JASMIN: A Visual Framework for Managing Applications in Service-oriented Grid Systems

Hayat Bendoukha, Abdelkader Benyettou
Department of Computer Science,
University USTO-MB Oran, Algeria
bendoukhyat,aek_benyettou@univ-usto.dz

Yahya Slimani
Department of Computer Science,
University of Tunis El Manar, Tunisia
yahya.slimani@fst.rnu.tn

***Abstract*— Both scientific and industrial applications are becoming more and more complex and need important computing and storage resources to be executed in an accepted time. Workflows associated to service-oriented grids allow to users the specification and the management of their most demanding and interdependent applications. In this paper, we propose a user-friendly framework JASMIN based on a refinement of UML to specify workflow models and on BPEL to generate and compose web and grid services.**

Keywords- Grid computing; Workflow; UML; BPEL; Service composition.

I. INTRODUCTION

Grids are able to aggregate a very big number of distributed and dispersed storage and computing nodes. They support huge databases and execute very demanding applications [1]. However, the most requiring applications in terms of storage and/or computing resources are, in the same time, composed of a set of sub-processes which are interdependent, share the same workspace and have to respect a particular execution scheme to achieve one same objective. Thus, new environments must be able not only to provide all needed resources but also specify, in an efficient way, the internal complexity of users' applications.

As service-oriented technology gains in popularity [2], it is normal that researchers try to design large scale solutions that incorporate web services. Current grids are mainly based on service-oriented architectures developed using grid service infrastructures enabling the invocation of services remotely across Internet [3]. The ability to define, deploy and invoke grid services remotely represents an important barrier for job submission and monitoring, staging, file transfer and data portal services [4]. Indeed, users are involved in many steps of the execution process of their respective applications. Also, in addition to fundamentals and tools of their exercising area, users are constrained to deal with formal languages and complex protocols requiring a very good master of grid technology, web and grid services composition and deployment. This can be noticed by observing the submission process in Globus Toolkit described in the programmer's tutorial of Borja Sotomayor [5].

We consider that it is increasingly necessary to reduce the complexity of the management of service-oriented grids. It is now necessary to associate user-oriented interfaces to large-scale and service-oriented systems in order to hide their complexity and make it easy to handle the services.

The goal of our work is to make grids more efficient and more transparent to individual users by making easy interaction between them and the grid execution environment. In this paper, we propose an approach which links efficiency of service-oriented grids and conviviality of user-friendly composition tools like workflows [6]. We define a workflow and service-based framework JASMIN responsible for submitting and visualizing user applications to a grid system. JASMIN is UML-based for the workflow specifications and BPEL-based for the service composition.

The remainder of the paper is organized as follows. Section II presents some related works and highlights the main contributions of our approach. Section III presents our framework, describes in details its architecture and the functionalities of its components. Section IV concludes the paper and outlines our future work.

II. RELATED WORKS

Workflow has emerged as a useful paradigm to describe, manage and share complex scientific analysis and business processes [7]. Workflows represent, declaratively, the components or codes that need to be executed in a complex application, as well as the data dependencies among those components [6]. Workflow systems address reproducibility by automatically managing the execution of the applications in distributed environments, and by assisting scientists to assemble the workflows and customize them to their particular data. Many researchers are interested, in their projects, to the field of grid computing and workflow [8]. These projects achieved to a variety of management systems for grid workflows, each dedicated to a particular application domain and based on concepts and specific models such as:

- Askalon [9] is a grid application development and computing environment which provides services for composing, scheduling and executing scientific workflows in the grid. Grid workflow applications can be composed using a UML-based workflow composition with Teuta workflow environment [10] or using the XML-based Abstract Grid Workflow Language (AGWL) [11].
- Kepler [12] is one of the most popular workflow systems with advanced features for composing scientific applications. Kepler allows Drag-Drop creation and execution of workflows for distributed applications. Workflows are modelled in MoML (Modeling Markup Language) [13].

- Taverna [14] is a collaboration between the European Bioinformatics Institute (EBI), IT Innovation and the Human Genome Mapping Project Resource Centre (HGMP). In Taverna, data models can be represented in XML based language called Simple Conceptual Unified Flow Language (SCUFL) [15].
- Triana [16] is a workflow-based graphical problem solving environment for data mining applications developed at Cardiff University. Triana provides a visual programming interface with functionalities represented by units.

Compared to these related works, our proposal has essentially 3 main characteristics. First, our framework's architecture is not related to any specific application domain contrarily to others like Triana which is dedicated to distributed data mining on grids or Taverna and Kepler which are both oriented to bioinformatics. Second, many frameworks like Triana are based on self defined notations to compose their workflow models. These notations require a learning phase to be mastered and generate models which are difficult to verify and to validate since corresponding toolkits do not provide any verification tool. We avoided these two above disadvantages by using standard tools such as: UML for workflow specification and BPEL for service and workflow composition. We also were widely inspired by the Workflow Management Coalition (WfMC) [6]. Third, our framework is mainly user-oriented. Users interact with our framework through a friendly graphical interface not requiring any specific expertise on formal languages. This can significantly increase the number of grid users. Workflow and UML make our tool much easier to handle than other frameworks that use exclusively XML-based languages as Taverna.

III. The JASMIN framework

We propose a grid workflow graphical framework composed of two major components. Each component is responsible for one or more specific task in the whole process of management of the distributed application. Our framework JASMIN is the user front-end of the distributed system. It interacts with other service-based and workflow enactment engines in order to accomplish the execution of users' applications. We aim to make grids more efficient and more transparent for different users by making easy interaction between the grid execution environment and the user. Figure 1 describes the architecture of JASMIN and presents its main interactions with the whole system.

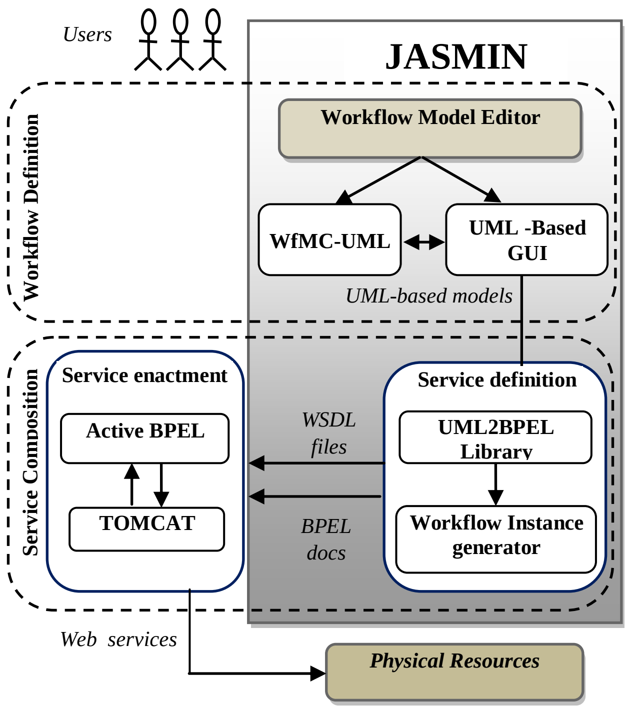


Figure 1. The architecture of JASMIN and its interactions

In order to gain in both efficiency and transparency, we separate between two main steps while composing and deploying processes: the generation of models and the generation of instances. Our framework is workflow-oriented in the first part and service-oriented in the second one. Our architecture is characterized by its capability to separate the specification of applications and their execution. This separation can help to:

- **Easy rewriting of repetitive processes**. Multiple uses of abstract models for a given process are possible without having to redefine it whenever users want to submit the repetitive actions to the grid. Users do not specify conditions on the physical nodes of the grid. This introduces a high degree of transparency.
- **Ease of communication with users**. Users submit their applications in form of diagrams made through a graphical interface easy to handle. This will undoubtedly increase the number of users of grids.
- **Time saving**. Users do not submit sequentially every unit of work separately but realize a global model corresponding to the whole process composed of a set of interdependent activities.
- **Following the evolution of the submitted applications**. Thanks to the graphical interface of the Workflow Model Editor, both submission and visualization of the applications are possible.

In the following, we describe the components of our architecture and their functionalities.

A. *Workflow Definition: The Workflow Model Editor*

The Workflow Model Editor is based on UML activity diagrams. Several editors of UML diagrams exist such as ArgoUML [17]. Our main contribution over these editors is that we focused on both workflow and services. Workflow concepts allow us to see applications through the flow of performed actions. Each application is defined as a set of interdependent activities. The routing rules describe the interdependencies as a control flow of a workflow model and define a formal view of a coordinated set of activities to accomplish the same goal. Besides, our proposal takes into account both the physical constraints of the execution platform and the users' skills. We consider that the execution environment is service-oriented and we suppose that users are not necessary expert and need to be assisted during the submission and the execution of their complex applications.

1) *The WfMC-UML Library*

The Workflow Definition tools of JASMIN support the main routing rules defined by the Workflow Management Coalition (WfMC) such as [6]: the parallel routing, the sequential routing, the AND-split, the OR-split, the AND-join, the OR-join and the iterative routing.

Figure 2, Figure 3, Figure 4 and Figure 5 show the most recurrent WFMC routings and their corresponding models in UML activity diagrams.

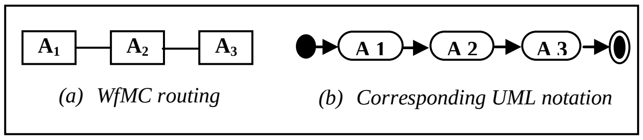


Figure 2. The sequential routing

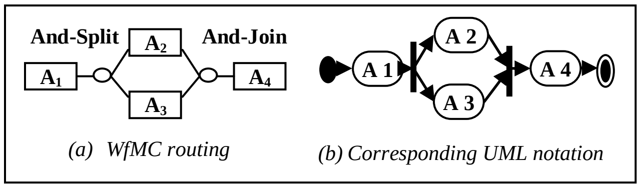


Figure 3. The parallel routing

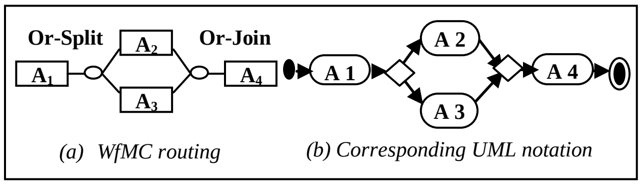


Figure 4. The selective routing

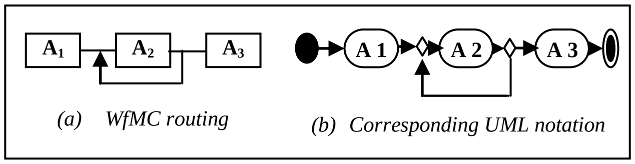


Figure 5. The iterative routing

2) *The UML-Based GUI*

JASMIN interacts with users through the graphical interface of the Workflow Model Editor. Standard UML formalism does not cover service-oriented applications. We decided to refine UML activity diagrams in order to support two main characteristics of new complex applications which are service-oriented and represent scientific workflow processes [18].

While composing new kind of applications, the importance of workflow concepts (rules, routes and roles) presented by the WfMC changes. In scientific workflows, rules expressing constraints and tasks' characteristics are more important than they are in business workflows. Contrary to rules, roles almost lose sense within new scientific processes since the complexity of applications is no more fixed by the human interactions during the process as in business management but by the interdependencies and the requirements of the activities. In addition, activities of grid workflows are often related to stateful services. Users have to compose a workflow of grid services unlike common workflows of business processes where a workflow of web stateless services is composed [19]. Also, UML-based user interfaces provide, usually, information about activities like name, shared objects, routing rules, dependencies, etc. In order to specify grid workflow applications, our interface provides additional information like activity type, activity communication ports with other activities, etc. Figure 6 shows the main window of the Workflow Model Editor.

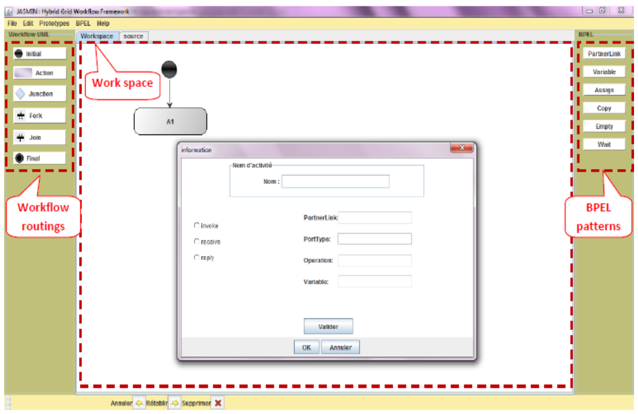


Figure 6. The Workflow Model Editor

Beside the usual patterns, additional toolbars are provided in JASMIN (as shown in figure 6). These ones represent the main refinements that we made on the UML notations. While refining UML, the most important challenge that faced us was: (i) to consider users' skills and provide a convivial interface, and (ii) to consider the service composition language which is BPEL and try to automate the translation of UML models into BPEL documents.

a) Users' skills related refinements: The first type of refinements on UML activity diagrams are related to users' skills. They are defined to ease the work of users and minimize their intervention while submitting and deploying their applications. We consider two different classes of users:

users who are familiar with workflow languages for process management (expert users) and those who have no expertise on UML and workflow (non-expert users).

Expert users represent the class of users from scientific or business fields who already deal with workflow technology and UML formalism. For these users, only standard patterns of UML activity diagrams are requested with no additional patterns or specifications. A tool bar gives access to the main patterns to compose a workflow such as: activity, transition, condition edge, synchronization bar, begin and end nodes, etc. A user can drag any node and drop it to compose a workflow by matching these different patterns together.

Non-expert users are those coming from different application areas and do not necessary have expertise about workflow and UML. However, they need important resources to execute their complex and demanding applications. For this kind of users, we propose a set of predefined prototypes and some dialog boxes to guide them while editing their UML model through JASMIN without forcing them to get a deep knowledge on UML. Users from this class are able to identify the different activities composing the whole process and their interdependencies. Users can use the predefined sub-workflows corresponding to different types of interdependencies to build their whole workflow. Since transparency is being our primary concern, we provide a set of *"prototypes"* corresponding to the most recurrent routings. These routings include, for example, sequential routing, parallel routing (Fork, Join) and selective routing (Switch), as shown in figure 7.

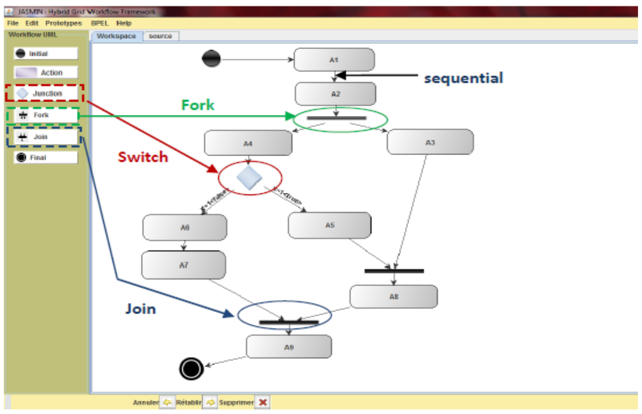
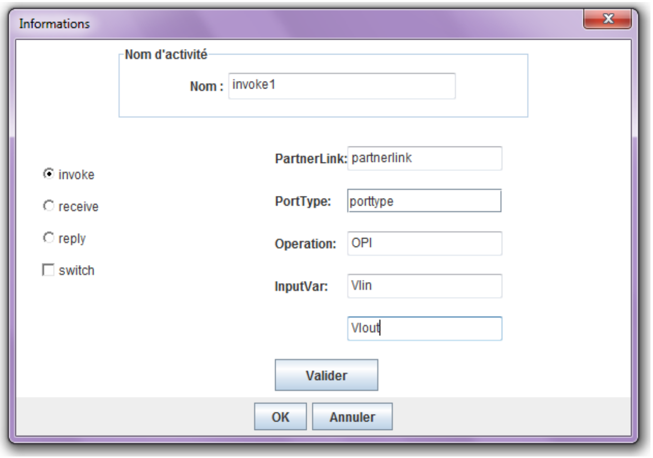


Figure 7. UML prototypes for non-expert users

b) BPEL related refinements: The second type of refinements are related to BPEL notations. Once made, UML activity diagrams have to be managed by service tools. This is possible by enhancing UML notations such as activities, transitions and routing rules with some other patterns like activity properties including, for example, types, variables, port types and partner links. These new patterns are introduced in the UML models in order to make easy the generation of BPEL instances.

Many BPEL patterns are generated automatically, for example, variables and port types in order to reduce users' intervention. However, users still have some informations to indicate like the activity type. While creating any activity, users have to select among a set of activities the type of the activity they wish to insert in the UML activity diagram (see Figure 8). The activity type can be invoke, reply, assign, etc.


Informations

Nom d'activité

Nom : invoke1

[✓ invoke] receive reply switch

PartnerLink: partnerlink

PortType: porttype

Operation: OPI

InputVar: VIin

VIout

Figure 8. BPEL activity types

B. The Service Definition

The workflow model editor helps to generate the UML models corresponding to a given process. However, even by refining UML diagrams, the execution of the workflow models is impossible, unless we translate these models into services. Service composition tools are responsible for the extraction of the executable jobs of workflow instances in form of services from the initial graphical models. In other words, these tools generate from the UML activities flow a set services written in a formal language. There are many workflow formal languages, but BPEL is the standard for describing the service composition. BPEL contains constraints for control flow and data manipulation as well as interaction activities which model the interaction with web services that implement tasks in a workflow [20].

1) The UML2BPEL Library

This library is a set of programs able to generate BPEL tags corresponding to any UML notation in the workflow model. For a portability purpose, the BPEL generation from UML diagrams was divided into two steps.

The first step consists on mapping UML diagrams into Java codes while the second one consists on mapping the obtained Java codes into BPEL documents. This intermediate java code corresponding to the behaviour of the sub-processes and their interdependencies may facilitate a future mapping of UML models into another formal language or creating BPEL documents from other semiformal notation when these ones are coded in java. The second part of the mapping process is from Java code to BPEL document. Each class of the mapping program generates a BPEL activity.

Thanks to the UML2BPEL library and the informations introduced by users while editing UML diagrams, BPEL documents corresponding to both the so-called *basic activities* and the *complex activities* are generated. The basic activities include, for example, invoke activity, receive activity and reply activity. The complex activities represent a set of basic activities grouped by workflow routing rules such as the switch and the flow.

2) The Workflow Instance Generator

In BPEL notation, both activities and interdependencies are supported. Comparing to UML, more notions are present in a BPEL definition. As examples, we can mention partner links, port types, variables and activity types [20].

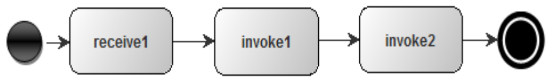
The Workflow Instance Generator is responsible for generating the BPEL documents corresponding to UML models, in coordination with the UML2BPEL library. When users define a new activity or introduce a new pattern related to any activity, the Workflow Instance Generator produces the corresponding code in BPEL. A BPEL document is filled gradually while editing UML diagrams.

Each time a user starts editing a UML activity diagram, a new java file is created. This file is filled while the workflow model is created (when activities are inserted in the diagram or their interdependencies are defined). At the end of the modelling step a Java code corresponding to the whole process is obtained. This mapping from UML to java is invisible to the user.

Beside the generation of BPEL patterns related to basic activities, we also made the generation of BPEL routings automatic. At this level, we proceeded as we did in the generation of the UML models. We implemented some rules to map workflow routing from UML activity diagrams into the so called control flows in BPEL documents. Our grid workflow framework provides a transparent manner to generate the BPEL tags corresponding to the most important WfMC routing rules already presented in the above sections (Sequential routing, parallel routing, selective routing and iterative routing).

3) An example of a sequential routing in JASMIN

The workflow definition and the service composition tools of JASMIN allow to generate, respectively, the UML activity diagram and the BPEL document corresponding to a a given application. In Figure 9, we show the UML model of a sequence of three activities: *receive1, invoke1 and invoke2* and a simplified syntax of its corresponding BPEL document as they are produced by JASMIN.



(a) Sequence in UML

```
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
      name="example"
       …
<sequence>
        <receive partnerLink="PartnerLink" portType="tns:PortType"
                operation="OPR1" variable="VR1" name="receive1">
        </receive>
         <invoke partnerLink="PartnerLink1" portType="tns:PortType1"
                operation="OPI1"  inputVariable="VI1in"
                outputVariable="VI1out" name="invoke1">
         </invoke>
        <invoke partnerLink="PartnerLink2" portType="tns:PortType2"
                operation="OPI2" inputVariable="VI2in"
                 outputVariable="VI2out" name="invoke2">
         </invoke>
 </sequence>
</process>
```

(B) Sequence in BPEL

Figure 9. An example of the sequential routing

C. *The Service enactment*

In order to deploy the workflow as a service in a grid environment, the behaviour description given by BPEL is not enough. It has to be completed by a static description of each activity (service) given in a WSDL file. In fact, the BPEL document shows, for example, which service interacts with which other services and when a given service is invoked. Two kinds of services need to be deployed:

- The web services representing the static description of the workflow and the grid services which are usually deployed on a grid service container, and,
- the workflow services related to BPEL which need a BPEL based workflow engine to be deployed such as ActiveBPEL [21] based on the "Apache Tomcat container".

We believe that it is possible if both kinds of services are deployed on the same service container. We chose to deploy web/grid services on Tomcat instead of the grid container and launch the ActiveBPEL services on Tomcat to allow the deployment of the final workflow services. At this level of our research, we consider that the WSDL documents corresponding to every single involved service available.

IV. CONCLUSION AND FUTURE WORK

Service-oriented grids provide environments to deploy and execute complex applications on distributed and heterogeneous nodes. Despite their performance, Grids stay underweight in terms of ease of use and conviviality. Currently, with the large use of service-oriented technology, workflow tools and languages of service composition are more and more converging. In this paper, we proposed a visual framework for managing applications in service-oriented grids. Our main objective is to take advantage of workflow techniques, service composition tools and grid infrastructures in an easy and transparent way for the user. Our framework JASMIN is based on UML activity diagrams to generate abstract workflow models and on BPEL to generate associated web and grid services to be deployed on a grid environment.

We intend to integrate our framework JASMIN in a service-oriented environment to test physical performances on systems like caGrid [22], Knowledge Grid [23] or Taverna.

ACKNOWLEDGMENT

The authors are pleased to acknowledge Professor Domenico Talia and the members of the GRID Computing Laboratory of the University of Calabria (Italy) for their precious contributions in this research work.

REFERENCES

[1] I. Foster and C. Kesselman, "The grid: blueprint for a new computing infrastructure", Morgan Kaufman, 2004.

[2] G. Baryannis, O. Danylevych, D. Karastoyanova, K. Kritikos, P. Leitner, F. Rosenberg and B. Wetzstein, "Service composition", in: M. Papazoglou, K. Pohl, M. Parkin and A. Metzger (Eds.), Proceedings of the service research challenges and solutions for the future internet, Vol. 6500, Springer, Heidelberg, 2010, pp. 55–84.

[3] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling and P. Vanderbilt, "Open grid services infrastructure (OGSI)", version 1.0., Technical report, Global grid forum, 2003.

[4] I. Foster, "Globus toolkit version 4: Software for service-oriented systems", in: Proceedings of the IFIP International conference on network and parallel computing, Vol. 3779, Springer-Verlag, Tokyo, Japan, 2006, pp. 2–13.

[5] B. Sotomayor, "The globus toolkit 4 programmer's tutorial", Technical report, University of Chicago, 2005.

[6] R. T. Marshak, "Workflow white paper: An overview of workflow software", in: Proceedings of the workflow'94 conference, San Jose, 1994.

[7] W. M. P. van der Aalst and K. Hee, "Workflow management: models, methods, and systems", MIT press, Cambridge, MA, 2002.

[8] J. Yu, R. Buyya, "A taxonomy of workflow management systems for grid computing", Journal of ACM SIGMOD record, vol. 34 (3), 2005, pp. 44–49.

[9] T. Fahringer, R. Prodan, R. Duan, J. Hofer, F. Nadeem, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H. L. Truong, A. Villazon and M. Wieczorek, "Askalon: A development and grid computing environment for scientific workflows", Springer Verlag, 2005, Ch. Workflows for escience, scientific workflows for grids, pp. 450–471.

[10] T. Fahringer, S. Pllana and J. Testori, "Teuta: Tool support for performance modeling of distributed and parallel applications", in: Proceedings of international conference on computational science, tools for program development and analysis in computational science, Springer-Verlag, Karakov, Poland, 2004, pp. 456–463.

[11] T. Fahringer, J. Qin and S. Hainzer, "Specification of grid workflow applications with agwl: An abstract grid workflow language", in: Proceedings of the IEEE international symposium on cluster computing and the grid, Vol. 2, Cardiff, UK, 2005, pp. 676–685.

[12] B. Luduscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao and Y. Zhao, "Scientific workflow management and the Kepler system", Concurrency and computation: practice and experience, Special issue on scientific workflows, vol. 18 (10), 2005, pp. 1039–1065.

[13] A. E. Lee and S. Neuendorffer, "MoML a modeling markup language in XML version 0.4.", Technical memorandum ERL/UCB M, University of California, Berkeley, 2000.

[14] T. M. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. P. Pocock, A. Wipat and P. Li, "Taverna: A tool for the composition and enactment of bioinformatics workflows", Bioinformatics, vol. 20 (17), 2004, pp. 3045–3054.

[15] G. Hobona, D. Fairbairn, H. Hiden and P. James, "Orchestration of grid-enabled geospatial web services in geoscientific workflows", IEEE transactions on automation science and engineering, vol. 7 (2), 2010, pp. 407–411.

[16] I. J. Taylor, M. S. Shields, I. Wang and O. F. Rana, "Triana applications within grid computing and peer to peer environments", Journal of grid computing, vol. 1 (2), 2003, pp. 199–217.

[17] ArgoUML, http://www.argouml.org

[18] M. Sonntag, D. Karastoyanova and F. Leymann, "The missing features of workflow systems for scientific computations", in: G. Engels, M. Luckey, A. P. and R. Reusner (Eds.), Proceedings of workshops on software engineering, Vol. 160 of LNI, GI, Hanoi, Vietnam, 2010, pp. 209–216.

[19] W. Dou, J. L. Zhao and S. Fan, "A collaborative scheduling approach for service-driven scientific workflow execution", Journal of computer and system sciences, vol. 76 (6), 2010, pp. 416–427.

[20] D. Jordan, J. Evdemon and A. Alves, "Web service business process execution language version 2.0.", Technical report, OASIS standard, 2007.

[21] The ActiveBPEL Project, http://www.activebprl.org

[22] J. H. Saltz, S. Oster, S. Hastings, S. Langella, T. M. Kurc¸, W. Sanchez, M. Kher, A. Manisundaram, K. Shanbhag and P. A. Covitz, "cagrid: design and implementation of the core architecture of the cancer biomedical informatics grid", Bioinformatics, vol. 22 (15), 2006, pp. 1910–1916.

[23] E. Cesario, M. Lackovic, D. Talia and P. Trunfio, "A visual environment for designing and running data mining workflows in the knowledge grid", In: Data mining: foundations and intelligent paradigms, D. Holmes, L. Jain (Editors), Springer, Intelligent systems reference library, vol. 24, 2012, pp. 57--75.