DELIVERABLE 1
SOFTWARE REQUIREMENTS AND ANALYSIS DOCUMENT (SRAD)

CEN5064 - Software Design

Team 1 – Edwin Malek, Dionny Santiago, Xiaoyu Dong

02/19/2015

Professor: Dr. Peter Clarke

Edwin Malek

Abstract

This document goes over the specifications for the Cloud Provisioning Graphical Editor System that team 1 is building, using of the EMF/GMF eclipse platform. This document defines the necessary functionalities that the future system will provide to our users. Through the use of UML diagrams, we will attempt to define our system and its responsibilities to our users (feature diagrams, class diagrams, sequence diagrams, activity diagrams, object diagrams, use case diagrams, etc…). We will also use a verbal description of each of our use cases along with specifically chosen scenarios for the most important use cases. Our system will be divided into a modeling environment and an Execution environment, both of which could in theory function independent of each other.

Cloud provisioning is still a fairly new technology, with great potential, but the systems today lack the visual representation in order for the less tech-savvy users to be able to see a complex system and tailor the service to their needs.

Table of Contents

1 Introduction

In this section we will discuss the purpose of our system, along with the scopes of the system, as well as a list of definitions, acronyms, and abbreviations used throughout the document.

1.1 *Purpose of System*

The purpose of our system will be to allow a user to create a visual representation of today's cloud provisioning services offers by Amazon, Google and others. While still retaining the possibilities to edit and configure his/her cloud environment and most important deploy the visually created system using the Amazon or google Cloud provisioning APIs.

1.2 *Scope of System*

The system will be in the form of a graphical editor. Similar to the UML style graphical editors already available on the market. The system will obey to a set of networking rules, and today's limitations of CPU power, Memory, and Storage. The system will be modified in the future if these limitations (CPU speed, Storage Capacity etc…) are no longer an issue.

1.3 *Definitions, Acronyms, and Abbreviations*

EE: Execution Environment

ME: Modeling Environment

XML: Extensible Markup Language

XML: file format respecting a certain syntax

CPROV: Cloud Provision system and subsystem

CPROV-XML: XML syntax for Cloud provisioning.

CPU: Computer Processing Unit

RAM: Random Access Memory

VPN: virtual private network

VPC: virtual private cloud

VM: Virtual Machine.

CPS: Cloud Provisioning System.

NODE: Any icon from the palette

GUI: Graphical User Interface

1.4 Overview of Document

Chapter 2:

This chapter will discuss how the current systems operate, and what the limitations and problems are reoccurring.

Chapter3:

This chapter will discuss the project plan along with the organization, software/hardware requirements, and work breakdown.

3.1 Organization

3.2 Software/hardware minimal requirements

3.3 Work breakdown.

Chapter4:

This chapter will discuss the system requirements both functional and non-functional, along with use case model(s) to support these requirements.

4.1 Overview

4.2 Functional requirements

4.3 Non-functional requirements

4.4 Use case models

Chapter 5:

This chapter will discuss the system's analysis using both the static and dynamic views of the system represented with use case scenarios, object models, Dynamic Models, and a simple view of the future system's interface.

5.1 scenarios

5.2 Object Models

5.3 Dynamic Models

5.4 Interface Model

Chapter 6:

This chapter contains the glossary of all associated terms used throughout the document for a quick reference for the any unfamiliar readers.

Chapter 7:

This chapter will contain an Appendix in the following order:

Appendix A - Project schedule (Gantt chart or PERT chart).

Appendix B – Use cases - use the template provided.

Appendix D – User Interface designs.

Appendix B – Diary of meeting and tasks.

2 Current System

The currently used cloud provisioning systems all have the same limitations. They are mostly designed with the website interface in mind. That is they all use the series of common interface items such as dropdown boxes, and/or selective buttons for the users to configure their systems. This forces the users to create a mental representation of the services they needs instead of visually allowing the users to manually configure their cloud using a click-and-drag style interface.

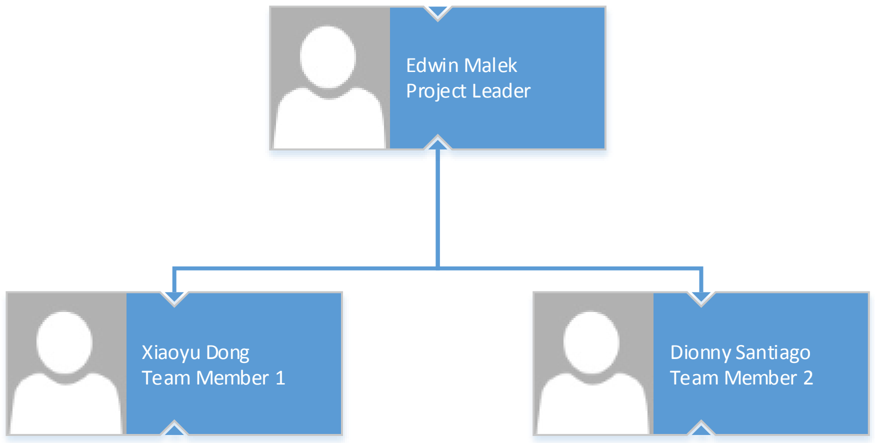
Our systems aims at filling the need for the less technology oriented users, and allowing them to configure and reconfigure their CPS before the deployment process. Another problem we aim to solve, is the validation of a user created cloud provisioning request. With the current system, the user cannot simply experiment with the design of a CPS our system aims at showing the final CPS model before any deployment.

3 Project Plan

This section presents the general organizational structure of the CPS development plan. This includes the various work distributions during each phase of the project, the hardware and software requirements to develop and test the system, as well as milestones and deliverables produced during each stage.

3.1 Project Organization

The following figure displays the organizational structure of the roles involved in the CPS project for each of the three phases of development:



Leader: Edwin Malek

Minute Taker: Xiaoyu Dong

Model Developer: Dionny Santiago

3.2 Hardware and Software Requirements

3.2.1 Necessary Hardware

Processor: Intel or AMD 2Ghz or faster

Memory: 1 GB of RAM (minimum), 2Gb (recommended)

Hard Drive: 80 GB

Headset with built-in microphone (optional)

3.2.2 Necessary Software

Windows XP Professional / Windows Vista/ Windows 7,8, 8.1

Microsoft Word 2007, 2010, 2013

Papyrus Plug-in

Microsoft PowerPoint 2007, 2010, 2013

Eclipse 3.4.1 IDE

Amazon EC2 API (optional, but recommended)

3.3 Work Breakdown

Tasks and Milestones: Refer to Project Plan in Appendix A, and Team Diary in Appendix D.

Milestone 1 consists of the completion of the Use Case specification and Analysis Phase. Do the abstraction. Analysis every problem the user may meet.

Milestone 2 consists of the completion of Design Phase, the creation of the meta model and the design and implementation of the CPS in general. This includes the implementation of the parsers and the validators as well. The result of this phase is presented in a design document handed in to professor.

Finally, **Milestone 3**, merge every member's result into final deliverable document includes the two previous deliverable plus the details of this last milestone.

4 System Requirements

4.1 Overview

The CPROV system is broken down into two made subsystems responsible for 2 different parts. The ME is the visual representation of the CPS chosen and designed by the user. It consists of the a graphical model and a tree representation of the graphical model.

The EE represents the engine that translates the user generated CPS into a CPROV-XML file. Independent of the ME. Allow for transportation and intermediate language for use on another graphical editor. The EE is also responsible for the CPS deployment phase once the user is satisfied with design of the CPS.

The CPROV system is designed for an advanced user, but could easily be learned and used by an intermediate user with basic knowledge of today's computer technical specification. The system consist of a workspace in the center of the interface, and a palette of icon located on the right side of the interface. The user typical use will consist of clicking an icon on the palette and place the icon on the workspace in the desired location. An icon in the CPS is referenced as a node, most Nodes taken from the palette will be able to be customized through changes made in the properties.

The nodes, once placed in the desired locations, can be connected with a variety of connection types also represented as icons situated on the palette. The user typically can select the connection icon, click on the source and destination nodes to create a connection. If the two nodes can somehow be linked without generating errors, a connection will be displayed between the nodes signifying that an association has been made.

On the execution environment side, the user will translate to CPROV-XML, and be able to deploy directly from the GUI.

4.2 Functional Requirements

1. The system shall allow cloud administrators to create CPS models based on the CPROV language.
   Use cases: All use cases related to General Modeling Environment.
   Use Case ID(s): CPROVME001 Create New Model, CPROVME002 Add Model Element, CPROVME003 Add Environment Node, CPROVME004 Add security group Node, CPROVME005 Add Instance Node, CPROVME006 add Storage Node, CPROVME007 Add Network Node, CPROVME008 Edit Node properties, CPROVME009 Edit Instance Properties, CPROVME010 Edit security Group Properties, CPROVME011 Edit Storage Properties, CPROVME012 Edit Network

Properties, CPROVME013 Delete Node, CPROVME014 Add Connection, CPROVME015 Delete Connection, CPROVME016 Load Existing Model, CPROVME017 Validate Model, CPROVME018 Save Model.

2. The system shall allow cloud administrators to translate the existing model into a CPROV-XML representation capable of deployment through the CPS appropriate API calls.
   Use cases: All use cases related to General Execution Environment.
   Use Case ID(s): CPROVEE001 Submit Request, CPROVEE002 Transformation to XML, CPROVEE003 Validation Against CPROV Schema, CPROVEE004 Transformation to Amazon EC2 Call(s), CPROVEE005 Print to XML, CPROVEE006 XML to Model.

3. The system shall allow cloud administrators to print their existing model to XML representation.
   Use cases: All use cases related to General Execution Environment.
   Use Case ID(s): CPROVEE001 Submit Request, CPROVEE002 Transformation to XML, CPROVEE003 Validation Against CPROV Schema, CPROVEE004 Transformation to Amazon EC2 Call(s), CPROVEE005 Print to XML, CPROVEE006 XML to Model.

4. The system shall allow cloud administrators to validate their CPS models against a set of rules based on current CPS systems such as the Amazon EC2.
   Use cases: All use cases related to General Execution Environment.
   Use Case ID(s): CPROVEE001 Submit Request, CPROVEE002 Transformation to XML, CPROVEE003 Validation against CPROV Schema, CPROVEE004 Transformation to Amazon EC2 Call(s),

5. The system shall allow cloud administrators to modify their models, specifically the properties of the nodes present on the diagram, before any deployment to CPS is necessary.
   Use cases: All use cases related to Modeling Environment.
   Use Case ID(s): CPROVME008 Edit Node properties, CPROVME009 Edit Instance Properties, CPROVME010 Edit security Group Properties, CPROVME011 Edit Storage Properties, CPROVME012 Edit Network

6. The system shall allow for a cloud administrator to import an CPROV XML file, and transform it to a CPS model (diagram and tree)
   Use cases: All use cases related to General Execution Environment.
   Use Case ID(s): CPROVEE006 XML to Model.

4.3 Non-Functional Requirements

1. Usability:

    a) No previous Training Time

    c) On average the user should be able to submit a deployment request in under 1 minute.

2. Reliability:

    a) Mean time to Failure – 5% failures for every 1000 attempts to add an instance.

    b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

    a) Deployment of the current workspace diagram should complete in under one minute.

    b) System handles only one request for deployment at a time.

4. Supportability:

    a) The execution environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

5. Implementation:

    a) The software will be implemented via a Eclipse plugin that the user can install and run.

4.4 Use Case Models

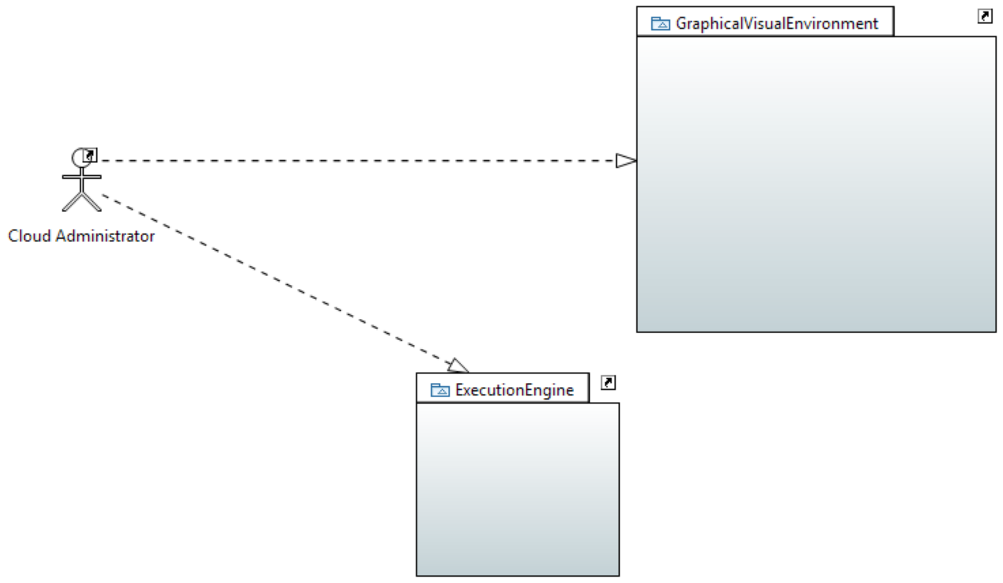


Figure 4-1 High level Use Case diagram based on the packages of the system.

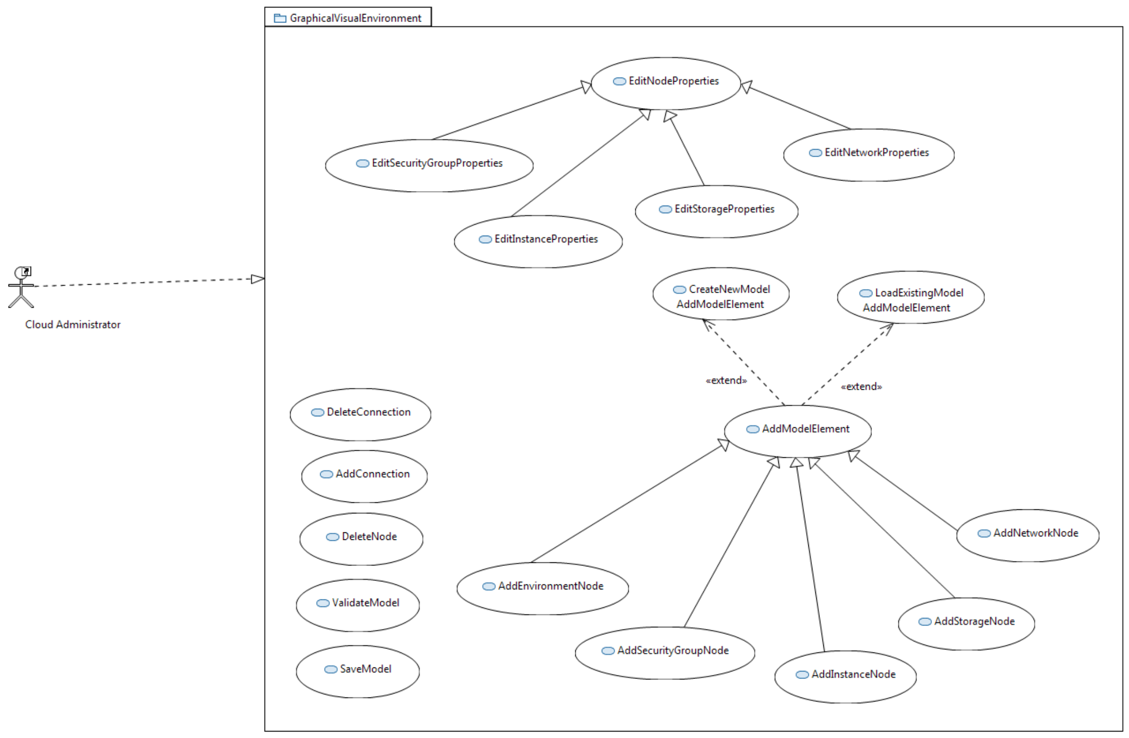


Figure 4-2 Use Case Diagram for the Modeling Environment of our system.

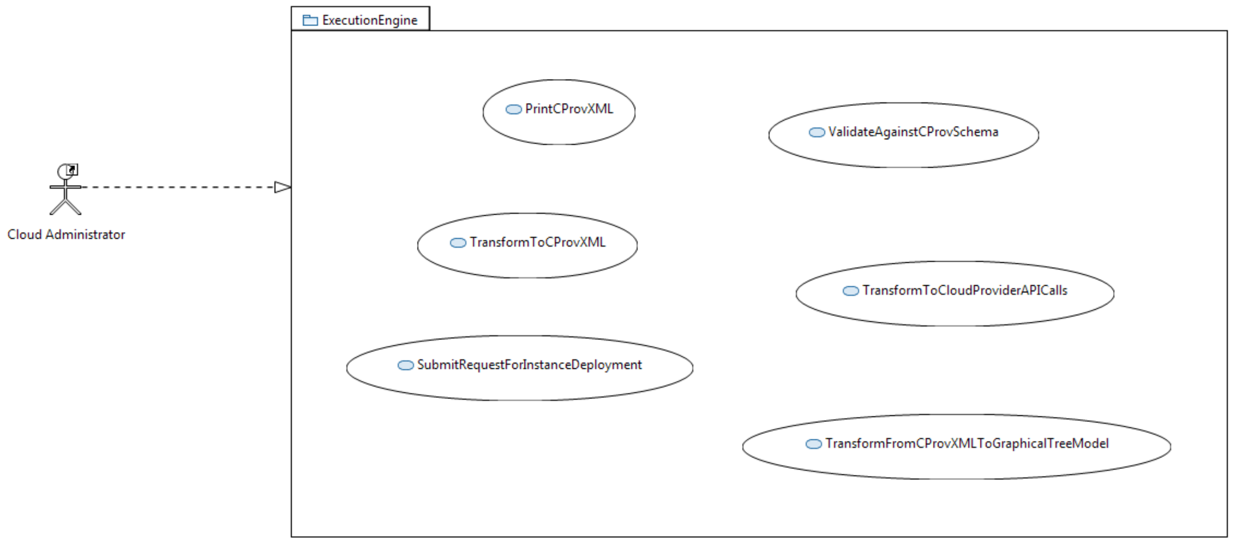

ExecutionEngine
Cloud Administrator
PrintCProvXML
ValidateAgainstCProvSchema
TransformToCProvXML
TransformToCloudProviderAPICalls
SubmitRequestForInstanceDeployment
TransformFromCProvXMLToGraphicalTreeModel

Figure 4-3 Use Case Diagram for the Execution Environment of our system.

Refer to Appendix B for full list of use cases.

5 System Analysis

5.1 Scenarios

<u>**Scenario 1: Jose Gonzalez submits a deployment request**</u>

Scenario ID: **SCE1-CPROVEE001 Submit request deployment**

Details:

Actor: Jose Gonzalez (Cloud Administrator)

Pre-conditions:

1. The graphical visual environment must be running on the system.
2. A model is active and contains an Environment connected to an Instance. The Instance is connected to a Storage node.

Description:

1. <u>Scenario begins</u> when Jose clicks on the deploy button on the toolbar.
2. The system provides Jose a confirmation message with two options: Confirm/Cancel.
3. Jose clicks on the Confirm button on the confirmation message.
4. The system translates the graphical representation of the server instance into a series of commands that utilize the Amazon EC2 API
5. The system executes the API calls to the Amazon EC2 service

Post-conditions:

1. A message appears stating that the deployment was successful.
2. The system returns focus to the graphical elements present on the workspace.

<u>**Scenario 2: Jose Gonzalez transforms the graphical model to the CPOV XML**</u>

Scenario ID: **SCE1-CPROVEE002 transform to CPROV XML**

Details:

Actor: Jose Gonzalez (Cloud Administrator)

Pre-conditions:

1. The graphical visualization environment must be running on the system.
2. A model is active and contains an Environment connected to an Instance. The Instance is connected to a Storage node.
3. The model must pass validation prior to transformation.

Description:

1. <u>Scenario begins</u> when Jose Gonzalez clicks on the “Transform to XML” button located in the file submenu.

2. The system provides Jose Gonzalez a confirmation message with two options: Confirm/Cancel.
3. Jose Gonzalez clicks on the Confirm button on the confirmation message.
4. The system translates the graphical representation of the server instance into an XML schema of the model.

Post-conditions:

1. A message appears stating that the transformation to XML was successful.
2. The system returns focus to the graphical elements present on the workspace.
3. The system saves the XML schema to a file in a location designated by Jose Gonzalez.

Scenario 3: Jose Gonzalez validates against the Cloud Provisioning Schema

Scenario ID: **SCE1-CPROVEE003 Validation against Cloud provisioning Schema**

Details:

Actor: Jose Gonzalez (Cloud Administrator)

Pre-conditions:

1. The graphical visualization environment must be running on the system.
2. A model is active and contains an Environment connected to an Instance, and a Storage node connected to the same Instance.

Description:

1. Scenario begins when Jose Gonzalez clicks right clicks on the workspace.
2. In the right click drop down menu, Jose Gonzalez clicks on the “Validate”
3. The system starts to validation process with a progress bar starting at 0 % and ends at 100% showing that the validation has been done.

Post-conditions:

1. A message appears stating that the validation process reported errors or no errors.
2. The system returns focus to the graphical elements present on the workspace.

Scenario 4: Henry submits a deployment request

Scenario ID: **SCE2-CPROVEE001 Submit request deployment**

Details:

Actor: Henry (Cloud Administrator)

Pre-conditions:

1. The graphical visual environment must be running on the system.

2. A model is active and contains an Environment connected to two different Instances.

Description:

1. <u>Scenario begins</u> when the Henry clicks on the deploy button on the toolbar.
2. The system provides Henry a confirmation message with two options: Confirm/Cancel.
3. Henry clicks on the Confirm button on the confirmation message.
4. The system translates the graphical representation of the server instance into a series of commands that utilize the Amazon EC2 API
5. The system executes the API calls to the Amazon EC2 service

Post-conditions:

1. A message appears stating that the deployment was successful.
2. The system returns focus to the graphical elements present on the workspace.

Scenario 5: Henry transforms the graphical model to the CPOV XML

Scenario ID: **SCE2-CPROVEE002 transform to CPROV XML**

Details:

Actor: Henry Smith (Cloud Administrator)

Pre-conditions:

1. The graphical visualization environment must be running on the system.
2. A model must be active and must contain an Environment node and two Instance nodes, each connected to the Environment.
3. The model must pass validation prior to transformation.

Description:

1. <u>Scenario begins</u> when Henry clicks on the “Transform to XML” button located in the file submenu.
2. The system provides Henry a confirmation message with two options: Confirm/Cancel.
3. Henry clicks on the Confirm button on the confirmation message.
4. The system translates the graphical representation of the server instance into an XML schema of the model.

Scenario 6: Henry validates against the Cloud Provisioning Schema

Scenario ID: **SCE2-CPROVEE003 Validation against Cloud provisioning Schema**

Details:

Actor: Henry Smith (Cloud Administrator)

Pre-conditions:

1. The graphical visualization environment must be running on the system.
2. A model is active and contains an Environment connected to two unique Instances.

Description:

1. Scenario begins when Henry clicks right clicks on the workspace.
2. In the right click drop down menu, Henry clicks on the “Validate”
3. The system starts to validation process with a progress bar starting at 0 % and ends at 100% showing that the validation has been done.

Post-conditions:

1. A message appears stating that the validation process reported errors or no errors.
2. The system returns focus to the graphical elements present on the workspace.

Scenario 7: Jose Gonzalez adds an instance node

Scenario ID: **SCE1-CPROVME005 Add Instance Node**

Details:

Actor: Jose Gonzalez (Cloud Administrator)

Pre-conditions:

1. The graphical visualization environment must be running on the system. the eclipse plug in must be running on Jose’s PC or MAC.

Description:

2. Jose clicks on the instance icon in the palette located on the right side of the interface.
3. Jose clicks on the workspace in the desired location.
4. The system shall provide the user with a text box to enter the instance name.
5. Jose renames the instance “JGServer1”.
6. An icon representing the instance is then added to the workspace.

Post-conditions:

1. The instance is then created and focus is switched to the properties window.
2. Jose is satisfied with the default values of the instance he just created and decides not to change it.

Scenario 8: Henry Smith creates a new model

Scenario ID: **SCE1-CPROVME001 Create New Model**

Details:

Actor: Henry Smith (Cloud Administrator)

Pre-conditions:

1. The graphical visualization environment must be running on the system.

Description:

1. Scenario begins when Henry Smith clicks on a File button located on the top left hand side of the workspace.
2. Henry Smith selects the “New CPROV Model” button.
3. The system opens a new diagram in the currently select Project folder.
4. The system displays a window for Henry Smith to enter a Name for the Model.
5. Henry Smith enters “MyNewModel” for the name.
6. Henry Smith clicks the “OK” button to confirm.
7. The system switches focus to the newly created model workspace.

Scenario 9: Jose Gonzalez adds a security group node

Scenario ID: **SCE1-CPROVME003 Add SecurityGroup**

Details:

Actor: Power user (Jose Gonzalez)

Pre-conditions:

1. The graphical visualization environment must be running on the system. The eclipse plug in must be running on Jose’s PC or MAC.
2. Jose must have at least 2 person/user icons must be present on the workspace for the SecurityGroup to be needed. If Jose’s deployment will not have more than 2 or 3 users than the SecurityGroup is probably not needed.

Description:

1. Scenario begins when Jose clicks on the SecurityGroup icon in the palette located on the right side of the interface.
2. Jose clicks on the workspace in the desired location.
3. The system shall provide the user with a text box to enter the SecurityGroup name.
4. Jose renames the SecurityGroup “network1”
5. An icon representing the SecurityGroup is then added to the workspace.

Post-conditions:

1. The SecurityGroup is then created and focus is switched to the properties window.
2. Jose is satisfied with the default values of the SecurityGroup he just created and decides not to change it.

Scenario 10: Jose Gonzalez adds a connection

Scenario ID: **SCE1-CPROVME004 Add Connection**

Details:

Actor: Power user (Jose Gonzalez)

Pre-conditions:

1. The graphical visualization environment must be running on the system. The eclipse plug in must be running on Jose's PC or MAC.
2. Jose has a model with 3 nodes: Environment, Instance, and Storage.
3. The Environment and Instance nodes are connected.

Description:

1. Jose clicks on the Connection icon in the palette located on the right side of the interface.
2. Jose clicks on the workspace on the Instance node.
3. Jose clicks on the workspace on the Storage node.
4. The system shall provide Jose with a text box to enter the connection's name.
5. Jose renames the connection "Instance/Storage Relation"
6. An arrow between the Instance and the Storage icon representing the connection is then added to the Jose's diagram.

<u>Scenario 10: Henry Smith adds an instance node</u>

Scenario ID: **SCE2-CPROVME005 Add Instance Node**

Details:

Actor: Henry Smith (Cloud Administrator)

Pre-conditions:

1. The graphical visual environment must be running on the system. The eclipse plug in must be running on Henry's PC or MAC.

Description:

1. Henry clicks on the instance icon in the palette located on the right side of the interface.
2. Henry clicks on the workspace in the desired location.
3. The system shall provide the user with a text box to enter the instance name.
4. Henry renames the instance "HSServer1".
5. An icon representing the instance is then added to the workspace.

Post-conditions:

1. The instance is then created and focus is switched to the properties window.
2. Henry is satisfied with the default values of the instance he just created and decides not to change it.

<u>Scenario 11: Henry Smith creates a new model</u>

Scenario ID: **SCE2-CPROVME001 Create New Model**
Details:

Actor: Jose Gonzalez (Cloud Administrator)

Pre-conditions:

1. The graphical visualization environment must be running on the system.

Description:

1. Trigger: the use case begins when Jose clicks on a File button located on the top left hand side of the workspace.
2. Jose selects the "New CPROV Model" button.
3. The system opens a new diagram in the currently select Project folder.
4. The system displays a window for Jose to enter a Name for the Model.
5. Jose enters "JGNewModel" for the name.
6. Jose clicks the "OK" button to confirm.
7. The system switches focus to the newly created model workspace.

Post-conditions:

1. The system creates a new Model along with its tree representation located on the left hand side of the workspace.

Scenario 12: Henry Smith adds a security group

Scenario ID: **SCE2-CPROVME003 Add SecurityGroup**

Details:

Actor: Henry Smith (Cloud Administrator)

Pre-conditions:

1. The graphical visualization environment must be running on the system. The eclipse plug in must be running on Henry's PC or MAC.
2. Henry must have at least 2 person/user icons must be present on the workspace for the SecurityGroup to be needed. If Henry's deployment will not have more than 2 or 3 users than the SecurityGroup is probably not needed.

Description:

1. Henry clicks on the SecurityGroup icon in the palette located on the right side of the interface.
2. Henry clicks on the workspace in the desired location.
3. The system shall provide the user with a text box to enter the SecurityGroup name.
4. Henry renames the SecurityGroup "H Network 1"
5. An icon representing the SecurityGroup is then added to the workspace.

Post-conditions:

1. The SecurityGroup is then created and focus is switched to the properties window.

2. Henry is satisfied with the default values of the SecurityGroup he just created and decides not to change it.

<u>Scenario 13: Henry Smith adds a connection</u>

Scenario ID: **SCE2-CPROVME004 Add Connection**

Details:

Actor: Henry Smith (Cloud Administrator)

Pre-conditions:

1. The graphical visualization environment must be running on the system. The eclipse plug in must be running on Henry's PC or MAC.
2. Henry has a model with 3 nodes: Environment, InstanceA, InstanceB.
3. The Environment and InstanceB nodes are connected.

Description:

1. Henry clicks on the Connection icon in the palette located on the right side of the interface.
2. Henry clicks on the workspace on the InstanceA node.
3. Henry clicks on the workspace on the InstanceB node.
4. The system shall provide the Henry with a text box to enter the connection's name.
5. Henry renames the connection "Server1 to Server2 conn"
6. An arrow between the InstanceA and InstanceB icons representing the connection is then added to the Henry's diagram.

Post-conditions:

1. The connection is then created and focus is switched to the properties window.
2. Henry is satisfied with the default values of the connection he just created and decides not to change it.

5.2 *Object Model*

Object Diagrams

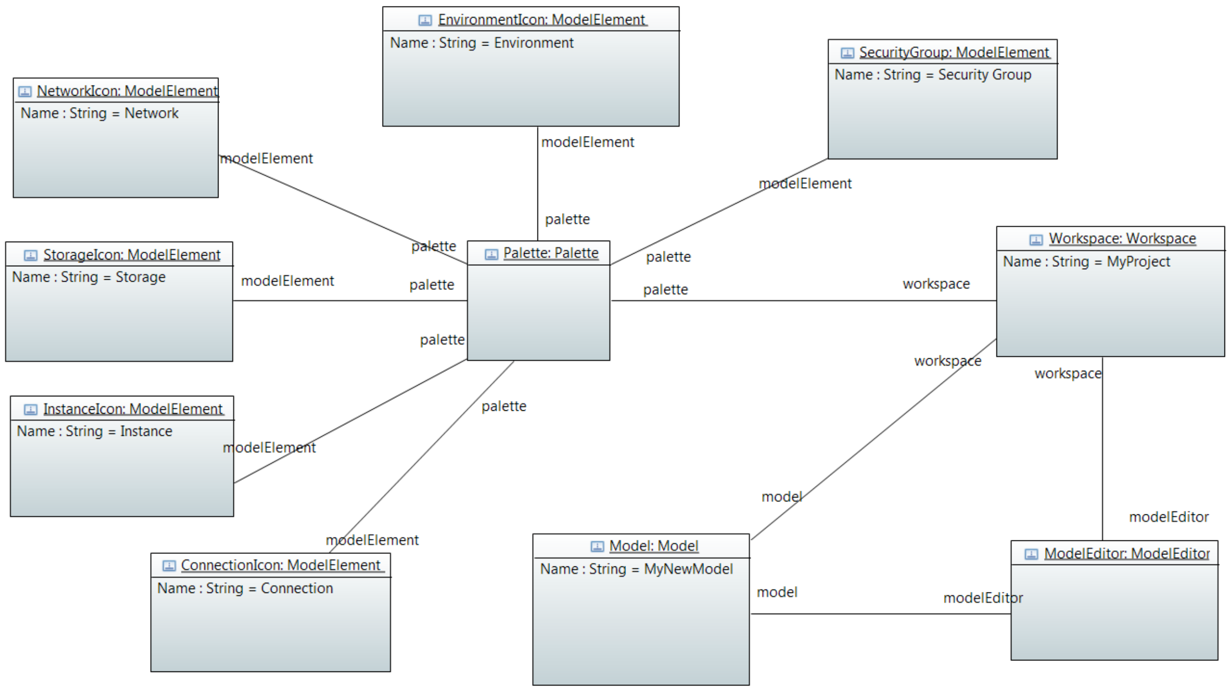

EnvironmentIcon: ModelElement
Name : String = Environment
SecurityGroup: ModelElement
Name : String = Security Group
NetworkIcon: ModelElement
Name : String = Network
StorageIcon: ModelElement
Name : String = Storage
InstanceIcon: ModelElement
Name : String = Instance
ConnectionIcon: ModelElement
Name : String = Connection
Palette: Palette
Workspace: Workspace
Name : String = MyProject
Model: Model
Name : String = MyNewModel
ModelEditor: ModelEditor
modelElement
palette
workspace
model
modelEditor

Figure 5-1 Object Diagram based on SCE1-CPROVME001 Create New Model

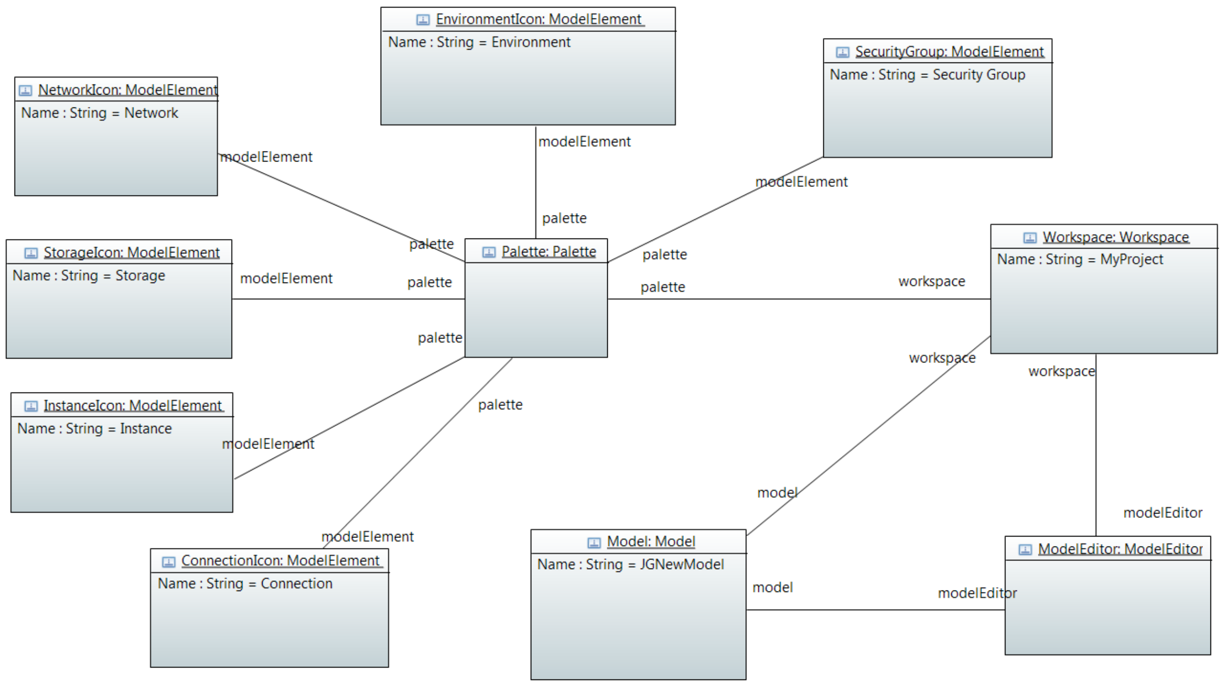


Figure 5-2 Object Diagram based on SCE2-CPROVME001 Create New Model

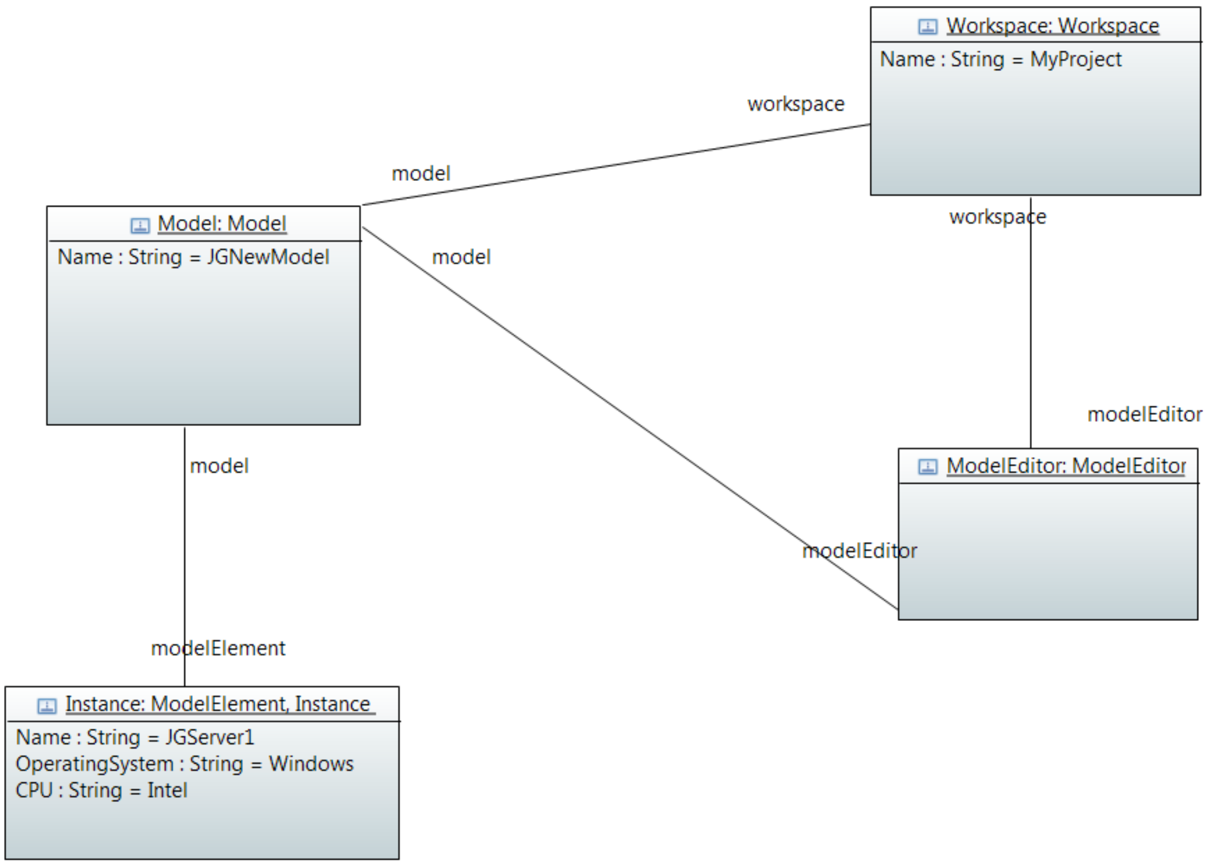


Figure 5-3 Object Diagram based on SCE1-CPROVME005 Add Instance Node

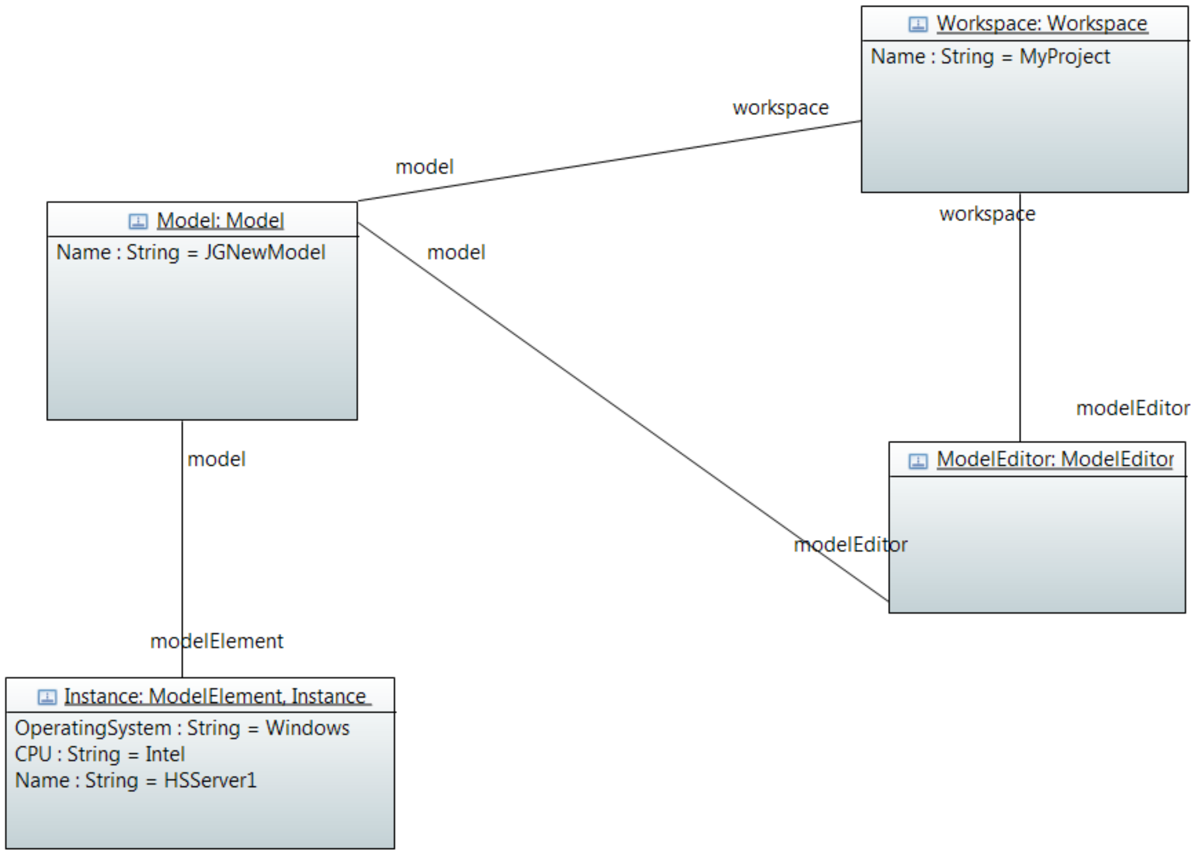


Figure 5-4 Object Diagram based on SCE2-CPROVME005 Add Instance Node

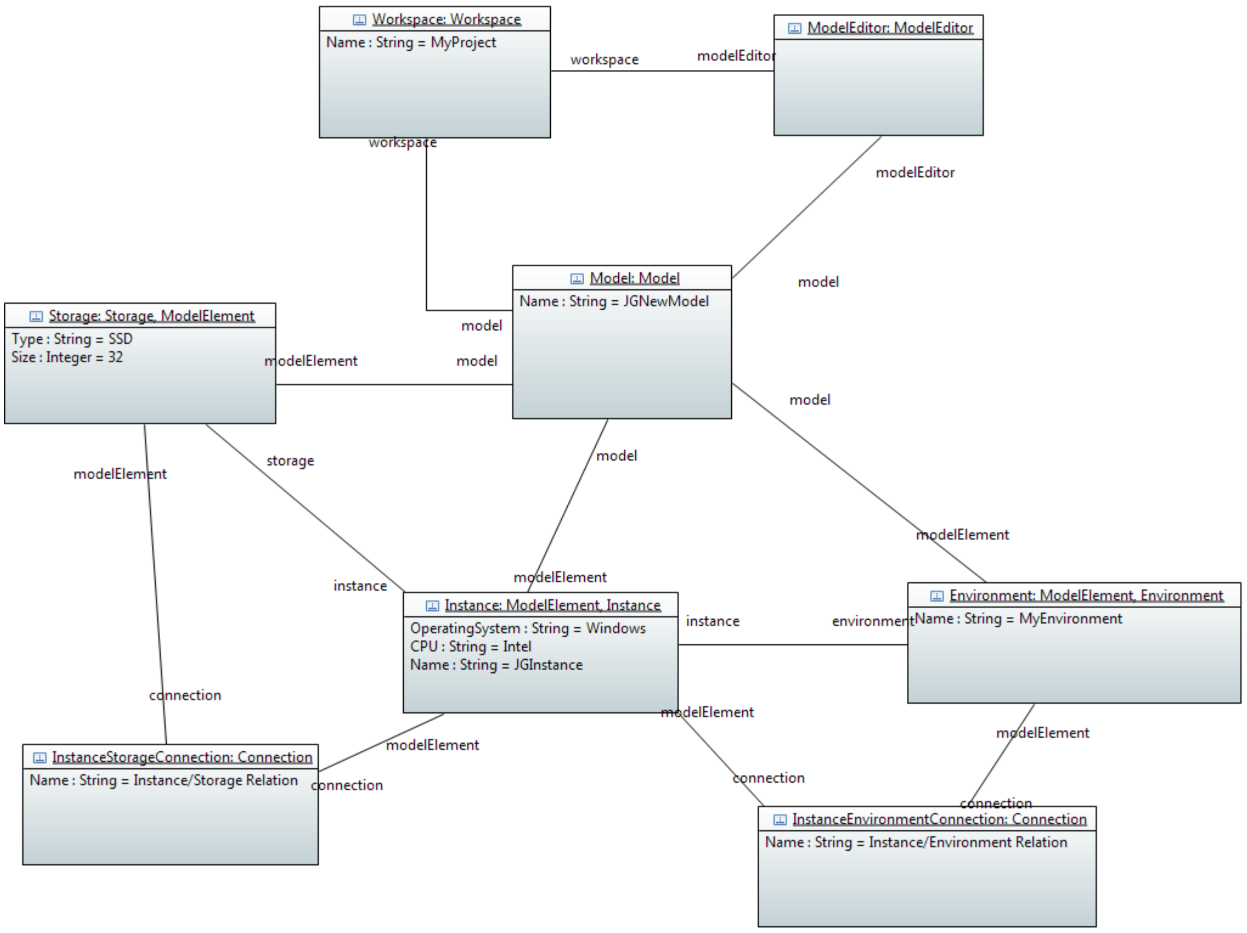

Workspace: Workspace
Name : String = MyProject
ModelEditor: ModelEditor
workspace
modelEditor
workspace
modelEditor
model
Model: Model
Name : String = JGNewModel
Storage: Storage, ModelElement
Type : String = SSD
Size : Integer = 32
model
modelElement
model
model
storage
model
modelElement
modelElement
instance
modelElement
modelElement
Instance: ModelElement, Instance
OperatingSystem : String = Windows
CPU : String = Intel
Name : String = JGInstance
instance
environment
Environment: ModelElement, Environment
Name : String = MyEnvironment
connection
modelElement
modelElement
modelElement
InstanceStorageConnection: Connection
Name : String = Instance/Storage Relation
connection
connection
connection
InstanceEnvironmentConnection: Connection
Name : String = Instance/Environment Relation

Figure 5-5 Object Diagram based on SCE1-CPROVME004 Add Connection

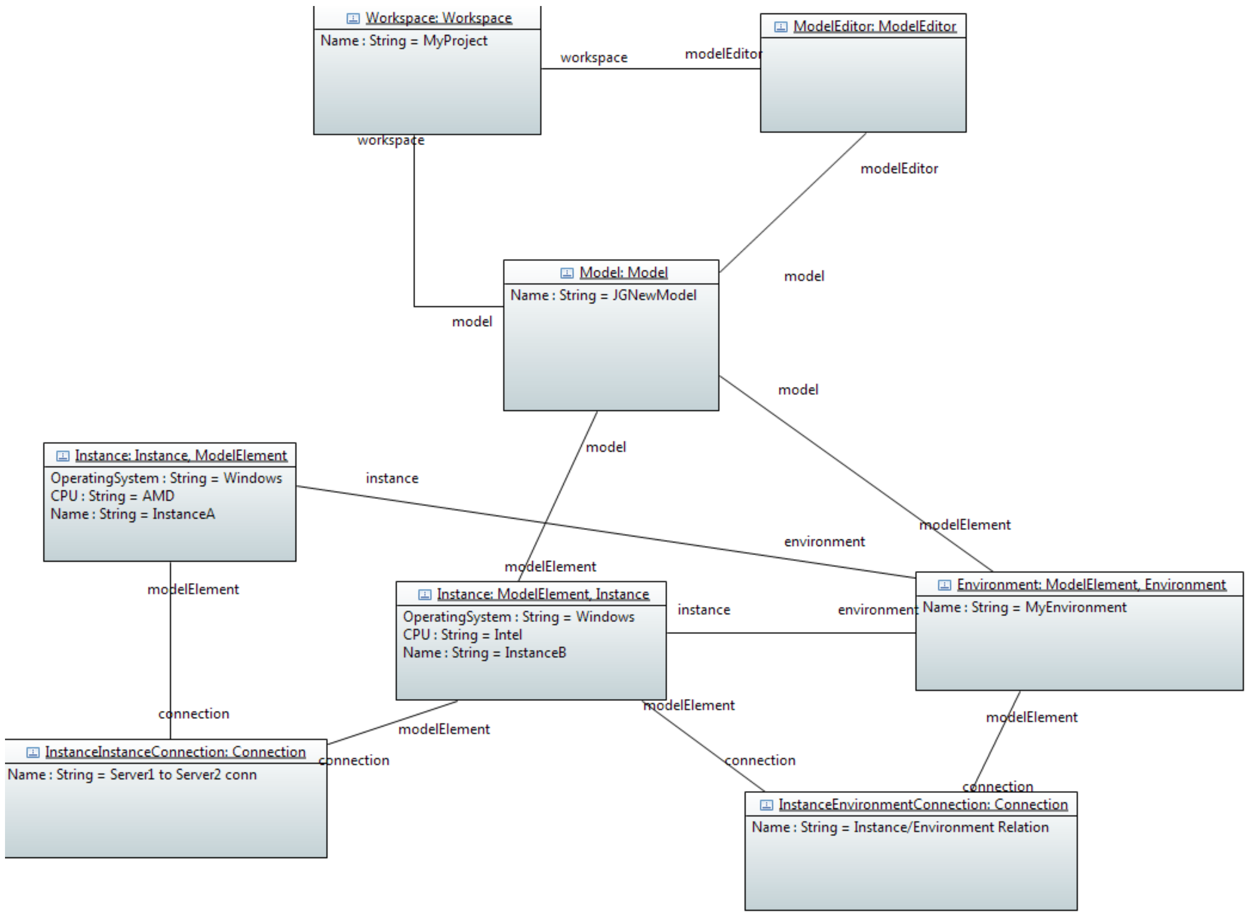

Workspace: Workspace
Name : String = MyProject
ModelEditor: ModelEditor
workspace
modelEditor
workspace
modelEditor
Model: Model
Name : String = JGNewModel
model
model
model
Instance: Instance, ModelElement
OperatingSystem : String = Windows
CPU : String = AMD
Name : String = InstanceA
instance
model
environment
modelElement
modelElement
modelElement
Instance: ModelElement, Instance
OperatingSystem : String = Windows
CPU : String = Intel
Name : String = InstanceB
instance
environment
Environment: ModelElement, Environment
Name : String = MyEnvironment
connection
modelElement
modelElement
modelElement
InstanceInstanceConnection: Connection
Name : String = Server1 to Server2 conn
connection
connection
connection
InstanceEnvironmentConnection: Connection
Name : String = Instance/Environment Relation

Figure 5-6 Object Diagram based on SCE2-CPROVME004 Add Connection

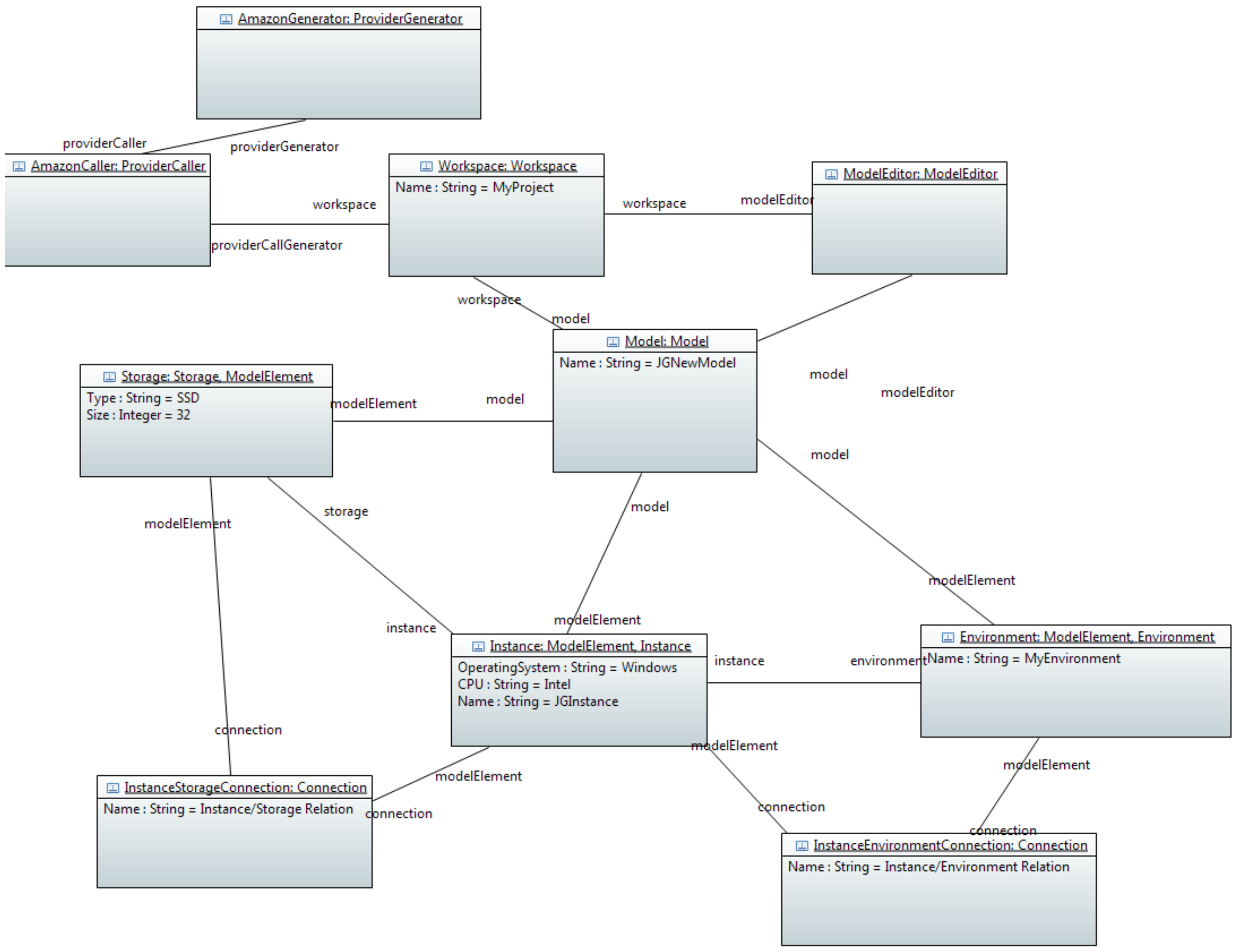

AmazonGenerator: ProviderGenerator
providerCaller
providerGenerator
AmazonCaller: ProviderCaller
workspace
providerCallGenerator
Workspace: Workspace
Name : String = MyProject
workspace
modelEditor
ModelEditor: ModelEditor
workspace
model
Model: Model
Name : String = JGNewModel
model
modelEditor
Storage: Storage, ModelElement
Type : String = SSD
Size : Integer = 32
modelElement
model
model
storage
modelElement
model
modelElement
instance
modelElement
Instance: ModelElement, Instance
OperatingSystem : String = Windows
CPU : String = Intel
Name : String = JGInstance
instance
environment
Environment: ModelElement, Environment
Name : String = MyEnvironment
connection
modelElement
modelElement
InstanceStorageConnection: Connection
Name : String = Instance/Storage Relation
connection
connection
modelElement
connection
InstanceEnvironmentConnection: Connection
Name : String = Instance/Environment Relation

Figure 5-7 Object Diagram based on SCE1-CPROVEE001 Submit request deployment

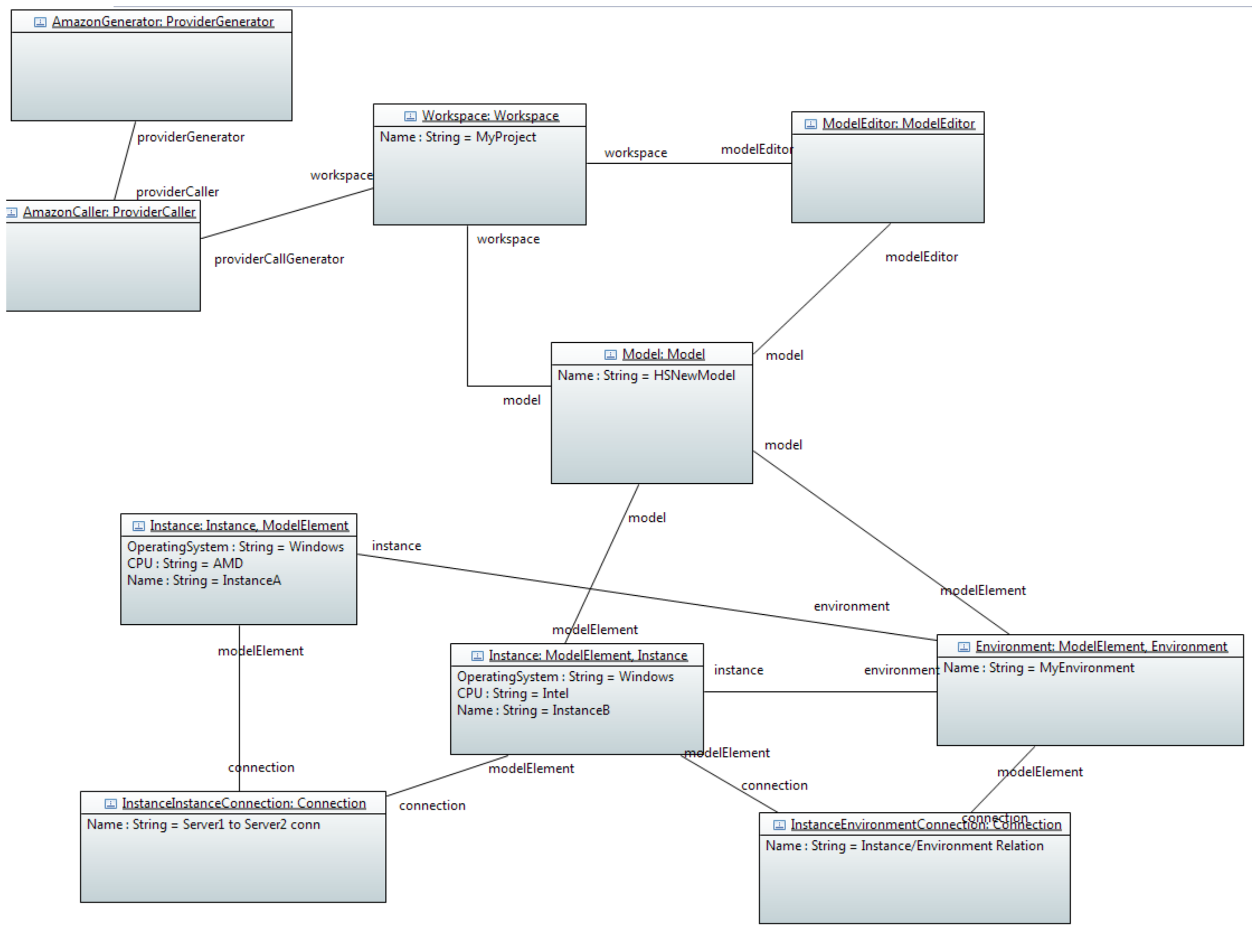

AmazonGenerator: ProviderGenerator
providerGenerator
providerCaller
AmazonCaller: ProviderCaller
providerCallGenerator
workspace
Workspace: Workspace
Name : String = MyProject
workspace
modelEditor
ModelEditor: ModelEditor
workspace
modelEditor
Model: Model
Name : String = HSNewModel
model
model
model
model
Instance: Instance, ModelElement
OperatingSystem : String = Windows
CPU : String = AMD
Name : String = InstanceA
instance
environment
modelElement
modelElement
modelElement
Instance: ModelElement, Instance
OperatingSystem : String = Windows
CPU : String = Intel
Name : String = InstanceB
instance
environment
Environment: ModelElement, Environment
Name : String = MyEnvironment
connection
modelElement
modelElement
connection
modelElement
InstanceInstanceConnection: Connection
Name : String = Serverl to Server2 conn
connection
InstanceEnvironmentConnection: Connection
connection
Name : String = Instance/Environment Relation

Figure 5-8 Object Diagram based on SCE2-CPROVEE001 Submit request deployment

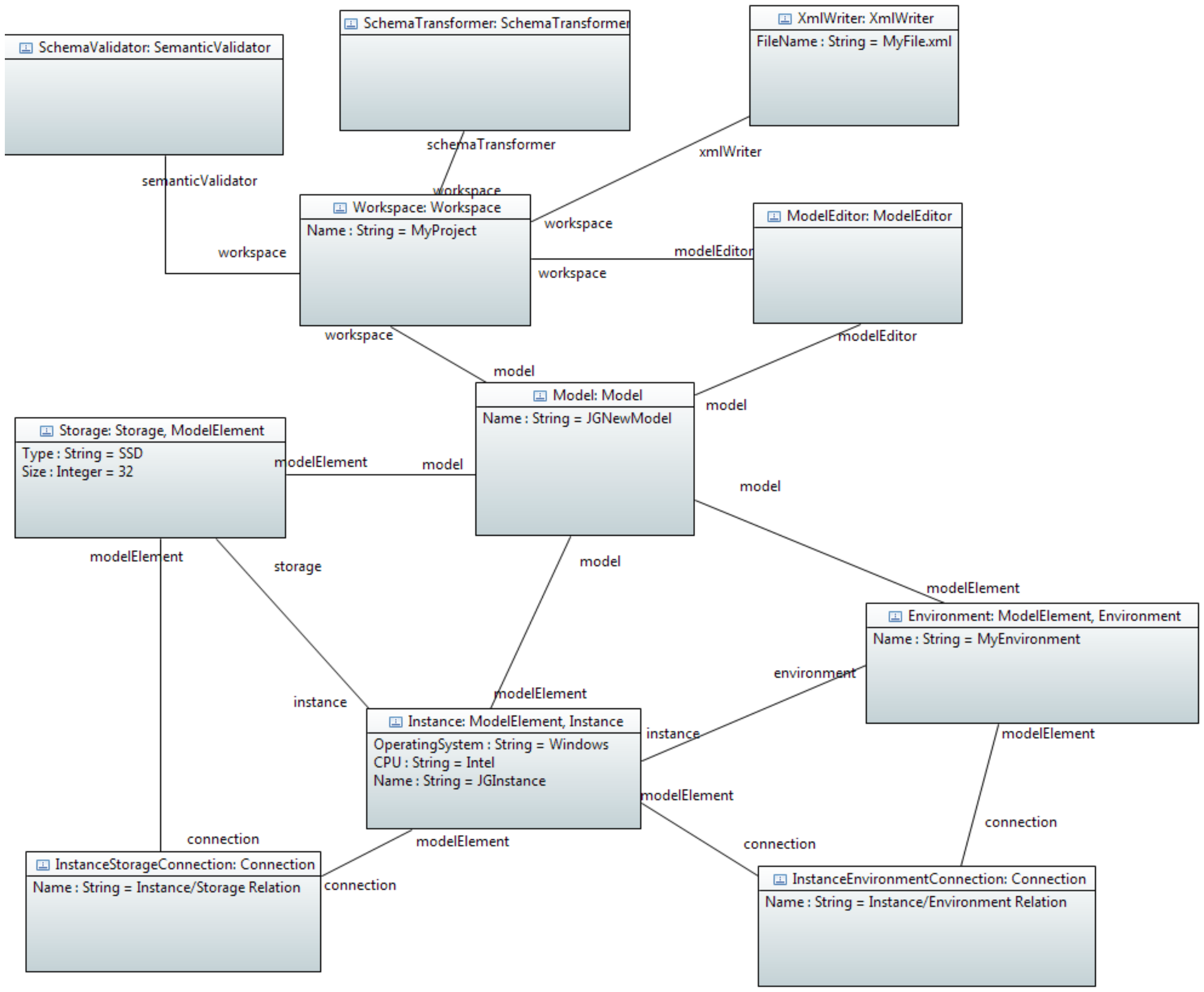

SchemaValidator: SemanticValidator
SchemaTransformer: SchemaTransformer
XmlWriter: XmlWriter
FileName : String = MyFile.xml
schemaTransformer
xmlWriter
semanticValidator
workspace
Workspace: Workspace
Name : String = MyProject
workspace
workspace
ModelEditor: ModelEditor
modelEditor
workspace
workspace
modelEditor
model
Model: Model
Name : String = JGNewModel
model
Storage: Storage, ModelElement
Type : String = SSD
Size : Integer = 32
modelElement
model
model
modelElement
storage
model
modelElement
Environment: ModelElement, Environment
Name : String = MyEnvironment
environment
instance
modelElement
Instance: ModelElement, Instance
OperatingSystem : String = Windows
CPU : String = Intel
Name : String = JGInstance
instance
modelElement
modelElement
connection
connection
modelElement
connection
InstanceStorageConnection: Connection
Name : String = Instance/Storage Relation
connection
InstanceEnvironmentConnection: Connection
Name : String = Instance/Environment Relation

Figure 5-9 Object Diagram based on SCE1-CPROVEE002 transform to CPROV XML

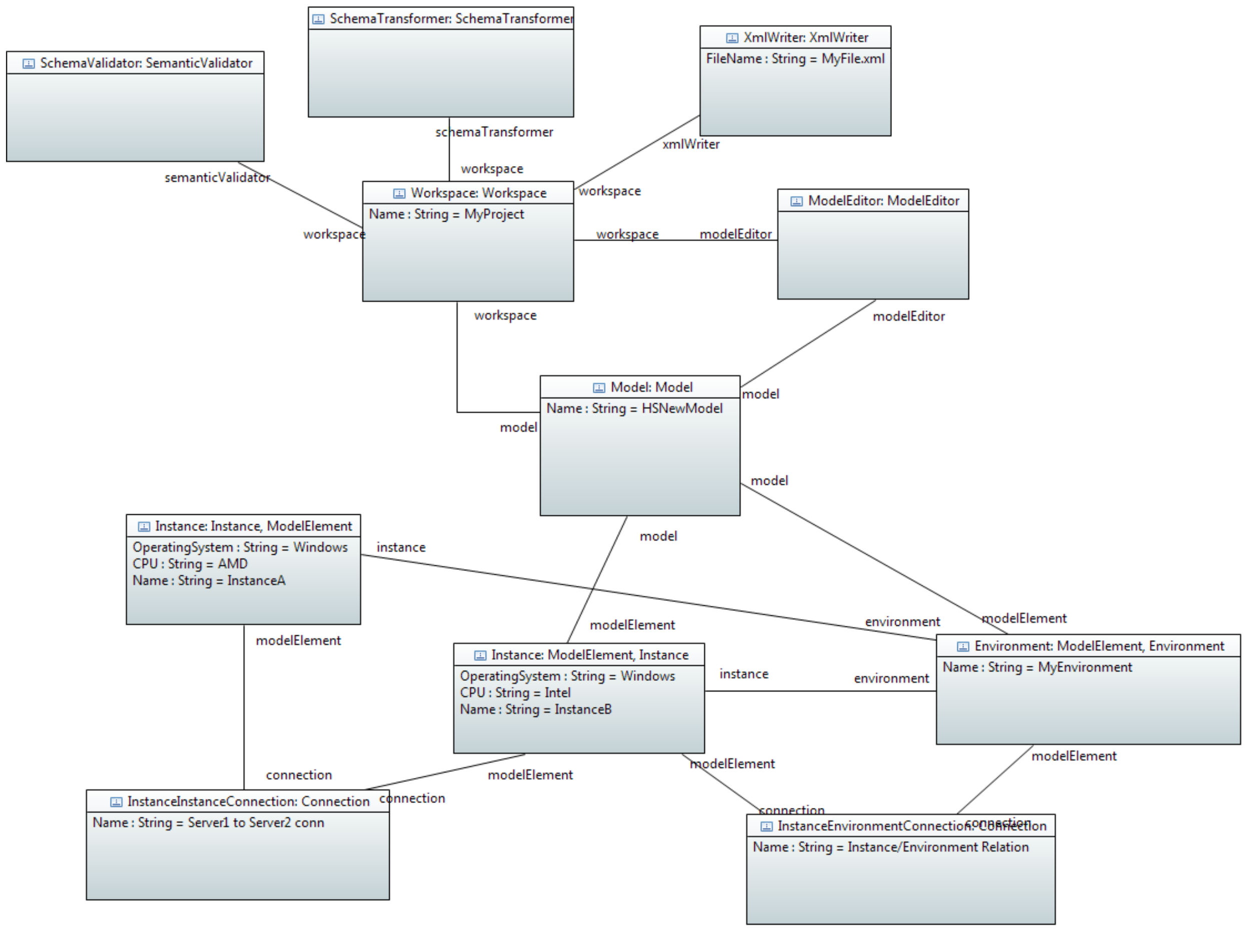

SchemaTransformer: SchemaTransformer
XmlWriter: XmlWriter
FileName : String = MyFile.xml
SchemaValidator: SemanticValidator
schemaTransformer
workspace
xmlWriter
semanticValidator
workspace
Workspace: Workspace
Name : String = MyProject
workspace
workspace
modelEditor
ModelEditor: ModelEditor
workspace
modelEditor
Model: Model
Name : String = HSNewModel
model
model
model
model
Instance: Instance, ModelElement
OperatingSystem : String = Windows
CPU : String = AMD
Name : String = InstanceA
instance
modelElement
environment
modelElement
modelElement
Instance: ModelElement, Instance
OperatingSystem : String = Windows
CPU : String = Intel
Name : String = InstanceB
instance
environment
Environment: ModelElement, Environment
Name : String = MyEnvironment
modelElement
modelElement
connection
modelElement
InstanceInstanceConnection: Connection
Name : String = Server1 to Server2 conn
connection
connection
InstanceEnvironmentConnection: Connection
connection
Name : String = Instance/Environment Relation

Figure 5-10 Object Diagram based on SCE2-CPROVEE002 transform to CPROV XML

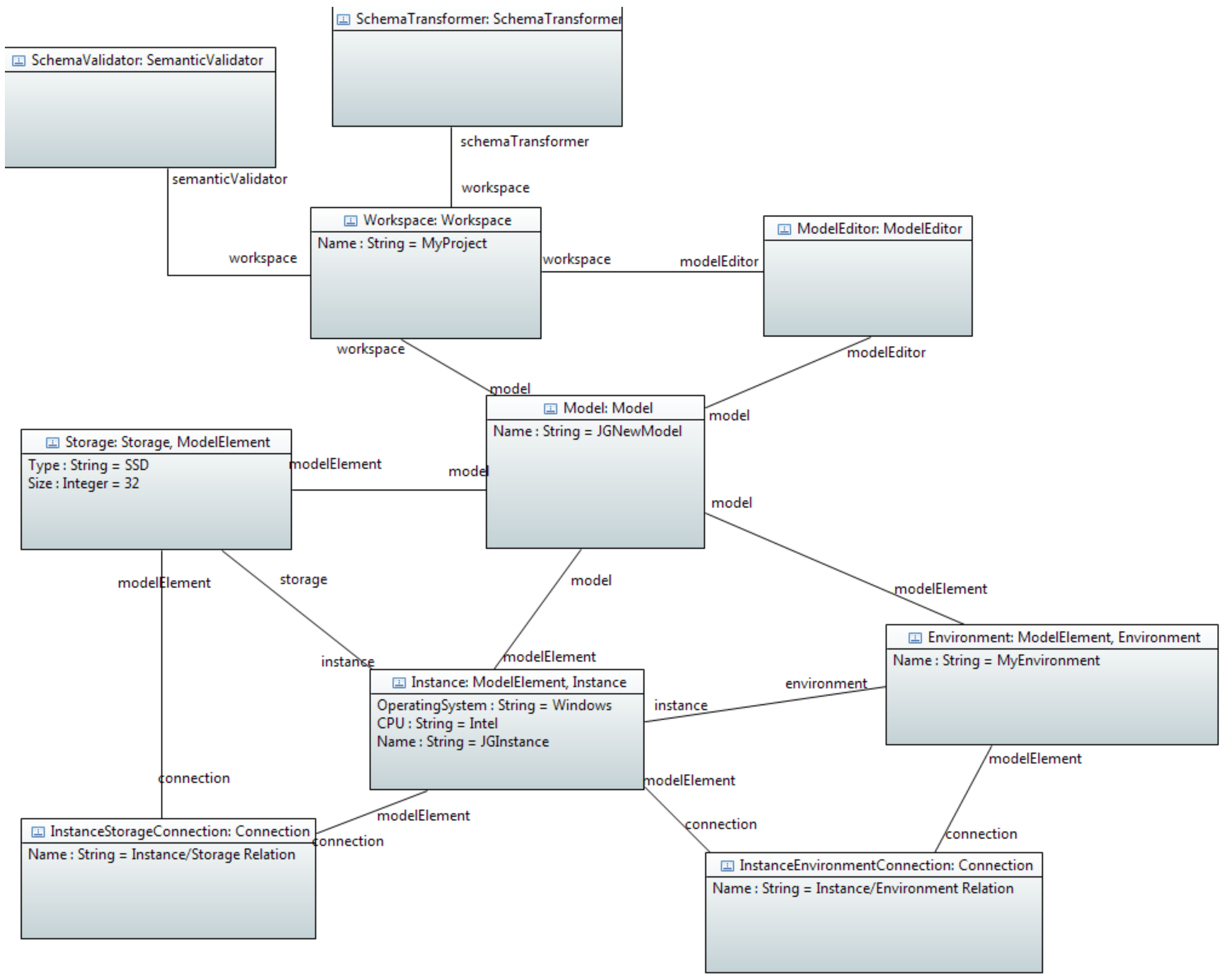

SchemaTransformer: SchemaTransformer
SchemaValidator: SemanticValidator
schemaTransformer
semanticValidator
workspace
workspace
Workspace: Workspace
Name : String = MyProject
workspace
modelEditor
ModelEditor: ModelEditor
workspace
modelEditor
model
Model: Model
Name : String = JGNewModel
model
Storage: Storage, ModelElement
Type : String = SSD
Size : Integer = 32
modelElement
model
model
modelElement
storage
model
modelElement
instance
modelElement
Environment: ModelElement, Environment
Name : String = MyEnvironment
Instance: ModelElement, Instance
OperatingSystem : String = Windows
CPU : String = Intel
Name : String = JGInstance
environment
instance
modelElement
connection
modelElement
modelElement
connection
InstanceStorageConnection: Connection
Name : String = Instance/Storage Relation
connection
connection
InstanceEnvironmentConnection: Connection
Name : String = Instance/Environment Relation

Figure 5-11 Object Diagram based on SCE1-CPROVEE003 Validation against Cloud Provisioning Schema

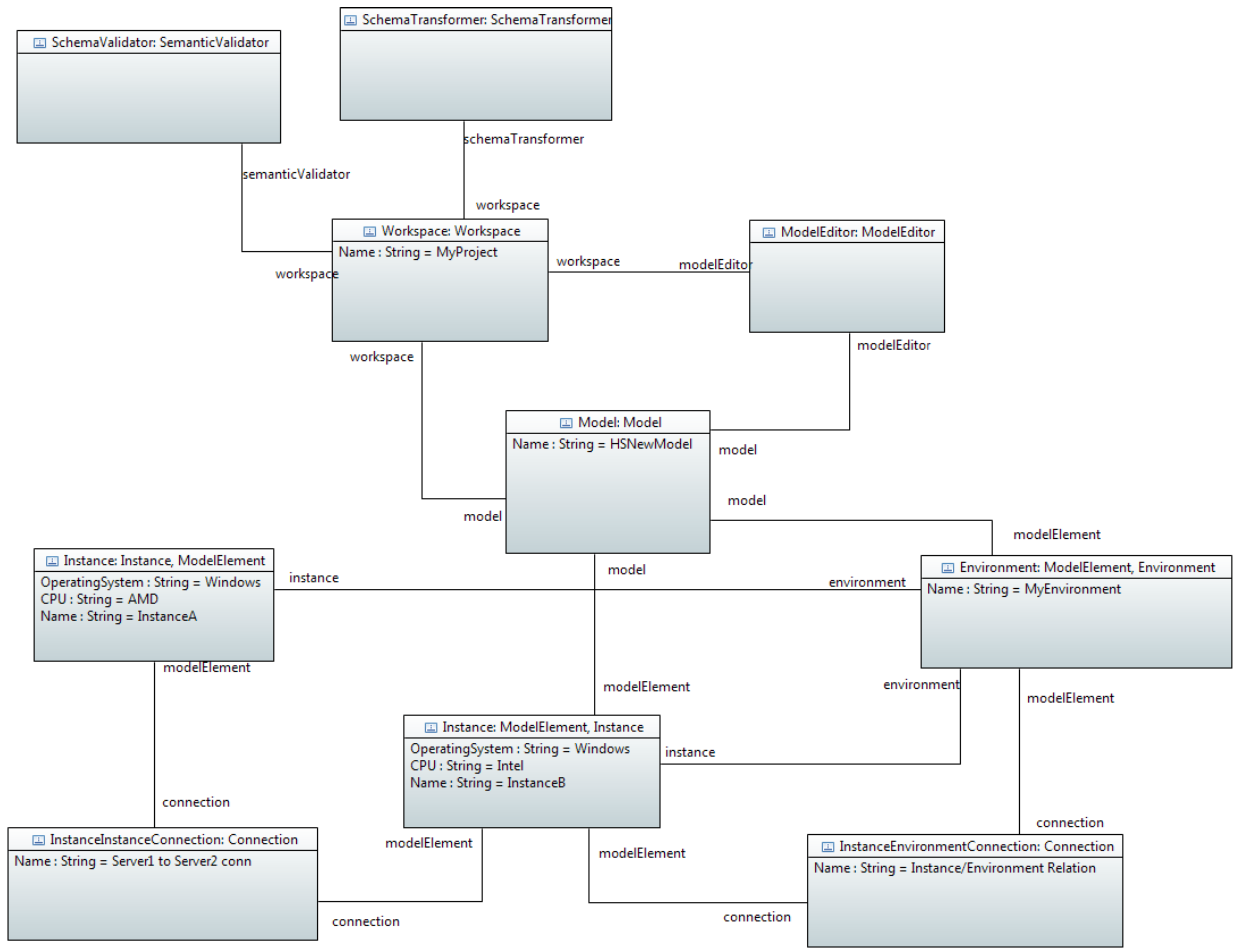

SchemaValidator: SemanticValidator
SchemaTransformer: SchemaTransformer
schemaTransformer
semanticValidator
workspace
Workspace: Workspace
Name : String = MyProject
workspace
workspace
modelEditor
ModelEditor: ModelEditor
workspace
modelEditor
Model: Model
Name : String = HSNewModel
model
model
model
modelElement
Instance: Instance, ModelElement
OperatingSystem : String = Windows
CPU : String = AMD
Name : String = InstanceA
instance
model
environment
Environment: ModelElement, Environment
Name : String = MyEnvironment
modelElement
modelElement
environment
modelElement
Instance: ModelElement, Instance
OperatingSystem : String = Windows
CPU : String = Intel
Name : String = InstanceB
instance
connection
connection
InstanceInstanceConnection: Connection
Name : String = Server1 to Server2 conn
modelElement
modelElement
InstanceEnvironmentConnection: Connection
Name : String = Instance/Environment Relation
connection
connection

Figure 5-12 Object Diagram based on SCE2-CPROVEE003 Validation against Cloud provisioning Schema

Class Diagram

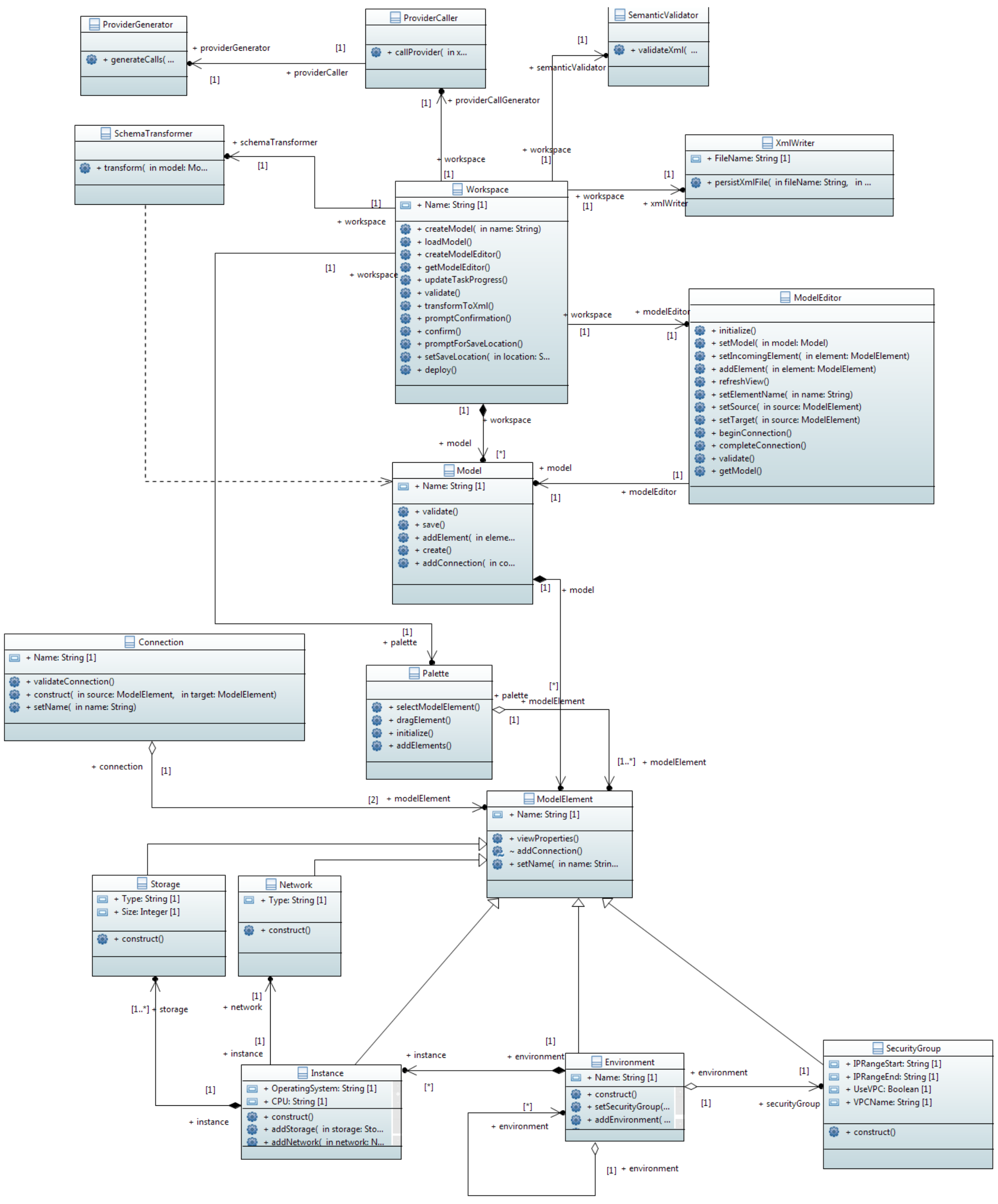


Figure 5-13 Class Diagram for our CProv System

5.3 Dynamic Model

<u>Sequence Diagrams</u>

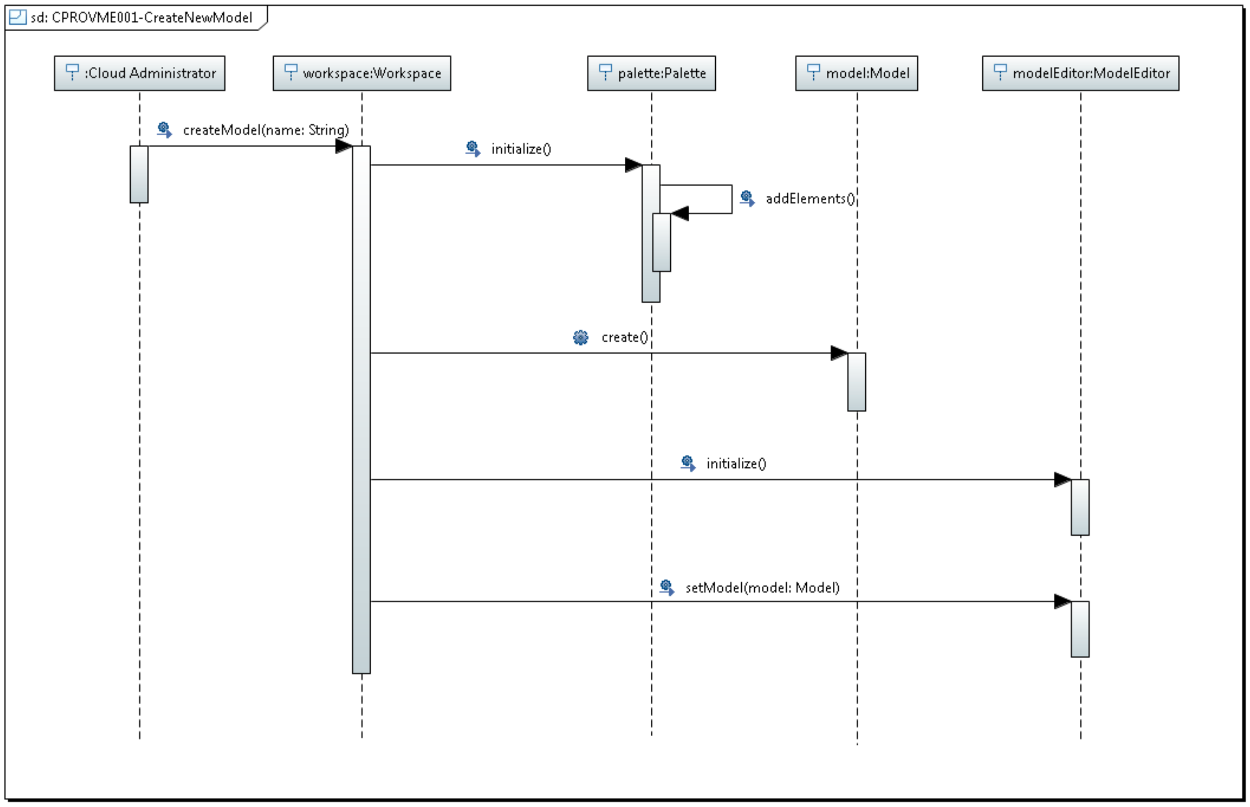


Figure 5-14 Sequence Diagram for use case CProv-CPROVME001-CreateNewModel

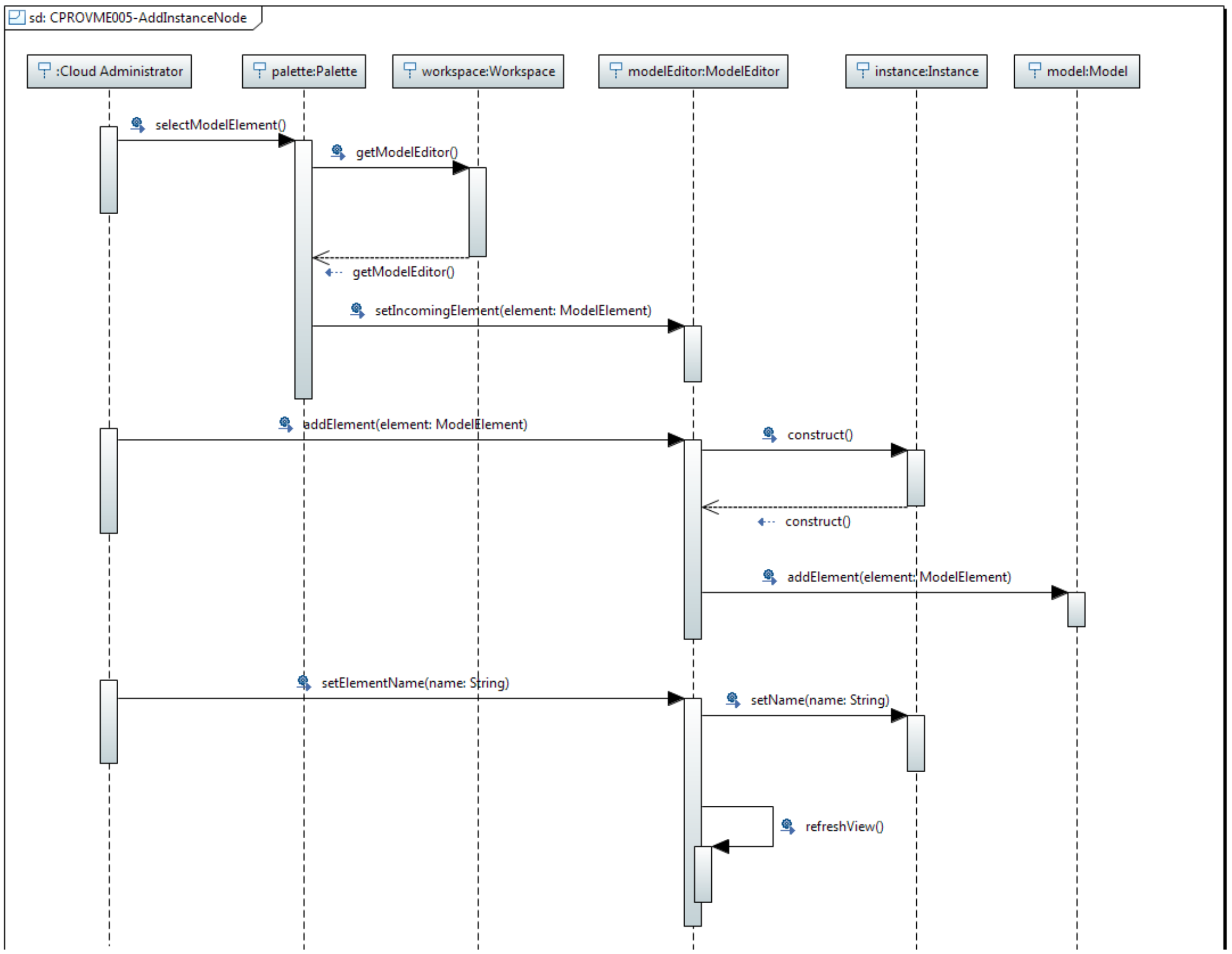


Figure 5-15 Sequence Diagram for use case CProv-CPROVME005-AddInstanceNode

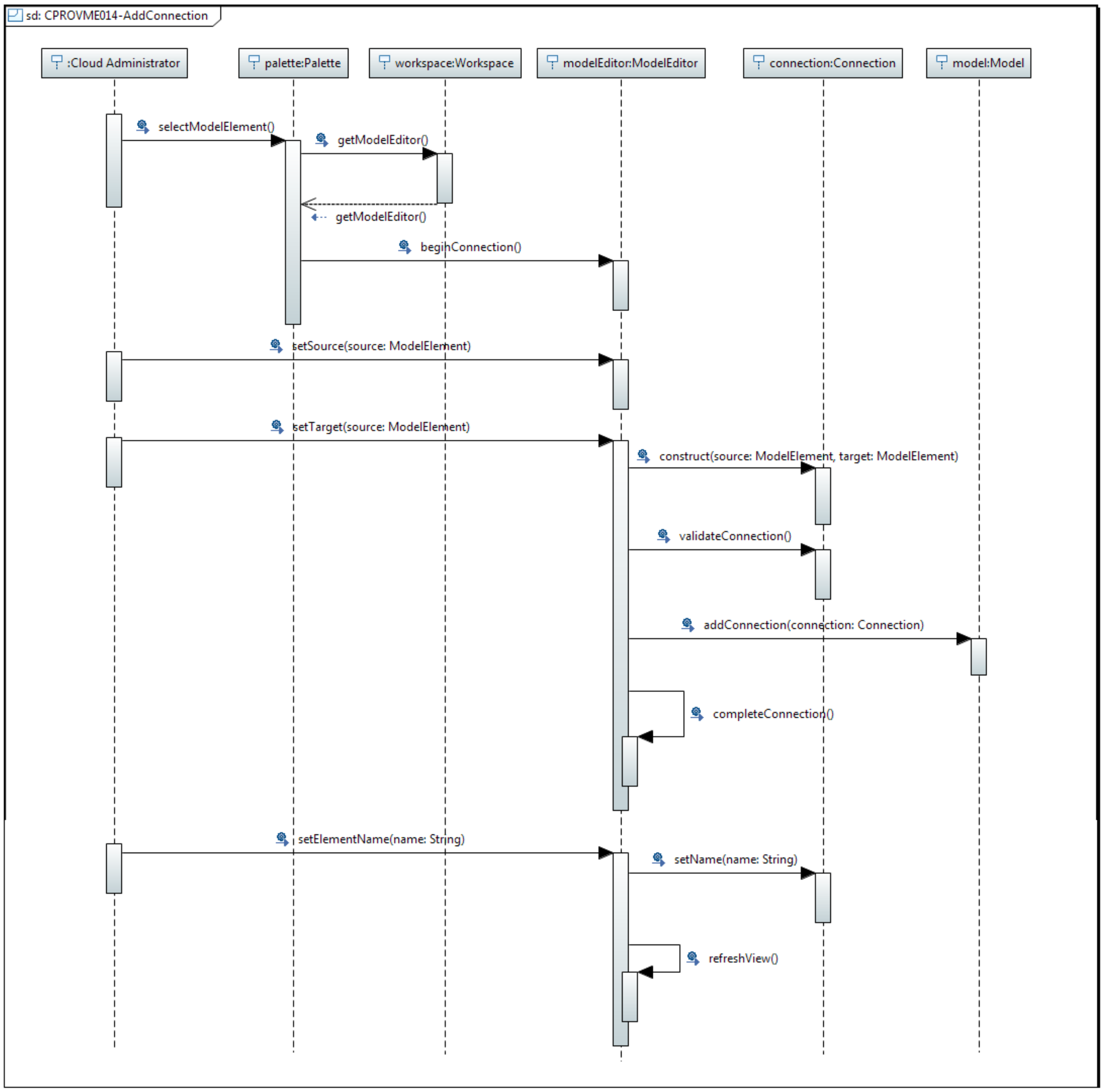


Figure 5-16 Sequence Diagram for use case CProv-CPROVME014-AddConnection

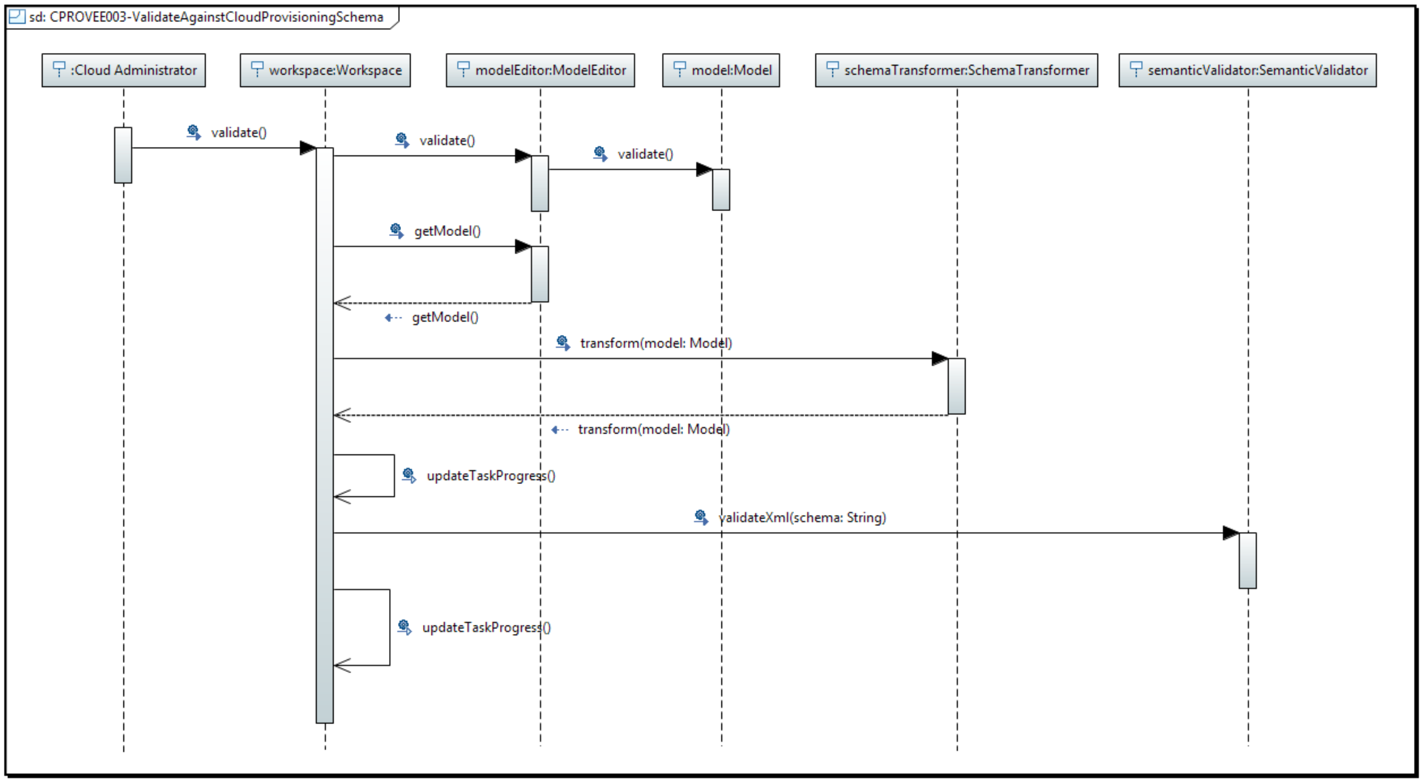


Figure 5-17 Sequence Diagram for use case CProv-CPROVEE003-ValidateAgainst CloudProvisioningSchema

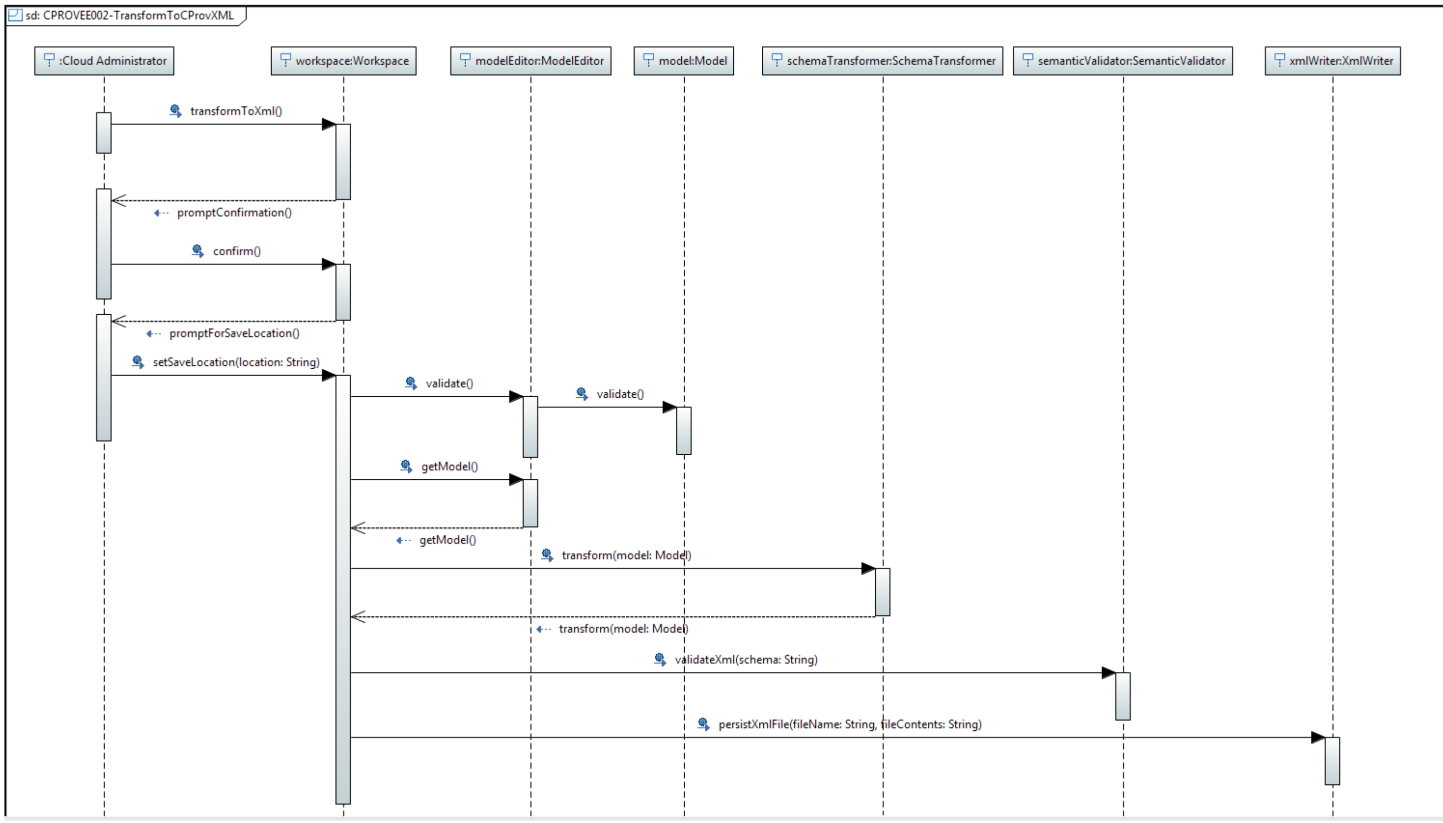


Figure 5-18 Sequence Diagram for use case CProv-CPROVEE002-TransformToCProvXML

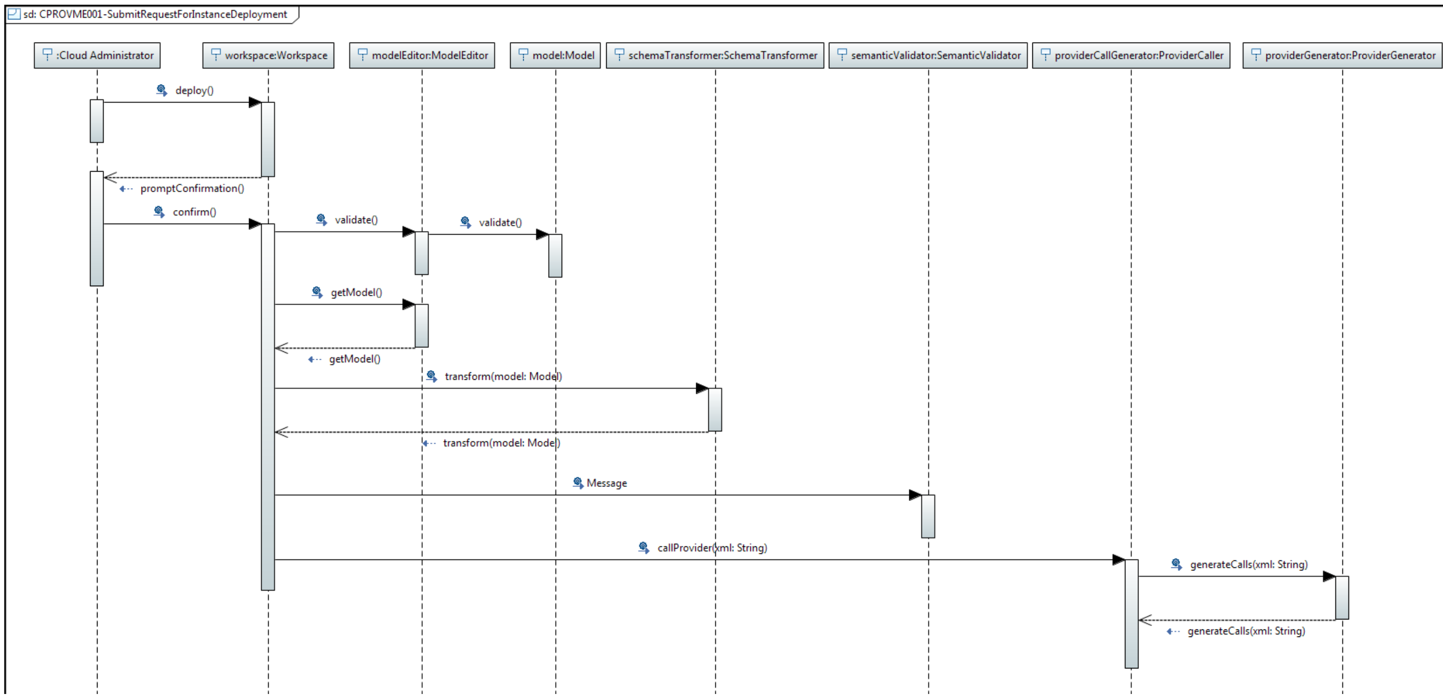

sd: CPROVME001-SubmitRequestForInstanceDeployment
:Cloud Administrator
workspace:Workspace
modelEditor:ModelEditor
model:Model
schemaTransformer:SchemaTransformer
semanticValidator:SemanticValidator
providerCallGenerator:ProviderCaller
providerGenerator:ProviderGenerator
deploy()
promptConfirmation()
confirm()
validate()
validate()
getModel()
getModel()
transform(model: Model)
transform(model: Model)
Message
callProvider(xml: String)
generateCalls(xml: String)
generateCalls(xml: String)

Figure 5-19 Sequence Diagram for use case CProv-CPROVME001-SubmitRequestForInstanceDeployment

State Machine Diagrams:

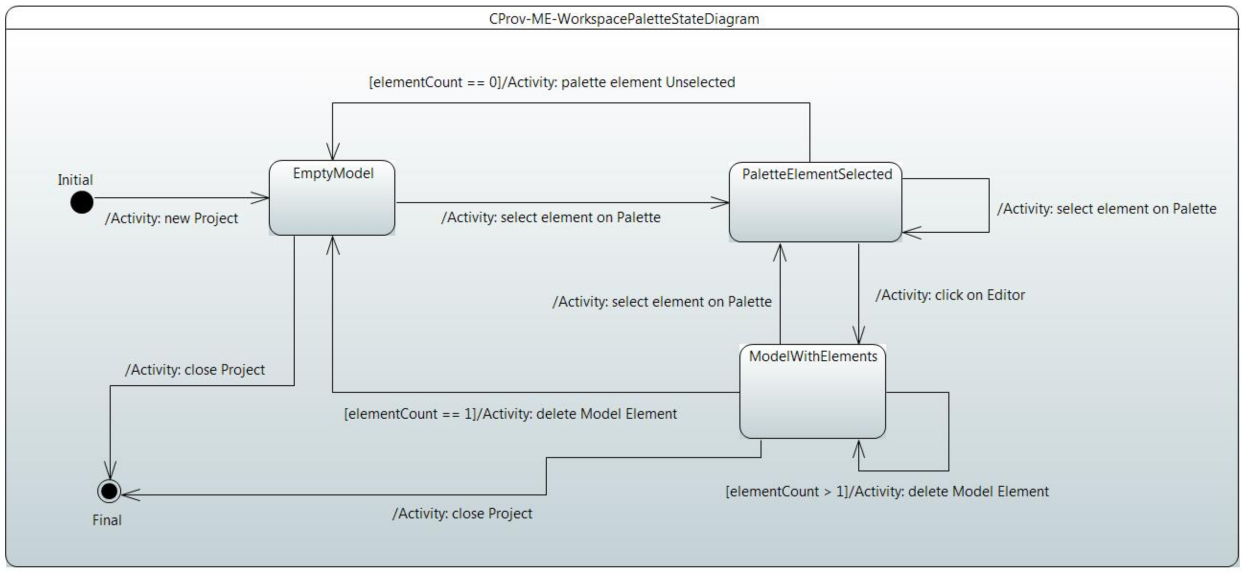

CProv-ME-WorkspacePaletteStateDiagram
[elementCount == 0]/Activity: palette element Unselected
Initial
EmptyModel
PaletteElementSelected
/Activity: select element on Palette
/Activity: new Project
/Activity: select element on Palette
/Activity: select element on Palette
/Activity: click on Editor
/Activity: close Project
ModelWithElements
[elementCount == 1]/Activity: delete Model Element
[elementCount > 1]/Activity: delete Model Element
/Activity: close Project
Final

Figure 5-20 State Machine Diagram for the Workspace Palette

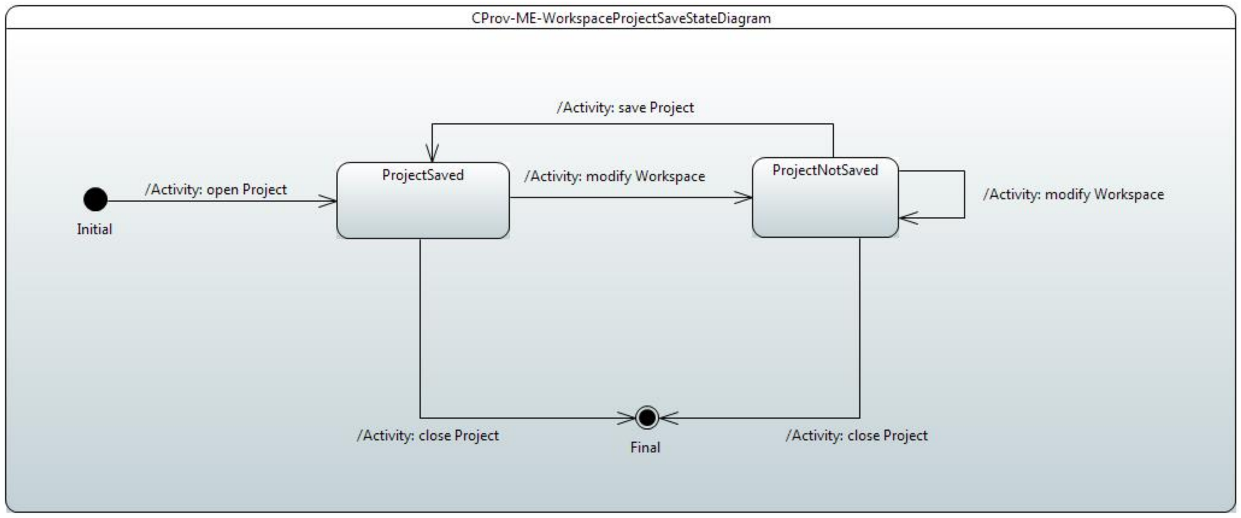


Figure 5-21 State Machine Diagram for Project save Management

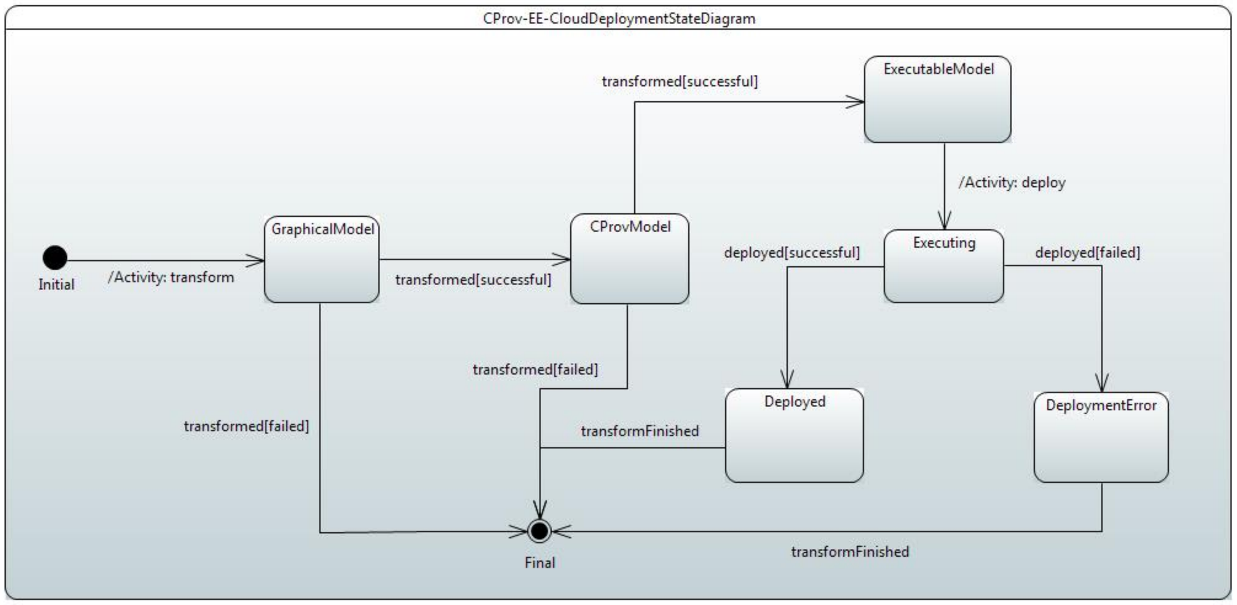


Figure 5-22 State Machine Diagram for Cloud Deployment (Execution Engine)

Activity Diagrams

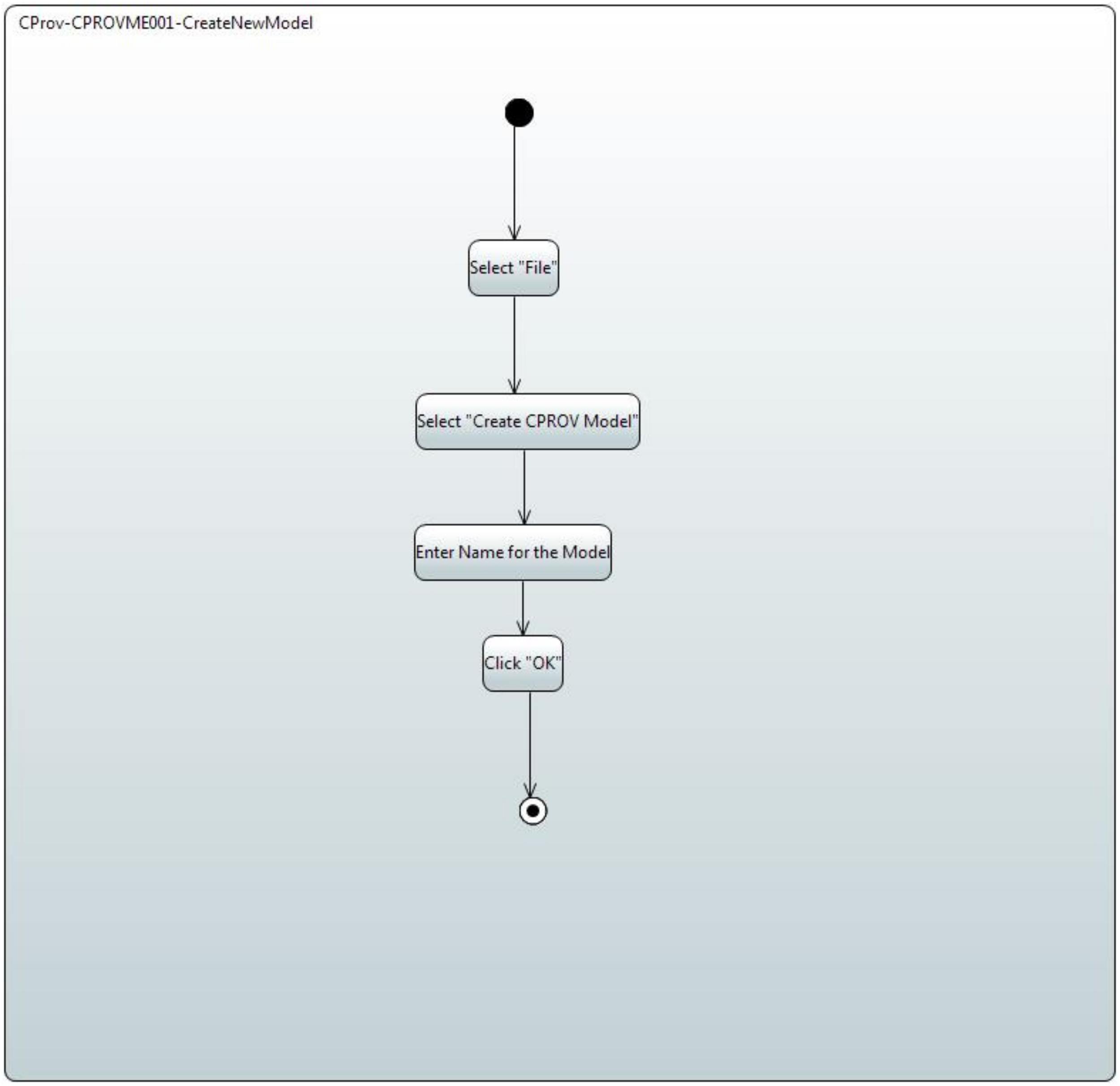


Figure 5-23 Activity Diagram for use case CProv-CPROVME001-CreateNewModel

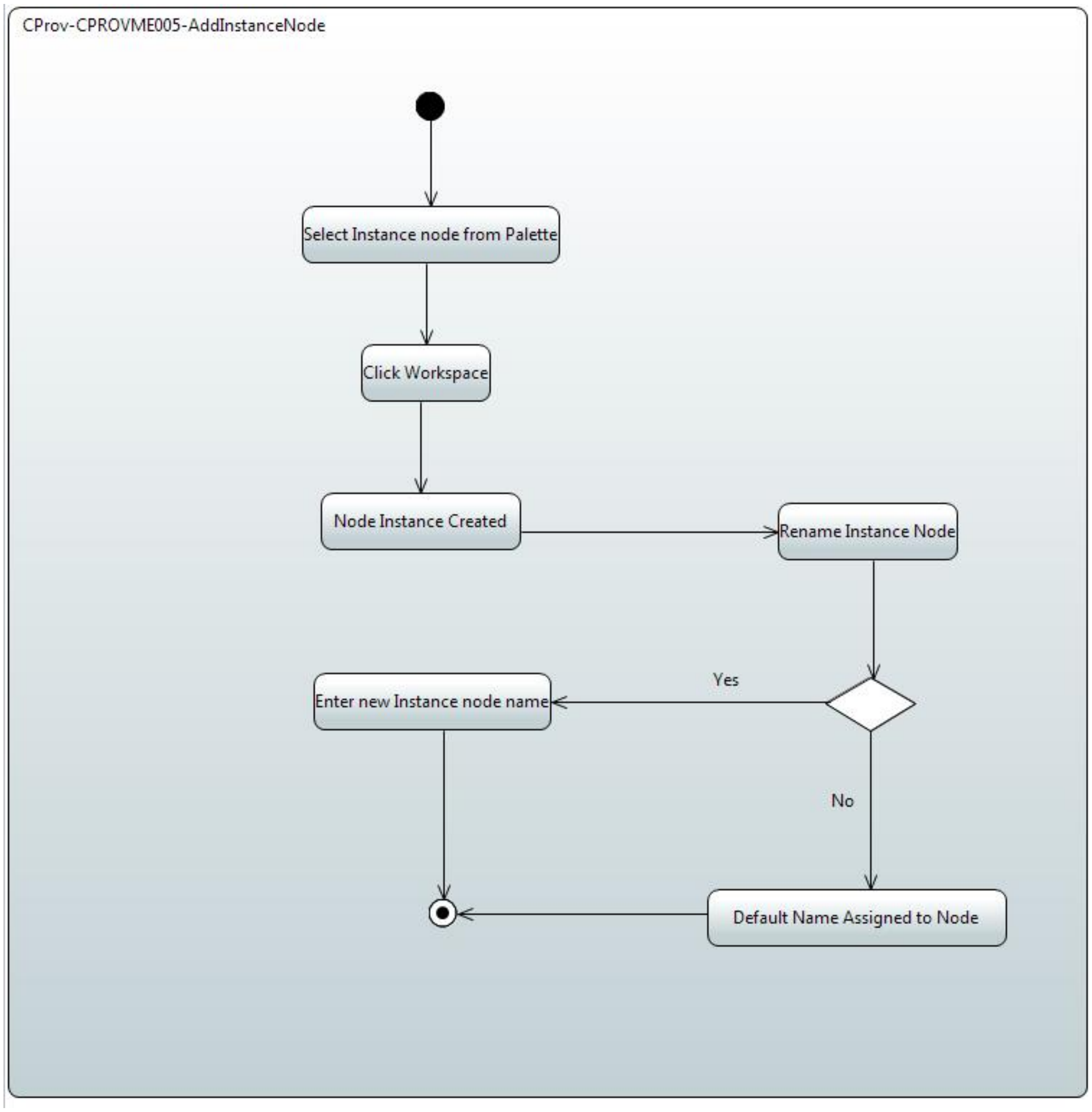

CProv-CPROVME005-AddInstanceNode
Select Instance node from Palette
Click Workspace
Node Instance Created
Rename Instance Node
Yes
Enter new Instance node name
No
Default Name Assigned to Node

Figure 5-24 Activity Diagram for use case CProv-CPROVME005-AddInstanceNode

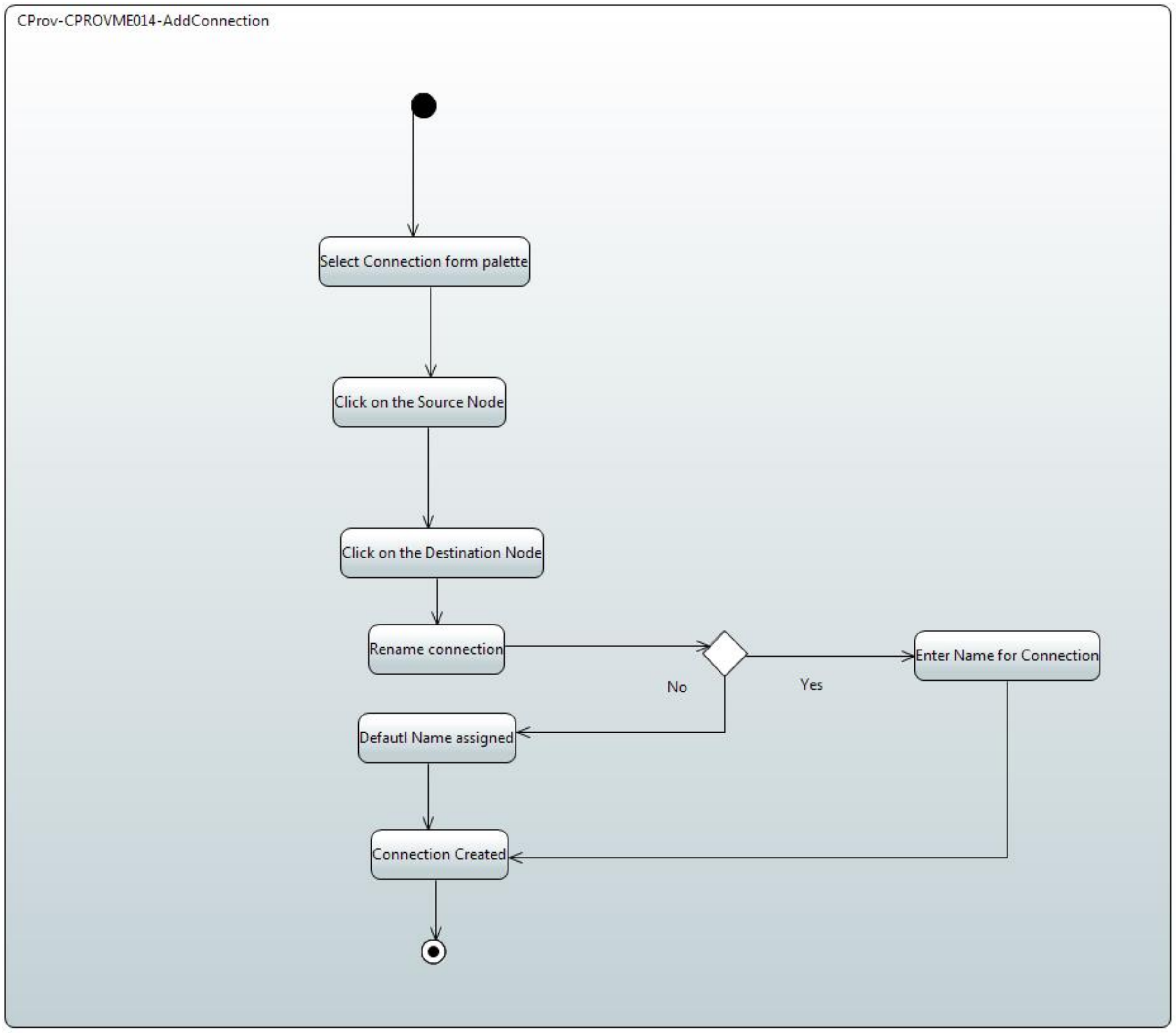

CProv-CPROVME014-AddConnection
Select Connection form palette
Click on the Source Node
Click on the Destination Node
Rename connection
No
Yes
Enter Name for Connection
Defautl Name assigned
Connection Created

Figure 5-25 Activity Diagram for use case CProv-CPROVME014-AddConnection

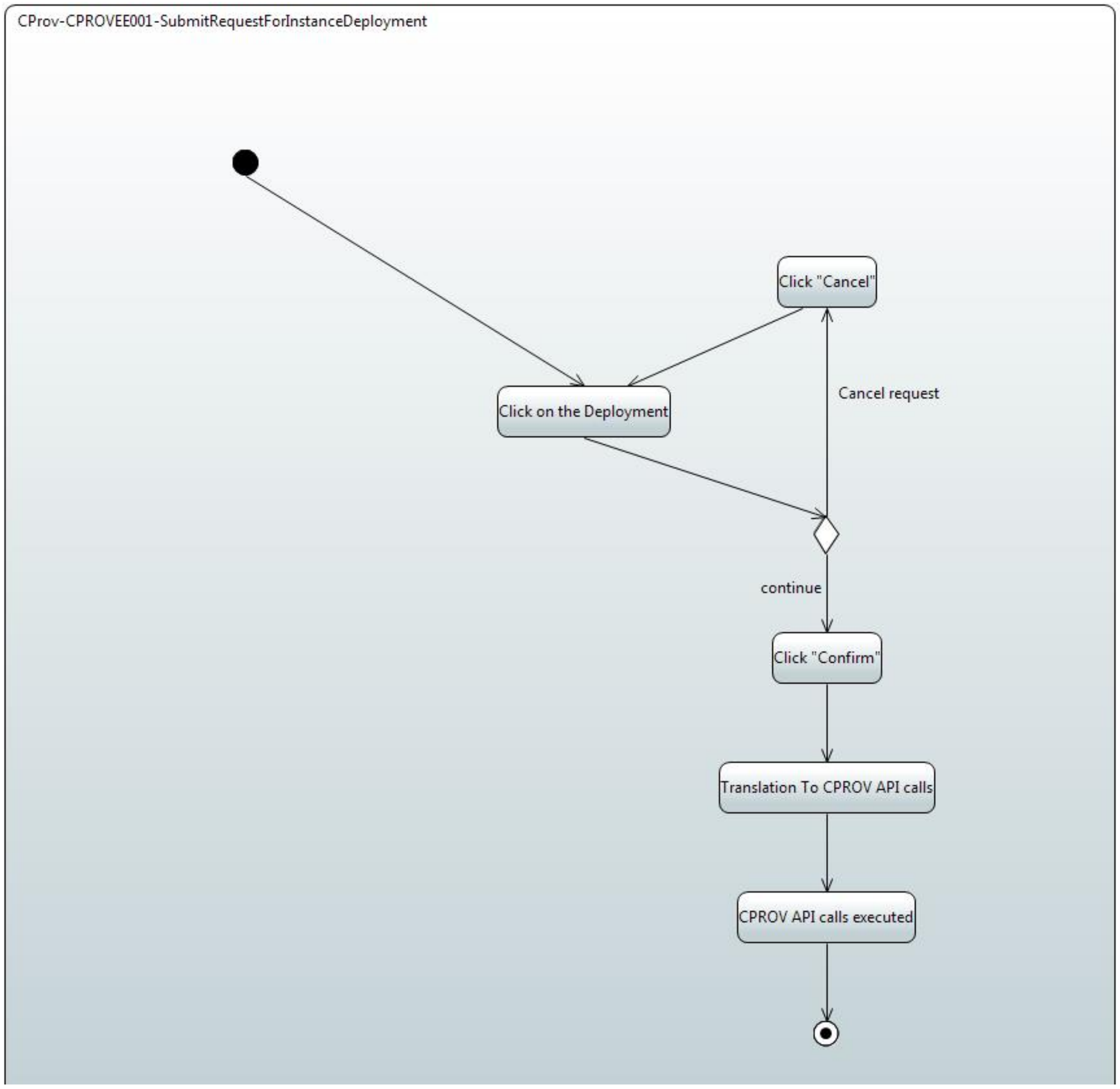


Figure 5-26 Activity Diagram for use case CProv-CPROVEE001-SubmitRequestDeployment

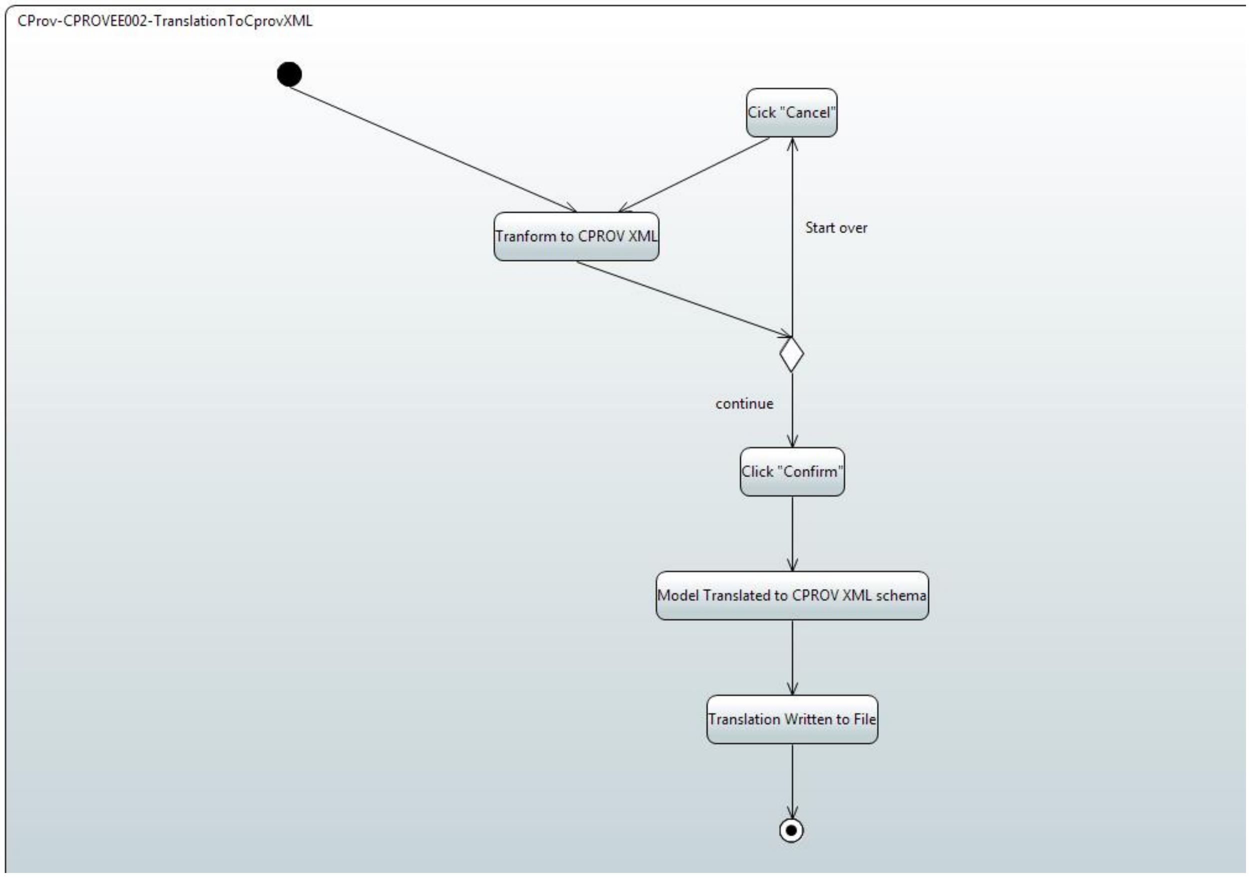

CProv-CPROVEE002-TranslationToCprovXML
Cick "Cancel"
Tranform to CPROV XML
Start over
continue
Click "Confirm"
Model Translated to CPROV XML schema
Translation Written to File

Figure 5-27 Activity Diagram for use case CProv-CPROVEE002-TranslationToCProvXML

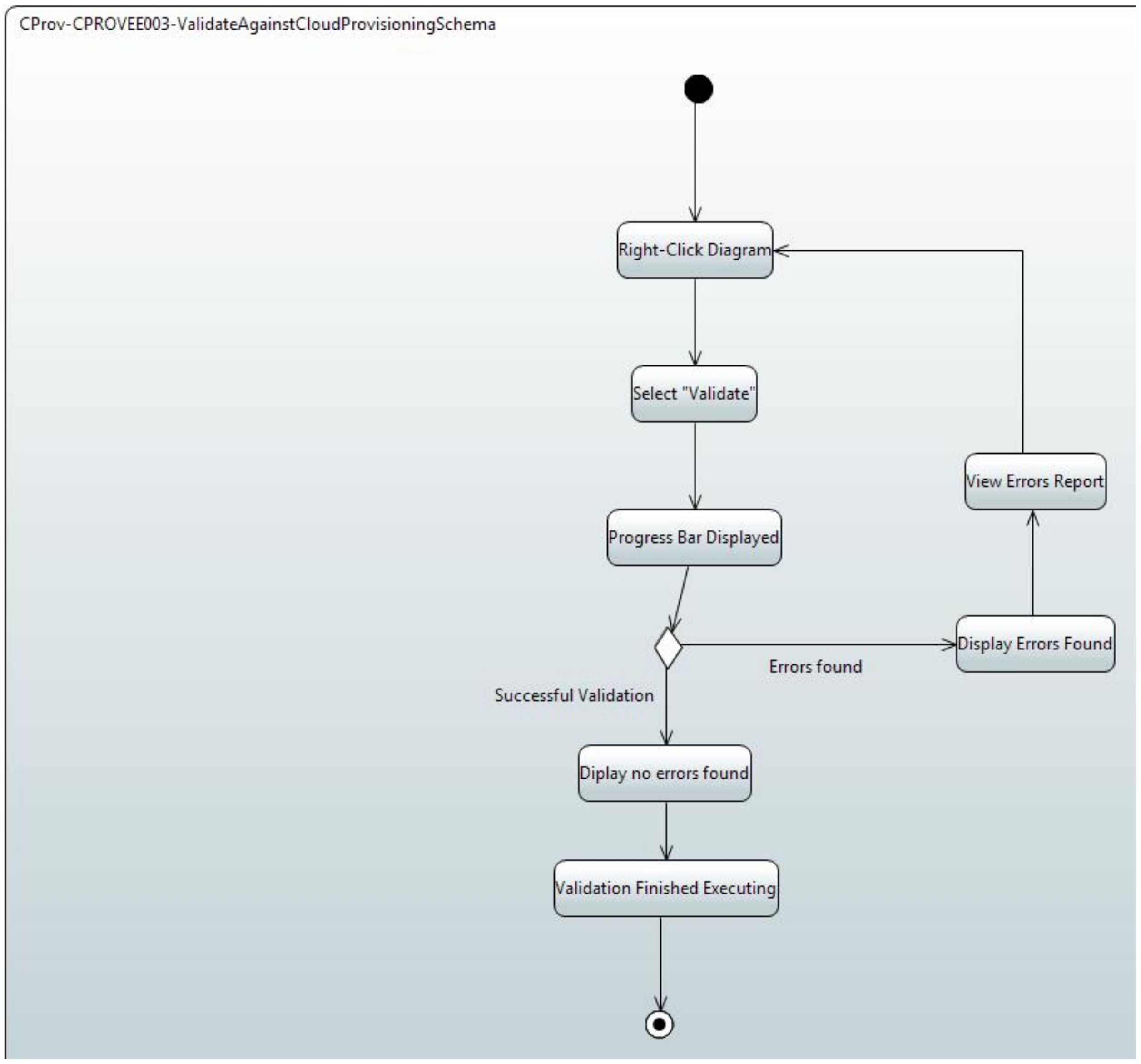


Figure 5-28 Activity Diagram for use case CProv-CPROVEE003-ValidationAgainstCProvSchema

5.4 User Interfaces

Refer to Appendix D for interface and GUI mock up designs.

6 Glossary

Class Diagram – A model representing the different classes within a software system and how they interact with each other.

Component – A physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces.

Model – an abstract representation of a system that enables us to answer questions about the system.

Post condition – A predicate that must be true after an operation is invoked.

Precondition – A predicate that must be true before an operation is invoked.

Sequence Diagram – A model representing the different objects and/or subsystems of a software project and how they relate to each other during different operations for a given use case.

Unified Modeling Language (UML) – A standard set of notations for representing models.

Use Case – A general sequence of events that defines all possible actions between one or many actors and the system for a given piece of functionality.

Model Driven Development Software – Software development paradigm that roots in software product line engineering, which is the discipline of designing and building families of applications.

Activity Diagram - A representation of actions, and activities that comprise a communication model that involve the usage of decision points that allow greater flexibility in describing models which can support more than one communication instance.

Object Diagram - A diagram that shows a complete or partial view of the structure of a modeled system or portion thereof.

Shape – An image that can be dragged into the canvas in the Visual Environment. It can be connected with other nodes using connections in order to build a communication model.

Connections – A line that connects one shape with another.

Participant – A special shape that represents a local or remote user of the communication system. One or more participants can establish a conversation using a communication model.

Network – A special shape that represents what the user is using to connect to the system.

Security Group – A special shape that represents what the user is using to connect to the system.

Storage – A special shape that represents a hard drive on an instance or outside an instance server.

Instance – A special shape that represents a server in the cloud.

Cloud – a term to define a group or single computer whose location is unimportant, but that the user can connect to and control.

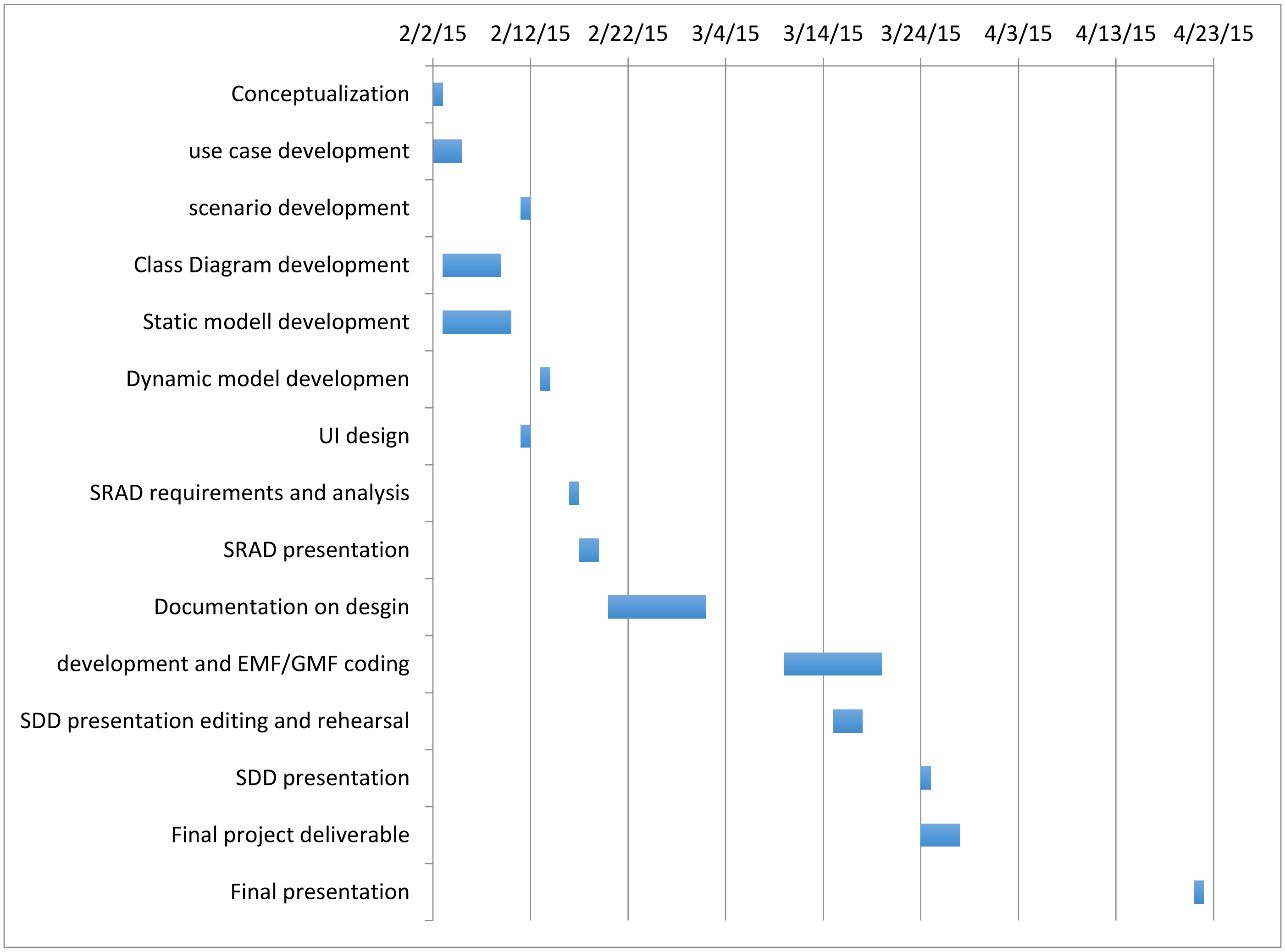
Palette – A location on the GUI where are the shapes are aggregated for the user to click and drag.

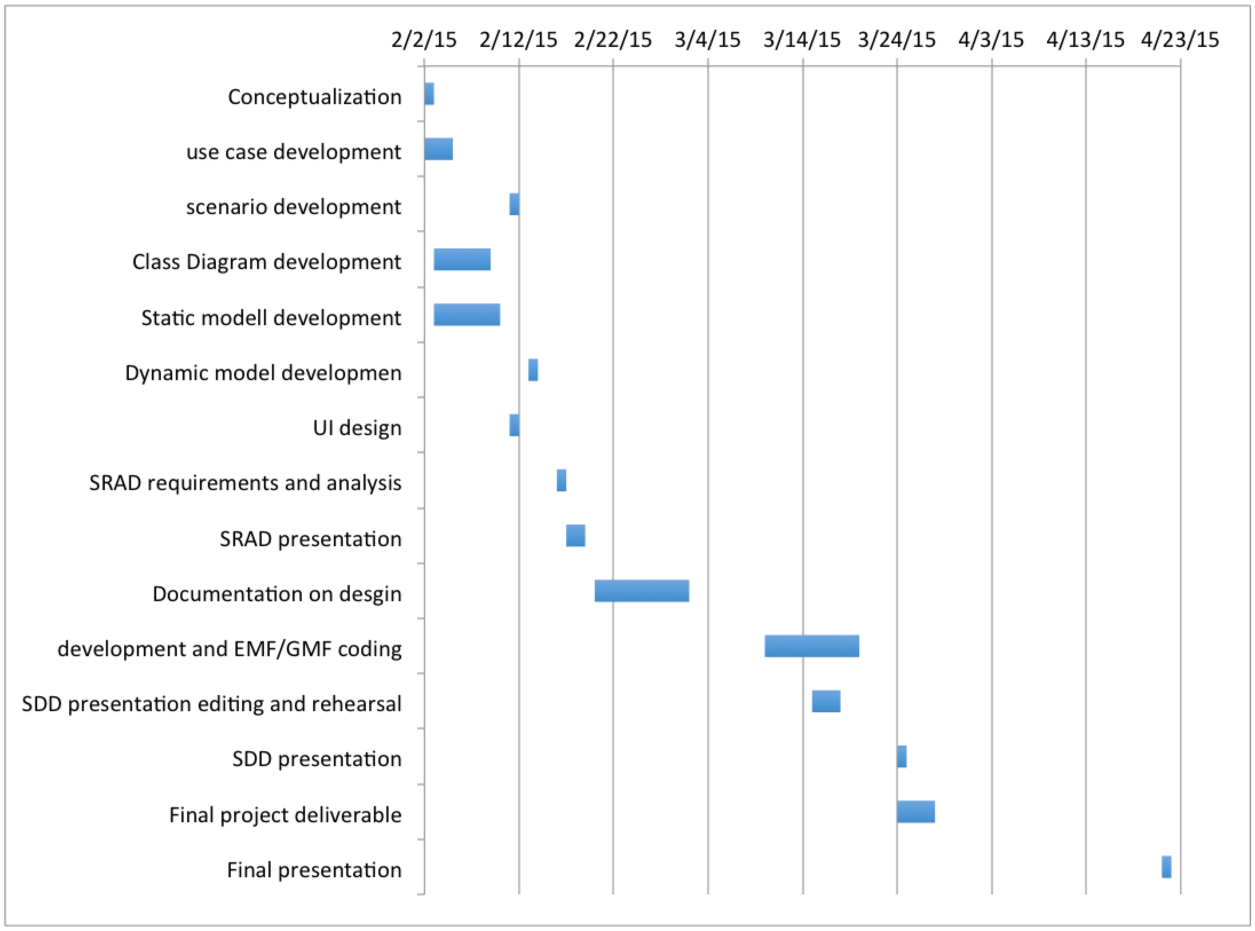
Workspace – The area where the user drags shapes and connections to build a communication model.

Constraints – A voluntary limitation imbedded into the design to obey syntax or semantic rule(s).

7 Appendix

7.1 *Appendix A – Project Schedule (Gantt chart, PERT chart)*





7.2 Appendix B – Use cases

Execution Engine Use Cases

<u>CPROV-EE Actors</u>

- Cloud Administrator - A cloud provisioning administrator who is interested in modeling a particular deployment.

CPROV-EE Use Cases

Use Case ID: **CPROVEE001 Submit request for instance deployment**

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visual environment must be running on the system.
2. A model must be active and must contain an instance of a server.

Description:

1. Use case begins when the user clicks on the deploy button on the toolbar.
2. The system shall provide the user a confirmation message with two options: Confirm/Cancel.
3. The user clicks on the Confirm button on the confirmation message.
4. The system translates the graphical representation of the server instance into a series of commands that utilize the Amazon EC2 API
5. The system executes the API calls to the Amazon EC2 service

Post-conditions:

1. A message appears stating that the deployment was successful.
2. The system returns focus to the graphical elements present on the workspace.

Alternative Courses of Action:

1. In step D.3 (Step 3 of Descriptions section), the user has the option to cancel the deployment of the selected diagram element present on the workspace.

Exceptions:

One or more element of the diagram cannot be translated. In this case an error message is returned to the user.

Related Use Cases:

CPROVME001 adding a Instance Node.

Decision Support

Frequency:

Each time the main user would like to deploy the diagram.
On average once per model.

Criticality:
High, it allows the user to establish the essence of the cloud deployment which provide virtual machines, and servers.

Risk:

Risk: medium risk, the system must communicate with a third party API and adhere to its contract. Since the API is always subject to change, our system must be maintained.

Constraints:

1. Usability:

   a) No previous Training Time

   b) On average the user should be able to submit a deployment request in under 1 minute.

2. Reliability:

   a) Mean time to Failure – 5% failures for every 1000 attempts to add an instance.

   b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

   a) Deployment of the current workspace diagram should complete in under one minute.

   b) System handles only one request for deployment at a time.

4. Supportability:

   a) The execution environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

5. Implementation:

   a) The software will be implemented via a Eclipse plugin that the user can install and run..

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVEE002 Transform to CPROV XML**

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.
2. A model must be active and must contain an instance of a server.
3. The model must pass validation prior to transformation.

Description:

1. Use case begins when the user clicks on the "Transform to XML" button on the toolbar.
2. The system shall provide the user a confirmation message with two options: Confirm/Cancel.
3. The user clicks on the Confirm button on the confirmation message.
4. The system prompts the user for a Save location.
5. The user types in a Save location.
6. The user clicks OK.

Post-conditions:

1. A message appears stating that the transformation to XML was successful.
2. The system returns focus to the graphical elements present on the workspace.
3. The system saves the XML schema to a file in a location designated by the user.

Alternative Courses of Action:

1. In step D.3 (step 3 of Descriptions section), the user has the option to cancel the transformation to XML by clicking on the "Cancel" button.

Exceptions:

One or more element of the diagram cannot be translated. In this case an error message is returned to the user.

Related Use Cases:

CPROVME005 Add Instance Node

--

Decision Support

Frequency: Each time the main user would like to transform the diagram into XML schema.

On average once per model.

Criticality: High, It allows the user to use save many XML schemas for use on another system, or for memory saving.

Risk:

Medium risk, the system must communicate with a third party API and adhere to its contract. Since the API is always subject to change, our system must be maintained.

Constraints:

1. Usability:
    a) No previous Training Time
    b) On average the user should be able to transform to XML request in under 1 minute.
2. Reliability:
    a) Mean time to Failure – 5% failures for every 1000 attempts to add an instance.
    b) Availability – The system shall be available as long as the user's machine is running.
3. Performance:
    a) transformation to XML of the current workspace diagram should complete in under 2 minute.
    b) System handles only one request for deployment at a time.
4. Supportability:
    a) The execution environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.
5. Implementation:
    a) The software will be implemented via a Eclipse plugin that the user can install and run..

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVEE003 Validation against Cloud provisioning Schema**

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.
2. A model must be active and must contain at least an instance of a server.

Description:

1. Trigger: the use case begins when the user clicks right clicks on the workspace.
2. In the right click drop down menu, the user clicks on the "Validate"
3. The system start to validation process with a progress bar starting at 0 % and ends at 100% showing that the validation has been done.

Post-conditions:

1. A message appears stating that the validation process reported errors or no errors.
2. The system returns focus to the graphical elements present on the workspace.

Alternative Courses of Action:

1. In step D.2 (Step 2 of Descriptions section), the user has the option select validate from the menu bar under "Validation".

Exceptions:

If one or more elements of the diagram does not respect our schema. In this case the error(s) are highlighted in the validation submenu located in the lower part of the screen.

Related Use Cases:

CPROVME005 Add Instance Node

Decision Support

Frequency: Each time the main user would like to validate the diagram.
On average once per changes applied to the model.

Criticality: High, It allows the user to verify is the diagram represents a sound model before the translation and deployment phases.

Risk: low risk, the system uses the EMF/GMF API to implement the validation.

Constraints:

1. Usability:

   a) No previous Training Time

b) On average the user should be able to validate a diagram in under 2 minute.

2. Reliability:

a) Mean time to Failure – 5% failures for every 1000 attempts to add an instance.

b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

a) Validation of the current workspace diagram should complete in under one minute.

b) System handles only one request for deployment at a time.

4. Supportability:

a) The execution environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

5. Implementation:

a) The software will be implemented via a Eclipse plugin that the user can install and run..

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVEE004** Transformation to Amazon EC2 format

Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.
2. A model must be active and must contain at least an instance of a server.

Description:
1. Trigger: the use case begins when the user clicks the “Transform” menu button location on the top menu bar.
2. in the top menu bar list, the user selects “Transform to EC2 format”

3. the system starts the validation process if the model has never been validated.
4. The system starts transformation process in the background.

Post-conditions:
1. A message appears stating that the transformation process has finished.
2. The system returns focus to the graphical elements present on the workspace.

Alternative Courses of Action:
1. In step D.2 (step 2 of Descriptions section), the user has the option select transform to EC2 format from the right click drop down menu.

Exceptions:
If the diagram does pass the validation process, Transformation to EC2 format cannot be done. In this case the error(s) are highlighted in the validation submenu located in the lower part of the screen.

Related Use Cases:
CPROVME005 Add Instance Node
CPROVEE003 Validation against Cloud provisioning Schema

Decision Support

Frequency: Each time the main user would like to transform the diagram to the EC2 format. On average once per changes applied to the model.

Criticality: High, It allows the user to Translate the diagram into the Amazon EC2 call which is the foundation of the plug in application.

Risk: High risk, Translating From a graphical language to EC2 format might cause ambiguity issues.

Constraints:

1. Usability:

a) No previous Training Time

b) On average the user should be able to translate to EC2 format a diagram in under 2 minute.

2. Reliability:

a) Mean time to Failure – 5% failures for every 1000 attempts to add an instance.

b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

   a) Translation process of the current workspace diagram should complete in under 2 minute.

   b) System handles only one request for deployment at a time.

4. Supportability:

   a) The execution environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

5. Implementation:

   a) The software will be implemented via a Eclipse plugin that the user can install and run..

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVEE005** Print to XML

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.
2. A model must be active and must contain at least an instance of a server.

Description:

1. Trigger: the use case begins when the user clicks the “Print” menu button location on the top menu bar.
2. in the top menu bar list, the user selects “Print to XML”
3. the system starts the validation process if the model has never been validated.
4. The system starts transformation process in the background.
5. The system outputs the print to the console in eclipse environment.

Post-conditions:

1. The system returns focus to the graphical elements present on the workspace.

Alternative Courses of Action:
No Alternative Courses.

Exceptions:
If the diagram does pass the validation process, The print to XML cannot be done. In this case the error(s) are highlighted in the validation submenu located in the lower part of the screen.

Related Use Cases:
CPROVME005 Add Instance Node
CPROVEE003 Validation against Cloud provisioning Schema

<u>Decision Support</u>

Frequency: Each time the main user would like to print to XML format.
On average once per changes applied to the model.

Criticality: Medium, It allows the user to see the XML version of the model currently on the workspace. Furthermore, it allows possibility of transport to another system.

Risk: High risk, Translating From a graphical language to XML format might cause ambiguity issues.

Constraints:

1. Usability:
   a) No previous Training Time
   b) On average the user should be able to print to XML format a diagram in under 1 minute.
2. Reliability:
   a) Mean time to Failure – 5% failures for every 1000 attempts to add an instance.
   b) Availability – The system shall be available as long as the user's machine is running.
3. Performance:
   a) Translation process of the current workspace diagram should complete in under 2 minute.
   b) System handles only one request for deployment at a time.
4. Supportability:
   a) The execution environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.
5. Implementation:

a) The software will be implemented via a Eclipse plugin that the user can install and run..

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVEE006** XML to Model

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.
2. A model must be active and must contain at least an instance of a server.

Description:

1. Trigger: the use case begins when the user clicks the “File” menu button location on the top menu bar.
2. in the top menu bar list, the user selects “load XML file”
3. A file picker windows opens.
4. the user selects the location of the XML file in the system.
5. The system starts transformation process in the background.
6. The system outputs diagram to the workspace.

Post-conditions:

1. The system returns focus to the newly created diagram on the workspace.

Alternative Courses of Action:

1. In D1 (step 1 of Description), the user can start the use case by right clicking the model in the tree view located on the left hand side interface.

Exceptions:
If the file loaded in the file picker is not in XML format the system shall reject the file, and send an error message to the user.

Related Use Cases:
CPROVME005 Add Instance Node
CPROVEE003 Validation against Cloud provisioning Schema

Decision Support

Frequency: Each time the main user would like to print to XML format.
On average once per changes applied to the model.

Criticality: Medium, It allows the user to create a diagram straight from an XML file. Furthermore, it allows possibility for the XML to come from another system or subsystem.

Risk: High risk, Translating from XML to a graphical language could produce errors in the positioning of the graphical elements in relation to the workspace grid.

Constraints:

1. Usability:

   a) No previous Training Time

   b) On average the user should be able load a diagram from XML filein under 3 minute.

2. Reliability:

   a) Mean time to Failure – 5% failures for every 1000 attempts to add an instance.

   b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

   a) Translation process of the XML file to workspace diagram should complete in under 2 minute.

   b) System handles only one request for deployment at a time.

4. Supportability:

   a) The execution environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

5. Implementation:

   a) The software will be implemented via a Eclipse plugin that the user can install and run..

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Modeling Environment Use cases:

<u>CPROV-ME Actors</u>

- <u>Cloud Administrator -</u> A cloud provisioning administrator who is interested in modeling a particular deployment.

<u>CPROV-ME Use Cases</u>

Use Case ID: **CPROVME001 Create New Model**
Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.

Description:
1. Trigger: the use case begins when the user clicks on a File button located on the top left hand side of the workspace.
2. The user selects the "New CPROV Model" button.
3. The system opens a new diagram in the currently select Project folder.
4. The system displays a window for the user to enter a Name for the Model.
5. The user enters a name for the model.
6. The user clicks the "OK" button to confirm.
7. The system switches focus to the newly created model workspace.

Post-conditions:
1. The system creates a new Model along with its tree representation located on the left hand side of the workspace.

Alternative Courses of Action:
1. In step D.2 (step 2 of Descriptions section) the user can choose to save a Model by selecting the model and pressing "CTRL-N".

Decision Support

Frequency: Each and every time the main user would like to save a model.

Criticality: High, It allows the user to create a new model.

Risk: Low risk, the system EMF/GMF framework handles Model file creation.

Constraints:

1. Usability:
    a) No previous Training Time
    b) On average the user should take no more than 2 min to create a new model.
2. Reliability:
    a) Mean time to Failure – 5% failures for every 1000 attempts to create a new model.
    b) Availability – The system shall be available as long as the user's machine is running.
3. Performance:
    a) To create a new model in general should complete in under 2 minutes.
    b) System handles only one user at time.
4. Supportability:
    a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME002 Add Model Element**
Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.

Description:
1. Trigger: the use case begins when the user click on a Model Element located in the palette.
2. The user clicks on the workspace in the desired location.

3. The system shall provide the user with a text box to enter the Model Element name.
4. The user types the name.
5. An icon representing the Model Element is then added to the workspace.

Post-conditions:
1. The Model Element is then created and focus is switched to the properties window.
2. The properties are now changed to their default value.

Alternative Courses of Action:
1. In step D.2 (Step 2 of Descriptions section), the user has the option to cancel the currently selected element on the palette.
2. In step D.4 the user can choose to not give the new Model Element a name. In this case, system will provide a default name.

Related Use Cases:
CPROVME003 Add Environment node

Decision Support

Frequency: Each and every time the main user would like to create a Model Element of a server. On average 5 Model Elements are added to a cloud deployment model.

Criticality: High, It allows the user to establish the essence of the cloud deployment which provide virtual machines, and servers.

Risk: Low risk, the system is using UML unambiguous modeling environment.

Constraints:

1. Usability:

   a) No previous Training Time

   b) On average the user should take no more than 1 min to add an Model Element to their model.

2. Reliability:

   a) Mean time to Failure – 5% failures for every 1000 attempts to add an Model Element.

   b) Availability – The system shall be available as long as the user’s machine is running.

3. Performance:

a) Model Element addition to the workspace should complete in under one second.

b) System handles only one user at time.

4. Supportability:

a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME003 Add Environment Node**
Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.

Description:
1. Trigger: the use case begins when the user click on the Environment Node icon located in the palette.
2. The user clicks on the workspace in the desired location.
3. The system shall provide the user with a text box to enter the Environment Node name.
4. The user types the name.
5. An icon representing the Environment Node is then added to the workspace.

Post-conditions:
1. The Environment Node is then created and focus is switched to the properties window.
2. The properties are now changed to their default value.

Alternative Courses of Action:
1. In step D.2 (step 2 of Descriptions section), the user has the option to cancel the currently selected element on the palette.
2. In step D.4 the user can choose to not give the new Environment Node a name. In this case, system will provide a default name.

Related Use Cases:
CPROVME003 Add Environment node

Decision Support

Frequency: Each and every time the main user would like to create a Environment Node of a server. On average 5 Environment Nodes are added to a cloud deployment model.

Criticality: High, It allows the user to establish the essence of the cloud deployment which provide virtual machines, and servers.

Risk: Low risk, the system is using UML unambiguous modeling environment.

Constraints:

1. Usability:
   a) No previous Training Time
   b) On average the user should take no more than 1 min to add an Environment Node to their model.
2. Reliability:
   a) Mean time to Failure – 5% failures for every 1000 attempts to add an Environment Node.
   b) Availability – The system shall be available as long as the user's machine is running.
3. Performance:
   a) Environment Node addition to the workspace should complete in under one second.
   b) System handles only one user at time.
4. Supportability:
   a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME004 Add Security Group Node**

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.
2. At least 2 person/user icons must be present on the workspace for the SecurityGroup to be needed. (it is not mandatory)

Description:

1. Trigger: the use case begins when the user click on the SecurityGroup icon located in the palette.
2. The user clicks on the workspace in the desired location.
3. The system shall provide the user with a text box to enter the person’s name.
4. The user types the name.
5. An icon representing the SecurityGroup is then added to the workspace.

Post-conditions:

1. The SecurityGroup is then created and focus is switched to its properties window.
2. The properties are now changed to their default value.

Alternative Courses of Action:

1. In step D.2 (step 2 of Descriptions section), the user has the option to cancel the currently selected element on the palette.
2. In step D.4 the user can choose to not give the new SecurityGroup a name. In this case, system will provide a default name.

Decision Support

Frequency: Each and every time the main user would like to create a SecurityGroup of a server. On average 2 SecurityGroup are added to a cloud deployment model.

Criticality: High, It allows the user to control the access of potential system users once the model is deployed for execution.
Risk: Low risk, the system is using UML unambiguous modeling environment.

Constraints:

1. Usability:

    a) No previous Training Time

    b) On average the user should take no more than 1 min to add an SecurityGroup to their model.

2. Reliability:

    a) Mean time to Failure – 5% failures for every 1000 attempts to add an SecurityGroup

    b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

    a) the SecurityGroup addition to the workspace should complete in under two second.

    b) System handles only one user at time.

4. Supportability:

    a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME005 Add Instance Node**

Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.

Description:
1. Trigger: the use case begins when the user click on the instance icon located in the palette.
2. The user clicks on the workspace in the desired location.
3. The system shall provide the user with a text box to enter the instance name.
4. The user types the name.

5. An icon representing the instance is then added to the workspace.

Post-conditions:
1. The instance is then created and focus is switched to the properties window.
2. The properties are now changed to their default value.

Alternative Courses of Action:
1. In step D.2 (step 2 of Descriptions section), the user has the option to cancel the currently selected element on the palette.
2. In step D.4 the user can choose to not give the new instance a name. In this case, system will provide a default name.

Decision Support

Frequency: Each and every time the main user would like to create a instance of a server. On average 5 instances are added to a cloud deployment model.

Criticality: High, It allows the user to establish the essence of the cloud deployment which provide virtual machines, and servers.

Risk: Low risk, the system is using UML unambiguous modeling environment.

Constraints:

1. Usability:

   a) No previous Training Time

   b) On average the user should take no more than 1 min to add an instance to their model.

2. Reliability:

   a) Mean time to Failure – 5% failures for every 1000 attempts to add an instance.

   b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

   a) Instance addition to the workspace should complete in under one second.

   b) System handles only one user at time.

4. Supportability:

   a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME006 Add Storage Node**
Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.

Description:
1. Trigger: the use case begins when the user click on the Storage Node icon located in the palette.
2. The user clicks on the workspace in the desired location.
3. The system shall provide the user with a text box to enter the Storage Node name.
4. The user types the name.
5. An icon representing the Storage Node is then added to the workspace.

Post-conditions:
1. The Storage Node is then created and focus is switched to the properties window.
2. The properties are now changed to their default value.

Alternative Courses of Action:
1. In step D.2 (step 2 of Descriptions section), the user has the option to cancel the currently selected element on the palette.
2. In step D.4 the user can choose to not give the new Storage Node a name. In this case, system will provide a default name.

Related Use Cases:
CPROVME003 Add Environment Node

Decision Support

Frequency: Each and every time the main user would like to create a Storage Node of a server. On average 5 Storage Nodes are added to a cloud deployment model.

Criticality: High, It allows the user to establish the essence of the cloud deployment which provide virtual machines, and servers.

Risk: Low risk, the system is using UML unambiguous modeling environment.

Constraints:

1. Usability:
   - a) No previous Training Time
   - b) On average the user should take no more than 1 min to add an Storage Node to their model.
2. Reliability:
   - a) Mean time to Failure – 5% failures for every 1000 attempts to add an Storage Node.
   - b) Availability – The system shall be available as long as the user's machine is running.
3. Performance:
   - a) Storage Node addition to the workspace should complete in under one second.
   - b) System handles only one user at time.
4. Supportability:
   - a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME007 Add Network Node**

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.

Description:

1. Trigger: the use case begins when the user click on the Network Node icon located in the palette.
2. The user clicks on the workspace in the desired location.
3. The system shall provide the user with a text box to enter the Network Node name.
4. The user types the name.
5. An icon representing the Network Node is then added to the workspace.

Post-conditions:

1. The Network Node is then created and focus is switched to the properties window.
2. The properties are now changed to their default value.

Alternative Courses of Action:

1. In step D.2 (step 2 of Descriptions section), the user has the option to cancel the currently selected element on the palette.
2. In step D.4 the user can choose to not give the new Network Node a name. In this case, system will provide a default name.

Related Use Cases:
CPROVME003 Add Environment Node

Decision Support

Frequency: Each and every time the main user would like to create a Network Node of a server. On average 5 Network Nodes are added to a cloud deployment model.

Criticality: High, It allows the user to establish the essence of the cloud deployment which provide virtual machines, and servers.

Risk: Low risk, the system is using UML unambiguous modeling environment.

Constraints:

1. Usability:

a) No previous Training Time

b) On average the user should take no more than 1 min to add an Network Node to their model.

2. Reliability:

a) Mean time to Failure – 5% failures for every 1000 attempts to add an Network Node.

b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

a) Network Node addition to the workspace should complete in under one second.

b) System handles only one user at time.

4. Supportability:

a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME008 Edit Node Properties**

Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.

Description:
1. Trigger: the use case begins when the user clicks on a Node icon located on the workspace.
2. The user clicks on the properties window.
3. The user selects the property to change by clicking on the property.
4. The user makes changes to the property selected.

Post-conditions:
1. The System applies the changes to the selected properties.
2. The properties are now changed to the chosen value(s).

Alternative Courses of Action:
1. In step D.3 (step 3 of Descriptions section) the user can choose to not to change any values in the properties.
2. In step D.4 (step 4 of Descriptions section) the user can choose undo the changes just made in the properties window by pressing CTRL-Z (undo)

Decision Support

Frequency: Each and every time the main user would like to edit the properties of a Node. Most Nodes have their properties edited prior to deployment.

Criticality: High, It allows the user to customize the settings of each of the Nodes present on the Model.

Risk: Low risk, the system is using UML unambiguous modeling environment. Editing the properties of a Node is merely using designed mutators for that object.

Constraints:

1. Usability:
    a) No previous Training Time
    b) On average the user should take no more than 1 min to edit the properties of a Node present on the Model
2. Reliability:
    a) Mean time to Failure – 5% failures for every 1000 attempts to add an Storage Node.
    b) Availability – The system shall be available as long as the user's machine is running.
3. Performance:
    a) Node editing in general should complete in under 2 minute.
    b) System handles only one user at time.
4. Supportability:
    a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME009 Edit Instance Properties**
Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.

Description:

1. Trigger: the use case begins when the user clicks on an Instance icon located on the workspace.
2. The user clicks on the properties window.
3. The user selects the property to change by clicking on the property.
4. The user makes changes to the property selected.

Post-conditions:

1. The System applies the changes to the selected properties.
2. The properties are now changed to the chosen value(s).

Alternative Courses of Action:

1. In step D.3 (Step 3 of Descriptions section) the user can choose to not to change any values in the properties.
2. In step D.4 (Step 4 of Descriptions section) the user can choose undo the changes just made in the properties window by pressing CTRL-Z (undo)

Decision Support

Frequency: Each and every time the main user would like to edit the properties of a Instance. Most Instances have their properties edited prior to deployment.

Criticality: High, It allows the user to customize the settings of each of the Instance Nodes present on the Model.

Risk: Low risk, the system is using UML unambiguous modeling environment. Editing the properties of a Instance is merely using designed mutators for that object.

Constraints:

1. Usability:

a) No previous Training Time

b) On average the user should take no more than 1 min to edit the properties of an Instance present on the Model

2. Reliability:

a) Mean time to Failure – 5% failures for every 1000 attempts to edit the properties of an Instance.

b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

a) Instance editing in general should complete in under 2 minute.

b) System handles only one user at time.

4. Supportability:

a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME010 Edit Security Group Properties**

Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.

Description:
1. Trigger: the use case begins when the user clicks on a Security Group icon located on the workspace.
2. The user clicks on the properties window.
3. The user selects the property to change by clicking on the property.
4. The user makes changes to the property selected.

Post-conditions:
1. The System applies the changes to the selected properties.
2. The properties are now changed to the chosen value(s).

Alternative Courses of Action:
1. In step D.3 (step 3 of Descriptions section) the user can choose to not to change any values in the properties.
2. In step D.4 (step 4 of Descriptions section) the user can choose undo the changes just made in the properties window by pressing CTRL-Z (undo)

Decision Support

Frequency: Each and every time the main user would like to edit the properties of a Security Group. Most Security Groups have their properties edited prior to deployment.

Criticality: High, It allows the user to customize the settings of each of the Security Group icons present on the Model.

Risk: Low risk, the system is using UML unambiguous modeling environment. Editing the properties of a Security Group is merely using designed mutators for that object.

Constraints:

1. Usability:
   a) No previous Training Time
   b) On average the user should take no more than 1 min to edit the properties of a Security Group present on the Model
2. Reliability:
   a) Mean time to Failure – 5% failures for every 1000 attempts to edit the properties of an Security Group.
   b) Availability – The system shall be available as long as the user's machine is running.
3. Performance:
   a) Security Group editing in general should complete in under 2 minute.
   b) System handles only one user at time.
4. Supportability:
   a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME011 Edit Storage Properties**

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.

Description:

1. Trigger: the use case begins when the user clicks on a Storage icon located on the workspace.
2. The user clicks on the properties window.
3. The user selects the property to change by clicking on the property.
4. The user makes changes to the property selected.

Post-conditions:

1. The System applies the changes to the selected properties.
2. The properties are now changed to the chosen value(s).

Alternative Courses of Action:

1. In step D.3 (step 3 of Descriptions section) the user can choose to not to change any values in the properties.
2. In step D.4 (step 4 of Descriptions section) the user can choose undo the changes just made in the properties window by pressing CTRL-Z (undo)

Decision Support

Frequency: Each and every time the main user would like to edit the properties of a Storage. Most Storages have their properties edited prior to deployment.

Criticality: High, It allows the user to customize the settings of each of the Storage icons present on the Model.

Risk: Low risk, the system is using UML unambiguous modeling environment. Editing the properties of a Storage is merely using designed mutators for that object.

Constraints:

1. Usability:

   a) No previous Training Time

   b) On average the user should take no more than 1 min to edit the properties of a Storage present on the Model

2. Reliability:

a) Mean time to Failure – 5% failures for every 1000 attempts to edit the properties of an Storage.

b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

a) Storage editing in general should complete in under 2 minute.

b) System handles only one user at time.

4. Supportability:

a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME012 Edit Network Properties**

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.

Description:

1. Trigger: the use case begins when the user clicks on a Network icon located on the workspace.
2. The user clicks on the properties window.
3. The user selects the property to change by clicking on the property.
4. The user makes changes to the property selected.

Post-conditions:

1. The System applies the changes to the selected properties.
2. The properties are now changed to the chosen value(s).

Alternative Courses of Action:

1. In step D.3 (step 3 of Descriptions section) the user can choose to not to change any values in the properties.

2. In step D.4 (step 4 of Descriptions section) the user can choose undo the changes just made in the properties window by pressing CTRL-Z (undo)

Decision Support

Frequency: Each and every time the main user would like to edit the properties of a Network. Most Networks have their properties edited prior to deployment.

Criticality: High, It allows the user to customize the settings of each of the Network icons present on the Model.

Risk: Low risk, the system is using UML unambiguous modeling environment. Editing the properties of a Network is merely using designed mutators for that object.

Constraints:

1. Usability:

   a) No previous Training Time

   b) On average the user should take no more than 1 min to edit the properties of a Network present on the Model

2. Reliability:

   a) Mean time to Failure – 5% failures for every 1000 attempts to edit the properties of an Network.

   b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

   a) Network editing in general should complete in under 2 minute.

   b) System handles only one user at time.

4. Supportability:

   a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME013 Delete Node**

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.

Description:

1. Trigger: the use case begins when the user clicks on a Node icon located on the workspace.
2. The user presses the “delete” key.
3. The system deletes the selected Node.

Post-conditions:

1. The System applies the changes to the Model

Alternative Courses of Action:

1. In step D.2 (Step 2 of Descriptions section) the user can choose to delete by right-clicking the Node and selecting Delete from the drop down menu.
2. In step D.2 (Step 2 of Descriptions section) the user can choose to delete by pressing the “BackSpace” key.

Decision Support

Frequency: Each and every time the main user would like to edit the properties of a Network. Most Networks have their properties edited prior to deployment.

Criticality: High, It allows the user to remove the desired Node from the Model Diagram.

Risk: Low risk, the system is using UML unambiguous modeling environment. Deleting an Icon is supported by the EMF/GMF framework..

Constraints:

1. Usability:

a) No previous Training Time

b) On average the user should take no more than 1 min to delete a Node present on the Model.

2. Reliability:

a) Mean time to Failure – 5% failures for every 1000 attempts to delete a Node from a Model.

b) Availability – The system shall be available as long as the user’s machine is running.

3. Performance:

a) Deleting a Node in general should complete in under 30 seconds.

b) System handles only one user at time.

4. Supportability:

a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME014 Add Connection**

Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.
2. At least 2 icons must be present on the workspace before a connection can be added.

Description:
1. Trigger: the use case begins when the user click on the connection icon located in the palette.
2. The user clicks on the workspace on the desired source icon.
3. The user clicks on the workspace on the desired destination icon.
4. The system shall provide the user with a text box to enter the connection's name.
5. The user types the name.
6. An arrow representing the connection between the 2 icons is then added to the workspace.

Post-conditions:
1. The connection is then created and focus is switched to its properties window.

2. The properties are now changed to their default value.

Alternative Courses of Action:

1. In step D.2 (step 2 of Descriptions section), the user has the option to cancel the currently selected element on the palette.
2. In step D.4 the user can choose to not give the new connection a name. In this case, system will provide a default name to the newly added connection

----

Decision Support

Frequency: Each and every time the main user would like to create a connection between two designated icons. On average between 5 and 20 connections are added to a cloud deployment model.

Criticality: High, It allows the user to determine the associations among different instances and users and/or sub-networks of the system.

Risk: Low risk, the system is using UML unambiguous modeling environment.

Constraints:

1. Usability:

    a) No previous Training Time

    b) On average the user should take no more than 1 min to add an connection to their model.

2. Reliability:

    a) Mean time to Failure – 5% failures for every 1000 attempts to add an person/user

    b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

    a) the connection addition to the workspace should complete in under 1 minute

    b) System handles only one user at time.

4. Supportability:

    a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME015 Delete Connection**

Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.

Description:
1. Trigger: the use case begins when the user clicks on a Connection icon located on the workspace.
2. The user presses the "delete" key.
3. The system deletes the selected Connection.

Post-conditions:
1. The System applies the changes to the Model.

Alternative Courses of Action:
1. In step D.2 (step 2 of Descriptions section) the user can choose to delete by right-clicking the Connection and selecting Delete from the drop down menu.
2. In step D.2 (step 2 of Descriptions section) the user can choose to delete by pressing the "BackSpace" key.

Decision Support

Frequency: Each and every time the main user would like to edit the properties of a Network. Most Networks have their properties edited prior to deployment.

Criticality: High, It allows the user to remove the desired Connection from the Model Diagram.
Risk: Low risk, the system is using UML unambiguous modeling environment. Deleting an Icon is supported by the EMF/GMF framework..

Constraints:

1. Usability:

a) No previous Training Time

b) On average the user should take no more than 1 min to delete a Connection present on the Model.

2. Reliability:

a) Mean time to Failure – 5% failures for every 1000 attempts to delete a Connection from a Model.

b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

a) Deleting a Connection in general should complete in under 30 seconds.

b) System handles only one user at time.

4. Supportability:

a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME016 Load Existing Model**

Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.

Description:
1. Trigger: the use case begins when the user clicks on a File button located on the top left hand side of the workspace.
2. The user selects the "Load Existing Model" button.
3. A file picker window opens for the use to select a File.
4. Once selected the user clicks the "open" button located on the File Picker window.
5. The system loads the selected Model and displays the diagram in a new window in the current workspace.

Post-conditions:

1. The System loads the Model and displays it on the workspace.

Alternative Courses of Action:

1. In step D.2 (step 2 of Descriptions section) the user can choose to load and existing model by clicking the open file button from toolbar menu located on the top of the workspace.
2. In step D.4 (step 4 of Descriptions section) the user can choose to cancel by pressing the "cancel" button.

Exceptions:

1. If the file type loaded in the File Picker is unrecognized by the system, a error message will display for the user, and the File Picker window will remain open.

Decision Support

Frequency: Each and every time the main user would like to load an existing file.

Criticality: High, It allows the user to load an existing model into the workspace. Without this function, transport from system to system would be rendered impractical.

Risk: Low risk, the system EMF/GMF framework handles file location, as well as File picker window.

Constraints:

1. Usability:

    a) No previous Training Time

    b) On average the user should take no more than 2 min to load an existing model in the current workspace.

2. Reliability:

    a) Mean time to Failure – 5% failures for every 1000 attempts to load an existing model.

    b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

    a) to load an existing model in general should complete in under 2 minutes.

    b) System handles only one user at time.

4. Supportability:

a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME017 Validate Model**

Details:
Actor: Cloud Administrator (Main user)

Pre-conditions:
1. The graphical visualization environment must be running on the system.
2. A model must be active and must contain at least an instance of a server.

Description:
1. Trigger: the use case begins when the user clicks right clicks on the workspace.
2. In the right-click drop down menu, the user clicks on the "Validate"
3. The system start to validation process with a progress bar starting at 0 % and ends at 100% showing that the validation has been done.

Post-conditions:
1. A message appears stating that the validation process reported errors or no errors.
2. The system returns focus to the graphical elements present on the workspace.

Alternative Courses of Action:
1. In step D.2 (step 2 of Descriptions section), the user has the option select validate from the menu bar under "Validation".

Exceptions:
If one or more elements of the diagram does not respect CProv modeling language, the error(s) are highlighted in the validation submenu located in the lower part of the screen.

Related Use Cases:
CPROVME005 Add Instance Node

Decision Support

Frequency: Each time the main user would like to validate the diagram.
On average once per changes applied to the model.

Criticality: High, It allows the user to verify is the diagram represents a sound model before the translation and deployment phases.

Risk: Low risk, the system uses the EMF/GMF API to implement the validation.

Constraints:

1. Usability:
   a) No previous Training Time
   b) On average the user should be able to validate a diagram in under 2 minute.
2. Reliability:
   a) Mean time to Failure – 5% failures for every 1000 attempts to add an instance.
   b) Availability – The system shall be available as long as the user's machine is running.
3. Performance:
   a) Validation of the current workspace diagram should complete in under one minute.
   b) System handles only one request for deployment at a time.
4. Supportability:
   a) The execution environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.
5. Implementation:
   a) The software will be implemented via a Eclipse plugin that the user can install and run.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

Use Case ID: **CPROVME018 Save Model**

Details:

Actor: Cloud Administrator (Main user)

Pre-conditions:

1. The graphical visualization environment must be running on the system.

Description:

1. Trigger: the use case begins when the user clicks on a File button located on the top left hand side of the workspace.
2. The user selects the "Save" button.
3. The system shows a brief progress bar at the bottom of the screen depicting the saving process.
4. The system displays a message to the user that the current model has been saved.

Post-conditions:

1. The System loads the Model and displays it on the workspace.

Alternative Courses of Action:

1. In step D.2 (step 2 of Descriptions section) the user can choose to save a Model by selecting the model and pressing "CTRL-S".

Decision Support

Frequency: Each and every time the main user would like to save a model.

Criticality: High, It allows the user to write the current model to the memory of the system.

Risk: Low risk, the system EMF/GMF framework handles file saving.

Constraints:

1. Usability:

   a) No previous Training Time

   b) On average the user should take no more than 2 min to save a model.

2. Reliability:

a) Mean time to Failure – 5% failures for every 1000 attempts to save a model.

b) Availability – The system shall be available as long as the user's machine is running.

3. Performance:

a) to save a model in general should complete in under 2 minutes.

b) System handles only one user at time.

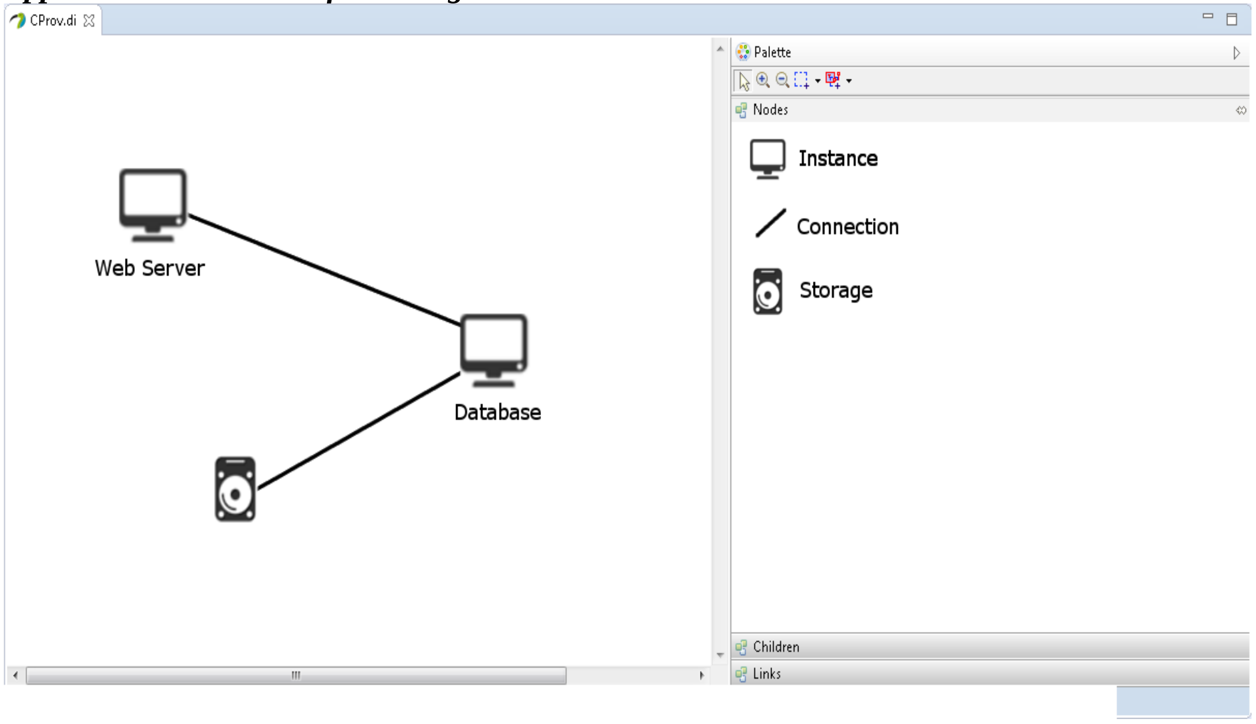
4. Supportability:

a) The modeling environment should be handled by all versions of Eclipse, Windows, Mac OS, and Linux.

Modification History:

Owner: Edwin Malek, Dionny Santiago, Xiaoyu Dong
Initiation date: 02/01/2015
Date last modified: 02/01/2015

7.3 *Appendix C – User Interface Designs*



7.4 *Appendix D- Diary of Meeting and Tasks*

<u>Diary of Meetings</u>

Meeting 1:

Place: Graduate lab

Date: 2/1/2015

Start: 11:00 AM

End: 2:00 PM

Facilitator: Edwin Malek

Attending: Dionny Santiago, Xiaoyu Dong

Minute Taker: Xiaoyu Dong

1. Objective

Establishing all project use cases

2. Status

Everyone participated in coming up with ideas for the project.

3. Discussion

We discussed the requirements and established use cases that we all agreed upon for the project

---

Meeting 2:

Place: Skype meeting

Date: 2/4/2015

Start: 8:00 PM

End: 9:00 PM

Facilitator: Edwin Malek

Attending: Dionny Santiago, Xiaoyu Dong

Minute Taker: Xiaoyu Dong

1. Objective

Creating our project feature diagram

2. Status

Everyone participated in elicitation of the features

3. Discussion

We discussed the requirements and established use cases that we all agreed upon for the project

---

Meeting 3

Place: Graduate lab

Date: 2/5/2015

Start: 7:40 PM

End: 8:00 PM

Facilitator: Edwin Malek

Attending: Dionny Santiago, Xiaoyu Dong

Minute Taker: Xiaoyu Dong

1. Objective

Creating our project creation of use case diagrams

2. Status

Everyone participated in elicitation of the features

3. Discussion

We discussed the requirements and established use cases that we all agreed upon for the project

---

Meeting 4

Place: Skype meeting

Date: 2/6/2015

Start: 7:40 PM

End: 9:40 PM

Facilitator: Edwin Malek

Attending: Dionny Santiago, Xiaoyu Dong

Minute Taker: Xiaoyu Dong

1. Objective

Creating our project creation of use case scenarios.

2. Status

Everyone participated in elicitation of the features

3. Discussion

We discussed use cases that we will use for an instance of those use cases. The most important use cases for the project.

--

Meeting 5

Place: Skype meeting

Date: 2/9/2015

Start: 7:00 PM

End: 9:00 PM

Facilitator: Edwin Malek

Attending: Dionny Santiago, Xiaoyu Dong

Minute Taker: Xiaoyu Dong

1. Objective

Creating our project creation of object diagrams according to our scenarios

2. Status

Everyone participated in the discussion.

3. Discussion

We discussed the requirements and established use cases that we all agreed upon for the project

----

Meeting 7

Place: Skype meeting

Date: 2/10/2015

Start: 7:40 PM

End: 8:00 PM

Facilitator: Edwin Malek

Attending: Dionny Santiago, Xiaoyu Dong

Minute Taker: Xiaoyu Dong

1. Objective

Creating our project creation of sequence diagrams

2. Status

Everyone participated in creation of sequence diagram for that meeting.

3. Discussion

We discussed the requirements and established use cases that we all agreed upon for the project. Dionny showed us how the sequence diagrams work in papyrus.

Meeting 7

Place: Skype meeting

Date: 2/12/2015

Start: 7:40 PM

End: 8:00 PM

Facilitator: Edwin Malek

Attending: Dionny Santiago, Xiaoyu Dong

Minute Taker: Xiaoyu Dong

1. Objective

Creating our project creation of activity diagrams

2. Status

Everyone participated in elicitation of the nodes of the diagrams, at least for the first diagram.

3. Discussion

We discussed how the rest of the activity diagrams will be created using papyrus.

Meeting 8

Place: Skype meeting

Date: 2/17/2015

Start: 7:40 PM

End: 9:40 PM

Facilitator: Edwin Malek

Attending: Dionny Santiago, Xiaoyu Dong

Minute Taker: Xiaoyu Dong

1. Objective

Creating our project creation of the presentation slides.

2. Status

Everyone participated in discussion.

3. Discussion

We discussed what would be the content of each slide according the presentation format.

Meeting 9

Place: Skype meeting

Date: 2/18/2015

Start: 7:00 PM

End: 9:00 PM

Facilitator: Edwin Malek

Attending: Dionny Santiago, Xiaoyu Dong

Minute Taker: Xiaoyu Dong

1. Objective

Edit the final document for deliverable 1.

2. Status

Everyone participated in fixed and editing the document.

3. Discussion

We discussed possible issues coming with formatting and resolution for the diagrams in the document.