UML BASED GRID WORKFLOW MODELING UNDER ASKALON*

Jun Qin[1], Thomas Fahringer[1], and Sabri Pllana[2]

[1]*Institute of Computer Science, University of Innsbruck*
Technikerstr. 21a, 6020 Innsbruck, Austria
{Jun.Qin, Thomas.Fahringer}@uibk.ac.at

[2]*Institute of Scientific Computing, University of Vienna*
Nordbergstr. 15/C308, 1090 Vienna, Austria
pllana@par.univie.ac.at

Abstract Most existing Grid workflow modeling tools are based on user-defined notations. Lack of standards hinders the collaboration among different Grid-related projects. The work presented in this paper introduces a graphical workflow editor Teuta, which has been implemented based on the latest standard UML 2.0 notations and tailored for specifying Grid workflows based on our Abstract Grid Workflow Language (AGWL). In Teuta, Grid workflows are composed by combining predefined UML modeling elements or user-defined constructs in a hierarchical fashion. Teuta can generates the corresponding AGWL representations and submit them to the ASKALON Grid runtime system for execution. We validate our approach for a real world hydrological application.

Keywords: Grid, workflow modeling, UML, ASKALON, AGWL

1. Introduction

In the recent years, significant research efforts have been involved in the development of tools support for Grid workflow modeling. Compared with textual-based modeling, graph-based modeling allows users to graphically define a Grid workflow through dragging and dropping the modeling elements of interest. However, most of the graph-based Grid workflow modeling tools are based on user-defined notations, which hin-

*This research has been partially supported by the Austrian Science Fund as part of the Aurora project under the contract SFBF1104 and the Austrian Federal Ministry for Education, Science and Culture as part of the Austrian Grid project under the contract GZ 4003/2-VI/4c/2004.

ders the collaboration among different Grid-related projects. Much remains to be done to streamline the task of Grid workflow modeling.

In this paper we present our graphical modeling tool Teuta. Compared with our previous work [Pllana et al., 2004], we have customized Teuta for the specification of scientific Grid workflows based on our Abstract Grid Workflow Language (AGWL) [Fahringer et al., 2005b], and integrated it with the ASKALON Grid environment [Fahringer et al., 2005a]. In Teuta, Grid workflows are composed based on the Unified Modeling Language (UML) 2.0 standard. Furthermore, in order to alleviate the complexity of composing large and complex Grid workflows, Teuta supports hierarchical workflow composition. This enables a simple view of the workflow being maintained at each level of abstraction. Teuta has been used as the main user interface to ASKALON, and applied to numerous real world applications.

The remainder of this paper is organized as follows. The related work is described and compared against our approach in the next section. Section 3 provides some background knowledge. Section 4 briefly describes our approach of UML based Grid workflow modeling. Our tool Teuta for modeling Grid workflow applications is introduced in Section 5. Section 6 applies Teuta for a real world application to evaluate our approach. In the last section, we draw our conclusions and outline the future work.

2. Related Work

GridFlow [Cao et al., 2003] uses Petri Nets to model Grid workflow. Fraunhofer Resource Grid (FhRG) [Hoheisel, 2004], which uses a hierarchical graph definition to model Grid workflows, is also built on Petri Nets. However, Petri Nets may be unable to model workflow activities accurately without extending its semantics [Eshuis and Wieringa, 2003]. And this drawback has been addressed in UML activity diagrams. The work presented in [Bastos et al., 2002] uses UML activity diagrams to model Grid workflows. However, the UML they used is UML1.x, in which the activity diagrams had several serious limitations in the types of flows that could be represented. Many of these limitations were due to the fact that activities were overlaid on top of the basic state machine formalism, and consequently constrained to the semantics of state machines [Bran Selic, 2005]. Rather than following the standard syntax and semantics of Petri Nets and UML, many Grid workflow editor tools create their own graphical representation of workflow components [Yu and Buyya, 2005], e.g. Triana [Taylor et al., 2005] and Kepler [Altintas et al., 2004]. However, lack of standards hinders the collaboration among different Grid-related projects. Much work is thus replicated such as different user interfaces developed by different projects for the same functionality.

In a word, most of existing work suffers from one or several of the following drawbacks: the use of self-defined notations, the use of old UML 1.x activity diagrams, or no adequate tool support. In contrast, we use the latest standard UML 2.0 activity diagram in Teuta to model Grid workflow applications. Teuta can model graphically any Grid workflow application that can be expressed textually using AGWL [Fahringer et al., 2005b]. Moreover, Teuta has been integrated with the ASKALON Grid environment as a user interface for Grid workflow composition, submission, controlling and monitoring.

3. Background

3.1 Abstract Grid Workflow Language (AGWL)

AGWL [Fahringer et al., 2005b] is an XML-based language for describing Grid workflow applications at a high level of abstraction. AGWL allows a programmer to define a graph of activities that refer to computational tasks or user interactions. Activities are connected by control and data flow links.

In AGWL, activities are described by activity types. An activity type is an abstract description of a group of activity instances deployed in the Grid which have the same input and output data structures. Activity types shield the implementation details of activity instances from the AGWL programmer. A rich set of control flow constructs is provided in AGWL to simplify the specification of Grid workflow applications, for example, `Sequence`, `If`, `Switch`, `While`, `For`, `ForEach`, `DAG`, `Parallel`, `ParallelFor` and `ParallelForEach`. AGWL also supports sub-workflows. Properties and constraints can be defined in AGWL to provide additional information for a workflow runtime environment to optimize and steer the execution of workflow applications.

3.2 UML Activity Diagrams

The UML Activity Diagram is one of the 13 UML diagrams of the UML 2.0 specification and it is used for *flow modeling* of various types of systems independently from their implementation (software or hardware). Hierarchical modeling capabilities of the UML Activity Diagram support modeling at arbitrary levels of detail and complexity. An *activity* is a flow graph, which consists of a set of *nodes* interconnected by directed *edges*. There are three types of nodes: *action nodes*, *control nodes*, and *object nodes*. *Action nodes* are basic units of the behavior

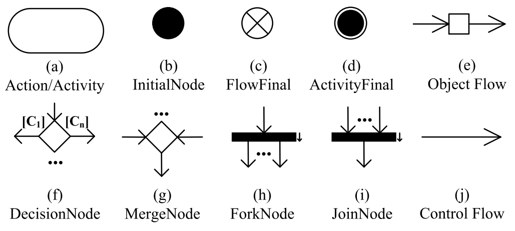

Figure 1. A subset of modeling elements of UML Activity Diagram

specification (see Figure 1(a)). Actions may contain *pins*, which represent input and output. *Control nodes* steer the control and data along the flow graph (see Figure 1(b,c,d,f,g,h,i)). *Object nodes* contain the data that flows through the graph. An edge of a UML Activity Diagram indicates either a control flow or an object flow. A *control flow edge* specifies the precedence relationship between two interconnected nodes (see Figure 1(j)). An *object flow edge* specifies the flow of objects along interconnected *action nodes* (see Figure 1(e)).

4. Modeling Grid Workflows with UML Activity Diagram

A Grid workflow Ψ is a pair (A, D), where A is a finite set of activities and D is a finite set of activity dependencies. Every activity dependency d_i, $d_i \in D$, is associated with an ordered pair of activities (a_m, a_n), where $a_m \in A \wedge a_n \in A$. An activity diagram Ω is a pair (N, E), where N is a finite set of nodes and E is a finite set of directed edges. Every directed edge is an ordered pair of nodes (n_k, n_j), where $n_k \in N \wedge n_j \in N$. The relationship between a Grid workflow $\Psi = (A, D)$ and an activity diagram $\Omega = (N, E)$ is defined by relations $R' = \{(a_i, n_i) \mid \text{for all } i,\ a_i \in A \wedge n_i \in N\}$ and $R'' = \{(d_j, e_j) \mid \text{for all } j,\ d_j \in D \wedge e_j \in E\}$. This means that each activity a_i of a Grid workflow is associated with a node n_i of a UML Activity Diagram, and each dependency d_j of a Grid workflow is associated with an edge e_j of a UML Activity Diagram.

In order to be able to model different types of systems, the UML specification provides several extension mechanisms to specialize semantics of modeling elements for a particular domain. Based on these mechanisms, we have extended the UML Activity Diagram by defining some new *stereotypes* with associated *tagged values* based on existing elements to model AGWL constructs (see Table 1). Figure 2 depicts an instance of the procedure, where we defined a model element *GridAction* by stereotyping the base class *Action* to model Grid workflow activities. The tagged value *type* specifies the *activity type* (e.g., Fast Fourier Transform

Table 1. Extending the UML Activity Diagram to model AGWL constructs

Base class	**Stereotype & Tags**	**Description**
Action	≪GridAction≫ type: string	Indicates that the Action represents a fundmental computation unit in the Grid
SequenceNode	≪Sequence≫	Indicates that the SequenceNode represents a group of Grid actions/activities that are executed sequentially
ConditionalNode	≪If≫ condition: boolean	Indicates that the ConditionalNode represents a conditional execution of Grid actions/activities in the *if-then-else* fashion
ConditionalNode	≪Switch≫ caseValue: integer	Indicates that the ConditionalNode represents a conditional execution of Grid actions/activities in the *switch* fashion
LoopNode	≪While≫ loopCondition: boolean	Indicates that the LoopNode represents a *while* loop. The loop body is executed zero or more times.
LoopNode	≪For≫ from, to, step: integer	Indicates that the LoopNode represents a *for* loop.
ExpansionRegion	≪ForEach≫	Indicates that the ExpansionRegion represents a loop that iterates over elements of a data collection sequentially
StructuredActivityNode	≪DAG≫	Indicates that the StructuredActivityNode represents a group of Grid actions/activities that are executed based on the order specified in the directed acyclic graph
StructuredActivityNode	≪Parallel≫	Indicates that the StructuredActivityNode represents a group of Grid actions/activities that are executed in parallel
LoopNode	≪ParallelFor≫ from, to, step: integer	Indicates that the LoopNode represents a *for* loop whose iterations are executed in parallel
ExpansionRegion	≪ParallelForEach≫	Indicates that the ExpansionRegion represents a loop that iterates over elements of a data collection in parallel
StructuredActivityNode	≪SubWorkflow≫	Indicates that the StructuredActivityNode represents a sub workflow that is invoked at a point of the main workflow

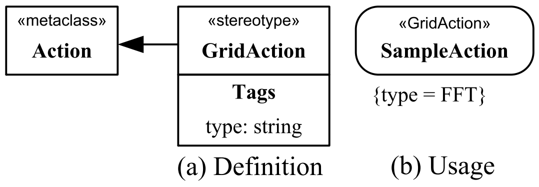

Figure 2. The definition and usage of the stereotype *GridAction*

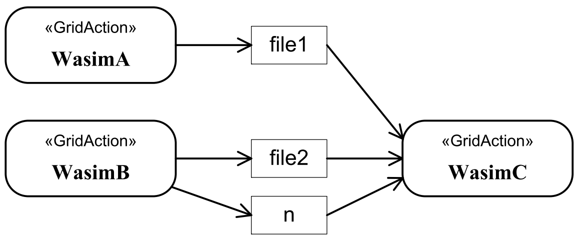

Figure 3. Modeling data flows

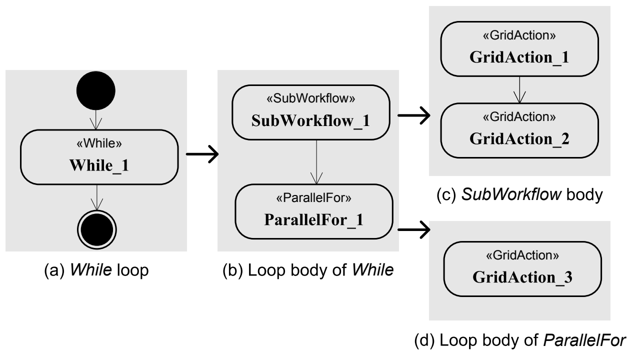

Figure 4. Modeling Grid workflow hierarchies

(FFT)), which is an abstract description of a group of activity instances (concrete implementations of computational entities) implementing the same functionality and having the same input and output data structure.

We use the *object flow* in UML Activity Diagrams to model the data flow in Grid workflows and *pins* to model input and output data ports, namely, dataIn and dataOut. Connecting one dataOut port of an activity to one dataIn port of another activity constitutes a data flow. Figure 3 illustrates three data flows. The output file *file1* of the GridAction *WasimA* and the output file *file2* and the number *n* of GridAction *WasimB* serves as the input of the GridAction *WasimC*.

Graphical representations of Grid workflows are very intuitive and can be handled easily even by a non-expert user. However, the layout of workflow components on a display screen can become very large and beyond the users control. Similar to [Hoheisel, 2004], our solution is to use hierarchical graph definition. A Grid workflow can be composed through several levels of abstraction, each of which is represented in a separate graph. All AGWL control flow constructs like `While`, `ParallelFor`, `SubWorkflow`, etc. can have lower level workflow graphs. Figure 4 shows three levels of abstraction of a Grid workflow which are represented in four graphs. The workflow contains a while loop *While_1* in the highest level (Figure 4(a)). The *While_1* loop contains two control flow constructs in its loop body: *SubWorkflow_1* and *ParallelFor_1* (Figure 4(b)). The SubWorkflow *SubWorkflow_1* is represented in detail in Figure 4(c), which contains two GridActions: *GridAction_1* and *GridAction_2*. The parallel loop *ParallelFor_1* contains a GridAction *GridAction_3* in its loop body (Figure 4(d)). With the hierarchical graphical definition, users can easily view and evaluate the structure of the entire workflow or change the local part (e.g. a loop body) without being aware of the details and complexity of other parts of the Grid workflow.

By AGWL constructs `subWorkflow`, the hierarchical graph definition directly supports the sub-workflow definition and invocation. The main workflow (caller) provides input data to sub-workflow and gets output data from it. The input data is processed in sub-workflow (callee). The sub-workflow can be saved and reused.

5. Teuta

Teuta is implemented as a platform independent tool in Java based on Model-View-Controller (MVC) paradigm. Teuta comprises three main components: Graphical User Interface (GUI), Model Traverser, and Model Checker. The *Model Traverser* provides the possibility to walk through the model, visit each modeling element, and access its properties. We use the model traverser for the generation of various model representations, e.g. an AGWL representation of a Grid workflow, which serves as the input for the ASKALON Grid environment. The *Model Checker* is responsible for the correctness of the model. Teuta serves for ASKALON as a user interface for workflow composition, submission, controlling and monitoring.

Figure 5 illustrates a Grid workflow model in Teuta which consists of several diagrams. One of the diagrams is *main* diagram, which can be compared to the *main* method in Java/C++ programs, the others are sub-diagrams, e.g. the loop body of the parallel loop *parallelFor1*. These diagrams constitute the hierarchy of the Grid workflow. As shown in Figure 5, the activity types, dataIn ports, dataOut ports and AGWL

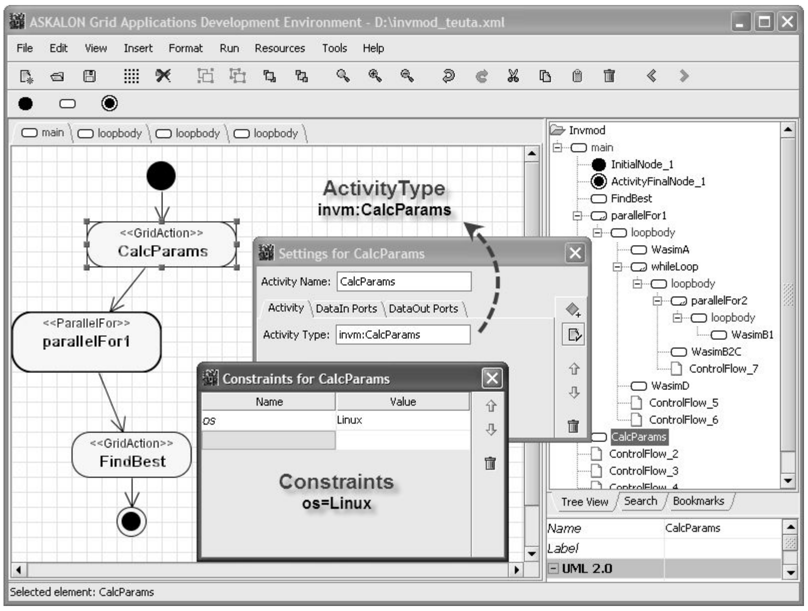

Figure 5. A Grid workflow model and the activity setting dialog in Teuta

properties and constraints can be added through the setting dialog for each modeling element. By specifying the source attributes of the data ports, users can create data flows. The corresponding AGWL representation of the Grid workflow can be generated automatically via the *Model Traverser* component.

6. Modeling a Real World Hydrological Workflow with Teuta

Invmod [Peter Rutschmann Dieter Theiner, 2005] is a hydrological application for river modeling. It has three levels of nested loops with variable number of inner loop iterations that depends on the actual convergence of the optimization process. Figure 6 illustrates the graphical representation of the Invmod Grid workflow application in Teuta. Since we adopt the hierarchical graph definition mechanism, the Invmod workflow looks very simple and can be easily understood: only one parallel loop *parallelFor1* and two atomic activities *CalcParams* and *FindBest* are shown in the *main* diagram of the workflow, because all the other activities are contained in the loop body of the *while* loop.

The code generation, implemented based on the Model Traverser, is done in the following steps: (1) put the activities in the *main* diagram into the object *AGWLWorkflow* as the workflow body; (2) put the acti-

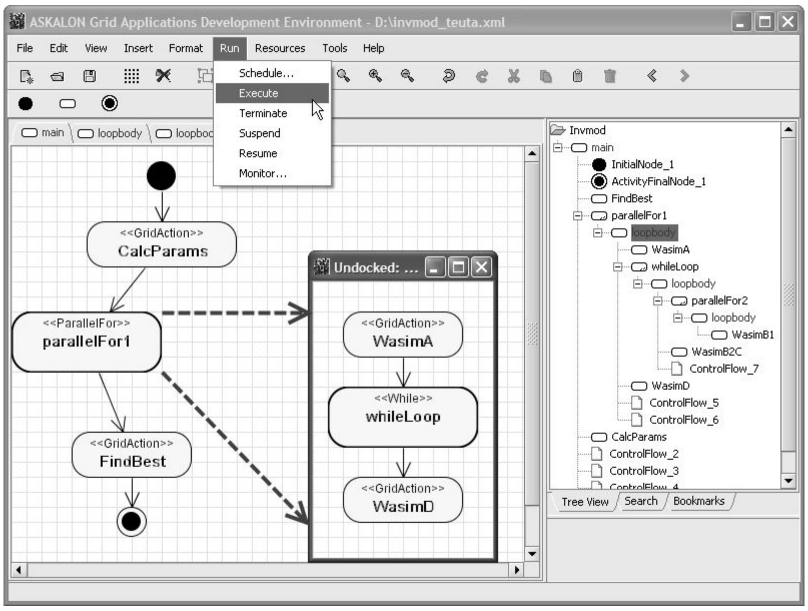

Figure 6. UML based graphical representation of the Invmod Grid workflow

vities in the other diagrams into the associated parent control flow constructs like the parallel loop *parallelFor1*; (3) invoke the *toXml()* method of the object *AGWLWorkflow* to generate the corresponding AGWL representation in XML. The workflow then is executed by the ASKALON *enactment engine* service, which takes the AGWL representation of the workflow and executes it based on the execution schedule made by the ASKALON *meta-scheduler* service. While the workflow is being executed, the *enactment engine* returns the execution status (represented by different background colors of activities) to Teuta for monitoring.

7. Conclusions and Future Work

There is a need for streamlining the process of Grid workflow modeling. We have tailored our graphical editor Teuta for the composition of Grid workflows based on the widely adopted standard UML 2.0. We have demonstrated our approach for a real world hydrological application, and showed that thanks to the hierarchical workflow composition a simple view of the workflow is maintained at each level of abstraction.

To further simplifying the specification of Grid workflows, our future work will focus on improving data flows modeling, e.g. to automatically fill the source attributes of data ports based on the model checking. We will also evaluate Teuta for large and complex Grid workflow applications in the future work.

References

[Altintas et al., 2004] Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B., and Mock, S. (2004). Kepler: An Extensible System for Design and Execution of Scientific Workflows. In *16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'04)*, Santorini Island, Greece. IEEE Computer Society Press.

[Bastos et al., 2002] Bastos, R., Dubugras, D., and Ruiz, A. (2002). Extending UML Activity Diagram for Workflow Modeling in Production Systems. In *Proceedings of 35th Annual Hawaii International Conference on System Sciences (HICSS02)*, Big Island, Hawaii. IEEE Computer Society Press.

[Bran Selic, 2005] Bran Selic (2005). What's New in UML 2.0. `ftp://ftp.software.ibm.com/software/rational/web/whitepapers/intro2uml2.pdf`.

[Cao et al., 2003] Cao, J., Jarvis, S., Saini, S., and Nudd, G. (2003). GridFlow: Workflow Management for Grid Computing. In *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003)*, Tokyo, Japan. IEEE Computer Society Press.

[Eshuis and Wieringa, 2003] Eshuis, R. and Wieringa, R. (2003). Comparing Petri Net and Activity Diagram Variants for Workflow Modelling A Quest for Reactive Petri Nets. In *Advances in Petri Nets: Petri Net Technology for Communication Based Systems; Lecture Notes in Computer Science (LNCS)*, volume 2472, pages 321–351, Heidelberg, Germany.

[Fahringer et al., 2005a] Fahringer, T., Prodan, R., Duan, R., Nerieri, F., Podlipnig, S., Qin, J., Siddiqui, M., Truong, H.-L., Villazon, A., and Wieczorek, M. (2005a). ASKALON: A Grid Application Development and Computing Environment. In *6th International Workshop on Grid Computing (Grid 2005)*, Seattle, USA. IEEE Computer Society Press.

[Fahringer et al., 2005b] Fahringer, T., Qin, J., and Hainzer, S. (2005b). Specification of Grid Workflow Applications with AGWL: An Abstract Grid Workflow Language. In *Proceedings of IEEE International Symposium on Cluster Computing and the Grid 2005 (CCGrid 2005)*, Cardiff, UK. IEEE Computer Society Press.

[Hoheisel, 2004] Hoheisel, A. (2004). User Tools and Languages for Graph-based Grid Workflows. In *Grid Workflow Workshop, GGF10*, Berlin, Germany.

[Peter Rutschmann Dieter Theiner, 2005] Peter Rutschmann Dieter Theiner (2005). An Inverse Modelling Approach for the Estimation of Hydrological Model Parameters. *Journal of Hydroinformatics*.

[Pllana et al., 2004] Pllana, S., Fahringer, T., Testori, J., Benkner, S., and Brandic, I. (2004). Towards an UML Based Graphical Representation of Grid Workflow Applications. In *The 2nd European Across Grids Conference*, Nicosia, Cyprus. ACM Press.

[Taylor et al., 2005] Taylor, I., Wang, I., Shields, M., and Majithia, S. (2005). Distributed computing with Triana on the Grid. *Concurrency and Computation: Practice and Experience*.

[Yu and Buyya, 2005] Yu, J. and Buyya, R. (2005). A Taxonomy of Workflow Management Systems for Grid Computing. Technical Report Technical Report GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia. `http://www.cis.uab.edu/gray/Pubs/grid-flow.pdf`.