SysML-Based Domain-Specific Executable Workflows

Vikas V. Patel, John D. McGregor and Sebastien Goasguen
Clemson School of Computing
Clemson University
Clemson, South Carolina, USA
{vikasp,mcgrego,sebgoa}@clemson.edu

Abstract—**This paper presents a tool called SysFlow Workflow Engine (SWE) that is being developed to execute a domain workflow defined using SysML's Activity Diagram. The paper also describes extensions added to the SysML semantics to make them SWE executable. SWE focuses on grid computing, cyberinfrastructure and related domains; however support for other domains can be easily added. SWE aims to provide a common interface to grid, cyberinfrastructure and other domain-specific software by abstracting their complexity and idiosyncrasies. To create a workflow, users can use SysML modelers such as Topcased, which allows them to create and validate SysML models. Before submitting a workflow to SWE for execution, users have to ensure that their workflow is not only a valid SysML model but also a valid SWE executable model. SWE receives a SysML workflow in XML Metadata Interchange (XMI) format and after performing certain validation checks it parses and executes the workflow.**

Keywords-SysML; Workflows; Executable models; Modeling;

I. Introduction

SysML (Systems Modeling Language), a standard defined by the Object Management Group (OMG), is a general purpose graphical modeling language for specifying, analyzing, designing and verifying complex systems [1]. SysML is a profile of UML 2.0 and adds its own diagrams. In this paper the activity diagram of SysML is used to compose executable workflows for grid computing [26] and related domains. The Grid infrastructure allows large scale scientific applications to run on distributed resources, however grid resources are not very easy to use and the software and middleware used to access them are diverse and intricate [27].

SWE provides a layer of abstraction over the supported domain software and provides a common and consistent way for workflow modelers to use these services and operations in their workflows. A workflow consists of multiple steps connected by control/data flow wherein every step represents an operation or computation. The notion of workflow is a popular and natural method of modeling complex scientific applications [25]. The SysML Block Definition Diagram (BDD) provided by SWE for every supported domain serves as an Application Programming Interface (API) for the workflow modeler. A BDD describes the system hierarchy and system/component specification [1]. This helps the workflow modeler identify the operations (behaviors) and attributes associated with a domain or its components, which they can use to model their workflows.

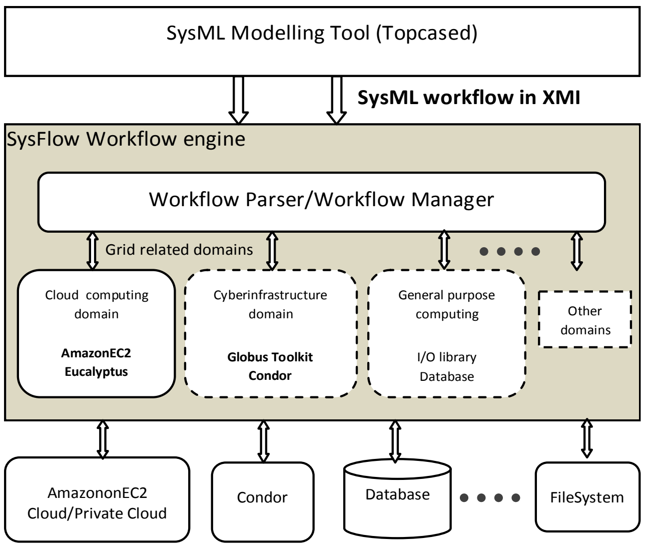
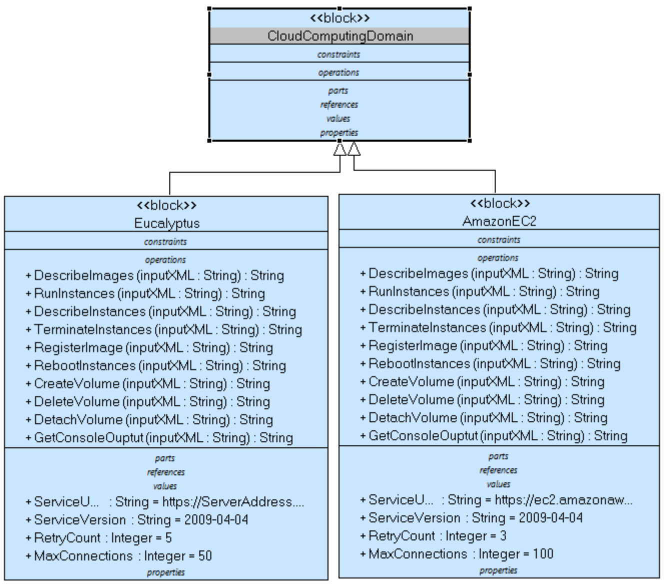


Fig. 1. A simplistic architecture of the SysFlow Workflow Engine (SWE)

The rest of the paper is organized as follows. In the next section we describe some related work, in section 3 we discuss the SWE architecture and some of the important extensions added to the abstract semantics of SysML Activity Diagram to make it executable by SWE. Section 4 and 5 present examples from the grid computing domain in order to discuss the steps required to orchestrate a SWE executable workflow in SysML and discuss some of the internal details of SWE. In Section 6 conclusions are drawn and future work is discussed.

II. Related Work

There have been several investigations into the use of UML/SysML to describe simulation models and executable domain workflow models. [7], [11] discuss extending SysML to include domain specific details and then transform them to arrive at simulation models. [12] and [13] describe tools to transform UML models to simulation models. Similarly [16] and [23] define extensions to the UML Activity diagram and then transform them to executable workflows. In the SWE approach, extensions are added to the SysML Activity Diagram to model executable workflows for the grid related



<<block>>
CloudComputingDomain
constraints
operations
parts
references
values
properties
<<block>>
Eucalyptus
constraints
operations
+ DescribeImages (inputXML : String) : String
+ RunInstances (inputXML : String) : String
+ DescribeInstances (inputXML : String) : String
+ TerminateInstances (inputXML : String) : String
+ RegisterImage (inputXML : String) : String
+ RebootInstances (inputXML : String) : String
+ CreateVolume (inputXML : String) : String
+ DeleteVolume (inputXML : String) : String
+ DetachVolume (inputXML : String) : String
+ GetConsoleOuptut (inputXML : String) : String
parts
references
values
+ ServiceU... : String = https://ServerAddress....
+ ServiceVersion : String = 2009-04-04
+ RetryCount : Integer = 5
+ MaxConnections : Integer = 50
properties
<<block>>
AmazonEC2
constraints
operations
+ DescribeImages (inputXML : String) : String
+ RunInstances (inputXML : String) : String
+ DescribeInstances (inputXML : String) : String
+ TerminateInstances (inputXML : String) : String
+ RegisterImage (inputXML : String) : String
+ RebootInstances (inputXML : String) : String
+ CreateVolume (inputXML : String) : String
+ DeleteVolume (inputXML : String) : String
+ DetachVolume (inputXML : String) : String
+ GetConsoleOuptut (inputXML : String) : String
parts
references
values
+ ServiceU... : String = https://ec2.amazonaw...
+ ServiceVersion : String = 2009-04-04
+ RetryCount : Integer = 3
+ MaxConnections : Integer = 100
properties

Fig. 2. Block definition diagram of the Cloud Comuting Domain (SWE). The block diagram does not show all the operations for brevity.

domains. The workflow models are directly executed by SWE, unlike other approaches which transform SysML models into simulation or executable models and other approaches which generate code from the system model. This work also involves abstracting various grid related software/services and providing a common interface to the workflow modeler.

[28] proposes the design of "Cyberaide Shell", a system shell, to allow easy access to advanced Cyberinfrastructure [24] resources by abstracting the complexities associated with resource, task and application management through a scriptable command line interface. In this paper, workflows described by SysML Activity Diagrams are used instead of command line scripts because activity diagrams provide an effective visual notation, facilitate easy analysis of workflows composition, and deliver functionality in a more natural way for a human user [16]. There are several articles such as [5], [6] and [15] that discuss the effectiveness of the UML Activity diagram as a workflow. The SysML Activity Diagram extends the Activity model from UML, therefore the aforementioned advantages of Activity Diagrams hold true for SysML Activity Diagrams. SysML is used because it supports more general description of systems and domains compared to UML which is software oriented.

III. SYSFLOW WORKFLOW ENGINE (SWE)

SWE is a Java based Workflow Engine which executes workflows modeled in SysML. Figure 1 shows the general architecture of SWE, the dashed boxes indicate the domains for which the support is planned but currently not implemented. Users create workflow models using modeling tools such as Topcased. Topcased is an open source visual modeling tool and supports many modeling languages including UML and SysML. Topcased supports limited validation of SysML Activity Diagrams and can save models in OMG XMI [3] format. OMG's XMI 2.1 format and ISO's 10303 STEP AP233 data interchange standard are the two main options for storing and exchanging SysML models. The majority of the SysML/UML modeling tools such as Topcased, IBM Rational Software Architect, and ARTiSAN Studio support the XMI format. The workflow in XMI format is then passed to SWE, where it is first parsed and validated before being executed.

SWE is extensible and support for related or new domains can be easily added. The supported domains can be seamlessly integrated into a single workflow. Consider a scenario, wherein a researcher has a local Condor [18] pool of 50 nodes and requires another 300 nodes to finish the submitted jobs in a reasonable time. Also assume that the results of the jobs should be saved on a local database. The researcher can make use of the AmazonEC2 [19] cloud from the Cloud Computing [21] domain to add another 300 nodes to his Condor pool and use the database adapter from the general purpose computing domain to save the results onto a local database. All this can be done seamlessly in the same workflow.

The domain systems and features supported by the SysFlow Workflow Engine (SWE) are modeled and available to the users as Block Definition Diagram (BDD). The users modeling the workflow use these predefined block definition diagrams to discover the services and operations available to them. Fig. 2 shows the BDD for the Cloud Computing Domain. At present SWE includes limited support for the cloud computing domain, in particular it supports the AmazonEC2 cloud and Eucalyptus [20] based private computing clouds. The general purpose computing domain support includes features such as reading/writing files, logging and database support, which do not fit into any particular domain.

The abstract semantics of the SysML Activity diagram are not sufficient to accurately describe a domain specific executable workflow. Extensions are added and constraints are imposed on the actual semantics of the activity diagram to make it a suitable executable model for SWE. Some of these extensions and constraints are listed below.

- Every Action node of an Activity should have a unique name. Furthermore an Action node can have at most one input pin and one output pin excluding the target pin, in the case of a Call Operation Action node.
- An output pin can send only Extensible Markup Language (XML) data and similarly an input pin can receive only XML. If an action does not have an input or output pin, the corresponding input or output is null.
- An Action node is considered as an execution step. The input and output of an Action node are treated as XML variables and are created when the Action node receives a token (XML data) and when it returns a result respectively. The scope of these variables is limited to the current activity.

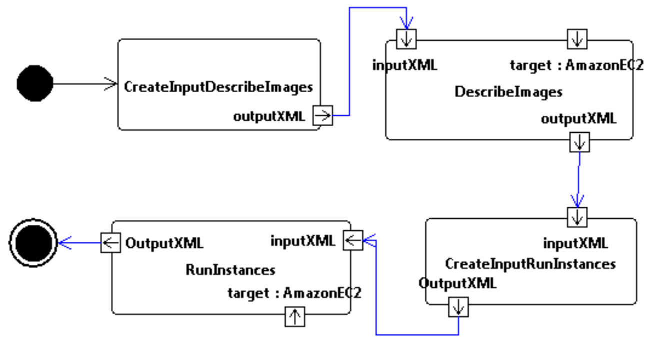


Fig. 3. A simple SWE executable workflow using OMG SysML Activity Diagram. The workflow fetches the list of images owned by a user and then runs an instance of the first image on AmazonEC2 cloud.

- The Call Operation Action is used to call an operation or invoke a service in a specific domain system. The predefined BDD defines the domain systems and services currently supported by SWE. However the BDD does not describe the XML structure of the input or the output, this is taken care by the XML schema's which are made available to the user along with the BDD.
- XPath [9] or XQuery [10] expressions can be used to select element or attribute from an XML variable.
- XQuery Boolean expressions can be used as a guard condition for edges and as decision input for decision nodes.
- Every SWE SysML model should have one Activity Diagram with name "Main". When a SysML model is executed with SWE, it starts with the Main activity and then executes the other activities as required.

A call operation action uses the target pin to specify the target object to which the request is sent; this object constitutes the context of the execution of the operation [2]. The type of target pin should be same as the type that owns the operation. In Fig. 3 the nodes DescribeImages and RunInstances are call operation actions. The target pins are of the type AmazonEC2, this indicates that the DescribeImages and RunInstances requests are sent to the AmazonEC2 domain.

IV. SWE WORKFLOW EXAMPLE 1

In this section, an example which involves running virtual machine (VM) instances on the Amazon Elastic Compute Cloud (AmazonEC2) is introduced. This example demonstrates the behavior of SWE and discusses the steps required to compose a workflow for SWE. AmazonEC2 is a web service that provides elastic compute capacity in the computing cloud, it allows users to quickly scale up or scale down their computing capacity as requirements change [19].

Fig. 3. shows a very simple SWE executable workflow modeled using SysML Activity diagram, the only thing the workflow does is to fetch a list of VM images owned by a user and then send a request to AmazonEC2 to boot the first VM image in the list. In this workflow, the initial node and the final node, like the name suggests, indicate the start and end of the workflow. The opaque action CreateInputDescribeImages creates XML input for the DescribeImages action. The DescribeImages action is a Call Operation action, which invokes the DescribeImages operation of the AmazonEC2 domain.

On seeing an AmazonEC2 DescribeImages operation call, SWE composes a request message using the XML input and sends a request to the AmazonEC2 server to get a list of images owned by the user. SWE then stores the XML version of the response in the output variable of DescribeImages action. The input variable of DescribeImages is referred to as $DescribeImagesRequest, the output as $DescribeImageResponse and the special variable which is created on error and holds an error response is referred to as $DescribeImageErrorResponse. The RunInstances Operation runs one or more instances of an image on the cloud. To start one or more instances, it requires the id of the image to launch (ImageId), the minimum number of instances to launch (MinCount) and the maximum number of instances to launch (MaxCount). The Opaque action CreateInputRunInstances is responsible for providing this information. The user enters this information in the Body property of CreateInputRunInstances action, below is the sample XML data.

```
<Input>
 <ImageId>
 {($DescribeImagesResponse//ImageId)[1]/text()}
  </ImageId>
  <MinCount>1</MinCount>
  <MaxCount>1</MaxCount>
</Input>
```

In the above XML, the curly braces indicate that the value of /Input/ImageId element is not a literal but an expression to be evaluated. The XQuery expression "($DescribeImagesResponse//ImageId)[1]" first searches for the list of ImageId elements in the $DescribeImagesResponse variable and then picks the first "ImageId" element from the list and returns its text value. When SWE finishes the execution of CreateInputRunInstances, it wraps the above XML in <CreateInputRunInstancesResponse> </CreateInputRunInstancesResponse> and copies it into the $CreateInputRunInstancesResponse variable. The XML below shows the possible contents of the $CreateInputRunInstancesResponse variable.

```
<CreateInputRunInstancesResponse>
  <Input>
    <ImageId>ami-dd59b8b </ImageId>
     <MinCount>1</MinCount>
     <MaxCount>1</MaxCount>
   </Input>
```

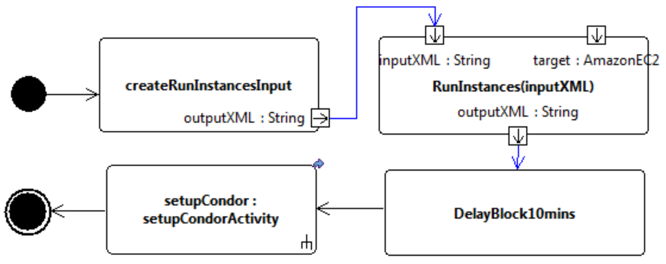


Fig. 4. A SysML SWE workflow which starts VMs on the AmazonEC2 cloud, waits for them to boot and then calls the sub-process in figure 5.

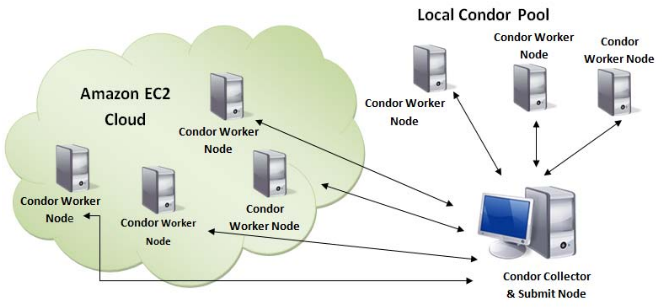


Fig. 5. A Condor pool with some worker nodes on the cloud. All the worker nodes report to the Condor Collector.

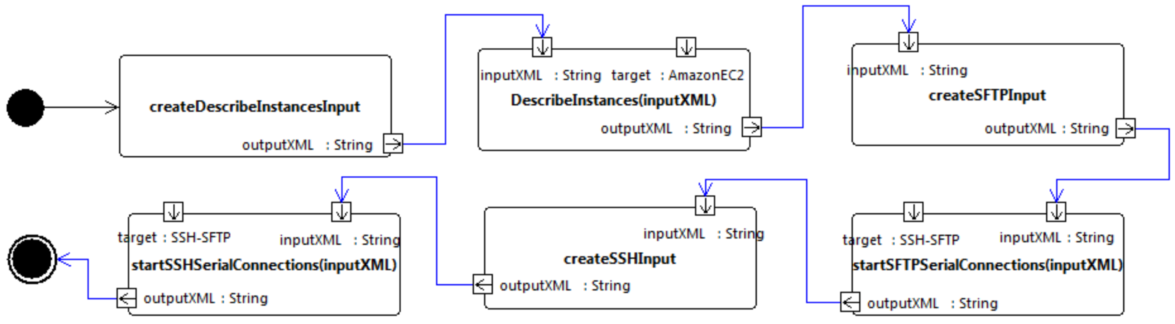


Fig. 6. A SysML SWE executable activity diagram responsible for setting up condor on virtual machines on the Amazon EC2 cloud.

</CreateInputRunInstancesResponse>

Currently, there is a limited support for XQuery built into SWE. However SWE has a number of inbuilt functions which can be used for simple to complex data manipulations. For instance SWE has a function called cirg:GetElementData(variableName, Xpath-Expression) which could have been used to retrieve the required ImageId value from $DescribeImagesResponse. The SWE custom function support, allows users to create and make available to SWE at runtime almost anything they can code in Java as SWE functions.

V. SWE Workflow Example 2

In this section, an example which involves starting Condor [18] worker nodes on the Amazon EC2 cloud and adding them to the local Condor pool is introduced. Fig. 5 shows a Condor pool which has worker nodes on the local network as well as on Amazon EC2 cloud. The SWE workflow consists of two SysML Activity Diagrams as shown in Fig. 4 and Fig. 5. The Activity Diagram in Fig. 4 is the "Main" activity diagram of the workflow and therefore the entry point with which SWE begins execution. In this Activity Diagram, the createRunInstancesInput opaque action and RunInstances call operation action are responsible for starting VMs on the AmazonEC2 cloud. The DelayBlock10mins is a OpaqueAction node which uses the custom XPath function cirg:sleep(milliseconds) to make the process sleep for 10 minutes, providing enough time for the VMs to boot.

```
<Input>
  <delay>{cirg:sleep(600000)}</delay>
</Input>
```

The setupCondor node is a Call Behavior Action node and it calls the Activity Diagram setupCondorActivity shown in Fig. 6. The setupCondorActivity Activity Diagram uses the DescribeInstancesInput and DescribeInstances to fetch the information about the instances running on the cloud. The createSFTPInput Opaque action and the startSFTPSerialConnection CallOperationAction nodes are responsible for copying the Condor setup and configuration script files to the VMs, information about which was obtained from the previous callOperationAction call. Below is a sample input to the startSFTPSerialConnection.

```
<Input>
 <Host>ec2-184-73-6-96.compute-1.amazonaws.com</Host>
 <Host>ec2-184-73-7-93.compute-1.amazonaws.com</Host>
 <KeyFile>../../condor.pem</KeyFile>
 <LocalFileName>../../condor.zip</LocalFileName>
 <RemotePath>/root</RemotePath>
 <Permissions>0777</Permissions>
 <User>root</User>
</Input>
```

The startSFTPSerialConnection node can transfer files to multiple VMs provided they are instances of the same image or use the same credentials. The createSSHInput and startSSHSerialConnections nodes are responsible for establishing SSH connections to the VMs and running the setup and configure scripts which were copied to the VMs by the previous action nodes. After this step the Condor daemons on the VMs report to the Condor Collector and are ready to accept jobs. The Final node marks the end of the sub-process and the control is transferred back to the Main Activity Diagram which terminates following the Final node.

VI. Conclusions And Future Work

This paper introduced SWE to execute workflows related to the Grid Computing domain and described in SysML. SWE provides a common interface to all the supported grid/cyberinfrastructure software and resources; it saves the users from the intricacies and quirks associated in dealing with multiple grid software. The paper also demonstrated that by adding simple extensions to the SysML Activity Diagram, it could be made executable by SWE. Steps required to create and execute a simple workflow on SWE have been illustrated with an example.

Currently, better support for error handling, XQuery, parallel execution and validation needs to be added. Work is also underway to improve the support for the Cloud Computing domain and adding support for the Globus Toolkit and the Condor batch scheduler. Future work includes adding support to more grid related software and extending the support to other domains.

References

[1] *OMG Systems Modeling Language Specification Version 1.1*, Object Management Group Std., 2008.

[2] *OMG Unified Modeling Language Version 2.2, Superstructure*, Object Management Group Std., 2009.

[3] *MOF 2.0/XMI Mapping Version 2.1.1*, Object Management Group Std., 2007.

[4] T. Weilkiens, *Systems Engineering with SysML/UML: Modeling, Analysis, Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[5] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and P. Wohed, "On the suitability of uml 2.0 activity diagrams for business process modelling," in *APCCM '06: Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling*. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2006, pp. 95–104.

[6] M. Dumas and A. H. M. t. Hofstede, "Uml activity diagrams as a workflow specification language," in *'01: Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*. London, UK: Springer-Verlag, 2001, pp. 76–90.

[7] E. Huang, R. Ramamurthy, and L. F. McGinnis, "System and simulation modeling using sysml," in *WSC '07: Proceedings of the 39th conference on Winter simulation*. Piscataway, NJ, USA: IEEE Press, 2007, pp. 796–803.

[8] R. Bastos, D. Dubugras, and A. Ruiz, "Extending uml activity diagram for workflow modeling in production systems," vol. 9. Los Alamitos, CA, USA: IEEE Computer Society, 2002, p. 291.

[9] *XML Path Language (XPath) Version 1.0*, World Wide Web Consortium (W3C) Std., 1999.

[10] *XQuery 1.0: An XML Query Language*, World Wide Web Consortium (W3C) Std., 2007.

[11] C. J. J. Paredis and T. Johnson, "Using omg's sysml to support simulation," in *WSC '08: Proceedings of the 40th Conference on Winter Simulation*. Winter Simulation Conference, 2008, pp. 2350–2352.

[12] L. B. Arief and N. A. Speirs, "A uml tool for an automatic generation of simulation programs," in *WOSP '00: Proceedings of the 2nd international workshop on Software and performance*. New York, NY, USA: ACM, 2000, pp. 71–76.

[13] O. Constant, W. Monin, and S. Graf, "A model transformation tool for performance simulation of complex uml models," in *ICSE Companion '08: Companion of the 30th international conference on Software engineering*. New York, NY, USA: ACM, 2008, pp. 923–924.

[14] Y. Jarraya, M. Debbabi, and J. Bentahar, "On the meaning of sysml activity diagrams," vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 95–105.

[15] S. Pllana, T. Fahringer, J. Testori, S. Benkner, and I. Brandic, "Towards an UML based graphical representation of grid workflow applications," in *2nd European Across Grids Conference, Nicosia, Cyprus*, January 2004. Springer-Verlag, 2004.

[16] Y. Hlaoui and L. BenAyed, "Extended uml activity diagram for composing grid services workflows," *in Risks and Security of Internet and Systems, 2008. CRiSIS '08. Third International Conference on*, Oct. 2008, pp. 207–212.

[17] M. Pantel, "The TOPCASED project: a toolkit in open source for critical applications & systems design," in *Model-Driven Development Tool Implementers Forum (MDD-TIF)*, Zurich, 2007.

[18] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor – a distributed job scheduler," in *Beowulf Cluster Computing with Linux*, T. Sterling, Ed. MIT Press, October 2001.

[19] Amazon Web Services, "Amazon elastic compute cloud (amazon EC2)." [Online]. Available: http://aws.amazon.com/ec2/

[20] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 124–131.

[21] A. Weiss, "Computing in the clouds," netWorker, vol. 11, no. 4, pp. 16–25, 2007.

[22] E. Walker and T. Minyard, "Orchestrating and coordinating scientific/engineering workflows using gridshell," in *High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on*, June 2004, pp. 270–271.

[23] A. D. Lucia, R. Francese, and G. Tortora, "Deriving workflow enactment rules from uml activity diagrams: a case study," *Human-Centric Computing Languages and Environments, IEEE CS International Symposium on*, vol. 0, pp. 211–218, 2003.

[24] J. Cao, *Cyberinfrastructure Technologies and Applications*. Commack, NY, USA: Nova Science Publishers, Inc., 2009.

[25] T. Gubala, D. Herezlak, M. Bubak, and M. Malawski, "Semantic composition of scientific workflows based on the petri nets formalism," *e-Science and Grid Computing, International Conference on*, vol. 0, p. 12, 2006.

[26] I. Foster, "The anatomy of the grid: Enabling scalable virtual organizations," vol. 15, no. 3, 2001, p. 2001.

[27] S. Krishnan, B. Stearn, K. Bhatia, K. K. Baldridge, W. W. Li, and P. Arzberger, "Opal: Simpleweb services wrappers for scientific applications," *Web Services, IEEE International Conference on*, vol. 0, pp. 823–832, 2006.

[28] G. von Laszewski, A. Younge, X. He, K. Mahinthakumar, and L. Wang, "Experiment and workflow management using cyberaide shell," *Cluster Computing and the Grid, IEEE International Symposium on*, vol. 0, pp. 568–573, 2009.

[29] I. G. Alonso, M. P. A. G. Fuente, and J. A. L. Brugos, "Using sysml to describe a new methodology for semiautomatic software generation from inferred behavioral and data models," *Systems, Second International Conference on*, vol. 0, pp. 210–215, 2009.