12-2010

# SysML-Based Domain-Specific Executable Workflows

Vikas Patel
*Clemson University*, vikas.patel.vidyalal@gmail.com

# SYSML-BASED DOMAIN-SPECIFIC EXECUTABLE WORKFLOWS

---

A Thesis
Presented to
the Graduate School of
Clemson University

---

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Science.

---

by
Vikas Vidyalal Patel
December 2010

---

Accepted by:
Dr. Sebastien Goasguen, Committee Chair
Dr. John D. McGregor
Dr. Wayne Goddard

# ABSTRACT

The Systems Modeling Language (SysML) is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems. This thesis presents a tool called SysFlow Workflow Engine (SWE) that is being developed to execute a domain workflow defined using SysML's Activity Diagram. The thesis 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 allow 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.

# DEDICATION

This thesis is dedicated to my parents, brother, friends and Lord Almighty.

# ACKNOWLEDGMENTS

I would like to acknowledge my advisor Dr Sebastien Goasguen for his ideas, support and motivation to pursue this work. I would also like to thank Dr John D. McGregor for his advice and invaluable feedback during the course of this work.

TABLE OF CONTENTS

Page

Table of Contents (Continued)

LIST OF LISTINGS

LIST OF FIGURES

List of Figures (Continued)

CHAPTER ONE

INTRODUCTION

## 1.1. SysFlow Workflow Engine (SWE)

SysML, an extension of UML 2.0 and 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]. In this thesis, the activity diagram of SysML is used to compose workflows for grid computing [26] and related domains. These workflows are executable by the SysFlow Workflow Engine (SWE). The outcome of executing these workflows could be something as simple as printing "Hello World" on the screen or something like instantiating a virtual machine on the AmazonEC2 cloud, depending upon the composition of the workflow.

The Grid infrastructure allows large-scale scientific applications to run on


Fig. 1.1. SysFlow Workflow Engine (SWE)

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.

### 1.2. Related Work

There have been several investigations into the use of UML/SysML to describe simulation models and executable domain workflow models. References [7], [11] discuss extending SysML to include domain-specific details and then transform them to arrive at simulation models. Reference [12] and [13] describe tools to transform UML models to simulation models. Similarly, reference [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 domains. The workflow models are directly executed by SWE, unlike other approaches that 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.

Reference [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 thesis, 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.

SysFlow has an interface to some of the cloud-computing web API's like AmazonEC2 and Eucalyptus that allows users to create complex executable workflows involving these cloud technologies. In recent years cloud computing has emerged as a hot technology that promises to simplify problems associated with maintaining large business applications and IT infrastructure by providing on demand access to hardware, software,

and data resources as a service [38]. Amazon EC2, VMware vCloud Express, Eucalyptus, and OpenNebula are some of the popular cloud computing solutions for building private and public clouds. One of the most important enablers of cloud Computing is virtualization, which provides infrastructure and management layer for the cloud. Virtualization allows running multiple virtual-machine instances simultaneously on a single machine and greatly increases resource utilization [21]. VMware and Citrix are the


Fig 1.2 Virtualization: VMware ESX Server

biggest providers of virtualization solutions, with 85 percent (as of August 2009) of all installed enterprise virtual machines being VMware based [39]. Fig 1.2 shows a simplified architecture of VMware's enterprise level virtualization technology called VMware ESX Server, which is bundled as a part of VMware vSphere. ESX Server is a bare metal hypervisor i.e. it runs directly on the physical hardware and partitions it into a number of virtual machines. Each of these virtual machines has its own processor, memory, networking etc., and can run simultaneously, completely unaware of each other.

The Amazon Web Services (AWS) offering, Elastic Cloud (EC2) is based on Citrix XenServer technology whereas the VMware vCloud Express is based on the VMware vSphere virtualization technology. Both VMware vCloud Express and AmazonEC2 have similar pricing but vCloud has more options related to instance provisioning such as amount of RAM or number of processors assigned per instance of a virtual machine. At present, vCloud Express offers more choices for operating system instances, in comparison, AWS has other offerings like Amazon Simple Storage Service (S3), Virtual Private Cloud and Simple Queue Service for which vCloud has no equivalent offerings [40]. Cloud services are usually exposed as web services and follow industry standards such as Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) and Universal Description Discovery and Integration (UDDI) [38]. Since most cloud services provide a web API, support for them can be easily added in SysFlow, which in turn enables straight forward executable workflow composition and cloud service orchestration.

### 1.3. Thesis Organization

The rest of the thesis is organized as follows. The next chapter includes topics on some of the prerequisites (SysML, XML, XPath, XQuery, Grid Computing, Cloud Computing, and Condor Batch Scheduler). In chapter 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. The chapter also includes sections that 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 chapter 4 conclusions are drawn and future work is discussed.

CHAPTER TWO

PREREQUISITES

## 2.1. Sytems Modeling Language (SYSML)

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]. 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. SysML is mainly designed for System Engineering applications, unlike Unified Modeling Language (UML) which focuses on

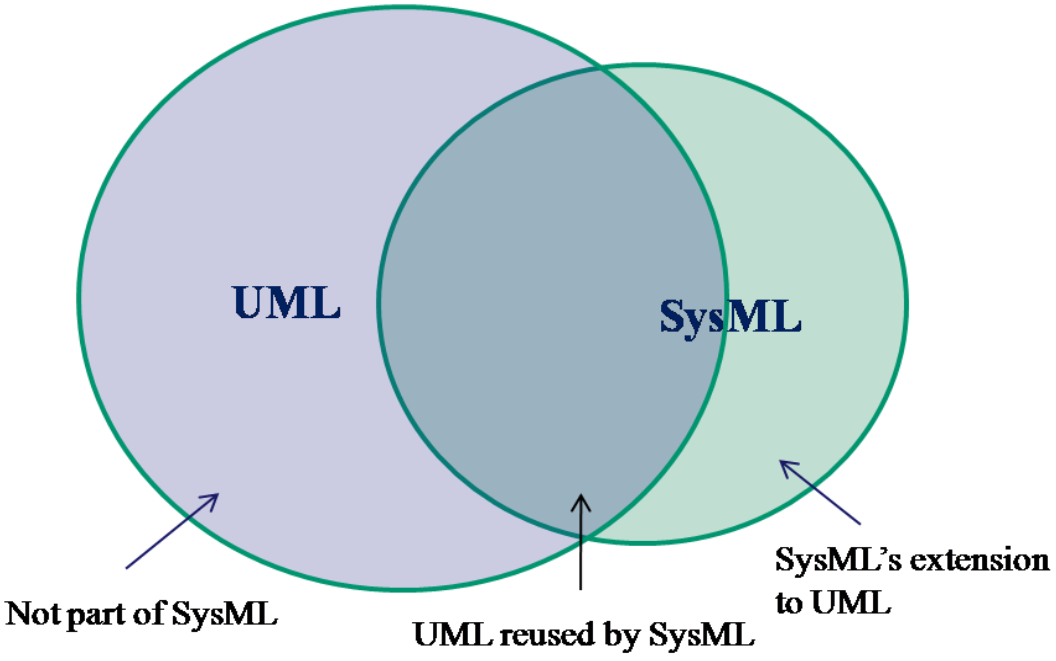


Figure 2.1: Venn Diagram, showing relationship between UML 2.0 and SysML

Software Engineering. SysML is a profile of UML 2.0 and adds its own diagrams as highlighted by the Venn diagram in the figure 2.2.

The Requirement Diagram and the Parametric Diagram are the two new diagram types added by SysML. SysML reuses Sequence Diagram, State Machine Diagram, Use Case Diagram and Package Diagram from UML 2.0 without any changes. The Activity Diagram, Block Definition Diagram and Internal Block Diagram are also reused from UML 2.0 but with certain modifications.

The SysML taxonomy is shown in the figure 2.2. Behavior Diagram, Requirement Diagram, Structure Diagram and Parametric Diagram types are the four pillars of OMG SysML.

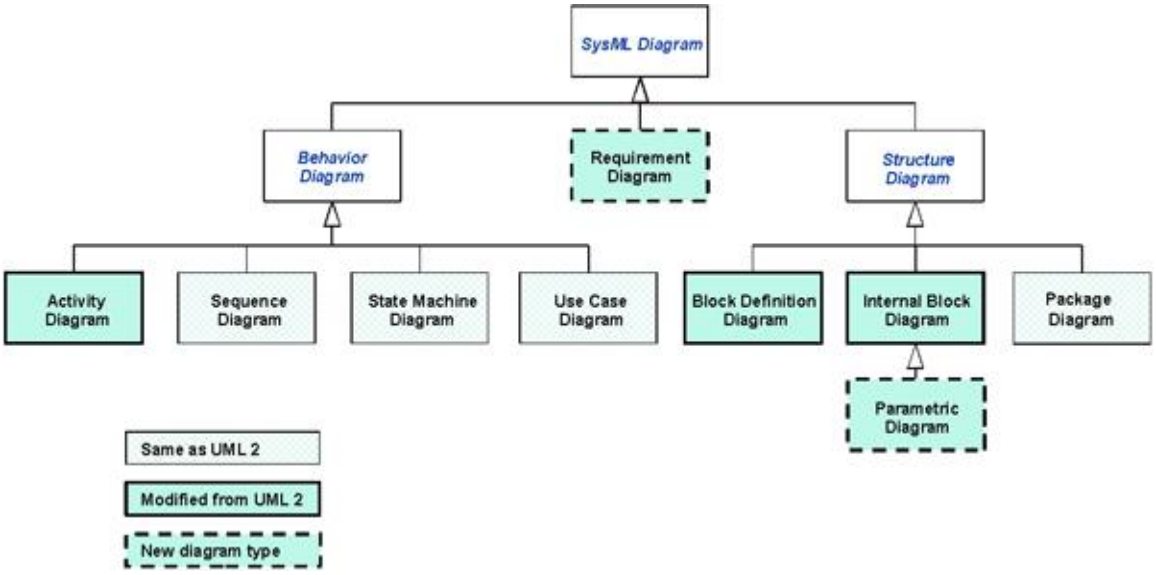


Figure 2.2: OMG SysML Diagram Taxanomy. Picture credit :http://www.omgsysml.org/

*Behavior Diagrams:*

- **Activity Diagram**: An Activity Diagram shows the flow from one activity to another. An Activity consists of one or more actions. An action may include calling an operation, sending a signal, any kind of computation, such as evaluating an expression [30]
- **Sequence Diagram**: A Sequence Diagram mainly shows interaction between objects in the order in which they occur i.e. in sequential order. A sequence diagram focuses on the time ordering of messages. In a typical sequence diagram the objects are arranged along the X-axis and messages are ordered in terms of increasing time along the Y axis.
- **State Machine Diagram**: A State Machine Diagram shows the flow of control from one state to another. A state is a situation or condition in the behavior of a system when a condition holds true.
- **Use Case Diagram**: Use Case Diagrams shows a set of use cases and actors and their relationships [30].

*Structure Diagram*

- **Block Definition Diagram**: A BDD describes the system hierarchy and system/component specification. It describes the relationship between blocks such as composition, association, and specialization.
- **Internal Block Diagram**: An Internal Block Diagram shows the internal structure of a block or the relationship between its constituent parts.

- **Parametric Diagram**: Parametric Diagrams show the parametric constraints between the structural elements. They use the constraint blocks of Block Definition Diagram to constrain the properties of other blocks.
- **Package Diagram**: A Package Diagram shows the organization of a model in terms of packages, views and viewpoints.

*Requirement Diagram*

A Requirement Diagram shows system requirements and their relationship with other elements [1].

## 2.2. Extensible Markup Language

Extensible Markup Language (XML) is a subset of Standard Generalized Markup Language (SGML). XML is a simple, very flexible text format and its goal is to enable generic SGML to be served, received and processed on the web in the way that is now possible with HTML [31]. XML is useful not only for describing documents for the web but also for describing structured data. XML can easily represent tabular data as well as semi-structured data such as web page or business document. XML is platform independent and not limited to any programming language, operating system, or software [32].

An XML document that satisfies all the syntax rules provided in the specification is called well-formed. In addition, a well-formed XML document is said to be valid if it has a reference to a valid DTD/XML schema and it conforms to the referenced DTD/XML

schema. Below is the example of a well-formed and valid XML. The XML schema called

```
<?xml version="1.0" encoding="UTF-8"?>
<SoftwareRepository xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.vikaspatel.org/vp" xsi:schemaLocation="http://www.vikaspatel.org/vp
soft.xsd">
        <Software>
                <Author>Vikas Patel</Author>
                <Title>Sysflow Workflow Engine</Title>
                <Type>Free</Type>
        </Software>
        <Software>
                <Author>Vikas Patel</Author>
                <Title>SiteWriter for Pegasus Workflow Engine</Title>
                <Type>Free</Type>
        </Software>
</SoftwareRepository>
```

Listing 2.1: An XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ns1="http://www.vikaspatel.org/vp" targetNamespace="http://www.vikaspatel.org/vp"
elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="SoftwareRepository">
                <xs:annotation>
                        <xs:documentation>The Software Repository holds the list of
software and their authors</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                        <xs:sequence>
                                <xs:element name="Software" minOccurs="0"
maxOccurs="unbounded">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="Author"/>
                                                        <xs:element name="Title"/>
                                                        <xs:element name="Type"/>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                        </xs:sequence>
                </xs:complexType>
        </xs:element>
</xs:schema>
```

Listing 2.2: An XML Schema for XML document in listing 2.1

“soft.xsd” given in listing 2.2 defines the elements of the xml document in the listing 2.1.

XML supports the use of Unicode characters other than the characters that are reserved by XML itself such as >, <, etc. Therefore, the XML containing Hindi characters in the listing 2.3 is well-formed.

```
<?xml version="1.0" encoding="UTF-8"?>
<हिन्दी-गाने>
        <गाना>दिल है की मानता नही ।</गाना>
        <गाना>मेरा नाम जोकर्</गाना>
</हिन्दी-गाने>
```

Listing 2.3: A well formed XML document with Hindi Unicode characters.

### 2.3. XML Path Language (XPath)

XML Path Language (XPath) is used to search and retrieve elements or attributes from an XML document. XPath includes numerous built-in functions that can be used for comparison and manipulation of strings, numbers and booleans. Xpath conforms to the XQuery/XPath Data Model (XDM) [9].

The XPath expressions below operate on the XML in listing 2.4. The results are given below the expressions and also highlighted in the listing 2.4. The prefix aws refers to the “http://ec2.amazonaws.com/doc/2009-04-04/” namespace

- **XPath Expression**: /aws:DescribeInstancesResponse

  **Result**: Selects the root element Selects DesribeInstancesResponse. The selected node is highlighted in blue in the listing 2.4.

- **XPath Expression**: /aws:DescribeInstancesResponse/aws:requestId/text()

  **Result*:*** Selects text value of the requestId element which is the child of DescribeInstancesResponse. The selected text value "f1402336-220c-47c2-87b6-d50675a03d25" is highlighted in red in the listing 2.4.

```
<DescribeInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2009-04-04/">
        <requestId>f1402336-220c-47c2-87b6-d50675a03d25</requestId>
        <instancesSet>
                <item>
                        <instanceId>i-b957dad2</instanceId>
                        <imageId>ami-cb13fea2</imageId>
                        <instanceState>
                                <code>16</code>
                                <name>running</name>
                        </instanceState>
                        <privateDnsName>domU-12-31-39-0F-85-17.compute-
1.internal</privateDnsName>
                        <dnsName>ec2-184-73-100-202.compute-
1.amazonaws.com</dnsName>
                </item>
                <item>
                        <instanceId>i-bf57dad4</instanceId>
                        <imageId>ami-cb13fea2</imageId>
                        <instanceState>
                                <code>16</code>
                                <name>running</name>
                        </instanceState>
                        <privateDnsName>domU-12-31-39-0B-25-87.compute-
1.internal</privateDnsName>
                        <dnsName>ec2-204-236-201-84.compute-
1.amazonaws.com</dnsName>
                </item>
        </instancesSet>
</DescribeInstancesResponse>
```

Listing 2.4: An XML document, highlighting element/values selected by the XPath expressions in section 2.3.

- **XPath Expression**:

  /aws:DescribeInstancesResponse/aws:instancesSet/aws:item[2]/aws:instanceState/aws:code/text()

**Result:** Selects the text value of "instanceState" element which is the child of the second "item". The result text "16" is highlighted in green in the listing 2.4.

## 2.4. XML Query Language (XQuery)

XQuery is a language used to query XML data. The queries in XQuery are concise and easy to understand; it can also a query broad range of XML information sources, including both databases and documents. XQuery operates on the abstract, logical structure of an XML document known as a data model defined in XQuery/XPath Data Model (XDM) [10]. XQuery is compatible with other W3C standards such as XML, Namespaces, XSLT, XPath and XML Schema. Listing 2.5 shows an example involving

```
<InstanceInfo xmlns:aws="http://ec2.amazonaws.com/doc/2009-04-04">
    <dns>{ for $x in doc("vp2.xml") return
$x/aws:DescribeInstancesResponse//aws:privateDnsName}</dns>
     <firstImageId></firstImageId>
</InstanceInfo>
```

Listing 2.5: XQuery Example

```
<InstanceInfo xmlns:aws="http://ec2.amazonaws.com/doc/2009-04-04">
        <dns>
                <privateDnsName xmlns="http://ec2.amazonaws.com/doc/2009-04-
04">domU-12-31-39-0F-85-17.compute-1.internal</privateDnsName>
                <privateDnsName xmlns="http://ec2.amazonaws.com/doc/2009-04-
04">domU-12-31-39-0B-25-87.compute-1.internal</privateDnsName>
        </dns>
        <firstImageId/>
</InstanceInfo>
```

Listing 2.6: XML obtained as a result of evaluating XQuery in listing 2.5

XQuery usage. The content of file vp2.xml used by the XQuery expression is shown in

listing 2.4. Listing 2.6 shows the resulting XML.

## 2.5. Grid Computing

Grid computing is a combination of computer resources from multiple administrative domains, shown in figure 2.3. The grid infrastructure allows large scale scientific applications to run on distributed resources. A grid consists of a number of Virtual Organizations (VO). A set of individuals and/or institutions defined by certain sharing

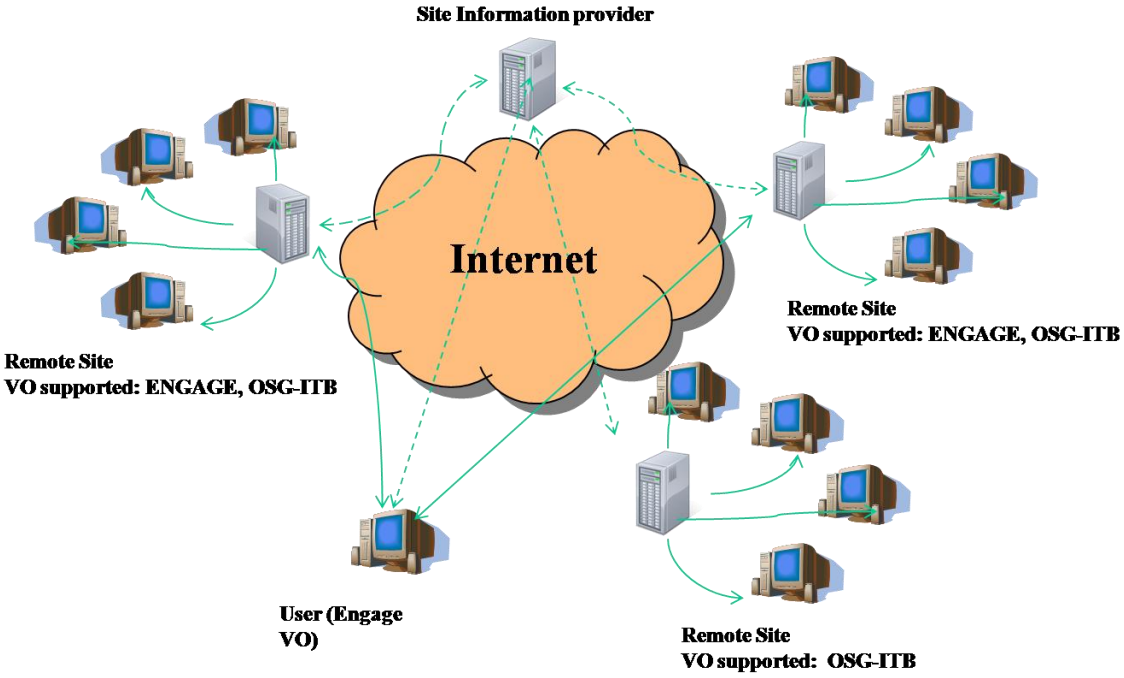


Figure 2.3 Grid Computing

rules form a VO. The sharing includes not only file exchange but rather direct access to computer, software, data and other resources as required by a range of collaborative

problem solving and resource-brokering strategies. The sharing is usually highly controlled by resource providers and consumers who clearly define what can be shared, who can share and under what conditions. [26]. There are number of grid middleware such as Globus Toolkit [26] and NoduGrid Middleware [35] .The Open Science Grid (OSG) [36] and TeraGrid [37] are some of the popular grid initiatives in the United States.

### 2.6. Cloud Computing

Cloud computing provides on-demand access to computational resources; it allows users to scale up and scale down their capacity as requirements change. Users don't have to invest in a new infrastructure, bother about maintenance or license new software.

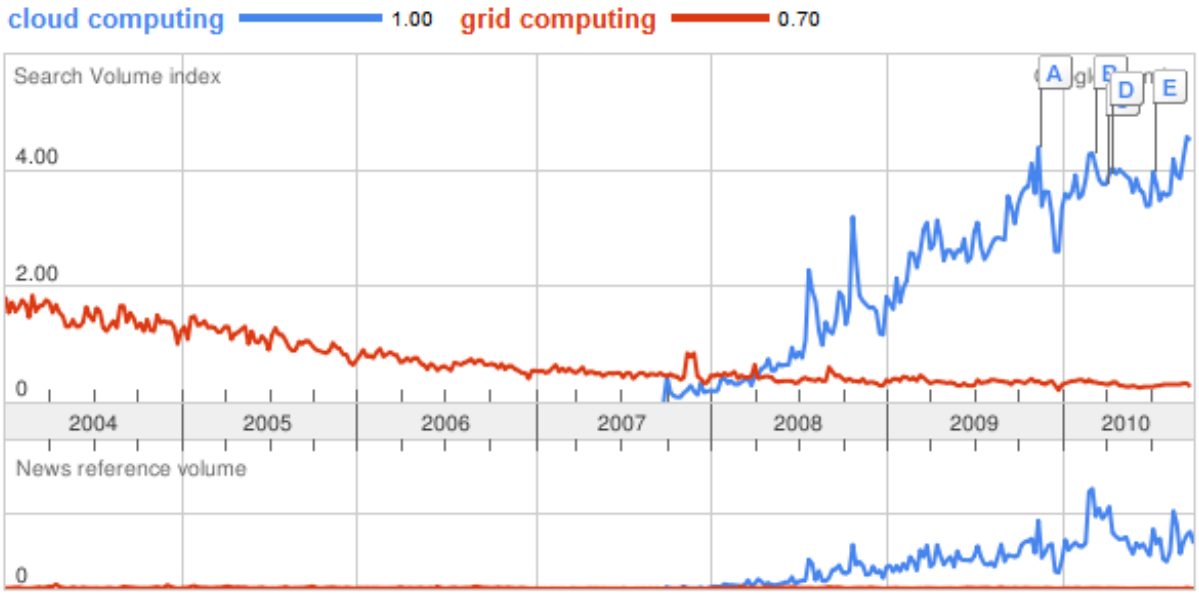


Fig 2.4 Comparison of Cloud Computing & Grid Computing in Google Trends

Since cloud computing is dynamically scalable, it allows users to draw as much computing power required on a per hour basis and therefore as demand for computational power, storage, or network capacity increases or diminishes, these resources can be easily added or subtracted. Cloud computing has incorporated the basic technologies of grid computing, utility computing, elastic computing and software as a service [33]. Cloud computing has become a popular technology in recent years due to its ability to provide on demand access to computational and storage resources. Fig 2.4 shows that in recent years, the popularity of cloud computing has surpassed grid computing. One of the main concerns associated with cloud computing is security. Many users are unwilling and unsure about hosting their data and applications on third-party infrastructure. In some cases, users are concerned about the location where their applications/data are hosted, since the laws of the host country applies to data/application hosted in that country.

Some of the popular Cloud computing providers are Amazon Elastic Cloud, Google App Engine, Eucalyptus, VMware vCloud and Windows Azure from Microsoft.

## 2.7. Condor Batch Scheduling System

Condor is a high-throughput distributed batch-scheduling system. Condor provides job management, scheduling policy, priority scheme, resource management and resource monitoring [34] Condor is designed for high throughput and opportunistic computing. High throughput computing involves the use of computational resources over long periods of time to solve a computational problem, whereas opportunistic computing

ensures the effective utilization of resources whenever they are available, without requiring one hundred percent availability.

Condor helps to make use of idle resources; it allows submission of multiple jobs at once allowing tremendous amounts of computation to be done with very little intervention from the user. Condor can be used on networks of various sizes, on a single machine. Condor executes the submitted job when the machine is idle, saves the state of the job when the machine becomes busy due user interaction and resumes the execution once resources are available again. It also restarts the job if the machine reboots. On a dedicated cluster condor behaves as a cluster submission tool.

CHAPTER THREE

# SYSFLOW, EXECUTABLE WORKFLOW ENGINE

## 3.1. Architecture of the SysFlow Workflow Engine

SWE is a Java-based Workflow Engine that executes workflows modeled in SysML. Figure 3.1 shows the general architecture of SWE, the dashed boxes indicate the domains for which 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.

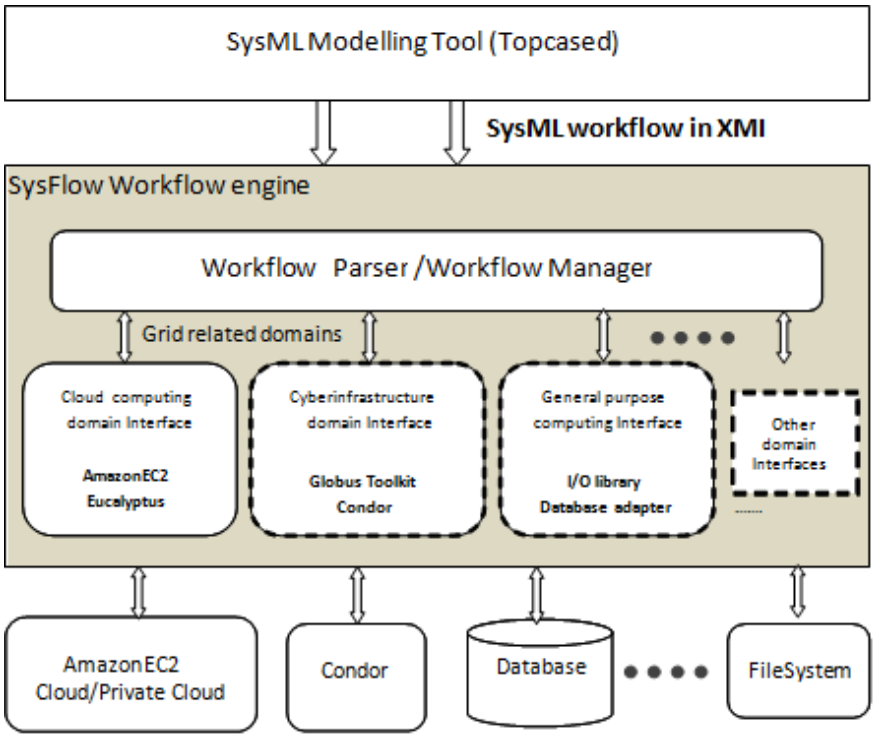


Fig. 3.1. A simplistic architecture of the SysFlow Workflow Engine (SWE)

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.

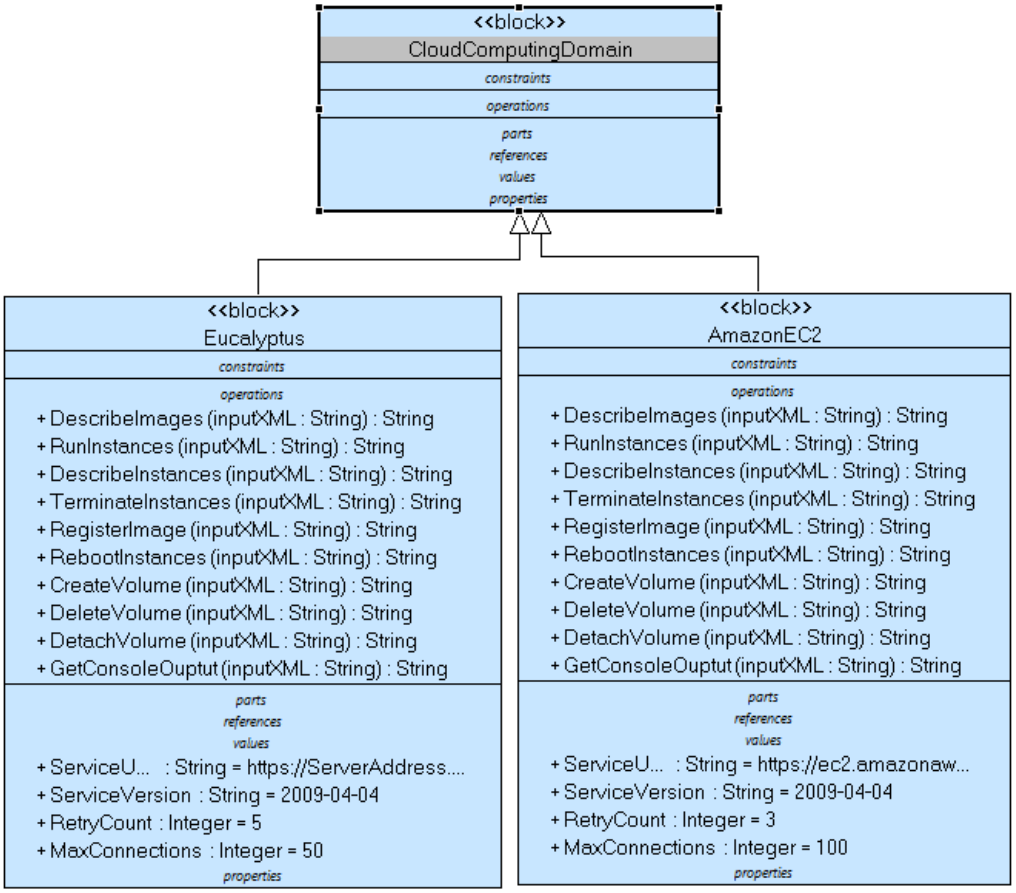


Fig. 3.2. Block definition diagram of the Cloud Comuting Domain (SWE). The block diagram does not show all the operations for brevity.

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.3.2 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.

### 3.2. SWE Workflow Semantics

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. The UML 2.0 specification defines a pin as a typed element and multiplicity element that provides values to actions, and accepts result values from them. An input pin is a pin that holds the input values to be consumed by an action, whereas an output pin is a pin that holds the values produced by an action [2].
- 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.
- 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. 3.3 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.

### 3.3. SWE Workflow Example 1

In this section, an example that 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].

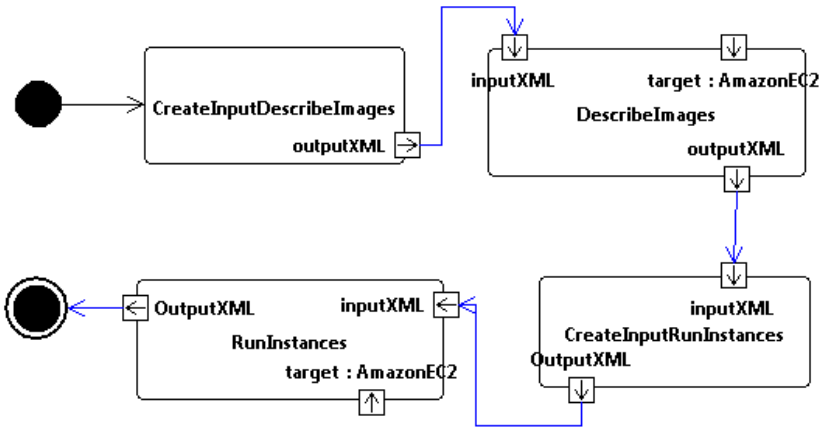


Fig. 3.3. A simple SWE executable workflow using OMG SysML Activity Diagram. The workflow fetches the list of images owned by a user and then runs an instance of the first image on AmazonEC2 cloud.

Fig. 3.3. 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. Opaque actions are type of actions that include implementation specific information.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>
</CreateInputRunInstancesResponse>
```

Currently, there is a limited support for XQuery built into SWE. However SWE has a number of inbuilt functions that can be used for simple to complex data manipulations. For instance, SWE has a function called cirg:GetElementData(variableName, Xpath-Expression) that 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.

### 3.4. SWE Workflow Example 2

In this section, an example that involves starting Condor [18] worker nodes on the Amazon EC2 cloud and adding them to the local Condor pool is introduced. Fig. 3.4 shows a Condor pool which has worker nodes on the local network as well as on Amazon EC2 cloud. The SWE workflow to achieve the configuration shown in Fig 3.4 consists of three SysML Activity Diagrams as shown in Figs. 3.5, 3.6 and 3.7.

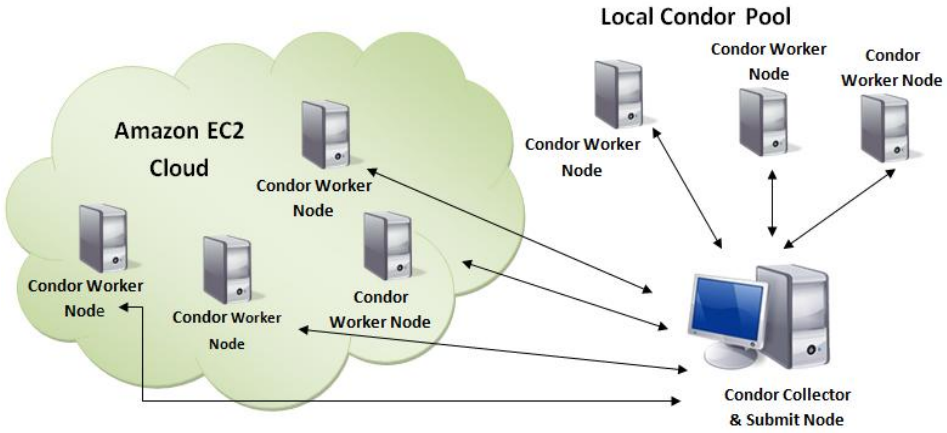


Fig. 3.4. A Condor pool with some worker nodes on the cloud. All the worker nodes report to the Condor Collector.

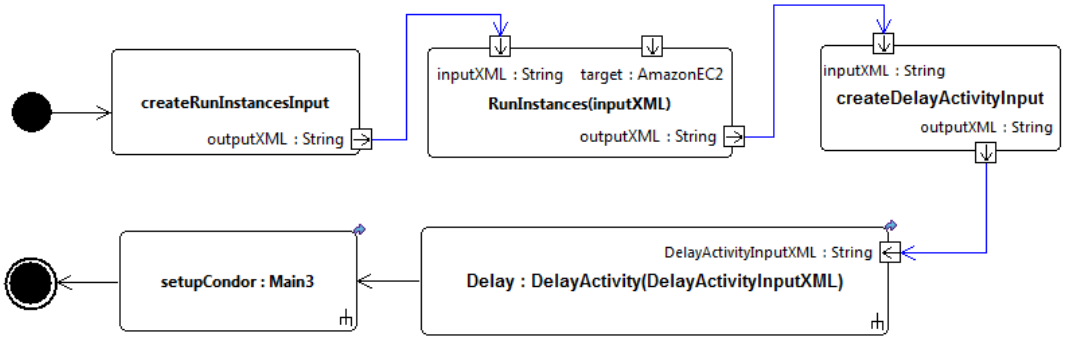


Fig 3 5. A SysML SWE workflow which starts VMs on the AmazonEC2 cloud, waits for them to boot and then calls the sub-process in figure 3.7.

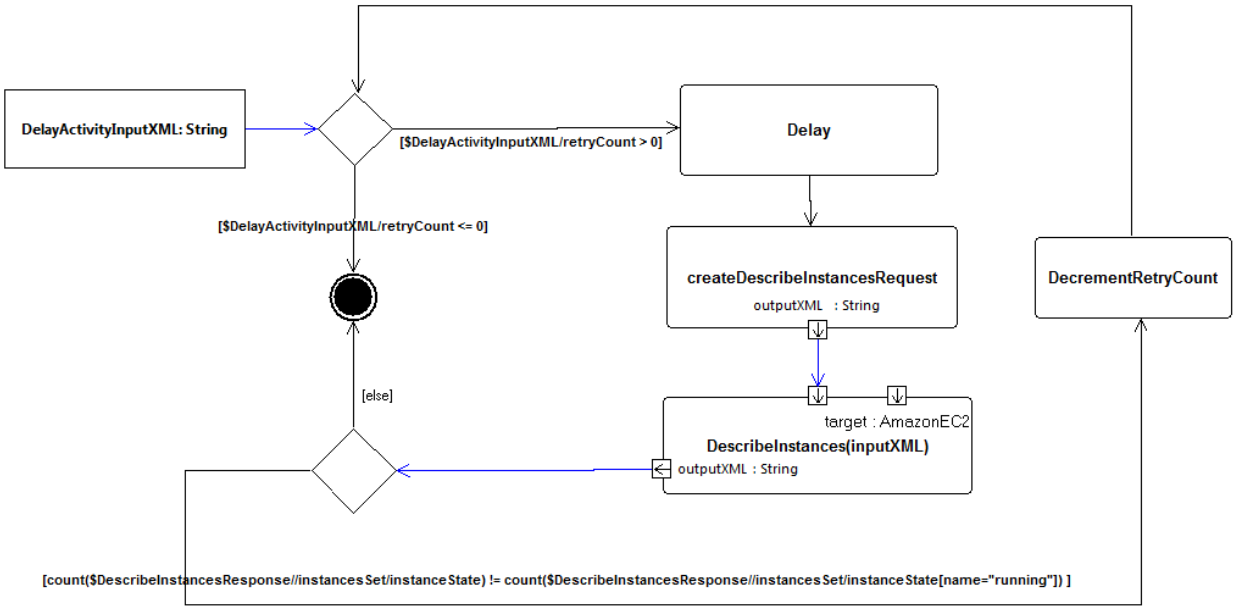


Fig. 3.6. A SysML SWE executable activity, which checks a specified number of times, every few minutes, if all virtual machines have finished booting or not.

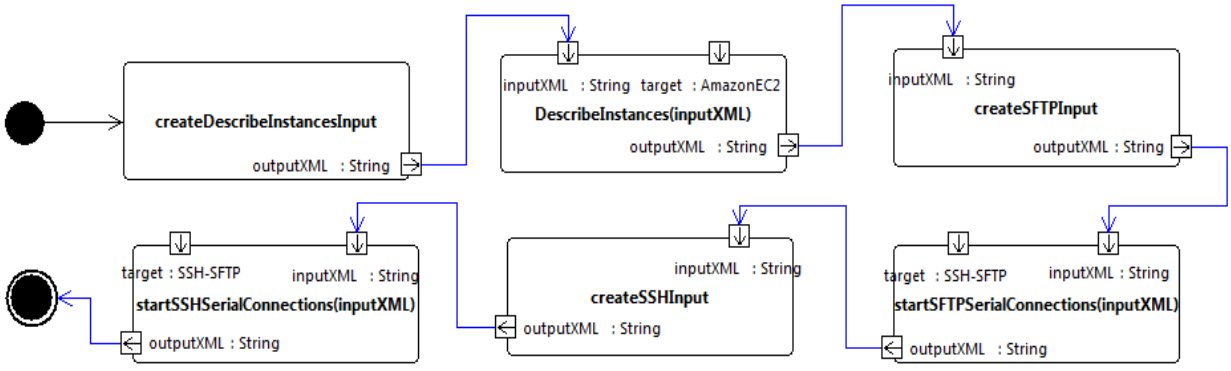


Fig. 3.7. A SysML SWE executable activity diagram responsible for setting up Condor on virtual machines on the Amazon EC2 cloud.

The workflow discussed in this section is fairly simplified and does not include

any kind of error checking or error handling. The Activity Diagram in Fig. 3.5 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 createRunInstancesInput node creates an XML message for RunInstances node, which in turn is responsible for composing a SOAP request, using this message and sending a webservice request to AmazonEC2 cloud to start virtual machine instances The flow then reaches the createDelayActivityInput opaque action node which creates an xml variable containing the following information.

*<Input>*

*<retryCount>10<retryCount>*

*<delay> 120000</delay>*

*<instanceIdList>{$RunInstancesResponse//item/instanceId}</instanceIdList>*

*</Input>*

This XML variable is then passed to the Delay CallBehaviorAction node which calls the DelayActivity activity, with the xml variable it received as the input parameter. The DelayActivity activity first waits for "x" seconds, as specified by delay element of the input parameter and then sends a DescribeInstances request (which specifies the instanceId's of virtual machine instances which were started by the Main activity) to the AmazonEC2 cloud. It then checks if the total number of instances returned by the previous call is equal to the total number of instances whose status is set to "running". If

this is true, all virtual machine instances have finished booting and are ready to use; otherwise, the activity decrements the retry count and repeats the process again until retry count becomes zero or all virtual machine instances reach the running state. The Delay node is a OpaqueAction node, which uses the custom XPath function cirg:sleep(milliseconds) to make the process sleep for specified number of milliseconds. The body of the Delay node is as below:-

```
<Input>
  <delay>{cirg:sleep($DelayActivityInputXML/delay/text())}</delay>
</Input>
```

The setupCondor node is a Call Behavior Action node and it calls the Activity Diagram setupCondorActivity shown in Fig.3.6. 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.

# CHAPTER FOUR

# CONCLUSIONS AND FUTURE WORK

This thesis 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 thesis 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 were illustrated with two examples.

Currently, better support for error handling, XQuery, parallel execution and validation needs to be added. Work is also under way to improve the support for the Cloud Computing domain and adding support for the VMware vCloud, Globus Toolkit and the Condor batch scheduler. Future work includes adding support to more grid-related software and extending the support to other domains. Future work also includes developing a SWE plugin for Topcased that can determine the input required by a node by referring to the corresponding XML schema and provide a user interface for the user to enter the values. This would save the user from typing XML. The SWE plugin would also support creating complex xml transformation by drag and drop feature. This would minimize the need to write complex Xpath/XQuery expressions. Another feature planned is to develop a web-based visual monitoring tool or a plugin for Topcased, which would show the current execution in real time. The tool will also display the workflow being

executed with suitable annotation and coloring, highlighting the parts that finished execution (with input and output data received, time consumed at each node) and parts that are pending execution

# APPENDICES

# Appendix A

## SWE Miscellaneous Topics

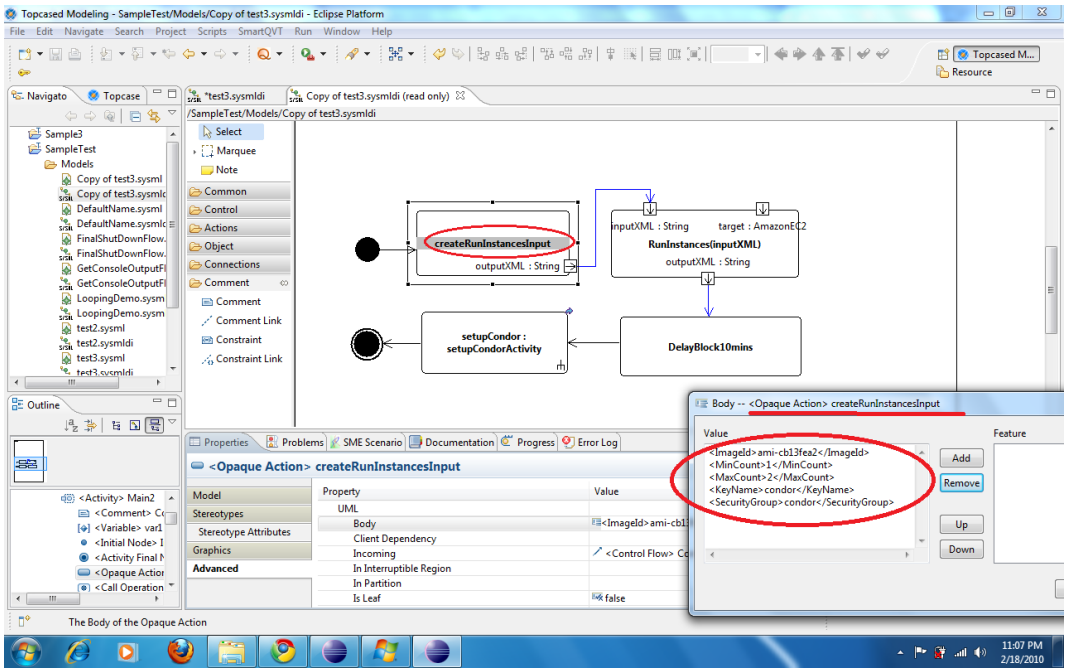


Figure A-1.1: Screenshot of the Topcased Modeler.

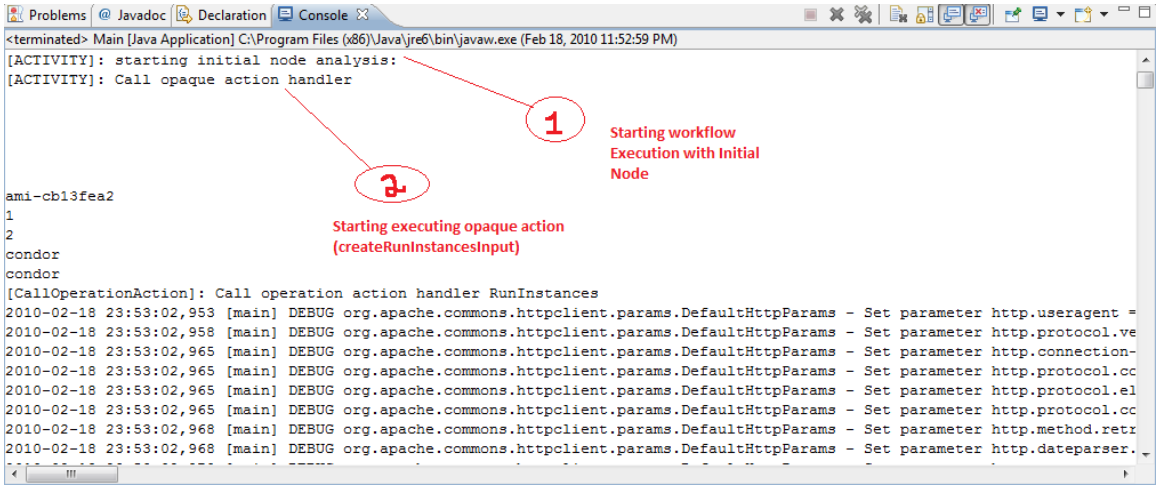


Figure A-1.2: SWE output on the console, on executing a workflow.

A sample SWE Workflow in XMI format is given below. This workflow is parsed and executed by SWE

```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Model xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
xmlns:sysML="http://www.topcased.org/2.0/sysML"
xmlns:uml="http://www.eclipse.org/uml2/3.0.0/UML" xmi:id="idModel" name="test3Model">
  <packagedElement xmi:type="uml:Package" xmi:id="idPackage" name="test3Package">
    <ownedComment xmi:type="uml:Comment" xmi:id="_98OjIBYHEd-nTpeT6OuvbQ">
      <body>SSH inputXML&#xD;
&#xD;
&lt;HostName>&#xD;
&lt;User>&#xD;
&lt;KeyFile>&#xD;
&lt;KeyFilePass>&#xD;
&lt;ShellCommand>uname -q &amp;&amp; /sbin/ifonfig&lt;/ShellCommand>&#xD;
</body>
    </ownedComment>
    <ownedComment xmi:type="uml:Comment" xmi:id="_3I-10BaYEd--GfqQ0FyhCA">
      <body>startSFTPSerialConnection&#xD;
&lt;HostName>{($DescribeInstancesResponse//PublicDnsName)[2]}&lt;/HostName>&#xD;
&lt;HostName>{($DescribeInstancesResponse//PublicDnsName)[1]}&lt;/HostName>&#xD;
&lt;ShellCommand>unzip condor.zip &amp;amp;&amp;amp;
./condor2.sh&lt;/ShellCommand>&#xD;
&lt;KeyFile>D:/vikas/891 RESEARCH/condor.pem&lt;/KeyFile>&#xD;
&lt;LocalFileName>D:/vikas/891 RESEARCH/condor-ec2/condor.zip&lt;/LocalFileName>&#xD;
&lt;RemotePath>/root&lt;/RemotePath>&#xD;
&lt;Permissions>0777&lt;/Permissions>&#xD;
&lt;User>root&lt;/User>&#xD;
</body>
    </ownedComment>
    <packageImport xmi:type="uml:PackageImport" xmi:id="_MdKiMBX2Ed-nTpeT6OuvbQ">
      <importedPackage xmi:type="uml:Model"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#_0"/>
    </packageImport>
    <packagedElement xmi:type="uml:Activity" xmi:id="idActivity" name="Main">
      <ownedComment xmi:type="uml:Comment" xmi:id="_84-YgBbaEd-GLPHJcCtbNQ">
        <body>Comment1</body>
      </ownedComment>
      <variable xmi:type="uml:Variable" xmi:id="_HbD2EBX9Ed-nTpeT6OuvbQ" name="var1">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
      </variable>
      <node xmi:type="uml:InitialNode" xmi:id="_3rZH8BXrEd-nTpeT6OuvbQ" name="InitialNode1"
outgoing="_d2dNcBXvEd-nTpeT6OuvbQ"/>
      <node xmi:type="uml:ActivityFinalNode" xmi:id="_6QQRYBXrEd-nTpeT6OuvbQ"
name="ActivityFinalNode1" incoming="_F7gVkBYCEd-nTpeT6OuvbQ"/>
      <node xmi:type="uml:OpaqueAction" xmi:id="_-lPUoBXrEd-nTpeT6OuvbQ"
name="createRunInstancesInput" incoming="_d2dNcBXvEd-nTpeT6OuvbQ">
        <outputValue xmi:type="uml:OutputPin" xmi:id="_Xr2P4BXwEd-nTpeT6OuvbQ"
name="outputXML" outgoing="_aqhAeBXwEd-nTpeT6OuvbQ">
```

<type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
<upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_Xr2P4RXwEd-nTpeT6OuvbQ" value="*"/>
</outputValue>
<body><ImageId>ami-cb13fea2</ImageId>
<MinCount>1</MinCount>
<MaxCount>2</MaxCount>
<KeyName>condor</KeyName>
<SecurityGroup>condor</SecurityGroup></body>
</node>
<node xmi:type="uml:CallOperationAction" xmi:id="_f2rYwBXvEd-nTpeT6OuvbQ" name="RunInstances" operation="_t7YCkBXvEd-nTpeT6OuvbQ">
<argument xmi:type="uml:InputPin" xmi:id="_PVJDoBXwEd-nTpeT6OuvbQ" name="inputXML" visibility="public" incoming="_aqhAeBXwEd-nTpeT6OuvbQ">
<type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
<upperBound xmi:type="uml:LiteralInteger" xmi:id="_ATI1UBX2Ed-nTpeT6OuvbQ" value="1"/>
</argument>
<result xmi:type="uml:OutputPin" xmi:id="_OkebsBXwEd-nTpeT6OuvbQ" name="outputXML" outgoing="_I_xAiBXxEd-nTpeT6OuvbQ">
<type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
<upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_OkebsRXwEd-nTpeT6OuvbQ" value="*"/>
</result>
<target xmi:type="uml:InputPin" xmi:id="_OkgQ4BXwEd-nTpeT6OuvbQ" name="target" visibility="private" type="_ohjeMBXvEd-nTpeT6OuvbQ">
<upperBound xmi:type="uml:LiteralInteger" xmi:id="_dQiYwBX2Ed-nTpeT6OuvbQ" value="1"/>
</target>
</node>
<node xmi:type="uml:OpaqueAction" xmi:id="_7obHwBX3Ed-nTpeT6OuvbQ" name="DelayBlock10mins" outgoing="_41L_kBYBEd-nTpeT6OuvbQ" incoming="_I_xAiBXxEd-nTpeT6OuvbQ">
<body><input>{cirg:sleep(600000)}</input></body>
</node>
<node xmi:type="uml:CallBehaviorAction" xmi:id="_kJndgBYDEd-nTpeT6OuvbQ" name="setupCondor" outgoing="_F7gVkBYCEd-nTpeT6OuvbQ" incoming="_41L_kBYBEd-nTpeT6OuvbQ" behavior="_kJqg0BYDEd-nTpeT6OuvbQ"/>
<node xmi:type="uml:CallOperationAction" xmi:id="_zr8BwBdJEd-xhvA75iYy6A" name="RunInstances_1" operation="_t7YCkBXvEd-nTpeT6OuvbQ">
<argument xmi:type="uml:InputPin" xmi:id="_8kq9ABdJEd-xhvA75iYy6A" name="inputXML" incoming="_ppPYUBdKEd-xhvA75iYy6A">
<type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
<upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_8kq9ARdJEd-xhvA75iYy6A" value="*"/>
</argument>
<result xmi:type="uml:OutputPin" xmi:id="_8ktZQBdJEd-xhvA75iYy6A" name="outputXML" outgoing="_wq42IBdKEd-xhvA75iYy6A">
<type xmi:type="uml:PrimitiveType" href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>

```
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_8kuAUBdJEd-xhvA75iYy6A"
value="*"/>
      </result>
      <target xmi:type="uml:InputPin" xmi:id="_8kvOcBdJEd-xhvA75iYy6A" name="target"
type="_ohjeMBXvEd-nTpeT6OuvbQ"/>
    </node>
    <node xmi:type="uml:OpaqueAction" xmi:id="_d1KtUBdKEd-xhvA75iYy6A"
name="OpaqueAction1" outgoing="_ppPYUBdKEd-xhvA75iYy6A"/>
    <node xmi:type="uml:OpaqueAction" xmi:id="_nzewkBdKEd-xhvA75iYy6A"
name="OpaqueAction1">
      <inputValue xmi:type="uml:InputPin" xmi:id="_vJ0E8BdKEd-xhvA75iYy6A"
name="InputPin1" incoming="_wq42IBdKEd-xhvA75iYy6A">
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_vJ0E8RdKEd-xhvA75iYy6A"
value="*"/>
      </inputValue>
    </node>
    <edge xmi:type="uml:ControlFlow" xmi:id="_d2dNcBXvEd-nTpeT6OuvbQ"
name="ControlFlow1" source="_3rZH8BXrEd-nTpeT6OuvbQ" target="_-lPUoBXrEd-
nTpeT6OuvbQ">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_d2dNcRXvEd-nTpeT6OuvbQ" value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_d2dNchXvEd-nTpeT6OuvbQ" value="1"/>
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_aqhAeBXwEd-nTpeT6OuvbQ"
name="ObjectFlow1" source="_Xr2P4BXwEd-nTpeT6OuvbQ" target="_PVJDoBXwEd-
nTpeT6OuvbQ">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_aqhAeRXwEd-nTpeT6OuvbQ"
value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_aqhAehXwEd-nTpeT6OuvbQ" value="1"/>
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_I_xAiBXxEd-nTpeT6OuvbQ"
name="ObjectFlow2" source="_OkebsBXwEd-nTpeT6OuvbQ" target="_7obHwBX3Ed-
nTpeT6OuvbQ">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_I_xAiRXxEd-nTpeT6OuvbQ" value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_I_xAihXxEd-nTpeT6OuvbQ" value="1"/>
    </edge>
    <edge xmi:type="uml:ControlFlow" xmi:id="_41L_kBYBEd-nTpeT6OuvbQ"
name="ControlFlow2" source="_7obHwBX3Ed-nTpeT6OuvbQ" target="_kJndgBYDEd-
nTpeT6OuvbQ">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_41L_kRYBEd-nTpeT6OuvbQ" value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_41L_khYBEd-nTpeT6OuvbQ" value="1"/>
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_F7gVkBYCEd-nTpeT6OuvbQ"
name="ObjectFlow3" source="_kJndgBYDEd-nTpeT6OuvbQ" target="_6QQRYBXrEd-
nTpeT6OuvbQ">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_F7gVkRYCEd-nTpeT6OuvbQ"
value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_F7gVkhYCEd-nTpeT6OuvbQ" value="1"/>
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_ppPYUBdKEd-xhvA75iYy6A"
name="ObjectFlow4" source="_d1KtUBdKEd-xhvA75iYy6A" target="_8kq9ABdJEd-
xhvA75iYy6A">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_ppPYURdKEd-xhvA75iYy6A" value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_ppPYUhdKEd-xhvA75iYy6A" value="1"/>
```

```
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_wq42IBdKEd-xhvA75iYy6A" name="ObjectFlow5"
source="_8ktZQBdJEd-xhvA75iYy6A" target="_vJ0E8BdKEd-xhvA75iYy6A">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_wq42IRdKEd-xhvA75iYy6A" value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_wq42IhdKEd-xhvA75iYy6A" value="1"/>
    </edge>
  </packagedElement>
  <packagedElement xmi:type="sysML:Block" xmi:id="_ohjeMBXvEd-nTpeT6OuvbQ"
name="AmazonEC2">
    <ownedOperation xmi:type="uml:Operation" xmi:id="_t7YCkBXvEd-nTpeT6OuvbQ"
name="RunInstances">
      <ownedParameter xmi:type="uml:Parameter" xmi:id="_zA5K4BXvEd-nTpeT6OuvbQ"
name="inputXML">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
      </ownedParameter>
      <ownedParameter xmi:type="uml:Parameter" xmi:id="_6S2VkBXvEd-nTpeT6OuvbQ"
name="outputXML" direction="out">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
      </ownedParameter>
    </ownedOperation>
    <ownedOperation xmi:type="uml:Operation" xmi:id="_sTJDABX7Ed-nTpeT6OuvbQ"
name="DescribeInstances">
      <ownedParameter xmi:type="uml:Parameter" xmi:id="_0Z0PMBX7Ed-nTpeT6OuvbQ"
name="inputXML">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
      </ownedParameter>
      <ownedParameter xmi:type="uml:Parameter" xmi:id="_UznoEBX8Ed-nTpeT6OuvbQ"
name="outputXML" direction="out">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
      </ownedParameter>
    </ownedOperation>
  </packagedElement>
  <packagedElement xmi:type="uml:Activity" xmi:id="_kJqg0BYDEd-nTpeT6OuvbQ"
name="Main3">
    <node xmi:type="uml:InitialNode" xmi:id="_p7kOABYDEd-nTpeT6OuvbQ"
name="InitialNode1" outgoing="_67caEBYEEd-nTpeT6OuvbQ"/>
    <node xmi:type="uml:ActivityFinalNode" xmi:id="_sBGXIBYDEd-nTpeT6OuvbQ"
name="ActivityFinalNode1" incoming="_3XGxwBbiEd-apKQ1plajsw"/>
    <node xmi:type="uml:CallOperationAction" xmi:id="_Lps8kBYEEd-nTpeT6OuvbQ"
name="DescribeInstances" operation="_sTJDABX7Ed-nTpeT6OuvbQ">
      <argument xmi:type="uml:InputPin" xmi:id="_ZODSYBYEEd-nTpeT6OuvbQ"
name="inputXML" incoming="_QMflsBbdEd-apKQ1plajsw">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_ZODSYRYEEd-
nTpeT6OuvbQ" value="*"/>
      </argument>
      <result xmi:type="uml:OutputPin" xmi:id="_ZOFuoBYEEd-nTpeT6OuvbQ"
name="outputXML" outgoing="_xJ28gBYKEd-nTpeT6OuvbQ">
```

```
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_ZOFuoRYEEd-
nTpeT6OuvbQ" value="*"/>
      </result>
      <target xmi:type="uml:InputPin" xmi:id="_ZOG8wBYEEd-nTpeT6OuvbQ" name="target"
type="_ohjeMBXvEd-nTpeT6OuvbQ"/>
    </node>
    <node xmi:type="uml:CallOperationAction" xmi:id="_f8DzwBYKEd-nTpeT6OuvbQ"
name="SSH" operation="_MSP4YBYGEd-nTpeT6OuvbQ">
      <argument xmi:type="uml:InputPin" xmi:id="_DdQ-oBZcEd--GfqQ0FyhCA"
name="inputXML" incoming="_H3HTwBZcEd--GfqQ0FyhCA">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_DdQ-oRZcEd--
GfqQ0FyhCA" value="*"/>
      </argument>
      <result xmi:type="uml:OutputPin" xmi:id="_DdYTYBZcEd--GfqQ0FyhCA"
name="outputXML" outgoing="_3XGxwBbiEd-apKQ1plajsw">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_DdYTYRZcEd--
GfqQ0FyhCA" value="*"/>
      </result>
      <target xmi:type="uml:InputPin" xmi:id="_Dddy8BZcEd--GfqQ0FyhCA" name="target"
type="_BYSFMBYGEd-nTpeT6OuvbQ"/>
    </node>
    <node xmi:type="uml:OpaqueAction" xmi:id="_jwLloBYKEd-nTpeT6OuvbQ"
name="createSSHInput">
      <inputValue xmi:type="uml:InputPin" xmi:id="_q8zkYBYKEd-nTpeT6OuvbQ"
name="inputXML" incoming="_up1CyBaZEd--GfqQ0FyhCA">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_q8zkYRYKEd-
nTpeT6OuvbQ" value="*"/>
      </inputValue>
      <outputValue xmi:type="uml:OutputPin" xmi:id="_2qxjkBZbEd--GfqQ0FyhCA"
name="outputXML" outgoing="_H3HTwBZcEd--GfqQ0FyhCA">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_2qyKoBZbEd--
GfqQ0FyhCA" value="*"/>
      </outputValue>

<body>&lt;input>&#xD;&#xA;{($DescribeInstancesResponse//PublicDnsName)[1]/text()}&#xD;&#
xA;&lt;ShellCommand>unzip condor.zip &amp;amp;&amp;amp;
./condor2.sh&lt;/ShellCommand>&#xD;&#xA;&lt;KeyFile>D:/vikas/891
RESEARCH/condor.pem&lt;/KeyFile>&#xD;&#xA;&lt;User>root&lt;/User>&#xD;&#xA;&lt;/input></
body>
    </node>
    <node xmi:type="uml:OpaqueAction" xmi:id="_UYpqcBaYEd--GfqQ0FyhCA"
name="createSFTPInput">
```

```
      <inputValue xmi:type="uml:InputPin" xmi:id="_UYpqcRaYEd--GfqQ0FyhCA"
name="inputXML" incoming="_xJ28gBYKEd-nTpeT6OuvbQ">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_UYpqchaYEd--
GfqQ0FyhCA" value="*"/>
      </inputValue>
      <outputValue xmi:type="uml:OutputPin" xmi:id="_UYpqcxaYEd--GfqQ0FyhCA"
name="outputXML" outgoing="_6cWg4BbhEd-apKQ1plajsw">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_UYpqdBaYEd--
GfqQ0FyhCA" value="*"/>
      </outputValue>

<body>&lt;HostName>{($DescribeInstancesResponse//PublicDnsName)[1]/text()}&lt;/HostName>
&#xD;&#xA;&lt;HostName>{($DescribeInstancesResponse//PublicDnsName)[2]/text()}&lt;/HostN
ame>&#xD;&#xA;&lt;ShellCommand>unzip condor.zip &amp;amp;&amp;amp;
./condor2.sh&lt;/ShellCommand>&#xD;&#xA;&lt;KeyFile>D:/vikas/891
RESEARCH/condor.pem&lt;/KeyFile>&#xD;&#xA;&lt;LocalFileName>D:/vikas/891
RESEARCH/condor-
ec2/condor.zip&lt;/LocalFileName>&#xD;&#xA;&lt;RemotePath>/root&lt;/RemotePath>&#xD;&#x
A;&lt;Permissions>0777&lt;/Permissions>&#xD;&#xA;&lt;User>root&lt;/User></body>
    </node>
    <node xmi:type="uml:CallOperationAction" xmi:id="_bi6cQBaZEd--GfqQ0FyhCA"
name="startSFTPSerialConnections" operation="_B9zQQBYKEd-nTpeT6OuvbQ">
      <argument xmi:type="uml:InputPin" xmi:id="_fA1UoBaZEd--GfqQ0FyhCA"
name="inputXML" incoming="_6cWg4BbhEd-apKQ1plajsw">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_fA1UoRaZEd--
GfqQ0FyhCA" value="*"/>
      </argument>
      <result xmi:type="uml:OutputPin" xmi:id="_fA3J0BaZEd--GfqQ0FyhCA" name="outputXML"
outgoing="_up1CyBaZEd--GfqQ0FyhCA">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_fA3J0RaZEd--GfqQ0FyhCA"
value="*"/>
      </result>
      <target xmi:type="uml:InputPin" xmi:id="_fA4X8BaZEd--GfqQ0FyhCA" name="target"
type="_BYSFMBYGEd-nTpeT6OuvbQ"/>
    </node>
    <node xmi:type="uml:OpaqueAction" xmi:id="_GWgNcBbcEd-apKQ1plajsw"
name="createDescribeInstancesInput" incoming="_67caEBYEEd-nTpeT6OuvbQ">
      <outputValue xmi:type="uml:OutputPin" xmi:id="_6AqcsBbcEd-apKQ1plajsw"
name="outputXML" outgoing="_QMflsBbdEd-apKQ1plajsw">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
        <upperBound xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_6AqcsRbcEd-apKQ1plajsw"
value="*"/>
      </outputValue>
    </node>
```

```
    <edge xmi:type="uml:ControlFlow" xmi:id="_67caEBYEEd-nTpeT6OuvbQ"
name="ControlFlow1" source="_p7kOABYDEd-nTpeT6OuvbQ" target="_GWgNcBbcEd-
apKQ1plajsw">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_67caERYEEd-nTpeT6OuvbQ"
value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_67caEhYEEd-nTpeT6OuvbQ" value="1"/>
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_xJ28gBYKEd-nTpeT6OuvbQ"
name="ObjectFlow1" source="_ZOFuoBYEEd-nTpeT6OuvbQ" target="_UYpqcRaYEd--
GfqQ0FyhCA">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_xJ28gRYKEd-nTpeT6OuvbQ" value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_xJ28ghYKEd-nTpeT6OuvbQ" value="1"/>
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_H3HTwBZcEd--GfqQ0FyhCA"
name="ObjectFlow2" source="_2qxjkBZbEd--GfqQ0FyhCA" target="_DdQ-oBZcEd--
GfqQ0FyhCA">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_H3HTwRZcEd--GfqQ0FyhCA"
value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_H3HTwhZcEd--GfqQ0FyhCA" value="1"/>
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_up1CyBaZEd--GfqQ0FyhCA"
name="ObjectFlow4" source="_fA3J0BaZEd--GfqQ0FyhCA" target="_q8zkYBYKEd-
nTpeT6OuvbQ">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_up1CyRaZEd--GfqQ0FyhCA" value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_up1CyhaZEd--GfqQ0FyhCA" value="1"/>
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_QMflsBbdEd-apKQ1plajsw" name="ObjectFlow5"
source="_6AqcsBbcEd-apKQ1plajsw" target="_ZODSYBYEEd-nTpeT6OuvbQ">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_QMflsRbdEd-apKQ1plajsw" value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_QMflshbdEd-apKQ1plajsw" value="1"/>
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_6cWg4BbhEd-apKQ1plajsw"
name="ObjectFlow3" source="_UYpqcxaYEd--GfqQ0FyhCA" target="_fA1UoBaZEd--
GfqQ0FyhCA">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_6cWg4RbhEd-apKQ1plajsw" value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_6cWg4hbhEd-apKQ1plajsw" value="1"/>
    </edge>
    <edge xmi:type="uml:ObjectFlow" xmi:id="_3XGxwBbiEd-apKQ1plajsw" name="ObjectFlow6"
source="_DdYTYBZcEd--GfqQ0FyhCA" target="_sBGXIBYDEd-nTpeT6OuvbQ">
      <guard xmi:type="uml:LiteralBoolean" xmi:id="_3XGxwRbiEd-apKQ1plajsw" value="true"/>
      <weight xmi:type="uml:LiteralInteger" xmi:id="_3XGxwhbiEd-apKQ1plajsw" value="1"/>
    </edge>
  </packagedElement>
  <packagedElement xmi:type="sysML:Block" xmi:id="_BYSFMBYGEd-nTpeT6OuvbQ"
name="SSH-SFTP">
    <ownedOperation xmi:type="uml:Operation" xmi:id="_MSP4YBYGEd-nTpeT6OuvbQ"
name="startSSHSerialConnections">
      <ownedComment xmi:type="uml:Comment" xmi:id="_AlwfQBYKEd-nTpeT6OuvbQ">
        <body>inputXML&#xD;
&#xD;
&lt;HostName>&#xD;
&lt;UserName>&#xD;
&lt;KeyFile>&#xD;
```

```
&lt;KeyFilePass&gt;&#xD;
&lt;ShellCommand&gt;</body>
      </ownedComment>
      <ownedParameter xmi:type="uml:Parameter" xmi:id="_t8kBcBYHEd-nTpeT6OuvbQ"
name="inputXML">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
      </ownedParameter>
      <ownedParameter xmi:type="uml:Parameter" xmi:id="_yKy8cBYHEd-nTpeT6OuvbQ"
name="outputXML" direction="out">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
      </ownedParameter>
    </ownedOperation>
    <ownedOperation xmi:type="uml:Operation" xmi:id="_B9zQQBYKEd-nTpeT6OuvbQ"
name="startSFTPSerialConnections">
      <ownedParameter xmi:type="uml:Parameter" xmi:id="_B9zQQRYKEd-nTpeT6OuvbQ"
name="inputXML">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
      </ownedParameter>
      <ownedParameter xmi:type="uml:Parameter" xmi:id="_B9zQQhYKEd-nTpeT6OuvbQ"
name="outputXML" direction="out">
        <type xmi:type="uml:PrimitiveType"
href="pathmap://UML_LIBRARIES/UMLPrimitiveTypes.library.uml#String"/>
      </ownedParameter>
    </ownedOperation>
    <ownedOperation xmi:type="uml:Operation" xmi:id="_YDoisBZ9Ed--GfqQ0FyhCA"
name="startSSHParallelConnections">
      <ownedComment xmi:type="uml:Comment" xmi:id="_YDoisRZ9Ed--GfqQ0FyhCA">
        <body>inputXML&#xD;
&#xD;
&lt;HostName&gt;&#xD;
&lt;UserName&gt;&#xD;
&lt;KeyFile&gt;&#xD;
&lt;KeyFilePass&gt;&#xD;
&lt;ShellCommand&gt;</body>
      </ownedComment>
      <ownedParameter xmi:type="uml:Parameter" xmi:id="_YDoishZ9Ed--GfqQ0FyhCA"
name="inputXML"/>
      <ownedParameter xmi:type="uml:Parameter" xmi:id="_YDoisxZ9Ed--GfqQ0FyhCA"
name="outputXML" direction="out"/>
    </ownedOperation>
  </packagedElement>
 </packagedElement>
 <profileApplication xmi:type="uml:ProfileApplication" xmi:id="idProfileApplication">
  <eAnnotations xmi:type="ecore:EAnnotation" xmi:id="idProfileAnnotation"
source="http://www.eclipse.org/uml2/2.0.0/UML">
    <references xmi:type="ecore:EPackage"
href="../../../plugin/org.topcased.modeler.sysml/templates/activitydiagram/profile/SysMLActivityExt
ensionsProfile.uml#ProfileContentId"/>
  </eAnnotations>
```

```
    <appliedProfile xmi:type="uml:Profile"
href="../../../plugin/org.topcased.modeler.sysml/templates/activitydiagram/profile/SysMLActivityExt
ensionsProfile.uml#ActivityProfileId"/>
  </profileApplication>
</uml:Model>
```

# Appendix B

## SWE Java Classes

This section contains some of the important classes of the SysFlow workflow Engine.

### Main.java

```
package edu.clemson;
import java.io.IOException;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.Namespace;
import org.jdom.Attribute;
import java.util.List;
import java.util.Iterator;


public class Main extends SysMLFlowModel{

	private SAXBuilder builder=null;
	private Element activityList[]=null;
	private Element blockList[]=null;

	public static void main(String[] arg)
	{
		try
		{
			Main mainProcess=new Main();
			String filename;
			if(arg!=null && arg.length>0)
				filename=arg[0];
			else
			{
				System.out.println("ERROR: SWE Workflow not
specified");
				return;
			}

			mainProcess.startProcessing(filename);
		}
		catch(JDOMException e)
		{
			e.printStackTrace();
		}
		catch(IOException e)
		{
			e.printStackTrace();
```

```
            }
            catch(Exception e)
            {
                  e.printStackTrace();
            }

      }

      private void startProcessing(String filename) throws
JDOMException,IOException,Exception
      {
            builder=new SAXBuilder();
            Document doc=builder.build(filename);
            Element rootElement=doc.getRootElement();
            setCorrectXMIVersion(rootElement);
            blockList=getPackagedElement(rootElement,"sysML:Block");

      activityList=getPackagedElement(rootElement,"uml:Activity");

            if(activityList==null && blockList==null)
                  System.out.println("Severe Error: Workflow is in
incorrect format");
            else
                  System.out.println("Success");

            SysMLFlowBlock blockAnalysis=new
SysMLFlowBlock(blockList,getXMIVersion());
            SysMLFlowActivity activityAnalyzer=new
SysMLFlowActivity(activityList,getXMIVersion());


      activityAnalyzer.startProcessing(blockAnalysis.getOperationHashMa
p());

            //activityAnalyzer.startProcessing(activity);

      //    listChildren(rootElement,0);
            //System.out.println("The element is
"+rootElement.getText());
      }

      private Element[] getPackagedElement(Element rootElement, String
nameWithNamespace)
      {
            List list=
rootElement.getChild("packagedElement").getChildren("packagedElement");
            Iterator iterator=list.iterator();
            Element result[]=new Element[list.size()];
            int resultCounter=0;

            while(iterator.hasNext())
            {
                  Element child=(Element)iterator.next();
```

```
		if(child.getAttributeValue("type",getXMINamespace()).equals(nameW
ithNamespace))
				{
					//System.out.println("Found PackagedElement");
					result[resultCounter]=child;
					resultCounter++;
				}
			}

			return result;

		}
}
```

## SysMLFlowActivity.java

```
package edu.clemson;
import org.jdom.Document;
import org.jdom.Element;
import org.jdom.Namespace;
import org.jdom.JDOMException;

import edu.clemson.eucalyptus.EucalyptusWrapper;
import edu.clemson.AmazonEC2.AmazonEC2Wrapper;
import edu.clemson.ssh.SSHManager;

import com.amazonaws.ec2.AmazonEC2Exception;

import java.util.List;
import java.util.HashMap;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Map;
import java.io.IOException;


public class SysMLFlowActivity extends SysMLFlowModel
{

		private Element activityList[]=null;
		private HashMap<String,ArrayList<Element>> activityMap=new
HashMap<String,ArrayList<Element>>();
		private HashMap<String,String> operationHashMap=null;
		private SysmlFlowActivityUtility activityUtility;

		private CurrentOutput currentOutput=new CurrentOutput();
		//private Map<String,String> outputValueMap=new
HashMap<String,String>();
		//private String currentOutputKey;
```

```
	SysMLFlowActivity(Element[] activityList, String XMIversion)
	{

		System.out.println("Activity Analysis...");
		this.activityList=activityList;
		this.setXMIVersion(XMIversion);
		activityUtility=new
SysmlFlowActivityUtility(getXMIVersion());
		handleMultipleActivityIntances();
	}


	public void handleMultipleActivityIntances()
	{
		for(Element activity:activityList)
		{
			if(activity==null) break;

			//To ensure that only the activity having the name
"Main" is indexed.

	//if(activity.getAttributeValue("name").equals("Main"))
				indexNodes(activity);
		}
	}

	/*
	 * nodes (elements) indexed against their "incoming" property
	 */
	public void indexNodes(Element activity)
	{
		String activityName=activity.getAttributeValue("name");
		List list= activity.getChildren();
		Iterator iterator=list.iterator();

		//String initialNodeOutgoing=null;
		boolean first=true,isMainActivity=false;

		if(activityName.equals("Main")) isMainActivity=true;

		while(iterator.hasNext())
		{
		   Element element=(Element)iterator.next();
		   String incoming=element.getAttributeValue("incoming");
		   String
type=element.getAttributeValue("type",getXMINamespace());

		  if(!isMainActivity &&
type.split(":")[1].equals("InitialNode"))
		   {
			ArrayList<Element> arrayList=new
ArrayList<Element>();
			arrayList.add(element);
```

```
      activityMap.put(activity.getAttributeValue("id",getXMINamespace()
), arrayList);
                  activityMap.put(incoming, arrayList);
               }

                // Detect initial node of main activity
                if (incoming==null && element.getName().equals("node")
&& isMainActivity
                              && type!=null &&
type.split(":")[1].equals("InitialNode"))
                {
                     // first=false;
                      incoming="InitialNode";
                      ArrayList<Element> arrayList=new
ArrayList<Element>();
                      arrayList.add(element);
                      activityMap.put(incoming, arrayList);

                      String nodeType=element.getAttributeValue("type",
getXMINamespace());
                      String temp[]=nodeType.split(":", 2);
                      //System.out.println("(index nodes)node name is:
"+temp[1]);

                     //
initialNodeOutgoing=element.getAttributeValue("outgoing");
                }
                else if(incoming==null &&
element.getName().equals("edge"))
                {
                     String id=element.getAttributeValue("id",
getXMINamespace());

                   //  if(initialNodeOutgoing!=null &&
initialNodeOutgoing.equals(id))
                   //      id="InitialNode";

                     String edgeId="(edge)"+id;

                     ArrayList<Element> arrayList=new
ArrayList<Element>();
                     arrayList.add(element);
                     activityMap.put(edgeId, arrayList);

                }
                else //other nodes
                {

                     if(incoming!=null)
                     {
                            String tempIncomings[]=incoming.split(" ");
                            int size=tempIncomings.length;
```

```
                        for(int i=0;i<size;i++) //if there are 2
incomings for a single element, make them 2 seperate keys
                        {
                              ArrayList<Element>
storedArrayList=activityMap.get(tempIncomings[i]);
                              if(storedArrayList==null)  //key is
not already existing, so create a new key a element to the map
                              {
                                    ArrayList<Element> arrayList=new
ArrayList<Element>();
                                    arrayList.add(element);

activityMap.put(tempIncomings[i], arrayList);
                              }
                               else
                               {
                                    storedArrayList.add(element);

activityMap.remove(tempIncomings[i]);

activityMap.put(tempIncomings[i], storedArrayList);
                               }
                        }
                  }

                  //If we detect a subprocess ... index its contents
                  if(element.getName().equals("ownedBehavior") &&
element.getAttributeValue("type", getXMINamespace())!=null &&
                              (element.getAttributeValue("type",
getXMINamespace()).split(":"))[1].equals("Activity") )
                  {
                      System.out.println("inside owned");
                        indexNodes(element);
                  }

                  /**
                   * get inside callOperation action to index
"inputPin"(argument), "outputPin (result)"
                   */
                  String nodeType=((element.getAttributeValue("type",
getXMINamespace())).split(":", 2))[1];
                  if(nodeType.equals("CallOperationAction"))
                  {
                        List<Element>
argList=element.getChildren("argument");
                        if(argList==null) return;

                        for(int i=0;i<argList.size();i++)
                        {
                              Element arg=(Element)argList.get(i);
                              String
argIncoming=arg.getAttributeValue("incoming");

                              if(argIncoming==null)
```

```
                                    continue;

                                String temp[]=argIncoming.split(" ");
                                  for(int j=0;j<temp.length;j++)
                                  {

                                        ArrayList<Element>
arrayList=activityMap.get(temp[j]);
                                        if(arrayList==null)
                                              arrayList=new
ArrayList<Element>();

                                        arrayList.add(arg);
                                        arrayList.add(element); // for
every input pin, we also send its parent callOperationAction
                                        activityMap.put(temp[j],
arrayList);
                                   }
                              }
                         }
                        else if(nodeType.equals("OpaqueAction"))
                        {
                              List<Element>
argList=element.getChildren("inputValue");
                              if(argList==null) return;

                              for(int i=0;i<argList.size();i++)
                              {
                                    Element arg=(Element)argList.get(i);
                                    String
argIncoming=arg.getAttributeValue("incoming");

                                    if(argIncoming==null)
                                        continue;

                                String temp[]=argIncoming.split(" ");
                                  for(int j=0;j<temp.length;j++)
                                  {

                                        ArrayList<Element>
arrayList=activityMap.get(temp[j]);
                                        if(arrayList==null)
                                              arrayList=new
ArrayList<Element>();

                                        arrayList.add(arg);
                                        arrayList.add(element); // for
every input pin, we also send its parent callOperationAction
                                        activityMap.put(temp[j],
arrayList);
                                   }
                              }
                        }
                } //end of while
```

```
		} //end of else
	}

	public void startProcessing(HashMap operationHashMap)
	{
		this.operationHashMap=operationHashMap;
		String incoming="InitialNode";
		startProcessing2(incoming);
	}

	private void startProcessing2(String incoming)
	{
		String outgoing=null;

			//step 1: Get the nodes(elements) mapped against the
incoming (key) and process it
			ArrayList<Element>
elementArray=activityMap.get(incoming);
			ArrayList<String> outgoingArray=new
ArrayList<String>();


		if(elementArray==null)
		{
		  System.out.println("ERROR: "+incoming +" not found in
activityMap hashMap...Ensure that it is mapped");
			return;
		}

				for(int i=0;i<elementArray.size();i++)
				{
					Element
element=(Element)elementArray.get(i);

					//only node and input node can have
"incoming"

if(activityUtility.isArgumentInputPin(element)||activityUtility.isInput
ValuePin(element))
					{
						i++; //because for every Input pin,
we store its parent CallOperationAction in the next location.
					Element
parent=(Element)elementArray.get(i);
						outgoing=elementHandler(parent);
					}
					else
					{
						outgoing=elementHandler(element);
						//
System.out.println("outgoingId:"+outgoing);
					}

					if(outgoing!=null)
```

```
                        {
                                String temp[]=outgoing.split(" ");
                                  for(int j=0;j<temp.length;j++)
                                        outgoingArray.add(temp[j]);
                        }

                        }

                  //step 2: look for outgoing property to select the
next node,
                  for(int i=0;i<outgoingArray.size();i++)
                  {
                        outgoing=outgoingArray.get(i);
                        incoming=outgoing;

                        if(incoming!=null)
                              startProcessing2(incoming);
                  }

      }

      private String elementHandler(Element element)
      {
            String name=element.getName();
            String outgoingId=null;

            if(name.equals("node"))
                  outgoingId=nodeHandler(element);

      /*    else if(name.equals("ownedParameter"))
            {
                System.out.println("[%%%%%%%%%%%%5%%%%%%%%%%%%%%%]");
                  ownedParameterHandler(element);
            }
            else if(name.equals("edge"))
            {
                  System.out.println("[%%%%%%%%%%%%5%%%%%%%%%%%%%%%]");
                  edgeHandler(element);
            }
            else if(name.equals("argument"))
            {
                  System.out.println("[%%%%%%%%%%%%5%%%%%%%%%%%%%%%]");
                  argumentHandler(element);
            }
            else if (name.equals("result"))
            {
                  System.out.println("[%%%%%%%%%%%%5%%%%%%%%%%%%%%%]");
                  resultHandler(element);
            }*/
            else
            {
                  System.out.println("Error: Token "+name+" not found
missed analysis.....");
            }
```

```
            return outgoingId;
      }
      private String nodeHandler(Element node)
      {
            String nodeType=node.getAttributeValue("type",
getXMINamespace());
            String outgoingId=node.getAttributeValue("outgoing");
            String childNodeOutgoingId=null;

            String temp[]=nodeType.split(":", 2);
            String prefix=temp[0];
            String nodeName=temp[1];

            if(nodeName.equals("InitialNode"))
                  initialNodeHandler(node);
            else if(nodeName.equals("CallOperationAction"))
                  childNodeOutgoingId=callOperationActionHandler(node);
            else if(nodeName.equals("ActivityFinalNode"))
                ActivityFinalNodeHandler(node);
            else if(nodeName.equals("OpaqueAction"))
                  childNodeOutgoingId=OpaqueActionNodeHandler(node);
            else if(nodeName.equals("DecisionNode"))
                   DecisionNodeHandler(node);
            else if(nodeName.equals("CallBehaviorAction"))
                  CallBehaviorActionHandler(node);
            else if(nodeName.equals("ownedBehavior"))
                  System.out.println("Owned Behavior...not
implemented");

            if(childNodeOutgoingId==null)
                  return outgoingId;
            else
                  return childNodeOutgoingId;

      }
      private void ownedParameterHandler(Element ownedParameter)
      {
            String nodeType=ownedParameter.getAttributeValue("type",
getXMINamespace());
            String temp[]=nodeType.split(":", 2);
            String prefix=temp[0];
            String nodeName=temp[1];

            if(nodeName.equals("Parameter"))
                  parameterHandler(ownedParameter);
      }

      private void edgeHandler(Element edge)
      {

            String nodeType=edge.getAttributeValue("type",
getXMINamespace());
            //System.out.println("node type"+nodeType);
```

```
            String temp[]=nodeType.split(":", 2);
            String prefix=temp[0];
            String nodeName=temp[1];

            if(nodeName.equals("ControlFlow"))
                  EdgeControlFlowHandler(edge);
            else if(nodeName.equals("ObjectFlow"))
                  EdgeObjectFlowHandler(edge);

      }
      private void argumentHandler(Element inputPin)
      {

      }
      private void resultHandler(Element inputPin)
      {

      }
      private void initialNodeHandler(Element node)
      {
            System.out.println("[ACTIVITY]: starting initial node
analysis:");
      }

      private String callOperationActionHandler(Element node)
      {
            //Get the input argument information from the
callOperationActionNode
            List<Element>
argumentNodeList=node.getChildren("argument");
            Map<String,String> valueArgumentMap=new
HashMap<String,String>();

            for(Element argument:argumentNodeList)
            {

if(argument.getAttributeValue("type",getXMINamespace()).equals(getUMLNa
mespace().getPrefix()+":"+"ValuePin"));
              {
                  String name=argument.getAttributeValue("name");
                  Element valueTmp=argument.getChild("value");
                  if(valueTmp==null)
                        break;

                  String value=valueTmp.getAttributeValue("value");
                  valueArgumentMap.put(name, value);
              }
            }

            String nodeName=node.getAttributeValue("name");
            System.out.println("[CallOperationAction]: Call operation
action handler "+nodeName);
            String operationId=node.getAttributeValue("operation");
```

```
            String operationFullName=operationHashMap.get(operationId);

            if(operationFullName==null)
            {
                  System.out.println("\nError: Could not find the
operation name in block definition diagram.\n");
                  return null;
            }

            String[] temp=operationFullName.split(":",2);
            String blockName=temp[0];
            String operationName=temp[1];


            Object
obj=getExecutingObject(blockName,operationName,valueArgumentMap);
            //ArrayList result=null;
            String result;

            if(obj instanceof EucalyptusWrapper)
            { try
                  {
                     System.out.println("Eucalyptus wrapper");

result=((EucalyptusWrapper)obj).runEucalyptusCommand(operationName);
                     System.out.println("The result is:\n"+result);

                   /*  if(result!=null)
                     {   for(int i=0;i<result.size();i++)
                           {
                                 System.out.println(result.get(i));
                           }
                     }*/
                   }
                catch(Exception e)
                {
                      e.printStackTrace();
                }
            }
            else if(obj instanceof SSHManager)
            {
                  System.out.println("SSH Manager...");
                  try
                  {
                        ((SSHManager)obj).startSession();
                  }
                  catch(JDOMException e)
                  {
                        e.printStackTrace();
                  }
                  catch(IOException e)
                  {
                        e.printStackTrace();
                  }
```

```
                  catch(Exception e)
                  {
                        e.printStackTrace();
                  }
            }
            else if(obj instanceof AmazonEC2Wrapper)
            {
                  String amazonEC2Output = null;
                  System.out.println("Amazon EC2 processing");

                  try
                  {
                        String name=node.getAttributeValue("name");
                        amazonEC2Output=((AmazonEC2Wrapper)
obj).runAmazonService();

                        if(amazonEC2Output!=null)

      currentOutput.putOutputValue(name,amazonEC2Output);
                        else
                              System.out.println("Warning: recieved
NULL response from Amazon Service");
                  }
                  catch(JDOMException e)
                  {
                        e.printStackTrace();
                  }
                  catch(IOException e)
                  {
                        e.printStackTrace();
                  }
                  catch(AmazonEC2Exception e)
                  {
                        e.printStackTrace();
                  }
                  catch(Exception e)
                  {
                        e.printStackTrace();
                  }

                  if(amazonEC2Output==null)
                        System.out.println("WARNING: The output
variable is NULL");
            }


            //If the outgoing node exists, get the outgoing id
            List<Element> allchildrenList=node.getChildren();
            for(Element child:allchildrenList)
            {
                  System.out.println(child.getName()+"
"+child.getAttribute("type",getXMINamespace()));
```

```
      if(child.getAttributeValue("type",getXMINamespace()).equals("uml"
/*getUMLNamespace().getPrefix()*/+":"+"OutputPin"))
                  {
                        String
outgoing=child.getAttributeValue("outgoing");
                        return outgoing;
                  }
            }

            return null;

      }
      private void CallBehaviorActionHandler(Element node)
      {

            System.out.println("[ACTIVITY]: Call Behavior action
..implementation pending");
            String behavior=node.getAttributeValue("behavior");
            startProcessing2(behavior);

      }
      private void ActivityFinalNodeHandler(Element node)
      {
            System.out.println("[ACTIVITY]: Workflow finished
execution");
      }
      private String OpaqueActionNodeHandler(Element node)
      {
            System.out.println("[ACTIVITY]: Call opaque action
handler");

            String name=node.getAttributeValue("name");
            Element body=node.getChild("body");

            if(body!=null)
            {
                  String xmlData=body.getText();
                  xmlData=xmlData.replaceAll("&lt;", "<");
                  xmlData=xmlData.replaceAll("&#xA;", "");
                  xmlData="<"+name+">"+xmlData+"</"+name+">";


                  XPathParser xpathParser=new
XPathParser(currentOutput);
                  try{
                  Document doc=xpathParser.getDocumentObject(xmlData);
                  Element root=doc.getRootElement();
                  xmlData=xpathParser.processXMLData(root);
            //    System.out.println(xmlData);
                  }
                  catch(Exception e)
                  {
```

```
                        System.out.println("ERROR: Check if your input
XML is correct for opaque action ");
                        e.printStackTrace();

                  }
                currentOutput.putOutputValue(name, xmlData);

            }
            else
                  System.out.println(" warning: Empty opaque action");

            String outgoing=null;

            //Ideally there will be just one outputValue. However
taking care of situtation where there are multiple output value.
            List<Element>
outputValueList=node.getChildren("outputValue");
            for(Element outputValue:outputValueList)
            {
                  outgoing=outputValue.getAttributeValue("outgoing");
                  if(outgoing!=null)
                        return outgoing;
            }

            if(outgoing==null)
            {
                  outgoing=node.getAttributeValue("outgoing");
                  if(outgoing!=null)
                        return outgoing;
            }

            return null;
      }
      private void DecisionNodeHandler(Element node)
      {
            System.out.println("[ACTIVITY]: Decision node");
      }

      private void parameterHandler(Element parameter)
      {

      }
      private void EdgeControlFlowHandler(Element element)
      {
            System.out.println("[EDGE]: Control Edge
"+element.getAttributeValue("name"));
      }
      private void EdgeObjectFlowHandler(Element element)
      {
            System.out.println("[EDGE]: Object Edge
"+element.getAttributeValue("name"));
      }
      private Object getExecutingObject(String blockName,String
operationName,Map<String,String> valueArgumentMap)
```

```
	{

		if(blockName.equals("Eucalyptus"))
		{
			EucalyptusWrapper eucalyptus=new EucalyptusWrapper();
			return eucalyptus;
		}
		else if(blockName.equals("AmazonEC2"))
		{
			AmazonEC2Wrapper amazonEC2 =new
AmazonEC2Wrapper(operationName,valueArgumentMap,currentOutput);
			return amazonEC2;
		}
		else if(blockName.equals("SSH-SFTP"))
		{
			SSHManager sshManager=new
SSHManager(operationName,currentOutput);
			return sshManager;
		}
		return null;
	}

}
```

## SysmlFlowActivityUtility.java

```
package edu.clemson;
import org.jdom.Element;


/**
 * This class holds the utility methods for a Activity
 * @author vikasp
 *
 */

public class SysmlFlowActivityUtility extends SysMLFlowModel
{

	SysmlFlowActivityUtility(String XMIversion)
	{
		this.setXMIVersion(XMIversion);
	}


	public boolean isCallOperationAction(Element element)
	{

		String name=element.getName();
```

```
            if(!name.equals("node"))
                  return false;

            String nodeType=element.getAttributeValue("type",
getXMINamespace());

            String temp[]=nodeType.split(":", 2);
            String nodeName=temp[1];
            if(nodeName.equals("CallOperationAction"))
                  return true;

            return false;
      }

      public String
getOutgoingFromResultNodeOfCallOperationAction(Element element) throws
NullPointerException
      {
         if(!isCallOperationAction(element))
               return null;

         Element result=element.getChild("result");
         String outgoing=result.getAttributeValue("outgoing");
         return outgoing;
      }


      public boolean isArgumentInputPin(Element element)
      {
        String name=element.getName();
        if(!name.equals("argument"))
              return false;

        String nodeType=element.getAttributeValue("type",
getXMINamespace());

            String temp[]=nodeType.split(":", 2);
            String nodeName=temp[1];

      if(nodeName.equals("InputPin")||nodeName.equals("ValuePin"))
                  return true;

            return false;


      }
      public boolean isInputValuePin(Element element)
      {
        String name=element.getName();
        if(!name.equals("inputValue"))
              return false;

        String nodeType=element.getAttributeValue("type",
getXMINamespace());
```

```
            String temp[]=nodeType.split(":", 2);
            String nodeName=temp[1];
            if(nodeName.equals("InputPin"))
                  return true;

            return false;


      }


}
```

## SysMLFlowBlock.java

```
package edu.clemson;

import org.jdom.Element;
import org.jdom.Namespace;

import java.util.HashMap;
import java.util.Iterator;
import java.util.List;

public class SysMLFlowBlock extends SysMLFlowModel
{

      private Element blockList[]=null;
      private HashMap<String,String> hashMap=new
HashMap<String,String>();

      SysMLFlowBlock(Element[] blockList,String XMIVersion)
      {
            this.blockList=blockList;
            this.setXMIVersion(XMIVersion);
            startProcessing();
      }
      public HashMap<String,String> getOperationHashMap()
      {
            return hashMap;
      }
      private void startProcessing()
      {

       for(Element block:blockList)
       {

            if(block==null) break;

            String blockName=block.getAttributeValue("name");
```

```
            List list= block.getChildren();
            Iterator iterator=list.iterator();

            while(iterator.hasNext())
            {
               Element element=(Element)iterator.next();
               String name=element.getName();
              // System.out.println(element.getName()+"\n");

              if(element.getAttributeValue("type",
getXMINamespace()).equals("uml:Operation"))
               {
                     String id=element.getAttributeValue("id",
getXMINamespace());
                     String
operationName=blockName+":"+element.getAttributeValue("name");
                     hashMap.put(id, operationName);
                }
              }
        }
      }
}
```

## SysMLModel.java

```
package edu.clemson;

import org.jdom.Namespace;
import org.jdom.Element;

public class SysMLFlowModel
{
      private String XMIversion;
      private final String STRING_NAMESPACE_XMI_2_1 =
"http://schema.omg.org/spec/XMI/2.1";
      private final String STRING_NAMESPACE_XMI_2_0 =
"http://www.omg.org/XMI";
      private final String
STRING_NAMESPACE_SYSML_2_0="http://www.topcased.org/2.0/sysML" ;
      private final String
STRING_NAMESPACE_UML_2_1="http://www.eclipse.org/uml2/2.1.0/UML" ;
      private final Namespace
NAMESPACE_XMI_2_1=Namespace.getNamespace("http://schema.omg.org/spec/XM
I/2.1");
      private final Namespace
NAMESPACE_XMI_2_0=Namespace.getNamespace("http://www.omg.org/XMI");
      private final Namespace
NAMESPACE_SYSML_2_0=Namespace.getNamespace("http://www.topcased.org/2.0
/sysML");
```

```
	private final Namespace
NAMESPACE_UML_2_1_0=Namespace.getNamespace("http://www.eclipse.org/uml2
/2.1.0/UML");

	/*
	private String STRING_NAMESPACE_UML;
	private String STRING_NAMESPACE_XMI;
	private String STRING_NAMESPACE_SYSML;
	private Namespace NAMESPACE_UML;
	private Namespace NAMESPACE_XMI;
	private Namespace NAMESPACE_SYSML; */

	public void setCorrectXMIVersion(Element root)
	{
		/**
		 * Old code, perhaps sometime in future I'll remove it to
support all XMI version.
		 *
		 */

		String version=null;
		version=root.getAttributeValue("version",
NAMESPACE_XMI_2_1);
		if(version==null)
			version=root.getAttributeValue("version",
NAMESPACE_XMI_2_0);

		if(version==null)
		{
			System.out.println("XMI Version not supported. This
version of sysml is not supported by this product. " +
					"Please contact vikasp@clemson.edu for
further details");
		}
		else
			XMIversion=version;



	}

	public void setXMIVersion(String XMIVersion)
	{
		this.XMIversion=XMIVersion;
	}
	public String getXMIVersion()
	{
		return this.XMIversion;
	}
	public Namespace getXMINamespace()
	{
		if(this.XMIversion.equals("2.0"))
			return NAMESPACE_XMI_2_0;
		else
```

```
                return NAMESPACE_XMI_2_1;

        }
      public Namespace getSYSMLnamespace()
      {
        return NAMESPACE_SYSML_2_0;
      }
      public Namespace getUMLNamespace()
      {
        return NAMESPACE_UML_2_1_0;
      }

}
```

## XPathParser.java

```
package edu.clemson;

import org.jdom.input.SAXBuilder;
import org.jdom.xpath.XPath;
import org.jdom.Document;
import org.jdom.JDOMException;
import org.jdom.Element;
import org.jdom.Namespace;
import org.jdom.output.XMLOutputter;
import org.jdom.Text;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import java.util.Iterator;

import java.util.List;
import java.util.ArrayList;


import edu.clemson.CurrentOutput;
import edu.clemson.cirgmethods.CirgMethodManager;

public class XPathParser
{

       SAXBuilder saxBuilder=new SAXBuilder();
       CurrentOutput currentOutput=null;

       /*The class asks for currentOutput so that it has capability to
refer to the output of
         of any "action". As the xpath may need not always refer to the
current ouput variable.
```

```
	*/
	public XPathParser(CurrentOutput currentOutput)
	{
		this.currentOutput=currentOutput;
	}

	public Document getDocumentObject(String inputString)
	{
		//get the inputStream from String
		byte[] byteArray=inputString.getBytes();
		ByteArrayInputStream byteArrayInputStream=new
ByteArrayInputStream(byteArray);
		InputStream inputStream=byteArrayInputStream;

		//get the document object
		Document doc=null;
		try
		{
			doc=saxBuilder.build(inputStream);
		}
		catch(JDOMException e)
		{
			System.out.println("JDOM Exception");
			e.printStackTrace();
		}
		catch(Exception e)
		{
			System.out.println("An unknown exception occured");
		}
		return doc;
	}

	private List<Object> getXpathResult(String inputString, String
xpathString) throws JDOMException
	{
		if(inputString==null||xpathString==null)
			return null;

		Document doc=getDocumentObject(inputString);
		XPath xpath=null;

		try
		 {
			xpath = XPath.newInstance(xpathString);
			//xpath.addNamespace(Namespace_AmazonEC2);
		 }
		catch(JDOMException e)
		{
			System.out.println("ERROR: Invalid xpath
expression");
			//e.printStackTrace();
		}
		catch(Exception e)
		{
```

```
                System.out.println("An unknown exception occured");
            }

            List<Object> list=xpath.selectNodes(doc);
            return list;
      }


      public List<String> getElementName(String inputString, String
xpathString) throws JDOMException, IOException
      {
            List<String> elementValuelist= new ArrayList();
            List<Object> list=getXpathResult(inputString,xpathString);

            for(Object el:list)
            {
                  String val=null;
                  if(el instanceof Element)
                  {
                        XMLOutputter xmlOutputter=new XMLOutputter();
                        val=xmlOutputter.outputString((Element)el);
                  }
                  else if(el instanceof Text)
                        val=((Text)el).getText();

                  if(val==null||val.equals(""))
                        continue;

                  System.out.println(val+"\n");
                  //If the element value has a xpath expression
(enclosed in {}), handle it here
                  if(val.charAt(0)=='{' && val.charAt(val.length()-
1)=='}')
                  {
                        String xpathInner=val.substring(1,val.length()-
1);

                    String[] result= getVariableAndPath(xpathInner);
                    String xmlVariableToProcess=result[0];
                    String xpathToProcess=result[1];


if(xmlVariableToProcess==null||xpathToProcess==null)
                        return null;

                        List<String>
innerList=getElementName(xmlVariableToProcess,xpathToProcess);
                      for(String strVal:innerList)
                      {
                        elementValuelist.add(strVal);
                      }
                  } //In the normal case
                  else if(val!=null & !val.equals(""))
                  {     elementValuelist.add(val);}
```

```
            }

            return elementValuelist;
}

      public String[] getVariableAndPath(String xpathInner)
      {
            String[] result=new String[2];

            String xmlVariableToProcess=null;
            String xpathToProcess=xpathInner;

            String xmlVariableName=null;
            String xpathInnerTmp=xpathInner;
            int paranthesisCount=0;

            while(xpathInnerTmp.charAt(0)=='(')
            {
                  xpathInnerTmp=xpathInnerTmp.substring(1);
                  paranthesisCount++;
            }

            //The user did not specify the name of the Output element
so assuming it is the previous element

      if(xpathInnerTmp.startsWith("//")||xpathInnerTmp.startsWith("/"))
            {

      xmlVariableToProcess=currentOutput.getPreviousValue();
            }
            else if(xpathInnerTmp.startsWith("$"))
            {
                  int pos1=xpathInnerTmp.indexOf('/',1);
                  int pos2=xpathInnerTmp.indexOf(")");
                  int pos=0;

                  if (pos1==-1 && pos2==-1)
                        System.out.println("Possible error in xpath
expression\n");
                  else if(pos2==-1)
                        pos=pos1;
                  else if(pos1==-1)
                        pos=pos2;
                  else if(pos1<pos2)
                      pos=pos1;
                  else
                        pos=pos2;

                  xpathToProcess=xpathInnerTmp.substring(pos1); //If
there are ')' for which there are no corresponding '(' in
xpathToProcess, this will produce a invalid xpath
                  xmlVariableName=xpathInnerTmp.substring(1, pos); //to
ignore the $ sign at the start of the string
```

```
      xmlVariableToProcess=currentOutput.getOutputValue(xmlVariableName
);
            }

            for(int k=0;k<xmlVariableName.length();k++)
                  if(xmlVariableName.charAt(k)==')')
                        paranthesisCount--;

            for(int k=0;k<paranthesisCount;k++)
                  xpathToProcess="("+xpathToProcess;

            result[0]=xmlVariableToProcess;
            result[1]=xpathToProcess;
            return result;
      }


      public String processXMLData(Element current) throws
JDOMException
      {
          XMLOutputter xmlOutputter=new XMLOutputter();
          //System.out.println(xmlOutputter.outputString(current));

            listChildren(current);
            String updatedXml=xmlOutputter.outputString(current);


            return updatedXml;
      }

      private void listChildren(Element current) throws JDOMException
      {
        //  System.out.print(current.getName());
         //
System.out.println("\t"+current.getAttributeValue("type",Namespace.getN
amespace("http://schema.omg.org/spec/XMI/2.1")));
          String value=current.getText();

          if(value!=null && value.length()!=0)
          {
            System.out.println(value);
            if(value.charAt(0)=='{' && value.charAt(value.length()-
1)=='}')
            {
                  String xpathInner=value.substring(1,value.length()-
1);
                  if(xpathInner.startsWith("cirg:"))
                  {
                        String
fnName=xpathInner.substring(xpathInner.indexOf(":")+1);
                        CirgMethodManager cirgmm=new
CirgMethodManager(fnName,this);
```

```
                        String
result=Integer.toString(cirgmm.processMethod());
                        current.setText(result);
                    }
                    else
                      {
                        String[] result=
getVariableAndPath(xpathInner);
                       String xmlVariableToProcess=result[0];
                          String xpathToProcess=result[1];

                          List<Object>
list=getXpathResult(xmlVariableToProcess,xpathToProcess);

                          int elementPos=0;
                          for(Object el:list)
                          {
                              String val=null;
                              if(el instanceof Element)
                              {
                                    List
listContent=current.getContent();
                                    //Still need to fix this one !!!
                                    /*listContent.add(el);
                                    current.addContent(listContent);*/
                              }
                              else if(el instanceof Text)
                              {

      current.setText(((Text)el).getText());
                              }
                              else if(el instanceof String)
                              {

      current.setText(((Text)el).getText());
                              }
                          }
            }}
          }
          List children = current.getChildren();
          Iterator iterator = children.iterator();
          while (iterator.hasNext())
          {
            Element child = (Element) iterator.next();
            listChildren(child);
          }
      }



}
```

## CurrentOutput.java

```
package edu.clemson;
import java.util.HashMap;

public class CurrentOutput
{
	private String currentKey=null;
	private String currentValue=null;
	private String previousKey=null;
	private String previousValue=null;
	private HashMap<String,String> map=new HashMap<String,String>();


	public String getcurrentOutputKey()
	{
		return currentKey;
	}

	public String getCurrentOutputValue()
	{
		return currentValue;
	}
	public String getPreviousValue()
	{
		return previousValue;
	}
	public String getPreviousKey()
	{
		return previousValue;
	}

	public String getOutputValue(String key)
	{
		return map.get(key);
	}

	public void putOutputValue(String key,String value)
	{
		key=key+"Response";
		if(!(value.startsWith("<"+key)))
			value="<"+key+">"+value+"</"+key+">";

		map.put(key, value);

		previousKey=currentKey;
		previousValue=currentValue;
		currentKey=key;
		currentValue=value;
	}
```

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.

[30] G. Booch, J. Rumbaugh and I. Jacobson, *Unified Modeling Language User Guide, the (2nd Edition) (Addison-Wesley Object Technology Series), 2005.*

[31] *Extensible Markup Language (XML) Version 1.1*, World Wide Web Consortium (W3C) Std., 2006.

[32] D. Obasanjo, "Understanding XML," http://msdn.microsoft.com/en-us/library/aa468558.aspx.

[33] R. Smith, "Computing in the cloud," *Research Technology Management, 2009.*

[34] D. Thain, T. Tannenbaum and M. Livny, "Distributed computing in practice: The condor experience," *Concurrency - Practice and Experience 17(2-4),* pp. 323-356, 2005.

[35] O. Smirnova, P. Eerola, T. Ekelof, M. Ellert, J. R. Hansen, A. Konstantinov, B. Konya, J. L. Nielsen, F. Ould-saada and A Waananen, "The NoduGrid Architecture and Middleware for Scientific Applications," *Lecture Notes in Computer Science*, 2003.

[36] R. Pordes, et al, "The Open Science Grid," *Journal of Physics: Conference Series 78:012057,* 2007.

[37] C. Catlett, “The Philosophy of TeraGrid: Building an Open, Extensible, Distributed TeraScale Facility,” in *CCGRID ’02: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 8.

[38] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl, "Scientific Cloud Computing: Early Definition and Experience," 2008.

[39] T. J. Bittman, “Server Virtualization: From Emerging to Mainstream at Light Speed,” Gartner Presentation, 2009.

[40] R. Vanover, “Key differences between Amazon EC2 and VMware vCloud Express.” [Online]. Available: http://blogs.techrepublic.com.com/networking/?p=2057.