Process* element specifies the overall workflow. It comprises all other elements of the workflow.

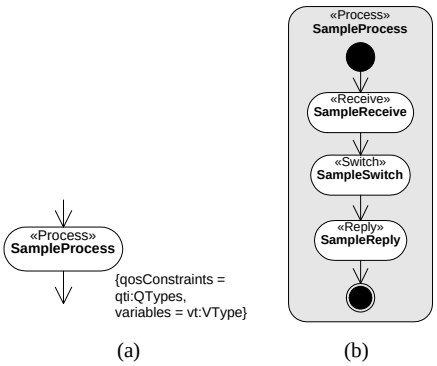


Figure 10. QoWL element *Process*

Figure 10(a) depicts an instance of *Process* element. The *qti:QTypes* attribute defines the QoS constraints. The *vt:VType* defines variables used for the data manipulation. Figure 10(b) shows the comprised activities within the *Process* element.

QoS information: The user may specify the QoS constraints for the *Process* element. The execution of the comprised activities should comply with the QoS constraints of *Process*.

3 QoS-aware Workflow System

In this section we describe our system that provides support for the whole workflow lifecycle from specification to execution.

3.1 Architectural Overview

Figure 11 shows the architecture of our system for QoS-aware Grid workflows. The main components include: (1) Teuta, which is a UML based graphical editor for workflow specification; (2) QWE, which is a QoS-aware workflow engine; and (3) VGE services, which are QoS-aware Grid services.

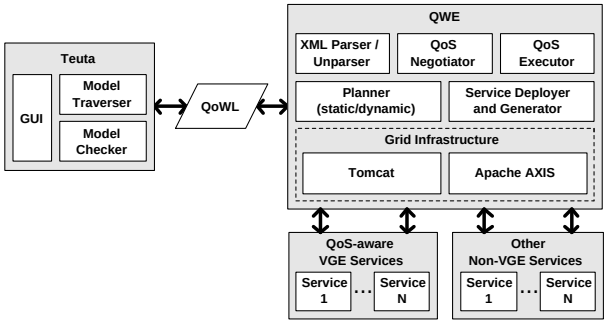


Figure 11. Architecture of the system for QoS-aware Grid workflows

A user may specify the workflow with Teuta by composing the predefined elements of UML-based DSL for QoS-aware workflows (see Section 2.3). Furthermore, for each workflow element different parameters (such as execution time, price, location affinity) may be specified that determine the user's QoS requirements. Thereafter, Teuta verifies whether the specified workflow is well defined. In the case that the workflow model is well defined, Teuta generates the corresponding QoWL representation. The QWE engine interprets the QoWL workflow, negotiates with specified services, applies the selected workflow planning strategy, selects appropriate services and finally executes the specified workflow. If the specified tasks need QoS guarantees we use VGE services, which are able to give certain QoS guarantees. In other cases (for instance in case that execution time of the service is negligible) the use of other non-VGE services may be considered. In what follows the main architectural components are explained in more detail.

3.2 Teuta

Teuta is a UML-based graphical editor. It is designed as a platform independent, configurable and extensible tool. Therefore, it is possible to extend Teuta with new types of diagrams and modeling elements for various domains. Examples of usage of Teuta include performance modeling of high performance programs [20] and specification of scientific workflows within the framework of Askalon project [2]. In order to provide tool-support for our approach described in this paper we have extended Teuta for QoS-aware workflows and integrated with QWE.

Teuta architecture is shown on the left-hand side of Figure 11. Teuta comprises three main components: Graphical User Interface (GUI), Model Checker, and Model Traverser. We illustrate the GUI of Teuta with examples of real-world workflows in Section 4.

The *Model Checker* verifies whether the model is well defined. The rules for model checking are specified by using our XML-based Model Checking Language (MCL). The model checker gets the model description from an MCL file. This MCL file contains a list of available diagrams, modeling elements and the set of rules that defines how the elements may be interconnected.

The *Model Traverser* provides the possibility to walk through the model, to visit each modeling element, and to access its properties (for instance QoS constraints). We use the model traversing for the generation of various model representations; for instance, a QoWL representation serves as input for QWE engine (see Figure 11).

3.3 QWE

The QoWL documents generated by the Teuta can be executed using the QoS-aware Grid Workflow Execution Engine (QWE). In this section we briefly describe the main components of QWE. A more comprehensive description of QWE can be found in [7].

QWE is depicted on the right-hand side of Figure 11. The main parts of the QWE engine are: (1) *XML parser and unparser* generates the intermediary representation of the QoWL workflow; (2) the *QoS Negotiator* queries the registries, generates necessary QoS requests and receives offers from services; (3) the *Planner* component calculates a workflow execution plan considering the selected workflow

planning strategy (*static, dynamic*) and the selected workflow planning technique (*Integer Programming, Genetic Algorithm, MCDM, etc.*); (4) the *Service Deployer and Generator* exposes a QoWL workflow as a Web service and the *QoS Executer* starts the execution of the QoWL workflow.

The proper execution of QoS-aware workflows demands the availability of QoS enabled services for the execution of tasks that need QoS guarantees. Such services are explained in the following section.

3.4 QoS-aware Grid Services

The Vienna Grid Environment (VGE) [4] is a service-oriented infrastructure for the provision of 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 support. VGE services are exposed using WSDL and securely accessed via SOAP/WS-Security.

In order to provide support for dynamic QoS negotiation, VGE services rely on a generic QoS module which comprises an *application-specific performance model*, a *pricing model*, a *compute resource manager* and several XML descriptors. Within the context of European Commission funded GEMSS project VGE has been successfully used for the development of a testbed for six medical simulation and image reconstruction Grid services [16].

4 Case Study

In this section we demonstrate the modeling of QoS-aware workflows using a real world application for maxillo facial surgery simulation.

4.1 Maxillo Facial Surgery Simulation

Maxillo facial surgery simulation (MFSS) is one of the six medical applications that we have used within the framework of GEMSS project [16]. The application facilitates the work of medical practitioners and provides the pre-operative virtual planning of maxillo-facial surgery. The application consists of a set of components which can run on a local machine or on different remote machines. These components may be organized as a Grid workflow in order to simplify the work of the end users. Cao et al. [8] describes the specification of MFSS workflow using the Triana tool [9], but the QoS requirements are not considered. In the following we describe the QoS-aware specification of the MFSS workflow.

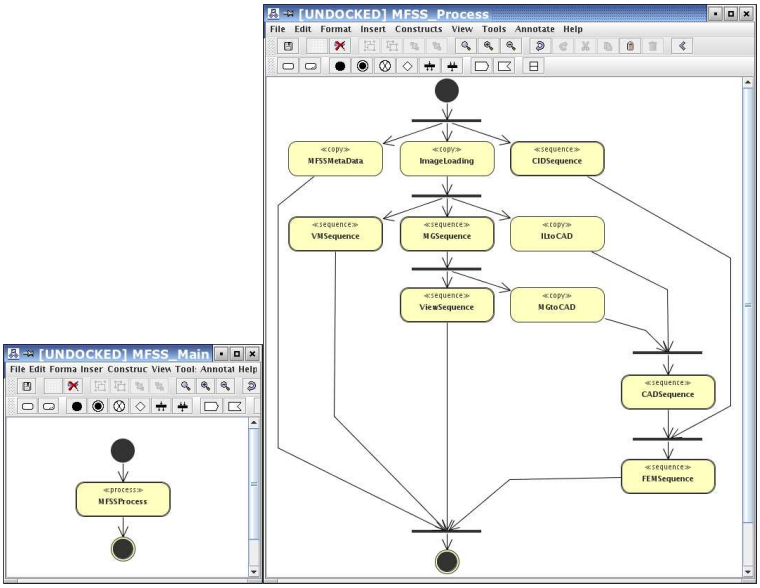
4.2 Workflow Specification

We have used our UML based workflow editor Teuta (see Section 3.2) for the specification of the MFSS workflow. The workflow specification process involved the definition of the flow of workflow activities and the association of the corresponding properties (such as QoS related properties).

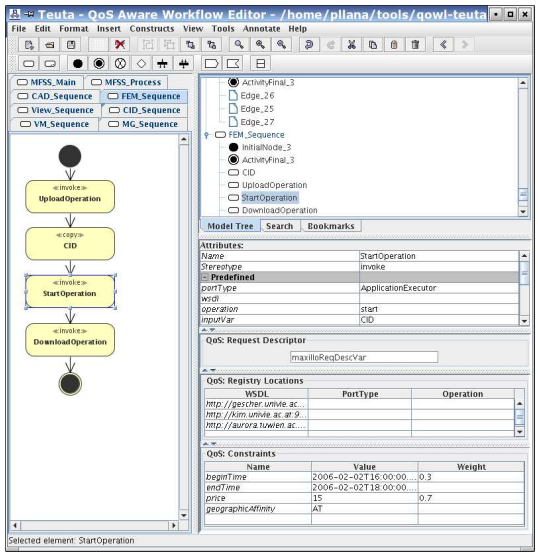
Figure 12 illustrates the process of specification of MFSS workflow with Teuta. The user may combine the predefined UML modeling elements, which are available in the Teuta toolbar, to specify the flow of workflow activities. The left hand side of the Figure 12(a) depicts *MFSSProcess* activity, which is an instance of the *process* element. A *process* element is indicated by stereotype `<<process>>`. The *MFSSProcess* activity indicates the root of the MFSS workflow. Complex activities, such as *process*, may comprise a group of activities. Teuta supports hierarchical modeling, by representing the body of a complex activity as a subgraph. The body of *MFSSProcess* activity is depicted on the right hand side of Figure 12(a). The UML element *InitialNode*, which is represented as a filled black circle, defines the starting point of a workflow or of a complex activity. We have used the *copy* element, which is indicated by stereotype `<<copy>>`, to express the data flow. For instance, *MFSSMetaData* activity copies the input data of the workflow to the corresponding activities. UML elements *Fork* and *Join*, which are represented as bold horizontal bars, express the split and join of multiple flow branches respectively. For instance, after the completion of *ImageLoading* activity, the flow is split into three branches. UML elements *Fork* and *Join* are mapped to the BPEL element *Flow*. The UML element *ActivityFinal*, which is represented as a circle surrounding a smaller solid filled circle, indicates the end of the workflow or the end of a complex activity.

The *MFSSProcess* activity comprises several complex activities of type *sequence*, which are indicated by the stereotype `<<sequence>>`. For instance, the body of *FEMSequence* activity, placed on the right-down corner of Figure 12(a)), is represented in Figure 12(b). An *invoke* element, which is indicated by stereotype `<<invoke>>`, specifies the invocation of a remote or local service operation. For instance, the activity *UploadOperation* invokes the *upload* operation of the remote service.

With each workflow element we have associated a set of properties by using the property panel, which is located on the right hand side of Teuta GUI (see Figure 12(b)). For instance, Figure 12(b) shows the properties of the invoke element *StartOperation*. The top compartment of the property panel allows the association of attributes such as *name*, *portType* and *operation*. We use the lower three compartments to specify QoS constraints such as *beginTime*, *endTime*, *price*, and *geographicAffinity*.



(a) UML representation of MFSS workflow



(b) Properties of *StartOperation* activity

```
...
<invoke inputVar="CID" name="StartOperation" operation="start" portType="ApplicationExecutor">
  <qos-constraints ReqDesc="maxilloReqDescVar">
    <registry wsdl="http://gescher.univie.ac.at:9357/registry/reg?wsdl" />
    <registry wsdl="http://kim.univie.ac.at:9357/registry/reg?wsdl" />
    <registry wsdl="http://aurora.tuwien.ac.at:9357/registry/reg?wsdl" />
    <qos-constraint name="beginTime" value="2006-02-02T16:00:00.000+02:00" weight="0.3" />
    <qos-constraint name="endTime" value="2006-02-02T18:00:00.000+02:00" />
    <qos-constraint name="price" value="15" weight="0.7" />
    <qos-constraint name="geographicAffinity" value="AT" />
  </qos-constraints>
</invoke>
...
```

(c) QoWL representation of *StartOperation* activity

Figure 12. Specification of MFSS workflow with Teuta editor

After the completion of MFSS workflow specification Teuta is able to generate automatically the corresponding QoWL representation, which is used as input for QWE for workflow execution (see Section 3.1). An excerpt of QoWL representation of MFSS workflow, which specifies the *StartOperation* activity, is depicted in Figure 12(c).

5 Related Work

Several projects are contributing to the establishment and improvement of the Grid workflow technology, each focusing on a specific research aspect. Triana [9], Askalon [2], JOpera [18], eXeGrid [15] are developing tools and languages for graphical workflow composition. The P-GRADE Portal is exploring the collaborative Grid workflows [21]. Pegasus [3] and LEAD [19] projects are focused on the development of workflow support for large scale Grid applications (such as galaxy morphology, tomography and mesoscale meteorology). The aspects of semantic grid workflows are investigated within the Taverna [23] and Kepler [5] projects.

There is not much related work focused on the development of a Grid services infrastructure that provides QoS guarantees. Moreover, not enough attention is paid to QoS-aware Grid workflows. Gridbus Project [12] is addressing the QoS-aware Grid workflows. Recent developments are following research problems on cost-based scheduling of scientific workflow applications [24]. However, within the framework of 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. In contrast to existing related work, we are developing a QoS-aware workflow system that supports time, cost, security, and legal constraints. Moreover, our system supports the graphical specification of QoS-aware workflows based on the latest UML standard.

6 Conclusions and Future Work

We consider that for the wide acceptance of Grid technology it is important that specification of tasks to be executed on the Grid is simple, and that the execution of these

tasks should meet the user's requirements. In this paper we have addressed the issue of high-level specification of QoS-aware Grid workflows. In order to streamline the process of workflow specification we have developed a Domain Specification Language (DSL) for QoS-aware Grid workflows based on UML2.0 standard. Furthermore, we have developed a system prototype that supports the whole workflow lifecycle from high-level specification to execution. Our system allows the specification of a comprehensive set of QoS requirements, that consider performance, economical, legal and security aspects. We evaluated our approach by modeling a real-world workflow for maxillo facial surgery simulation, and showed the hierarchical modeling capabilities of our approach for modeling complex activities. In the future we plan to extend our approach with workflow adaptivity and optimization mechanisms.

References

[1] 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"*, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/ 2003

[2] Askalon Project. http://dps.uibk.ac.at/projects/askalon/

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

[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] C. Berkley, S. Bowers, M. Jones, B. Ludäscher, M. Schildhauer, J. Tao. *Incorporating Semantics in Scientific Workflow Authoring.* 17th International Conference on Scientific and Statistical Database Management, University of California, Santa Barbara, CA, USA, 2005

[6] I. Brandic, S. Benkner, G. Engelbrecht, R. Schmidt. *Towards Quality of Service Support for Grid Workflows.* Proceedings of the European Grid Conference 2005 (EGC2005), Amsterdam, The Netherlands, February 2005.

[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] J. Cao, G. Berti, J. Fingberg, J. G. Schmidt. *Implementation of Grid-enabled medical simulation applications using workflow techniques.* The Second International Workshop on Grid and Cooperative Computing, Shanghai, China, 2003.

[9] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor and I. Wang. *Programming Scientific and Distributed Workflow with Triana Services.* In Grid Workfbw 2004 Special Issue of Concurrency and Computation: Practice and Experience, 2005.

[10] T. Fahringer, S. Pllana, and A. Villazon. *AGWL: Abstract Grid Workflow Language*. International Conference on Computational Science, Programming Paradigms for Grids and Metacomputing Systems. Krakow, Poland, June 2004

[11] I. Foster, C. Kesselman, and S. Tuecke. *The Anatomy of the Grid - Enabling Scalable Virtual Organizations*. The International Journal of High Performance Computing Applications, 15(3):200-222, 2001.

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

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

[14] GEMSS Consortium. *Report on COTS Security Technologies and Authorisation Services*. Deliverable D2.2c. GEMSS Project, European Commission Framework V Project No. IST-2001-37153. February 2004

[15] A. Hoheisel. *User Tools and Languages for Graph-based Grid Workflows*. In: Special Issue of Concurrency and Computation: Practice and Experience, Wiley, 2004.

[16] 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.

[17] Object Management Group (OMG). UML 2.0 Superstructure Specifi cation. http://www.omg.org, August 2005.

[18] C. Pautasso. *JOpera: Visual Composition of Grid Services* In: ERCIM News No. 59, October 2004

[19] 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

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

[21] G. Sipos, G. J. Lewis, P. Kacsuk, V. N. Alexandrov. *Workflow-Oriented Collaborative Grid Portals*. Proceedings of the European Grid Conference 2005 (EGC2005), Amsterdam, The Netherlands, February 2005.

[22] Web Services Policy (WS-Policy). http://ifr.sap.com/ws-policy/index.html

[23] K. Wolstencroft, T. Oinn, C. Goble, J. Ferris, Ch. Wroe, P. Lord, K. Glover, R. Stevens. *Panoply of Utilities in Taverna*. Proceedings 1st IEEE International Conference on eScience and Grid Computing, Melbourne, Australia, December, 2005.

[24] J. Yu, R. Buyya, and Ch. K. Tham. *QoS-based Scheduling of Workflow Applications on Service Grids*. Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing, 2005, Melbourne, Australia.

[25] 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/GridWorkfbwTaxonomy.pdf