Software Requirements and Analysis for Cloud Provisioning System

CEN 5064 Software Design

Dr. Peter Clarke

Team 1:

Rachel Chávez

Rudy Padrón

Roly Vicaría

Javier Luque

February 18, 2014

Abstract

With the ever increasing popularity of cloud computing there has also been a surge in the number of tools that try to simplify the process of provisioning resources in the cloud. Unfortunately, most of these tools fall short of their promise. This is primarily because the tools require their users to have an intimate knowledge of programming.

In this document we will outline how we intend to build a system that truly simplifies the process of provisioning resources in the cloud by exploiting the expressive powers of models. That is, one that allows domain experts to easily provision resources in the cloud by simply creating a model that represents the configuration they would like to have in the cloud. Then with a click of a button this model will be interpreted, executed and deployed onto the cloud provider of choice.

Table of Contents

1. Introduction

As an introduction to this document we will begin by describing the purpose of the system. That will be followed by the scope of the system, then by some definitions, acronyms and abbreviations. Lastly, an overview of the document will prepare the reader for the topics to be covered in more detail in the remainder of the document.

1.1 Purpose of System

The purpose of this system will be to facilitate the provisioning of resources in the cloud by providing an intuitive and expressive approach that abstracts away the complexity normally involved with these operations and lets the user focus on the task at hand. The system will accomplish this by leveraging the expressive power of models and the simplicity of a Graphical Modeling Environment (GVE) that will allow domain experts to model a cloud provisioning scenario simply by dragging components onto a canvas and connecting these components as necessary.

1.2 Scope of System

When delivered the system will allow Cloud Provisioning Administrators and other users to easily provision resources in the cloud through the use of modeling. The system will allow users to model the scenario they would like to implement. Once the user has created a model that is representative of the cloud provisioning setup that they would like to have, with the click of a button this model will be interpreted, executed and deployed on the cloud provider.

1.3 Definitions, acronyms, and abbreviations

CPS - Cloud Provisioning System
GVE - Graphical Modeling Environment
DSL - Domain Specific Language
CProvML - Cloud Provisioning Modeling Language
G-CProvML - Graphical Cloud Provisioning Modeling Language
X-CProvML - XML Cloud Provisioning Modeling Language
API - Application Programming Interface
Meta-model - defines the language and processes from which to form a model

1.4 Overview of document

In Section 2 we will analyze the current system to understand it's limitations and problems. Section 3 will provide a description of the project plan, where you will find the assignment of roles, hardware and software requirements, and the identified milestones and deliverables. Section 4 contains a high level description of the proposed functionality that will include non-functional requirements relating to the areas of usability, reliability, performance, supportability and implementation. Additionally, there will be a use case diagram depicting how the different users of the system interact with the functionality provided by the system. Moreover, in section 5, you will find a variety of models depicting the static and dynamic views of the system. Lastly, you will find a glossary of terms used in this document and an appendix with more details on the project schedule, uses cases, user interface designs and diary of meetings and tasks.

2. Current System (limitations and problems)

Presently, there are several prominent players in the public cloud market, among them are Amazon and IBM. Other notable vendors include Rackspace and VMWare. One of the main challenges created by having so many options for cloud provisioning is the lack of standard APIs and tools to facilitate the use of each of these cloud providers' services. Scripting languages such as Puppet and Chef have emerged as the tool of choice in many enterprises for managing cloud deployments.

These scripting languages can best be described as DSLs for configuration management. They are able to provide a layer of abstraction that facilitates interoperability with different cloud providers. This alleviates the challenge of managing resources in the cloud without coupling oneself to a particular vendor. However, the hurdle that still remains is that these scripting languages are not readily accessible to non-programmers.

Both Chef and Puppet are DSLs built on Ruby, using Ruby or Ruby-like syntax. Chef, in particular, is an internal DSL therefore any user writing a Chef "cookbook" essentially has the entire Ruby language at their disposal, a full-fledged, general purpose dynamic language - not for the faint of heart. It is because of this that many enterprises have begun to merge their IT departments with software developers, commonly referred to as DevOps teams.

3. Project Plan

The project plan is explained in this part of the document, this includes the roles assigned to every team member, the hardware and software needed to complete the project and the work breakdown that will be presented in a Gantt chart. The tasks presented in this chart may change and new tasks might be added and other might be deleted or modified.

3.1 Project Organization

Deliverable/Name	**Team Leader**	**Minute Taker**	**Time Keeper**
Deliverable 1	Rachel Chavez	Roly Vicaria	Rudy Padron
Deliverable 2	Rudy Padron	Javier Luque	Rachel Chavez
Deliverable 3	Roly Vicaria	Rachel Sanchez	Javier Luque

3.2 Hardware and software requirements

Hardware: this projects require any of the following OS: Windows XP, Linux, MacOS, Windows 7, windows 8; space requirements: at least 315 MB; Memory Requirements: minimum 256 MB for Eclipse 2.x and 512 for Eclipse 3.x - Recommended 1GB of memory for best performance.

Software: This project requires Eclipse with the papyrus plugin installed, a JVM and a JDK.

3.3 Work breakdown

See Appendix A.

4. System Requirements

In this section we will describe the functional and nonfunctional requirements of the system so we can justify our solution. In addition we will talk about specific functions that the system must perform together with client-defined constraints.

4.1 Overview

In the following two subsections we will discuss the overall functionality of the system, what it should do and how it should do it.

4.2 Functional Requirements

- The system shall allow users to define VM instances
- The system shall allow users to set/edit properties of those instances
- The system shall allow users to define their own environment which contains the instances
- The system shall allow users to save/edit/delete/open an environment configuration
- The system shall allow the user to validate the in-progress configuration

4.3 Non-functional Requirements

In order to complete our solution we must use a specific tool called Papyrus. The reason for this is client requested.

4.4 Use case model

Refer to Appendix B.

5. System Analysis

5.1 Scenarios

Create VM (Scenario: Create Small Windows VM)

Actors: Kurt, Cloud provisioning administrator
Pre-conditions: Kurt has opened the system.
Description:

1. The use case begins when Kurt selects the “New VM” button.
2. Kurt chooses to create a User-defined VM.
3. System responds by prompting Kurt to confirm that he would create a User-defined VM.
4. Kurt Confirms.
5. The system prompts Kurt to enter the name of the new VM.
6. Kurt creates the KurtVM001 VM.
7. The system list all the Hardware properties (CPU, RAM, Storage and Network) with their values.
8. Kurt chooses a 32-bit CPU, 1GB of storage, a medium level of network performance and 3.75 GB of RAM.

9. Kurt chooses Windows 7 as a OS and no applications are added.
10. Kurt save the configuration.
11. System prompts Kurt that KurtVM001 has been created.

Post-conditions:
1. KurtVM001 is created.

Create VM (Scenario: Create Large Windows VM)

Actors: Sean, Cloud provisioning administrator
Pre-conditions:
Description:
1. The use case begins when Sean selects the “New VM” button.
2. Sean chooses to create a User-defined VM.
3. System responds by prompting Sean to confirm that he would create a User-defined VM.
4. Sean Confirms.
5. The system prompts Sean to enter the Name of the new VM.
6. Sean creates the SQLWINVM01 VM.
7. The system list all the Hardware properties (CPU, RAM, Storage and Network) with their values.
8. Sean chooses 2 64-bit CPU, 32GB of storage, a medium level of network performance and 7.5 GB of RAM.
9. Sean chooses Windows 7 as a OS and choose to install Microsoft SQL Server 2008.
10. Sean saves the configuration.
11. System alerts Sean that SQLWINVM01 has been created.
12. System alerts Sean that Microsoft SQL Server 2008 has been installed

Post-conditions:
1. The SQLWINVM01 is created.

Create VM from Template (Scenario:Create VM From Linux-64-2-32.7.5-H Template)

Actors: Jean, Cloud provisioning administrator
Pre-conditions:
Description:
1. The use case begins when Jean clicks the “New VM” button.
1. Jean chooses to create the VM from a template.
2. System responds by prompting Jean to confirm that she would create a VM from a template.
3. Jean confirms.
4. The system prompts Jean to set the Name for the new VM.
5. Jean sets JeanVM123 as the Name
6. The system lists all the templates available.
7. Jean chooses the Linux-64-2-32.7.5-H template
8. The system prompts Jean to confirm the selection of the template.
9. The system creates the VM with the name jeanVM123 and the properties related to the template.

Post-conditions:

1. jeanVM123 is created is created and properly stored.

Create VM (Scenario:Create VM From Windows-64-8-68.8-H Template)

Actors: Susan, Cloud provisioning administrator
Pre-conditions:
Description:

1. The use case begins when Susan clicks the “New VM” button
2. Jean chooses to create the VM from a template.
3. System responds by prompting Susan to confirm that she would create a VM from a template.
4. Susan clicks the “OK” button.
5. The system prompts Susan to set the Name for the new VM.
6. Susan sets LabVM001 as the Name
7. The system lists all the templates available.
8. Susan chooses the Windows-64-8-68.8-H template
9. The system prompts Susan to confirm the selection of the template.
10. The system creates the VM with the Name “LabVM001” and the properties related to the template.

Post-conditions:
“LabVM001” is created and properly stored.

Delete an Existing VM (Scenario: Delete VM)

Actors: Juan, Cloud provisioning administrator
Pre-conditions:

1. Juan has opened the GVE.
2. Juan has an open model with at least one VM created.

Description:

1. Juan clicks the comlabdev03 VM on the canvas.
2. The GVE responds by highlighting comlabdev03 VM
3. Juan presses the “Delete VM” button.
4. GVE shows a confirmation dialog with the message “The following VM is about to be deleted” and the comlabdev03 is listed in the message.
5. Juan clicks the “OK” button.
6. The GVE deletes all the relations that comlabdev03 has with other artifacts.
7. The system removes the VM from the canvas.

Post-conditions:

1. comlabdev03 VM is deleted and removed from the canvas.

Delete Two Existing VM’s (Scenario: Delete Two VM’s)

Actors: Maria, Cloud provisioning administrator
Pre-conditions:

1. Maria has opened the GVE.
2. Maria has an open model with at least two VM created.

Description:
1. Maria clicks the comlabdev15 VM and the comlabdev16 VM on the canvas.
2. The GVE responds by highlighting comlabdev15 VM and the comlabdev16 VM
3. Juan presses the “Delete VM” button.
4. GVE shows a confirmation dialog with the message “The following VM are about to be deleted” and the comlabdev15 VM and the comlabdev16 VM are listed in the message.
5. Maria clicks the “OK” button.
6. The GVE deletes all the relations that comlabdev15 has with other artifacts and then the relations of the comladev16.
7. The system removes both VM’s from the canvas.

Post-conditions:
1. comlabdev15 and comlabdev16 are deleted and removed from the canvas.

Modify Hardware (Scenario: Increase RAM)

Actor: Tom, Cloud provisioning administrator

Pre-conditions:
1. User has opened the Graphical Visual Environment
2. User has model with existing VM

Description:
1. The use case begins when Tom clicks on the VM component on the canvas labeled “comvmsql10”.
2. The system responds by highlighting the VM component on the canvas labeled “comvmsql10” and displaying it’s properties.
3. Tom modifies the RAM property by changing the value to 8 GBs.
4. The use case ends when Tom clicks on the Save button.

Post-conditions:
1. The RAM property of “comvmsql10” should display 8 GBs.

Modify Hardware (Scenario: Increase Storage)

Actor: Tom, Cloud provisioning administrator

Pre-conditions:
1. User has opened the Graphical Visual Environment
2. User has model with existing VM

Description:
1. The use case begins when Tom clicks on the VM component on the canvas labeled “comvmdev09”.

2. The system responds by highlighting the VM component on the canvas labeled “comvmdev09” and displaying it’s properties.
3. Tom modifies the Storage property by changing the value to 95 GBs.
4. The use case ends when Tom clicks on the Save button.

Post-conditions:

1. The RAM property of “comvmdev09” should display 95 GBs.

Create Model (Scenario: Blank Cloud Provisioning Model)

Pre-conditions:

1. The graphical visual modeling environment has to be opened.

Description:

1. Use case begins when Tom clicks on the create ‘New Model’ button.
2. The system responds by display a list of available modeling templates.
3. Tom selects “Blank Model” template
4. The system responds by prompting Tom to enter a name for the model.
5. Tom enters “Amazon Cloud” and clicks on the ‘OK” button
6. The system responds by loading the toolbar with the following components: Environment and VM.

Post-conditions:

1. A new blank cloud provisioning model is created.

Create Model (Scenario: Single VM Model)

Pre-conditions:

1. The graphical visual modeling environment has to be opened.

Description:

1. Use case begins when Bob clicks on the create ‘New Model’ button.
2. The system responds by display a list of available modeling templates.
3. Bob selects “Single VM” template
4. The system prompts Bob to enter a name for the model.
5. Bob enters “Single VM” and clicks on the “OK” button
6. The system responds by loading the toolbar with the following components: Environment and VM.

Post-conditions:

1. A new communication model is created with a single VM component on the canvas.

Add Elements (Scenario: Add an Enviroment)

Actors: Kim, Cloud provisioning administrator

Pre-conditions:

1. Kim has opened the GVE.
2. Kim has an open model.

Description:

1. Kim clicks the environment button.
2. Kim drags the environment and drops it into the canvas.
3. The system prompts Kim to enter the name of the environment.
4. Kim enters kimsenvironment.
5. The system sets the name for the environment and creates it.
6. The system shows a confirmation message, “kimsenvironment is created”.

Post-conditions:

1. kimsenvironment is created.

Add Elements (Scenario: Add a VM)

Actors: Britney, Cloud provisioning administrator
Pre-conditions:

1. Britney has opened the GVE.
2. Britney has an open model.

Description:

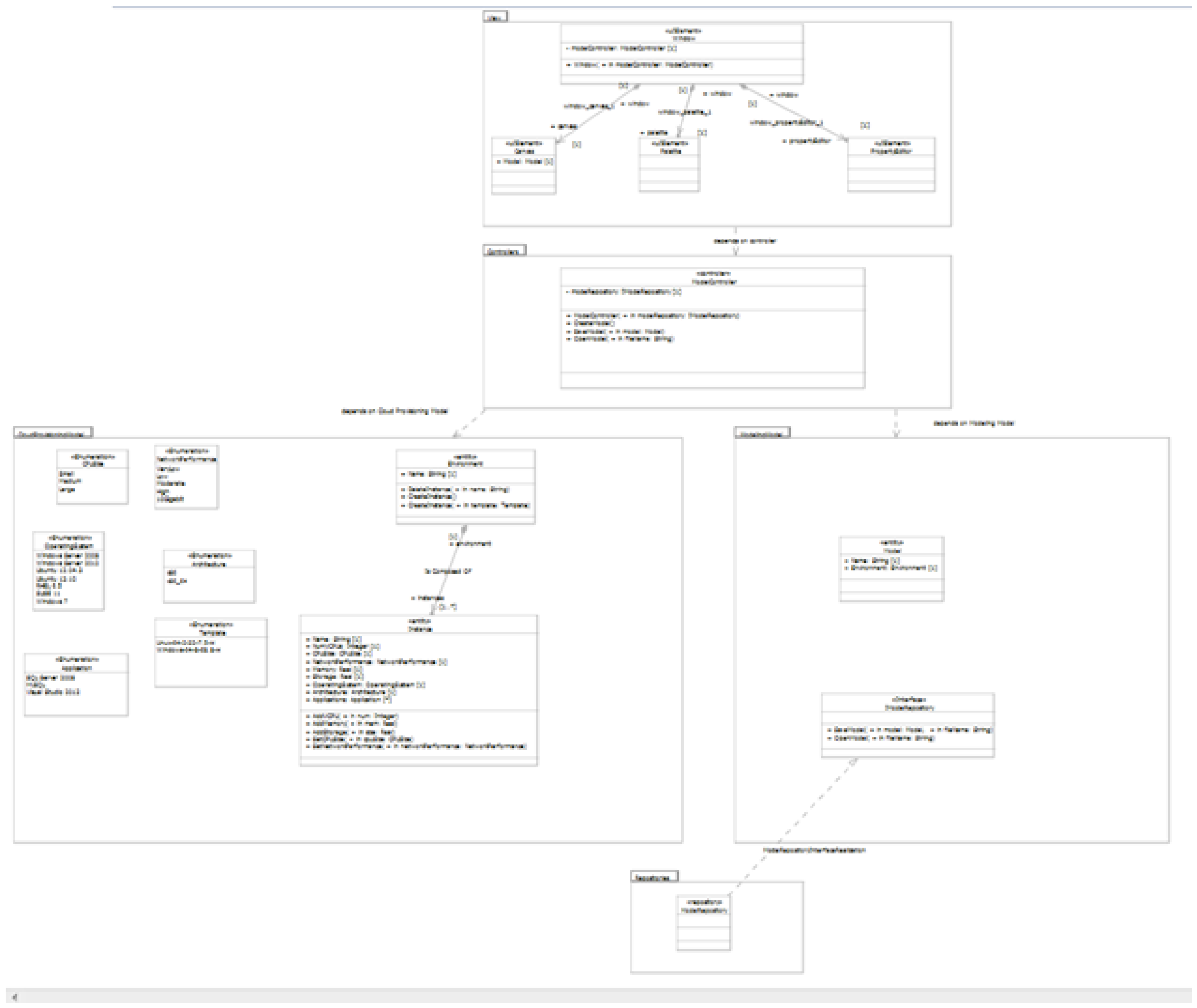
1. Britney clicks the VM button.
2. Britney drags the VM and drops it into the canvas.
3. The system prompts Britney to enter the name of the VM and the properties.
4. Britney enters labVM01 and chooses the Linux-64-2-32.7.5-H template for her new VM.
5. The system sets the name for the VM and creates it.
6. The system shows a confirmation message, “labVM01 is created”.

Post-conditions:

1. LabVM01 is created.

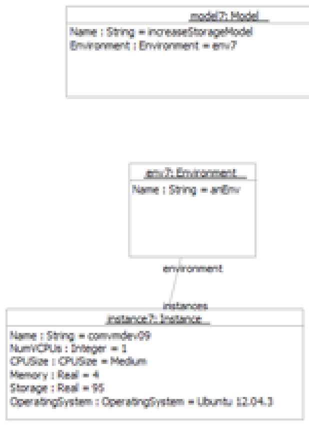
5.2 Object model

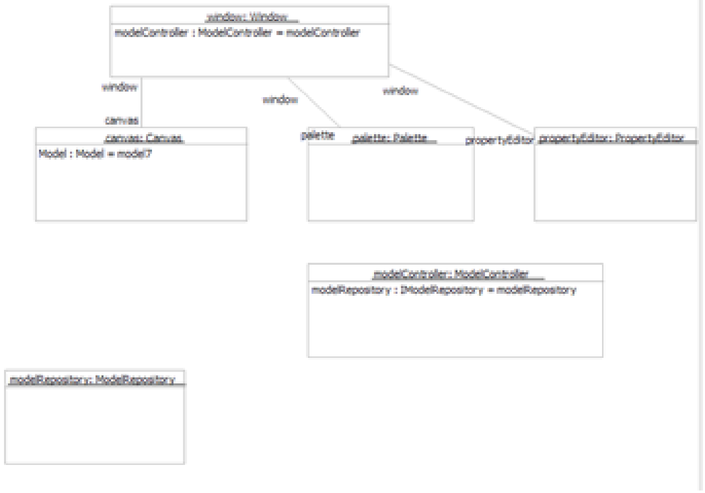
Class Diagram



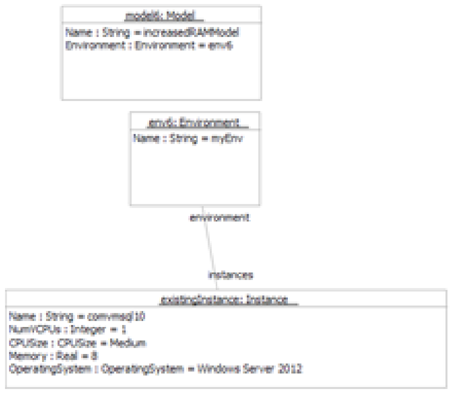
Object Diagrams

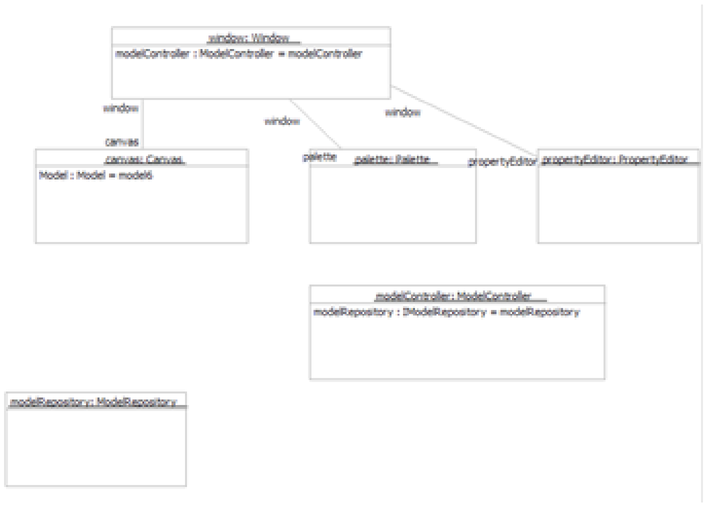
IncreaseStorage


model7: Model
Name : String = increaseStorageModel
Environment : Environment = env7
env7: Environment
Name : String = anEnv
environment
instances
instance7: Instance
Name : String = comvmdev09
NumVCPUs : Integer = 1
CPUSize : CPUSize = Medium
Memory : Real = 4
Storage : Real = 95
OperatingSystem : OperatingSystem = Ubuntu 12.04.3

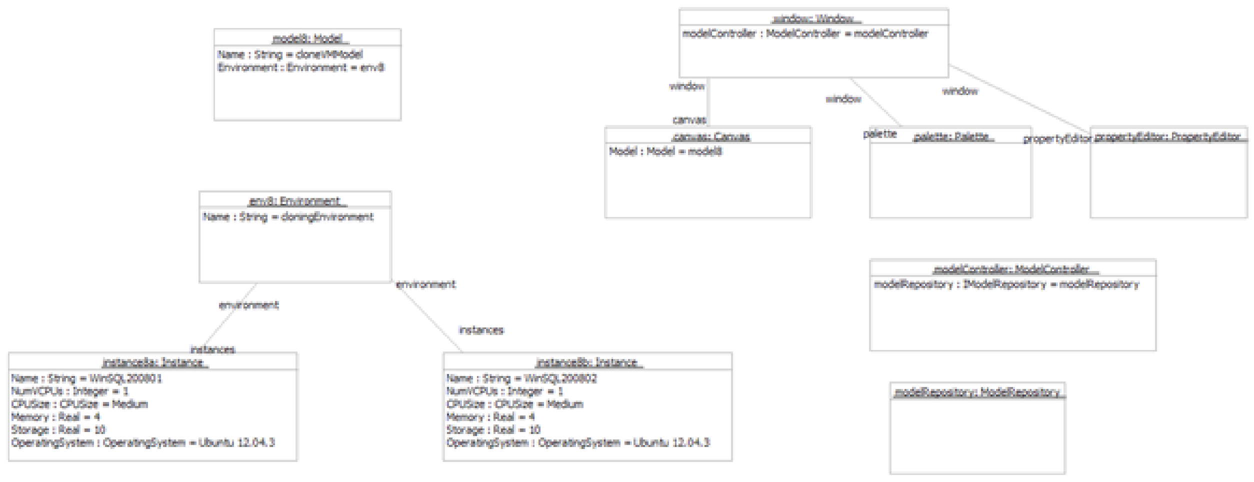

window: Window
modelController : ModelController = modelController
window
canvas
window
window
palette
canvas: Canvas
Model : Model = model7
palette: Palette
propertyEditor
propertyEditor: PropertyEditor
modelController: ModelController
modelRepository : IModelRepository = modelRepository
modelRepository: ModelRepository

IncreaseRam

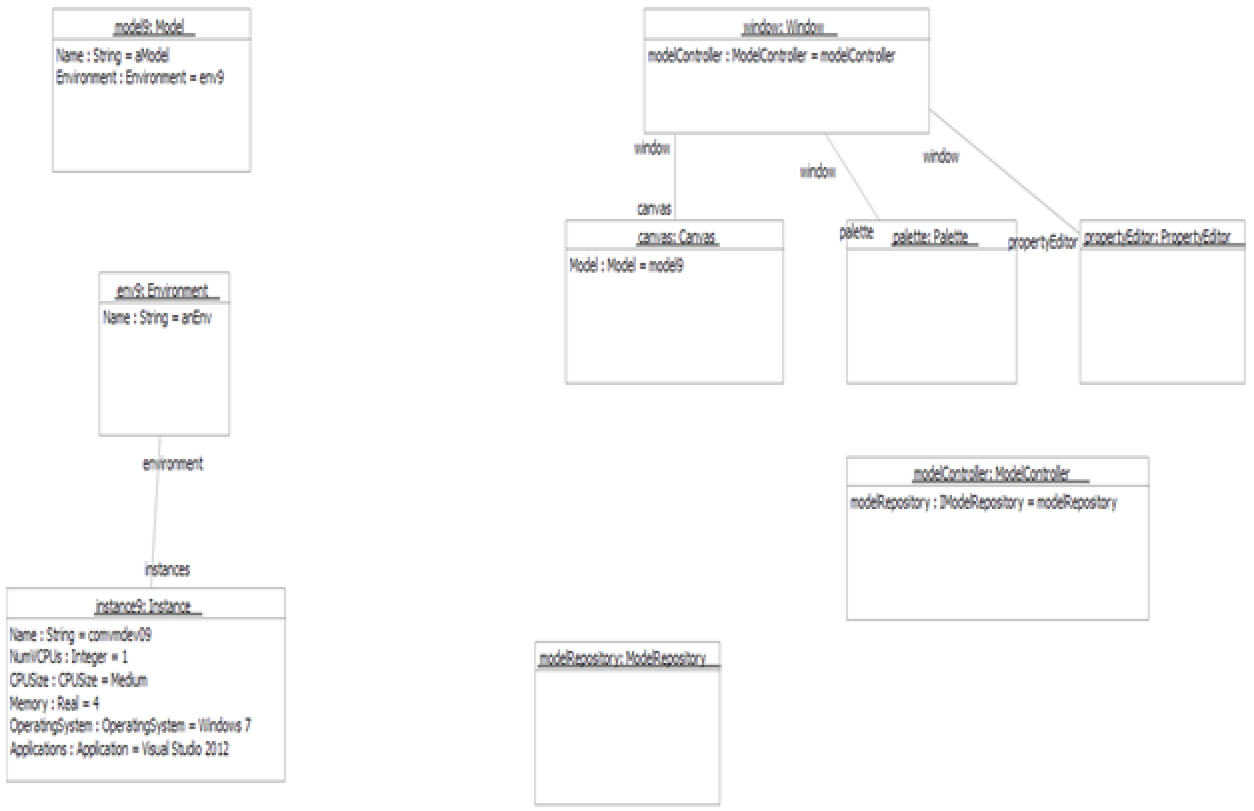

model6: Model
Name : String = increasedRAMModel
Environment : Environment = env6
env6: Environment
Name : String = myEnv
environment
instances
existingInstance: Instance
Name : String = comvmsql10
NumVCPUs : Integer = 1
CPUSize : CPUSize = Medium
Memory : Real = 8
OperatingSystem : OperatingSystem = Windows Server 2012


window: Window
modelController : ModelController = modelController
window
canvas
window
window
palette
canvas: Canvas
Model : Model = model6
palette: Palette
propertyEditor
propertyEditor: PropertyEditor
modelController: ModelController
modelRepository : IModelRepository = modelRepository
modelRepository: ModelRepository

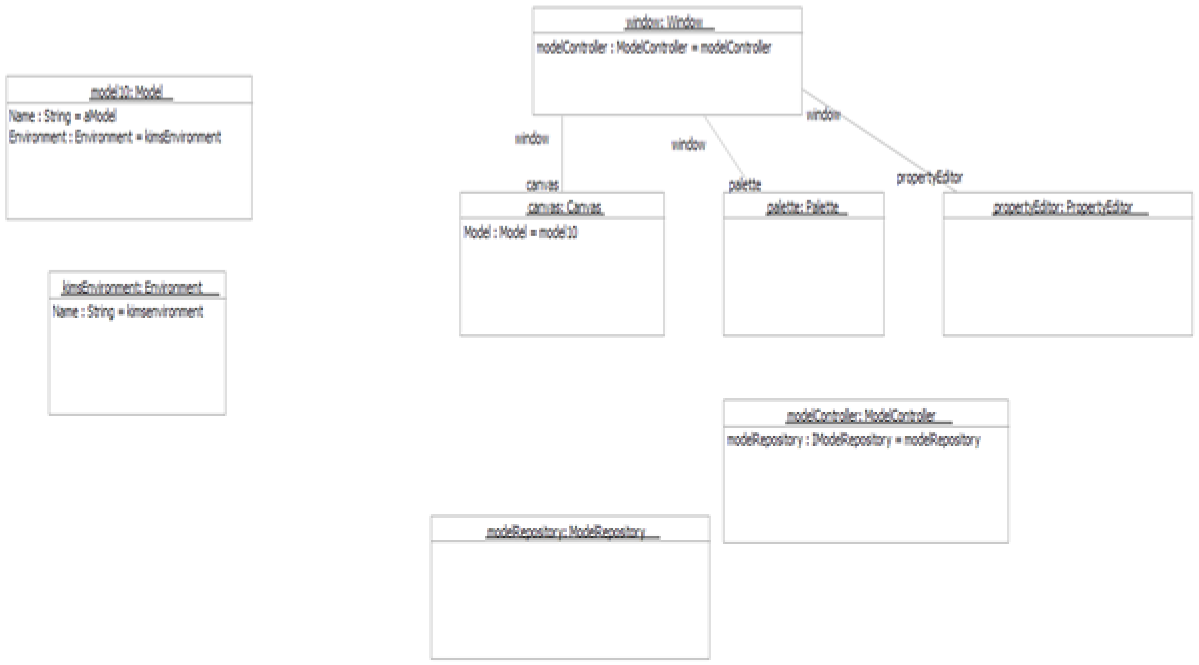
CloneSingleVM


model8: Model
Name : String = cloneVMModel
Environment : Environment = env8
window: Window
modelController : ModelController = modelController
window
canvas
window
palette
window
propertyEditor
canvas: Canvas
Model : Model = model8
palette: Palette
propertyEditor: PropertyEditor
env8: Environment
Name : String = cloningEnvironment
environment
instances
environment
instances
modelController: ModelController
modelRepository : IModelRepository = modelRepository
instance8a: Instance
Name : String = WinSQL200801
NumVCPUs : Integer = 1
CPUSize : CPUSize = Medium
Memory : Real = 4
Storage : Real = 10
OperatingSystem : OperatingSystem = Ubuntu 12.04.3
instance8b: Instance
Name : String = WinSQL200802
NumVCPUs : Integer = 1
CPUSize : CPUSize = Medium
Memory : Real = 4
Storage : Real = 10
OperatingSystem : OperatingSystem = Ubuntu 12.04.3
modelRepository: ModelRepository

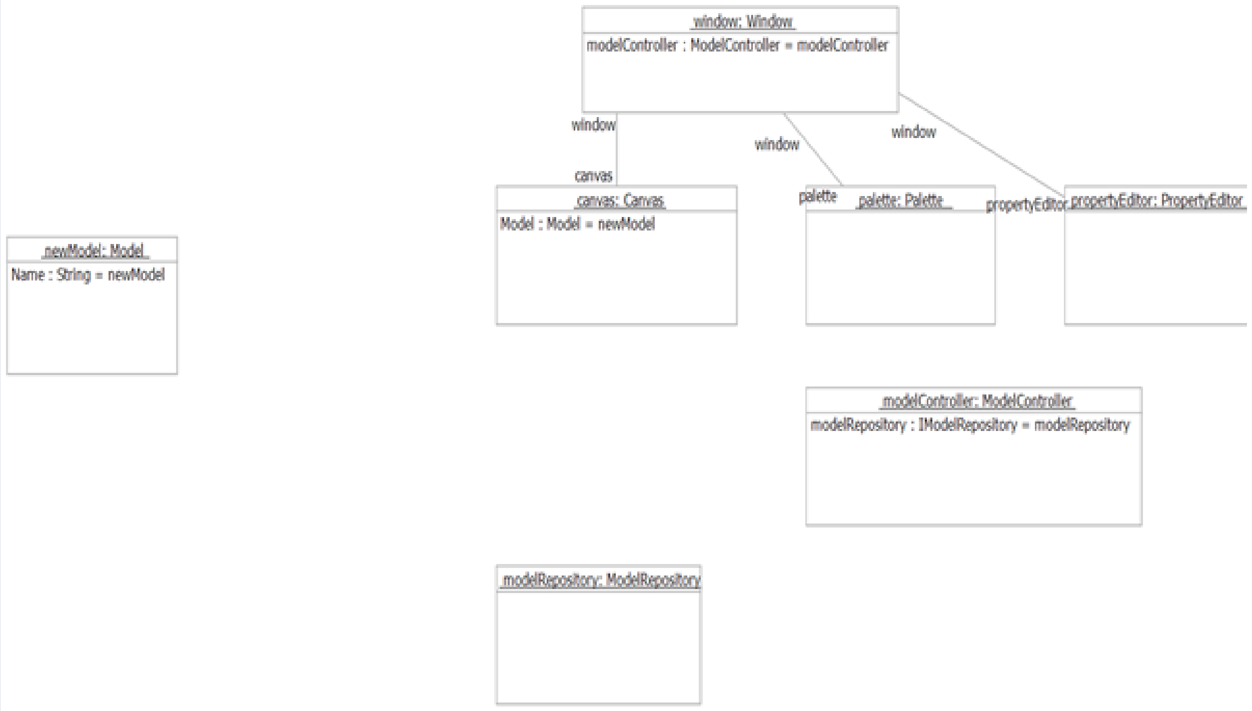
UninstallSoftware


model9: Model
Name : String = aModel
Environment : Environment = env9
window: Window
modelController : ModelController = modelController
window
canvas
window
palette
window
propertyEditor
canvas: Canvas
Model : Model = model9
palette: Palette
propertyEditor: PropertyEditor
env9: Environment
Name : String = anEnv
environment
instances
modelController: ModelController
modelRepository : IModelRepository = modelRepository
instance9: Instance
Name : String = convmdev09
NumVCPUs : Integer = 1
CPUSize : CPUSize = Medium
Memory : Real = 4
OperatingSystem : OperatingSystem = Windows 7
Applications : Application = Visual Studio 2012
modelRepository: ModelRepository

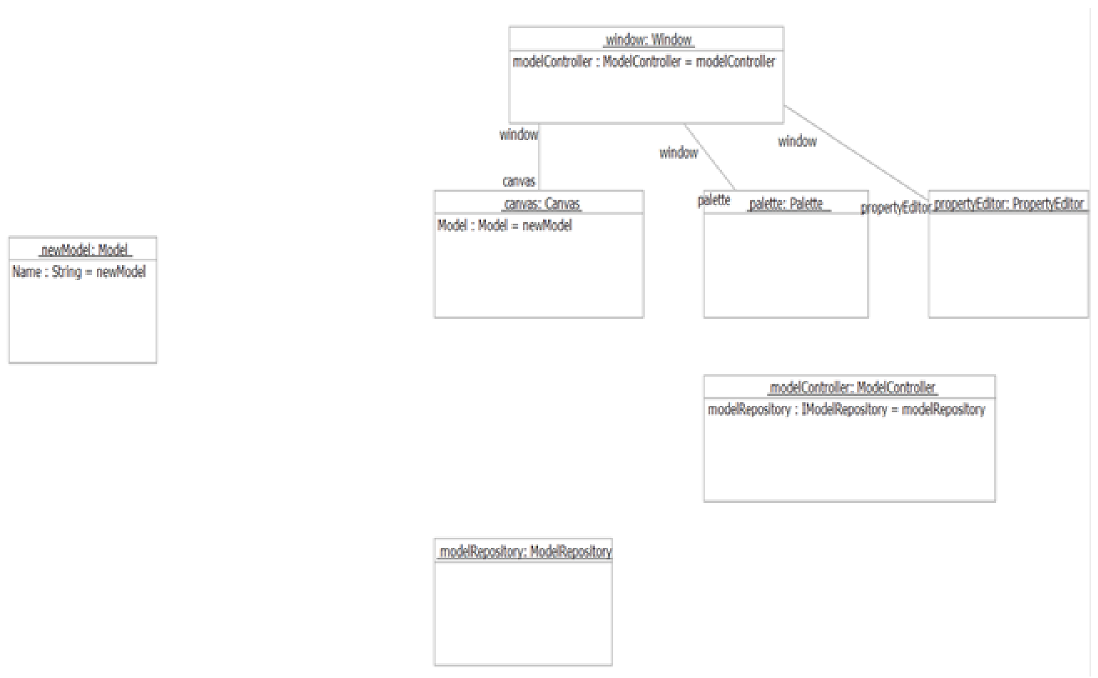
AddEnvironment


model10: Model
Name : String = aModel
Environment : Environment = kimsEnvironment
kimsEnvironment: Environment
Name : String = kimsenvironment
window: Window
modelController : ModelController = modelController
window
canvas
window
palette
window
propertyEditor
canvas: Canvas
Model : Model = model10
palette: Palette
propertyEditor: PropertyEditor
modelController: ModelController
modelRepository : IModelRepository = modelRepository
modelRepository: ModelRepository

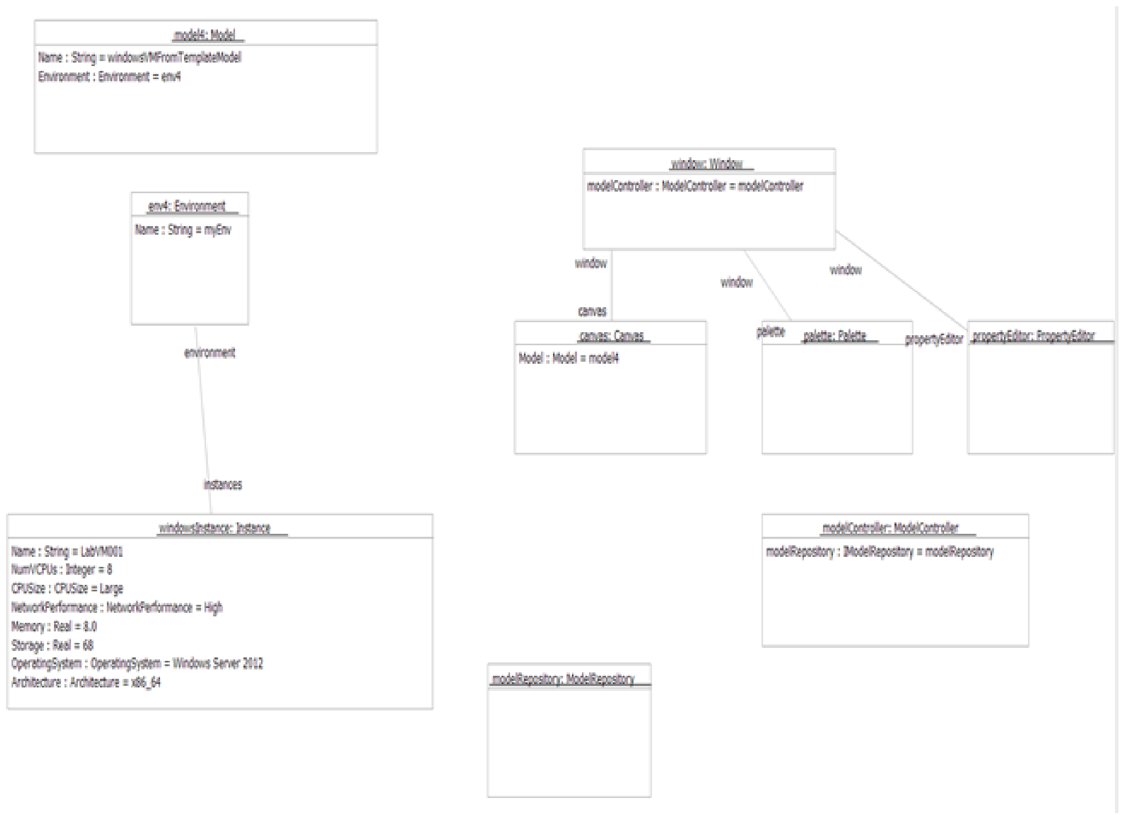
CreateNewModel


window: Window
modelController : ModelController = modelController
window
canvas
window
palette
window
propertyEditor
canvas: Canvas
Model : Model = newModel
palette: Palette
propertyEditor: PropertyEditor
newModel: Model
Name : String = newModel
modelController: ModelController
modelRepository : IModelRepository = modelRepository
modelRepository: ModelRepository

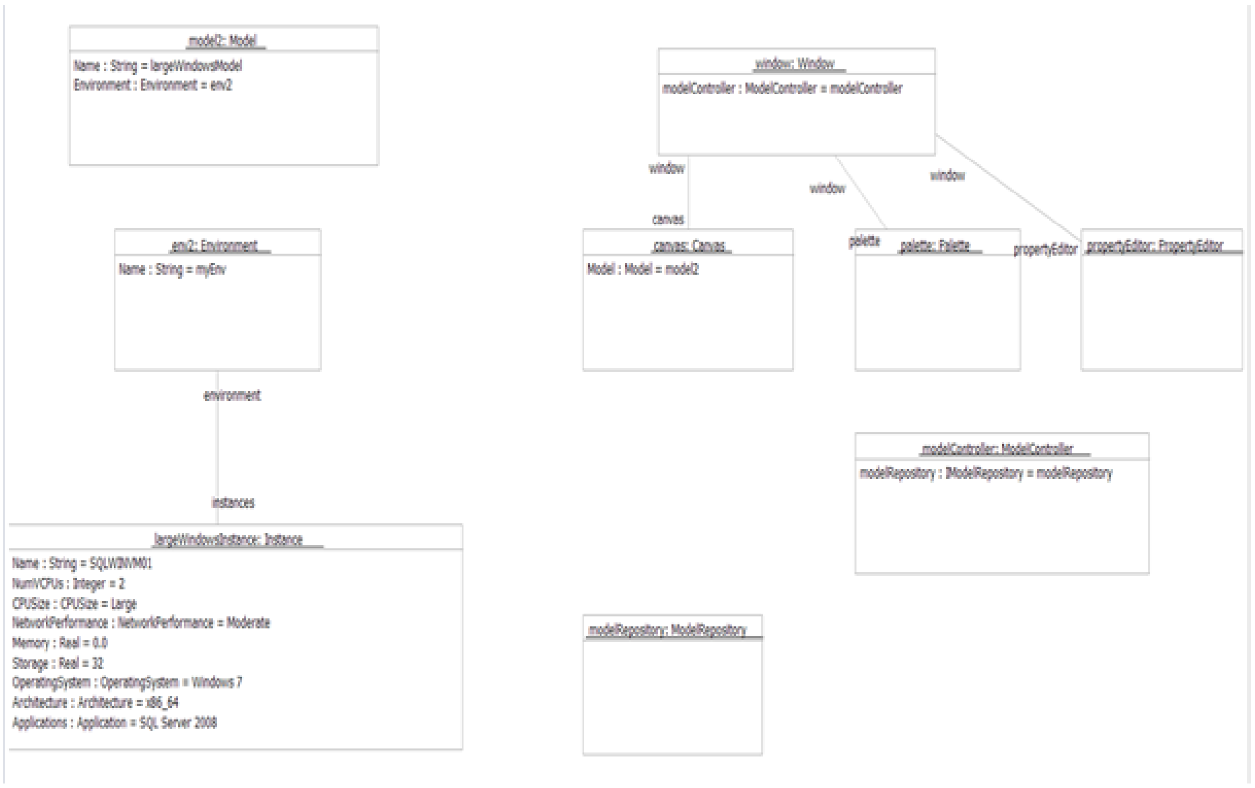
DeleteVM


window: Window
modelController : ModelController = modelController
window
canvas
window
palette
window
propertyEditor
canvas: Canvas
Model : Model = newModel
palette: Palette
propertyEditor: PropertyEditor
newModel: Model
Name : String = newModel
modelController: ModelController
modelRepository : IModelRepository = modelRepository
modelRepository: ModelRepository

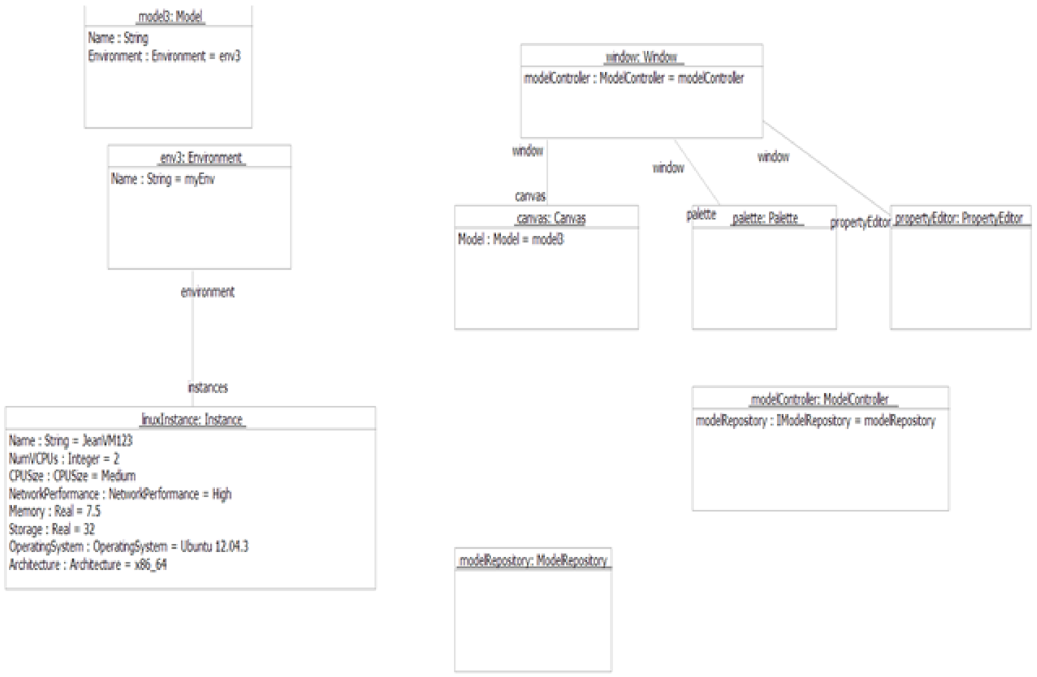
CreateWindowsVMFromTemplate


model4: Model
Name : String = windowsVMFromTemplateModel
Environment : Environment = env4
env4: Environment
Name : String = myEnv
environment
instances
windowsInstance: Instance
Name : String = LabVM001
NumVCPUs : Integer = 8
CPUSize : CPUSize = Large
NetworkPerformance : NetworkPerformance = High
Memory : Real = 8.0
Storage : Real = 68
OperatingSystem : OperatingSystem = Windows Server 2012
Architecture : Architecture = x86_64
window: Window
modelController : ModelController = modelController
window
canvas
window
palette
window
propertyEditor
canvas: Canvas
Model : Model = model4
palette: Palette
propertyEditor: PropertyEditor
modelController: ModelController
modelRepository : IModelRepository = modelRepository
modelRepository: ModelRepository

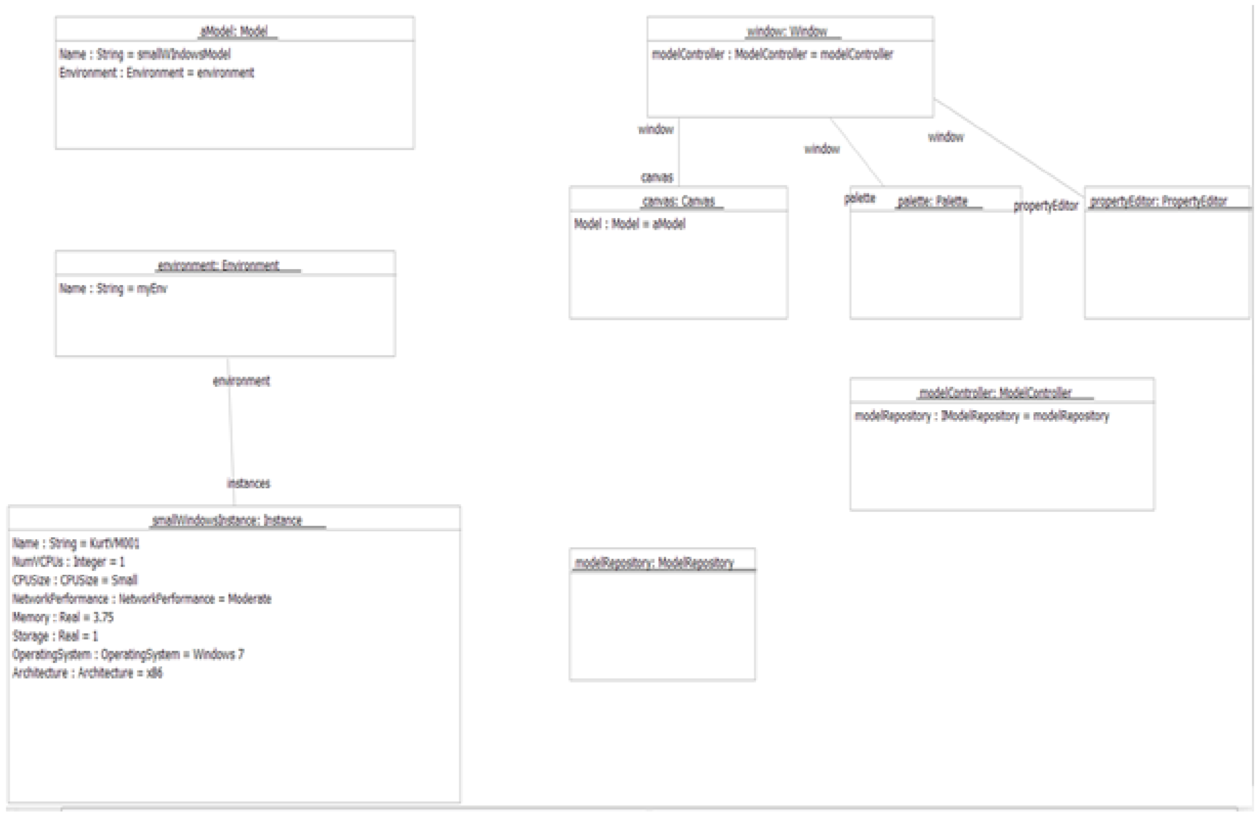
CreateLinuxFromVMTemplate


model2: Model
Name : String = largeWindowsModel
Environment : Environment = env2
env2: Environment
Name : String = myEnv
environment
instances
largeWindowsInstance: Instance
Name : String = SQLWINVM01
NumVCPUs : Integer = 2
CPUSize : CPUSize = Large
NetworkPerformance : NetworkPerformance = Moderate
Memory : Real = 0.0
Storage : Real = 32
OperatingSystem : OperatingSystem = Windows 7
Architecture : Architecture = x86_64
Applications : Application = SQL Server 2008
window: Window
modelController : ModelController = modelController
window
window
window
canvas
canvas: Canvas
Model : Model = model2
palette
palette: Palette
propertyEditor
propertyEditor: PropertyEditor
modelController: ModelController
modelRepository : IModelRepository = modelRepository
modelRepository: ModelRepository

CreateLargeWindowsVMTemplate

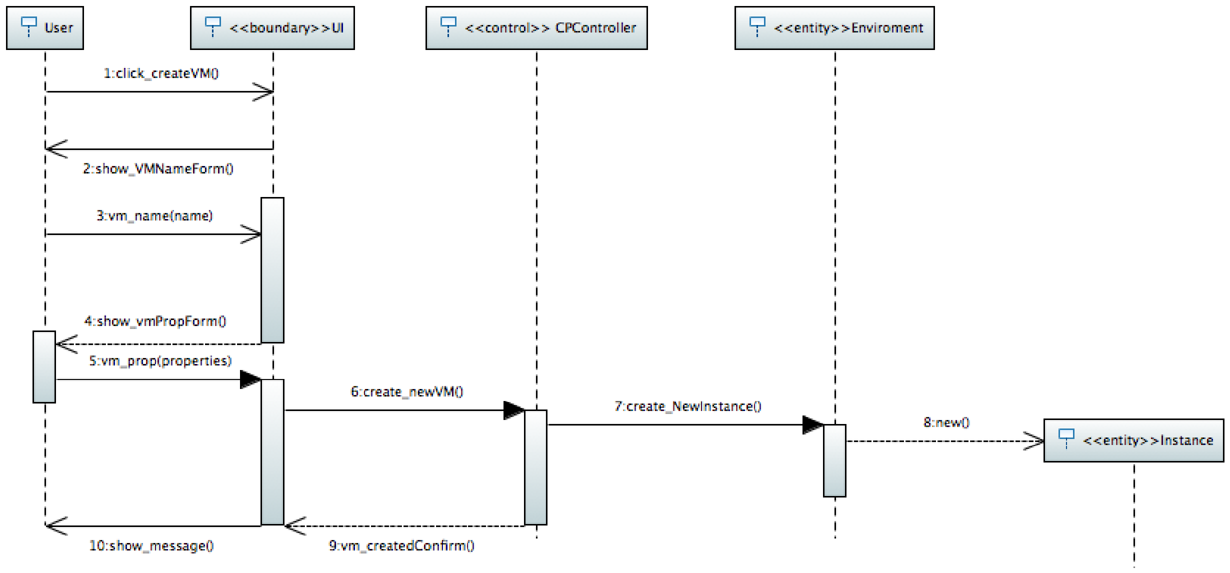

model3: Model
Name : String
Environment : Environment = env3
env3: Environment
Name : String = myEnv
environment
instances
linuxInstance: Instance
Name : String = JeanVM123
NumVCPUs : Integer = 2
CPUSize : CPUSize = Medium
NetworkPerformance : NetworkPerformance = High
Memory : Real = 7.5
Storage : Real = 32
OperatingSystem : OperatingSystem = Ubuntu 12.04.3
Architecture : Architecture = x86_64
window: Window
modelController : ModelController = modelController
window
window
window
canvas
canvas: Canvas
Model : Model = model3
palette
palette: Palette
propertyEditor
propertyEditor: PropertyEditor
modelController: ModelController
modelRepository : IModelRepository = modelRepository
modelRepository: ModelRepository

CreateSmallWindowsVMObject

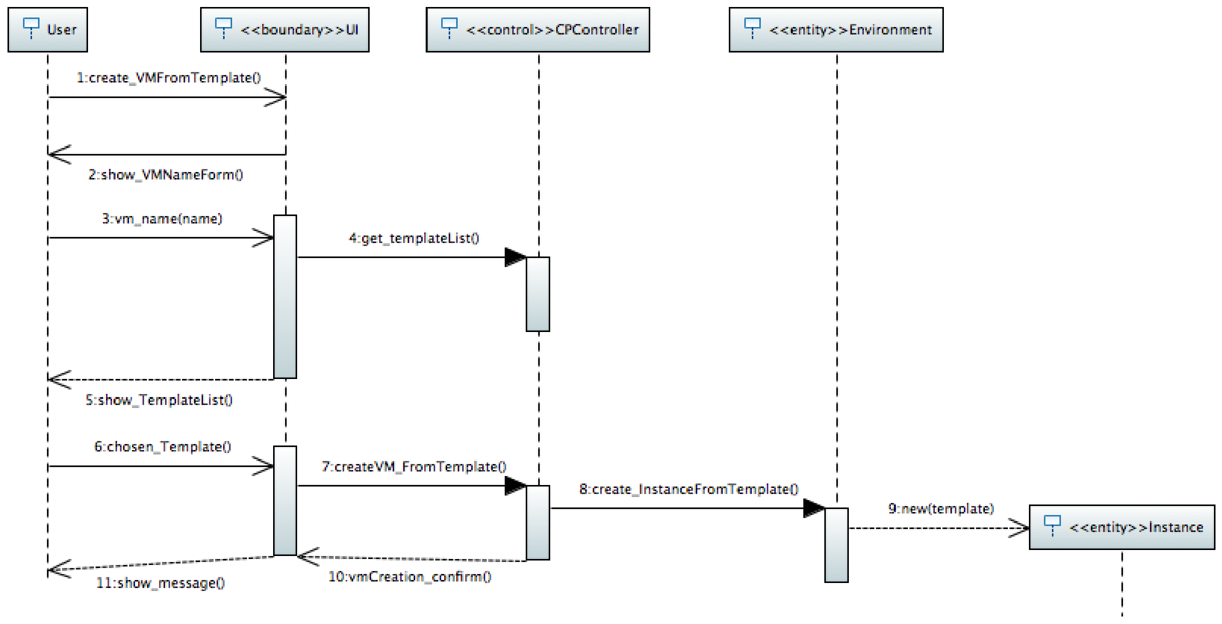

aModel: Model
Name : String = smallWindowsModel
Environment : Environment = environment
window: Window
modelController : ModelController = modelController
window
window
window
canvas
canvas: Canvas
Model : Model = aModel
palette
palette: Palette
propertyEditor
propertyEditor: PropertyEditor
environment: Environment
Name : String = myEnv
environment
instances
modelController: ModelController
modelRepository : IModelRepository = modelRepository
smallWindowsInstance: Instance
Name : String = KurtVM001
NumVCPUs : Integer = 1
CPUSize : CPUSize = Small
NetworkPerformance : NetworkPerformance = Moderate
Memory : Real = 3.75
Storage : Real = 1
OperatingSystem : OperatingSystem = Windows 7
Architecture : Architecture = x86
modelRepository: ModelRepository

5.3 Dynamic model

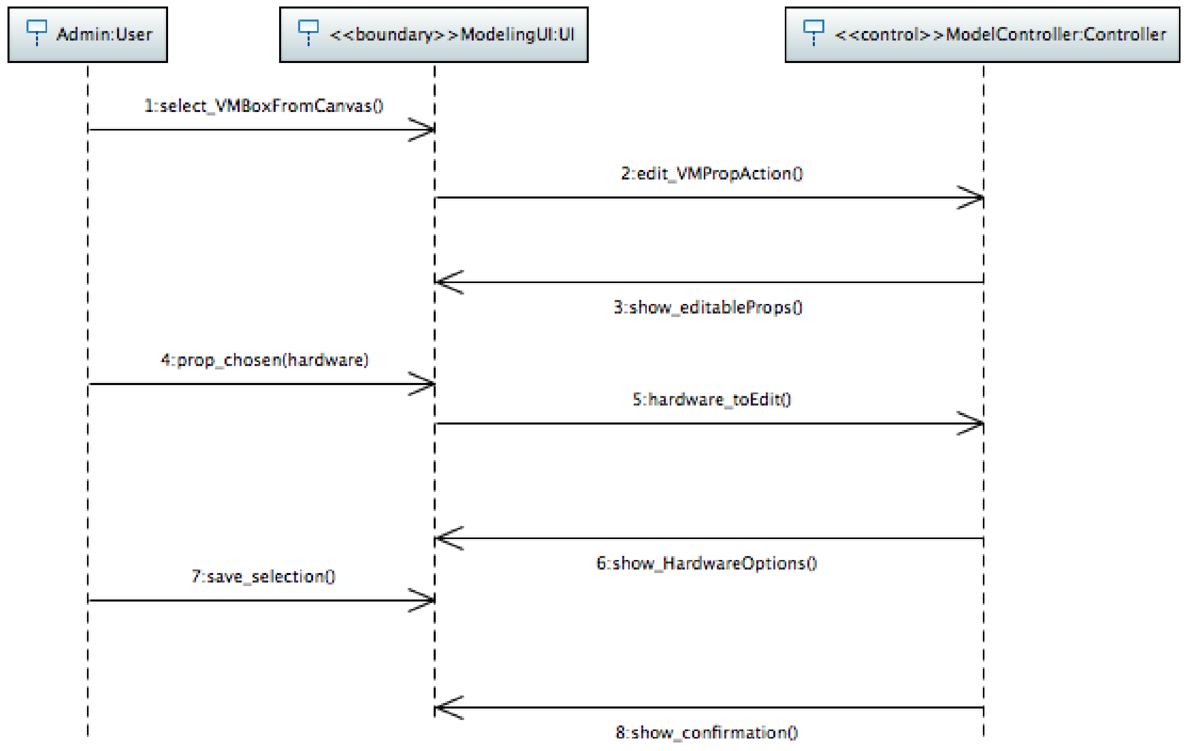
Create User-Defined VM



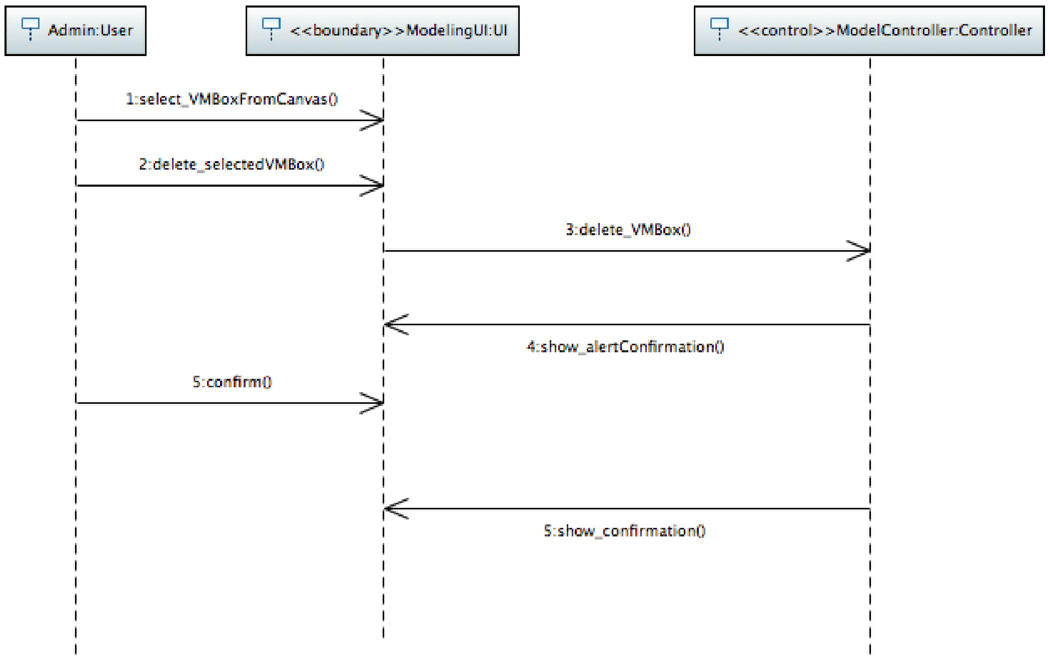
Create VM From Template



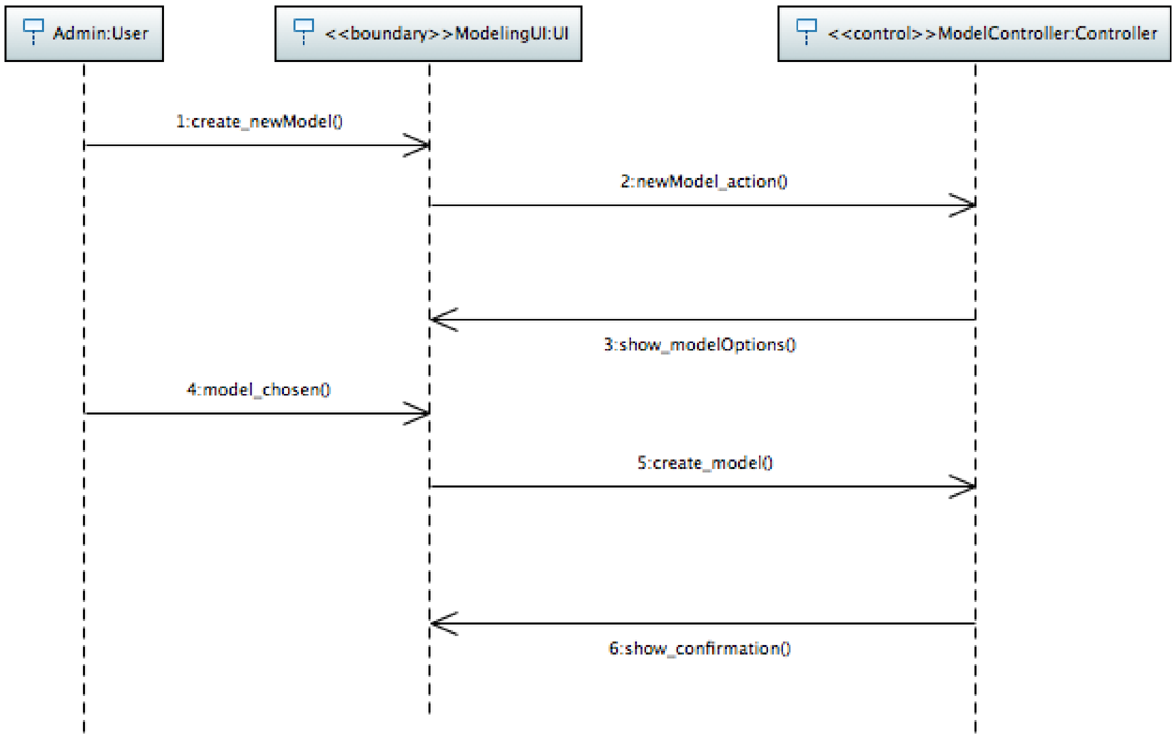
Edit Hardware Property (GVE)



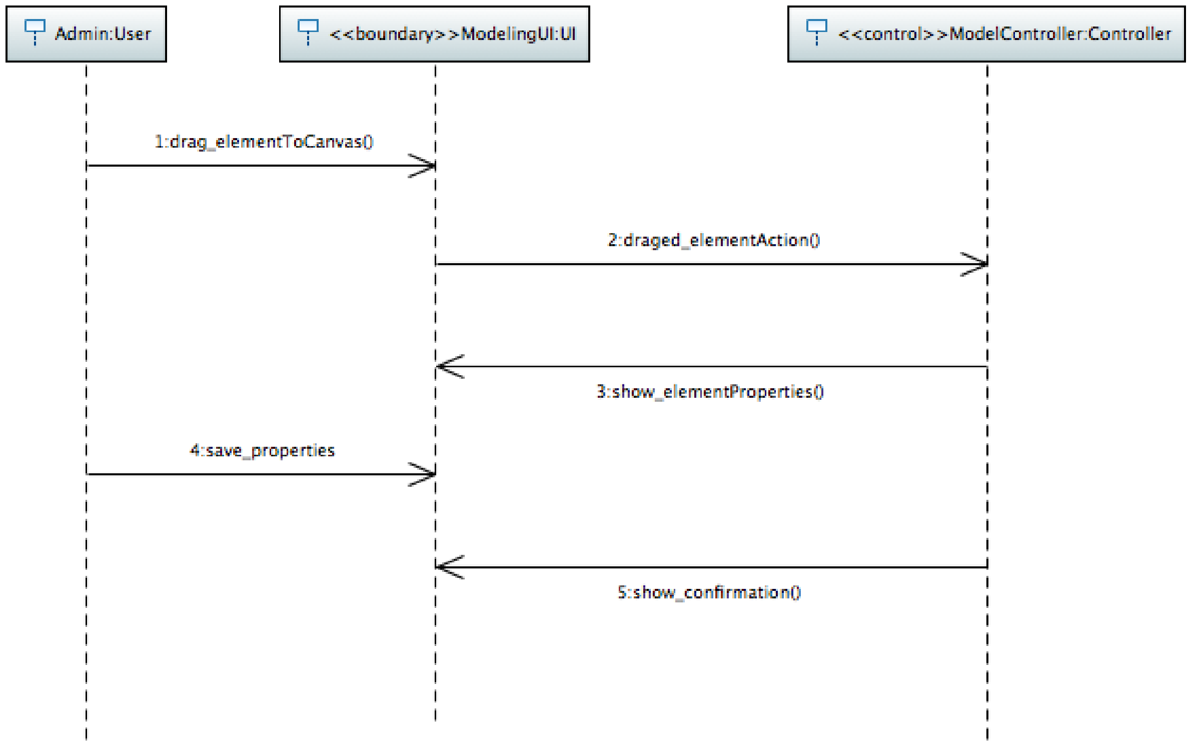
Delete VM (GVE)


Admin:User
<<boundary>>ModelingUI:UI
<<control>>ModelController:Controller
1:select_VMBoxFromCanvas()
2:delete_selectedVMBox()
3:delete_VMBox()
4:show_alertConfirmation()
5:confirm()
5:show_confirmation()

Create New Model

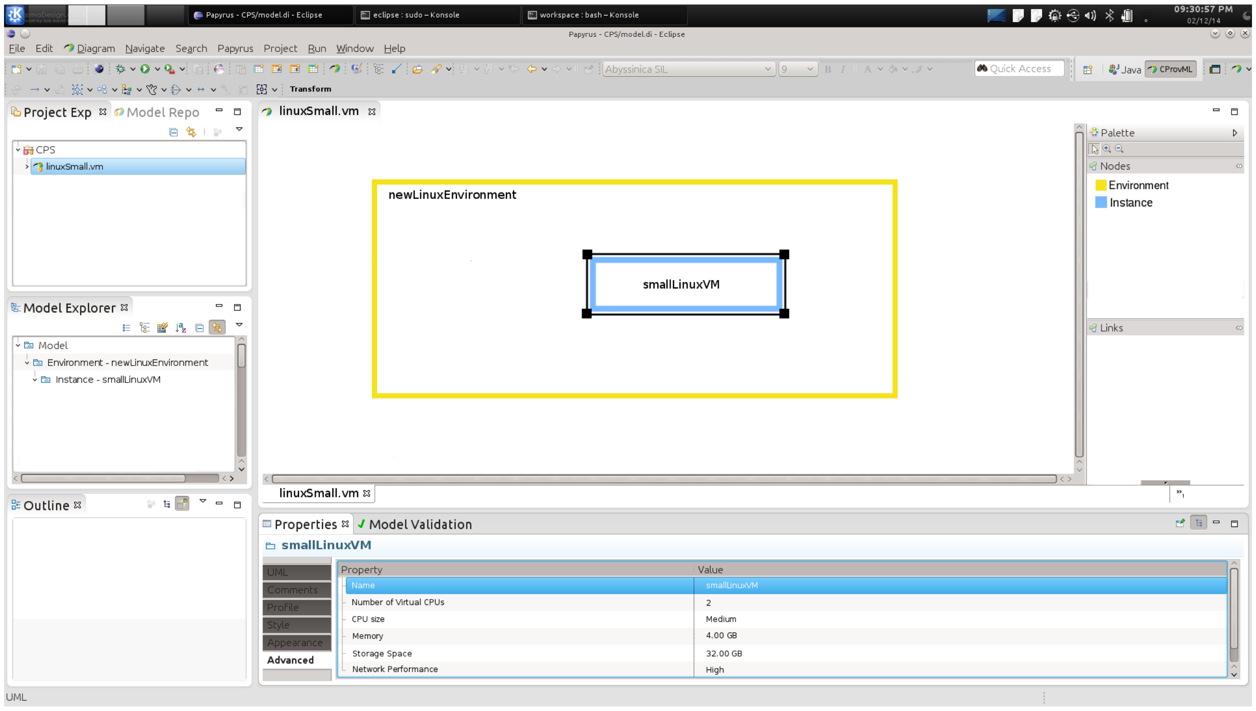


Add Element To Model



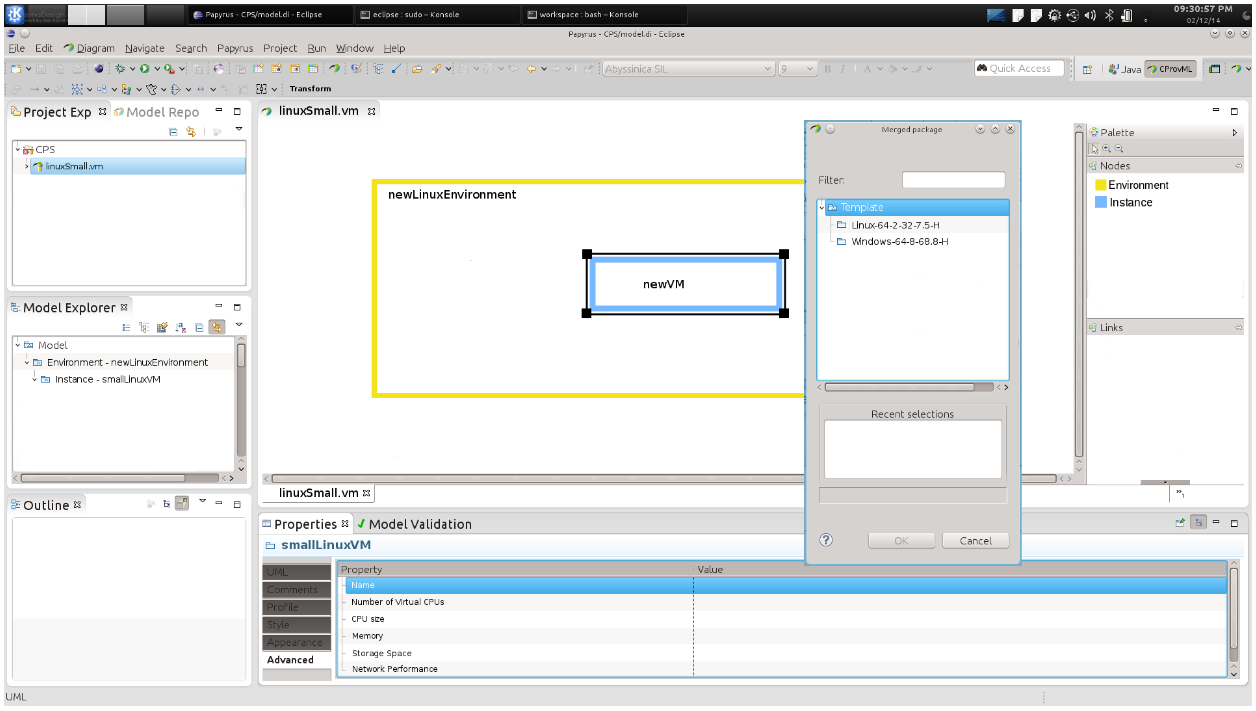
5.4 User Interfaces

Create VM Mockup



Property	Value
Name	smallLinuxVM
Number of Virtual CPUs	2
CPU size	Medium
Memory	4.00 GB
Storage Space	32.00 GB
Network Performance	High

Create VM From Template Mockup



Property	Value
Name	
Number of Virtual CPUs	
CPU size	
Memory	
Storage Space	
Network Performance	

6. Glossary

TERM	DEFINITION
Cloud Provisioning System (CPS)	Is the allocation of a cloud provider's resources to a customer. When a cloud provider accepts a request from a customer, it must create the appropriate number of virtual machines (VMs) and allocate resources to support them.
Virtual Machine (VM)	Is a software implementation of a computing environment in which an operating system (OS) or program can be installed and run.
Graphical Visual Environment (GVE)	Software which allows the use of visual expressions (such as graphics, drawings, animation or icons) in the process of programming.
Domain Spacific Language (DSL)	Is a computer language specialized to a particular application domain.
CPROVML	Cloud Provisioning Language where a graphicallanguage is specified.
G-CProvML	Especification of a graphical visual CPS Model.
X-CProvML	Transformation generated by the GVE from the G-CProvML Model
Aplicattion Programming Interface (API)	Specifies a set of functions or routines that accomplish a specific task or are allowed to interact with a specific software component.

7. Appendix

Appendix A - Project Schedule
Appendix B - Use Cases
Appendix C - User Interface Designs
Appendix D - Diary of meetings and tasks

7.1 Appendix A - Project Schedule

#	Task Name	Duration	Start	Finish	Pred
1	**Cloud Provivioning System**	**58 days**	**Fri 1/24/14**	**Tue 4/15/14**	
2	**Deliverable 1**	**18 days**	**Fri 1/24/14**	**Tue 2/18/14**	
3	**System Requirements**	**11 days**	**Fri 1/24/14**	**Fri 2/7/14**	
4	Feature Diagram Design	1 day	Fri 1/24/14	Fri 1/24/14	
5	Functional Requirements	7 days	Mon 1/27/14	Tue 2/4/14	4
6	Non-Functional Requirements	7 days	Mon 1/27/14	Tue 2/4/14	4
7	Use Cases Definitions and Diagram	10 days	Mon 1/27/14	Fri 2/7/14	4
8	**System Analysis**	**6 days**	**Mon 2/10/14**	**Mon 2/17/14**	
9	Scenario Description for Use Cases	5 days	Mon 2/10/14	Fri 2/14/14	7
10	Object Model	5 days	Mon 2/10/14	Fri 2/14/14	7
11	Dynamic Model	5 days	Mon 2/10/14	Fri 2/14/14	7
12	User Interface	1 day	Mon 2/17/14	Mon 2/17/14	11
13	Presentation	1 day	Tue 2/18/14	Tue 2/18/14	12
14	**Deliverable 2**	**20 days**	**Wed 2/19/14**	**Tue 3/18/14**	**2**
15	**Software Architecture**	**6 days**	**Wed 2/19/14**	**Wed 2/26/14**	
16	Package Diagram Design	6 days	Wed 2/19/14	Wed 2/26/14	13
17	UML Profile	6 days	Wed 2/19/14	Wed 2/26/14	13
18	Subsystem Decomposition	6 days	Wed 2/19/14	Wed 2/26/14	13
19	**Object Design**	**13 days**	**Thu 2/27/14**	**Mon 3/17/14**	
20	Class Diagram for the Subsystem	6 days	Thu 2/27/14	Thu 3/6/14	18
21	Object Interaction	6 days	Thu 2/27/14	Thu 3/6/14	18
22	Detail Class Design	6 days	Fri 3/7/14	Fri 3/14/14	21
23	OCL Constraints	7 days	Fri 3/7/14	Mon 3/17/14	21
24	Presentation	1 day	Tue 3/18/14	Tue 3/18/14	23
25	**Deliverable 3**	**20 days**	**Wed 3/19/14**	**Tue 4/15/14**	**14**
26	**Validation**	**5 days**	**Wed 3/19/14**	**Tue 3/25/14**	
27	Validation of the Use Case Model	5 days	Wed 3/19/14	Tue 3/25/14	24
28	Validation of the Analysis Model	5 days	Wed 3/19/14	Tue 3/25/14	24
29	Validation of the System Model	5 days	Wed 3/19/14	Tue 3/25/14	24
30	Validation of the Detail Design Model	5 days	Wed 3/19/14	Tue 3/25/14	24
31	Implementation	14 days	Wed 3/26/14	Mon 4/14/14	30
32	Presentation	1 day	Tue 4/15/14	Tue 4/15/14	31

Milestones and Deliverables

Task Name	Duration	Start	Finish
Cloud Provisioning System	**58 days**	**Fri 1/24/14**	**Tue 4/15/14**
Deliverable 1	**18 days**	**Fri 1/24/14**	**Tue 2/18/14**
Presentation	1 day	Tue 2/18/14	Tue 2/18/14
Deliverable 2	**20 days**	**Wed 2/19/14**	**Tue 3/18/14**
Presentation	1 day	Tue 3/18/14	Tue 3/18/14
Deliverable 3	**20 days**	**Wed 3/19/14**	**Tue 4/15/14**
Presentation	1 day	Tue 4/15/14	Tue 4/15/14

7.2 Appendix B - Use cases

Cloud Provisioning Use Cases

Create a new VM

Use case ID: T1-CPS-CreateVM
Details:

Actor: Cloud provisioning administrator

Pre-conditions:
1. The system has to be open.

Description:
1. Use case begins when the user select the "New VM" button.
2. The user selects the "Create User-Defined VM" option
3. The system prompts the user to enter the name of the VM.
4. The user click 'OK" button
5. The system display all the properties that the user has to set. (Hardware {CPU, RAM, Storage & Network} and Software {OS, Apps}).
6. The user confirms the configuration.
7. The system confirms the creation of the VM.

Relevant requirements:

N/A

Post-conditions:
1. A new VM is created.

Alternative Courses of Action:
1. In step 2, the user can choose "Create a VM from a Template" (T1-CPS-CreateVMFromTemplate)

Exceptions:

N/A

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-AddElementsToModel
4. T1-GVE-ModifyElementPropertiesOnModel
5. T1-GVE-TransformModel
6. T1-GVE-ValidateModel

Decision Support

Frequency: On average 5 request by day is made by the user
Critically: High , allows the user to create a VM.
Risk: High.

Constraints

1. *Usability:*
    a. Not enough knowledge of a Cloud Provisioning System.
    b. On average an user should take 2 minutes to create the new VM.
2. *Reliability:*
    a. 5% failures for every twenty four hours of operation is acceptable.
3. *Performance:*
    a. Request should be saved within 5 secs.
4. *Supportability:*
    a. The application will need to supported across all main platforms (PC, Mac, etc).

Modification History:

Owner: Team 1

Initiation date: 02/01/2014

Date last modified: 02/08/2014

Create VM From Template

Use case ID: T1-CPS-CreateVMFromTemplate

Details:

Actor: Cloud provisioning administrator

Pre-conditions:

1.

Description:

1. Use case begins when the user clicks the "Create VM from Template" button
2. The system responds by generating a list of all the templates available.

3. User clicks on any of the templates listed.
4. User accepts the selection.
5. The system responds by showing all the properties associated to the template selected by the user.
6. The user confirms the selection
7. Use case ends when the system creates the VM with the template's properties

Relevant requirements:

N/A

Post-conditions:

1. A new VM is added to the model with all the specifications listed in the properties of the template chosen.

Alternative Courses of Action:

1. In step 3, the user has the option to cancel the selection of the template.
2. In step 5, the user has the option to change the template before confirming the selection..

Exceptions:

1. No Templates are shown to the user.

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-AddElementsToModel
4. T1-GVE-ModifyElementPropertiesOnModel
5. T1-GVE-TransformModel
6. T1-GVE-ValidateModel

Decision Support

Frequency: On average 5 request by day is made by the user
Critically: High.
Risk: High.

Constraints

1. Usability:
    a. Not enough knowledge of a Cloud Provisioning System.
    b. On average an user should take 3 minutes to create the new VM based on the template..
2. Reliability:
    a. 5% failures for every twenty four hours of operation is acceptable.
3. Performance:
    a. Request should be sent and saved within 5 secs.
4. Supportability:
    a. The application will need to supported across all main platforms (PC, Mac, etc).

Modification History:

Owner: Team 1

Initiation date: 02/01/2014

Date last modified: 02/08/2014

Delete existing VM(s)

Use case ID: T1-CPS-DeleteExistingVM

Details:

Actor: Cloud provisioning administrator

Pre-conditions:

1. The user has an existing VM

Description:

1. Use case begins when the user selects the VM(s) they would like to delete.
2. The user then clicks on the “Delete” button
3. The system responds by prompting the user to confirm that they would like to delete the VM(s).
4. User confirms that they would like to delete the VM
5. Use case ends when the system has responded back with confirmation that the VM has been deleted.

Relevant requirements:

N/A

Post-conditions:

1. The deleted VM(s) is no longer listed in the list of VMs.

Alternative Courses of Action:

1. In Step 4 the user can choose to cancel.

Exceptions:

N/A

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-ValidateModel

Decision Support:

Frequency: Low. Less than 1 time per year.
Criticality: Low. Operation is not integral to the day to day operations.
Risk: Low.

Constraints

1. Usability:
    a. No prior training is required.
2. Reliability:
    a. 5% failures for every twenty four hours of operation is acceptable.
3. Performance:
    a. The existing VM should be deleted within 1 second of the user confirming deletion.
4. Supportability:
    a. The application will need to supported across all main platforms (PC, Mac, etc).

Modification History:
Owner: Team 1
Initiation date: 02/01/2014
Date last modified: 02/08/2014

Modify Hardware

Use case ID: T1-CPS-ModifyHardware

Details:

Actor: Cloud provisioning administrator

Pre-conditions:

1. User has an existing VM

Description:

1. Use case begins when user initiates this action by selecting a VM
2. System responds by presenting the user a list of components that can be modified.
3. The user select the hardware component they would like to modify
4. The system list the options available for the selected hardware component.
5. The user makes his selections.
6. The user saves his changes
7. Use case ends when the system confirms that the changes have been applied.

Relevant requirements:

N/A

Post-conditions:

1. Selected VM's properties have changed

Alternative Courses of Action:

1. In step 6, the user can choose to cancel and go back to step 2.

Exceptions:

N/A

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-ModifyElementPropertiesOnModel
4. T1-GVE-ValidateModel

Decision Support

Frequency: On average 1 request by day is made by the user
Critically: Medium.
Risk: Medium.

Constraints

1. Usability:
    a. On average an user should take 2 minutes to modify the hardware properties.
2. Reliability:
    a. 5% failures for every twenty four hours of operation is acceptable.
3. Performance:
    a. Request should be sent and saved within 5 secs.
4. Supportability:
    a. The application will need to supported across all main platforms (PC, Mac, etc).

Modification History:

Owner: Team 1

Initiation date: 02/01/2014

Date last modified: 02/08/2014

Clone VM

Use case ID: T1-CPS-CloneVM

Details:

Actor: Cloud provisioning administrator

Pre-conditions:

1. User has opened the system.
2. There is at least one VM created in the environment.

Description:

1. Use case begins when the user clicks the “Clone VM” button.
2. The system list all the VM’s that have been created on the current environment.
3. The user selects the VM’s that he wants to clone.
4. The system prompts the user to enter the name for the cloned VM.
5. The user enters the name and clicks “OK”

6. The system sets the name.
7. Use case ends when the system clones the VM on the current environment.

Relevant requirements:

N/A

Post-conditions:

1. The cloned VM is created

Alternative Courses of Action:

1. On step 3, the user can select more than one VM.

Exceptions:

1. The user enters a name that already exist. The system will alert the user to choose another name

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-AddElementsToModel
4. T1-GVE-ModifyElementPropertiesOnModel
5. T1-GVE-TransformModel
6. T1-GVE-ValidateModel

Decision Support

Frequency: On average 5 request by day is made by the user
Critically: Medium.
Risk: Medium.

Constraints

1. Usability:
   a. On average an user should take 2 minutes to clone the VM based on the template.
2. Reliability:
   a. 5% failures for every twenty four hours of operation is acceptable.
3. Performance:
   a. Request should be sent and saved within 5 secs.
4. Supportability:
   a. The application will need to supported across all main platforms (PC, Mac, etc).

Modification History:

Owner: Team 1

Initiation date: 02/01/2014

Date last modified: 02/08/2014

Modify Software

Use case ID: T1-CPS-ModifySoftware
Details:

Actor: Cloud provisioning administrator

Pre-conditions:

1. User has an existing VM with some software installed

Description:

1. The use case begins when the user selects the VM whose software they would like to modify from the list of VM's that exist in the current environment.
2. The system will respond by showing a list of software.
3. The user will then select the software they would like to modify.
4. System responds by prompting to user to confirm that they would like to perform the action
5. User confirms that they would like to proceed.
6. The use case ends when system responds by alerting user that the software modification has been completed.

Relevant requirements:

N/A

Post-conditions:

1. The selected VM's software has changed.

Alternative Courses of Action:

1. In step 2 the user can instead choose to modify the operating system.
    a. In step 4 the user can choose to cancel.

Exceptions:

N/A

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-ValidateModel

Decision Support

Frequency: Low - 4 times per year.
Criticality: Medium. The software to be modified might be a mission critical application.
Risk: Low

Constraints

Usability: No previous training is needed.
Reliability: Mean time to failure - 1 failure for every 100 times the operation is performed
Performance: The software modification should occur within 1 sec of the user confirming.
Supportability: The application will need to supported across all main platforms (PC, Mac, etc).

Cloud Provisioning - Graphical Visual Environment (CPS-GVE) Use Cases

Create a new VM

Use case ID: T1-CPS-GVE-CreateVM

Details:

Actor: Cloud provisioning administrator

Pre-conditions:

1. User has opened the Graphical Visual Environment

Description:

1. Use case begins when user initiates this action by clicking on the “New Model” menu option
2. The system responds by presenting the user with a blank canvas and a palette of modeling components and relationships
3. User drags a Cloud Provider block onto canvas
4. User sets the Provider block’s properties (e.g. SSH key name, auth credentials)
5. User drags a VM block on top of the Provider block
6. User sets VM block’s properties (e.g. type, name, specs)
7. Use case ends when user saves model (see Use Case: T1-GVE-SaveModel)

Relevant requirements:

N/A

Post-conditions:

1. A new G-CProvML file is created on the file system.

Alternative Courses of Action:

1. Instead of step D1, the user has option to initiate a new model via clicking the “New Model” button from the toolbar
2. Instead of step D1 and D2, the user can create a new VM on an existing model.

Exceptions:

1. See Exceptions from Use Case: T1-GVE-SaveModel

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-AddElementsToModel
4. T1-GVE-ModifyElementPropertiesOnModel
5. T1-GVE-TransformModel
6. T1-GVE-ValidateModel

Decision Support

Frequency: On average 1 request is made by the user

Critically: High, allows the user to create a VM.
Risk: High, implementing this use case will imply to communicate with the provider.

Constraints

1. Usability:
    a. Not enough knowledge of a Cloud Provisioning System.
    b. On average a user should take 3 minutes to create the new VM.
2. Reliability:
    a. 5% failures for every twenty four hours of operation is acceptable.
3. Performance:
    a. Request should be sent and saved within 10 secs.
4. Supportability:
    a. The request should be correctly handled by the GVE

Modification History:
Owner: Team 1
Initiation date: 02/01/2014
Date last modified: 02/08/2014

Create VM from Template

Use case ID: T1-CPS-GVE-CreateVMFromTemplate

Details:

Actor: Cloud provisioning administrator

Pre-conditions:

1. User has opened the Graphical Visual Environment

Description:

1. Use case begins when the user clicks the “Create VM from Template” button
2. The system responds by generating a list of all the templates available.
3. User clicks on any of the templates listed.
4. The system responds by showing all the properties associated to the template selected by the user.
5. The user confirms the selection
6. Use case ends when the system creates the VM with the template’s properties and select it on the canvas.

Relevant requirements:

N/A

Post-conditions:

1. A new VM is added to the model with all the specifications listed in the properties of the template chosen.

Alternative Courses of Action:

1. Instead of step D5, the user has the option to cancel the creation of the VM from the model.

Exceptions:

1. No Templates are shown to the user.

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-AddElementsToModel
4. T1-GVE-ModifyElementPropertiesOnModel
5. T1-GVE-TransformModel
6. T1-GVE-ValidateModel

Decision Support

Frequency: On average 1 request is made by the user
Critically: High, allows the user to create a VM from a template.
Risk: High.

Constraints

1. Usability:
    a. Not enough knowledge of a Cloud Provisioning System.
    b. On average a user should take 3 minutes to create the new VM based on the template.
2. Reliability:
    a. a. 5% failures for every twenty four hours of operation is acceptable.
3. Performance:
    a. Request should be sent and saved within 5 secs.
4. Supportability:
    a. The request should be correctly handled by the GVE

Modification History:
Owner: Team 1
Initiation date: 02/01/2014
Date last modified: 02/08/2014

Delete an existing VM

Use case ID: T1-CPS-GVE-DeleteExistingVM

Details:

Actor: Cloud provisioning administrator

Pre-conditions:

1. User has opened the Graphical Visual Environment
2. User has model with existing VM

Description:

1. Use case begins when user initiates this action by selecting an existing VM on the canvas
2. The system responds by highlighting the user's selection
3. User clicks on the "Delete" menu option
4. System responds by showing a confirmation dialog with the name of the VM selected.
5. User confirms deletion.
6. The system deletes all the relations that this VM has with other artifacts.
7. Use case ends when the selected VM is removed from the canvas

Relevant requirements:
N/A

Post-conditions:

1. Selected VM is removed from model

Alternative Courses of Action:

1. Instead of step 3, the user has option to initiate a delete via right-clicking on the selected VM element.
2. In step 5, the user has the option to cancel the deletion.

Exceptions:
N/A

Related Use Cases:

1. T1-GVE-Open
2. T1-GVE-SaveModel
3. T1-GVE-ValidateModel

Decision Support
Frequency: On average 1 request is made by the user
Critically: High, allows the user to delete a VM from the model.
Risk: High, implementing this use case will imply to communicate with the provider.

Constraints

1. Usability:
   a. Not enough knowledge of a Cloud Provisioning System.
   b. On average a user should take 3 minutes to create the new VM based on the template.
2. Reliability:
   a. 5% failures for every twenty four hours of operation is acceptable.
   b. Down time for Login Back-up 30 minutes in a 24 hour period.
3. Performance:
   a. Request should be sent and saved within 5 sec.
4. Supportability:
   a. The request should be correctly handled by the GVE

Modification History:
Owner: Team 1

Initiation date: 02/01/2014
Date last modified: 02/08/2014

Modify Hardware

Use case ID: T1-CPS-GVE-ModifyHardware
Details:

Actor: Cloud provisioning administrator

Pre-conditions:
1. User has opened the Graphical Visual Environment
2. User has model with existing VM

Description:
1. Use case begins when user initiates this action by selecting an existing VM on the canvas
2. The system responds by highlighting the user's selection and displaying the properties for the selected VM.
3. The user modifies any of the hardware properties.
4. Use case ends when the user saves his changes

Relevant requirements:
N/A

Post-conditions:
1. Selected VM's properties have changed

Alternative Courses of Action:
N/A

Exceptions:
1. No values for the properties are shown.

Related Use Cases:
1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-ModifyElementPropertiesOnModel
4. T1-GVE-ValidateModel

Decision Support
Frequency: On average 1 request is made by the user
Critically: Medium, allows the user to modify the hardware of the VM.
Risk: Medium,

Constraints
1. Usability:
   a. On average an user should take 3 minutes to create the new VM based on the template

2. Reliability:
    a. 5% failures for every twenty four hours of operation is acceptable.
3. Performance:
    a. Request should be sent and saved within 5 sec.
4. Supportability:
    a. The request should be correctly handled by the GVE

Modification History:
Owner: Team 1
Initiation date: 02/01/2014
Date last modified: 02/08/2014

Create environment consisting of multiple VMs

Use case ID: T1-CPS-GVE-CreateEnvironment
Details:

Actor: Cloud provisioning administrator

Pre-conditions:

1. User has opened the Graphical Visual Environment

Description:

1. Use case begins when user initiates this action by clicking on the “New Model” menu option
2. The system responds by presenting the user with a blank canvas and a palette of modeling components and relationships
3. User drags a Cloud Provider block onto canvas
4. User sets the Provider block’s properties (e.g. SSH key name, auth credentials)
5. User drags an Environment block on top of the Provider block
6. User sets Environment block’s properties (e.g. type, name, specs)
7. Use case ends when user saves model (see Use Case: T1-GVE-SaveModel)

Relevant requirements:

N/A

Post-conditions:

1. A new G-CProvML file is created on the file system.

Alternative Courses of Action:

1. Instead of step D1, the user has option to initiate a new model via clicking the “New Model” button from the toolbar
2. Instead of step D1 and D2, the user can create a new environment on an existing model.

Exceptions:

1. See Exceptions from Use Case: T1-GVE-SaveModel

Related Use Cases:
1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-AddElementsToModel
4. T1-GVE-ModifyElementPropertiesOnModel
5. T1-GVE-TransformModel
6. T1-GVE-ValidateModel

Decision Support

Frequency: On average 5 request by day is made by the user
Critically: High , allows the user to create a VM.
Risk: High.

Constraints
1. *Usability:*
    a. Not enough knowledge of a Cloud Provisioning System.
    b. On average an user should take 2 minutes to create the new VM.
2. *Reliability:*
    a. 5% failures for every twenty four hours of operation is acceptable.
3. *Performance:*
    a. Request should be saved within 5 secs.
4. *Supportability:*
    a. The application will need to supported across all main platforms (PC, Mac, etc).

Create Template from VM
Use case ID: T1-CPS-GVE-ModifySoftware
Details:

Actor: Cloud provisioning administrator

Pre-conditions:
1. User has opened the Graphical Visual Environment
2. User has model with existing VM or Environment

Description:
1. Use case begins when user initiates this action by dragging a Snapshot block onto the canvas
2. User then creates a “Of” relationship between the existing VM/Environment and the new Snapshot block
3. Use case ends when the user saves the model

Relevant requirements:

N/A

Post-conditions:
1. A new Snapshot element and corresponding relationship is added to model

Alternative Courses of Action:

N/A

Exceptions:

N/A

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-ValidateModel

Graphical Visual Environment (GVE) Use Cases

Create a new Model

Use case ID: T1-GVE-CreateModel
Details:

Actor: User

Pre-conditions:

1. The graphical visual modeling environment has to be opened.

Description:

1. Use case begins when the user clicks on the create ‘New Model’ button.
2. The system responds by display a list of available modeling templates.
3. The user selects the template
4. The system prompts the user to enter a name for the model.
5. The user enters name and clicks “OK” button
6. The system display all the objects that can be dragged into the canvas in order to form the model.

Relevant requirements:

N/A

Post-conditions:

1. A new model is created.

Alternative Courses of Action:

1. In step 1, the user can choose to create Model in a specific folder by clicking the File button.

Exceptions:

N/A

Related Use Cases:

1. T1-GVE-SaveModel

2. T1-GVE-AddElementsToModel
3. T1-GVE-ModifyElementPropertiesOnModel
4. T1-GVE-ValidateModel

Decision Support

Frequency: On average 5 request by day is made by the user
Critically: High , allows the user to create a new Model.
Risk: High.

Constraints

1. Usability:
   a. Not enough knowledge of a Graphical Visual Environment.
   b. On average an user should take 1 second for a user to create a Model.
2. Reliability:
   a. 2% failures for every twenty four hours of operation is acceptable.
3. Performance:
   a. Request should be saved within 5 secs.
4. Supportability:
   a. The application will need to be supported across all main platforms (PC, Mac, etc).

Modification History:
Owner: Team 1
Initiation date: 02/12/2014
Date last modified: 02/12/2014

Open existing Model

Use case ID: T1-GVE-OpenModel
Details:

Actor: User

Pre-conditions:

1. The graphical visual modeling environment has to be opened.

Description:

1. Use case begins when the user clicks on the 'Open Model' button.
2. The system prompts the user to choose the path where the model is stored.
3. The user click 'OK" button
4. The system display in the canvas all the objects and connections that form the Model.

Relevant requirements:

N/A

Post-conditions:

1. An existing Model is opened.

Alternative Courses of Action:

1. In step 1, the user can choose to open Model in a specific folder by clicking the File dropdown menu item.
2. The system shows the list of items in the dropdown list.
3. User clicks 'Open Model' item from the list.

Exceptions:

N/A

Related Use Cases:

1. T1-GVE-SaveModel
2. T1-GVE-AddElementsToModel
3. T1-GVE-ModifyElementPropertiesOnModel
4. T1-GVE-ValidateModel

Decision Support

Frequency: On average 5 request by day is made by the user
Critically: High , allows the user to open an existing Model.
Risk: High.

Constraints

1. Usability:
   a. Not enough knowledge of a Graphical Visual Environment.
   b. On average an user should take 1 second for a user to open a Model.
2. Reliability:
   a. 2% failures for every twenty four hours of operation is acceptable.
3. Performance:
   a. Request should be saved within 5 secs.
4. Supportability:
   a. The application will need to be supported across all main platforms (PC, Mac, etc).

Modification History:

Owner: Team 1
Initiation date: 02/12/2014
Date last modified: 02/12/2014

Save a Model

Use case ID: T1-GVE-SaveModel
Details:

Actor: User

Pre-conditions:

1. The graphical visual modeling environment has to be opened.
2. A Model has to be opened.

Description:

    1. Use case begins when the user clicks on the ‘Save Model’ button.
    2. The system displays a confirmation message that the Model has been saved.

Relevant requirements:
N/A

Post-conditions:
    1. A Model is saved.

Alternative Courses of Action:
N/A

Exceptions:
N/A

Related Use Cases:
1. T1-GVE-AddElementsToModel
2. T1-GVE-ModifyElementPropertiesOnModel
3. T1-GVE-ValidateModel

Decision Support
Frequency: On average 5 request by day is made by the user
Critically: High , allows the user to save a Model.
Risk: High.

Constraints
1. Usability:
    a. Not enough knowledge of a Graphical Visual Environment.
    b. On average an user should take 1 second for a user to create a Model.
2. Reliability:
    a. 2% failures for every twenty four hours of operation is acceptable.
3. Performance:
    a. Request should be saved within 5 secs.
4. Supportability:
    a. The application will need to be supported across all main platforms (PC, Mac, etc).

Modification History:
Owner: Team 1
Initiation date: 02/12/2014
Date last modified: 02/12/2014

Add Elements To Model
Use case ID: T1-GVE-AddElementsToModel
Details:
Actor: User

Pre-conditions:

1. The graphical visual modeling environment has to be opened.
2. A Model has to be opened.

Description:

1. Use case begins when the user drags an element to the canvas.
2. The system prompts the user to fill the properties of the elements.
3. The user clicks Create.
4. The system displays a confirmation message that the properties have been applied to the object.

Relevant requirements:

N/A

Post-conditions:

1. An element has been added to the Model.

Alternative Courses of Action:

N/A

Exceptions:

N/A

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-ModifyElementPropertiesOnModel
4. T1-GVE-ValidateModel

Decision Support

Frequency: On average 5 request by day is made by the user

Critically: High , allows the user to add an element to a Model.

Risk: High.

Constraints

1. Usability:
    a. Not enough knowledge of a Graphical Visual Environment.
    b. On average an user should take 1 second for a user to add an element to a Model.
2. Reliability:
    a. 2% failures for every twenty four hours of operation is acceptable.
3. Performance:
    a. Request should be saved within 5 secs.
4. Supportability:
    a. The application will need to be supported across all main platforms (PC, Mac, etc).

Modification History:

Owner: Team 1

Initiation date: 02/12/2014

Date last modified: 02/12/2014

Modify Element Properties On Model

Use case ID: T1-GVE-ModifyElementPropertiesOnModel

Details:

Actor: User

Pre-conditions:

1. The graphical visual modeling environment has to be opened.
2. A Model has to be opened.

Description:

1. Use case begins when the user clicks over an element in the canvas.
2. The system displays to the user the properties of the element.
3. The user edits the properties and clicks OK.
4. The system displays a confirmation message that the properties have been successfully edited.

Relevant requirements:

N/A

Post-conditions:

1. An element’s properties has been edited.

Alternative Courses of Action:

N/A

Exceptions:

N/A

Related Use Cases:

1. T1-GVE-OpenModel
2. T1-GVE-SaveModel
3. T1-GVE-ValidateModel

Decision Support

Frequency: On average 5 request by day is made by the user
Critically: High , allows the user to edit an element’s properties in a Model.
Risk: High.

Constraints

1. Usability:
    a. Not enough knowledge of a Graphical Visual Environment.
    b. On average an user should take 1 second for a user to edit an element’s properties in a Model.
2. Reliability:

    a. 2% failures for every twenty four hours of operation is acceptable.
3. Performance:
    a. Request should be saved within 5 secs.
4. Supportability:
    a. The application will need to be supported across all main platforms (PC, Mac, etc).

Modification History:
Owner: Team 1
Initiation date: 02/12/2014
Date last modified: 02/12/2014

Validate Model

Use case ID: T1-GVE-ValidateModel
Details:

Actor: Cloud provisioning administrator

Pre-conditions:

1. The graphical visual modeling environment has to be opened.
2. A Model has to be opened.
3. The model has to be saved.

Description:

1. The use case begins when the user press the “Validate Model” button.
2. The system checks that all the relations among artifacts are correctly done.
3. The system checks that all the artifacts attributes are properly set.
4. The use case ends when the system display a “Validation success, 0 errors” message

Relevant requirements:

N/A

Post-conditions:

1. A validated model is saved.

Alternative Courses of Action:

1. In step 2 the system can find a relation that is not correctly done, the system takes the artifact’s name and saves it to the list of errors that will be shown to the user after the execution of the validation process.
2. In step 3 the system can find a property that is not correctly set, the system takes the artifact’s name and saves it to the list of errors that will be shown to the user after the execution of the validation process.

Exceptions:

N/A

Related Use Cases:

1. T1-GVE-CreateModel

2. T1-GVE-OpenModel
3. T1-GVE-SaveModel
4. T1-GVE-ModifyElementPropertiesOnModel

Decision Support

Frequency: On average 5 request by day is made by the user
Critically: High.
Risk: High.

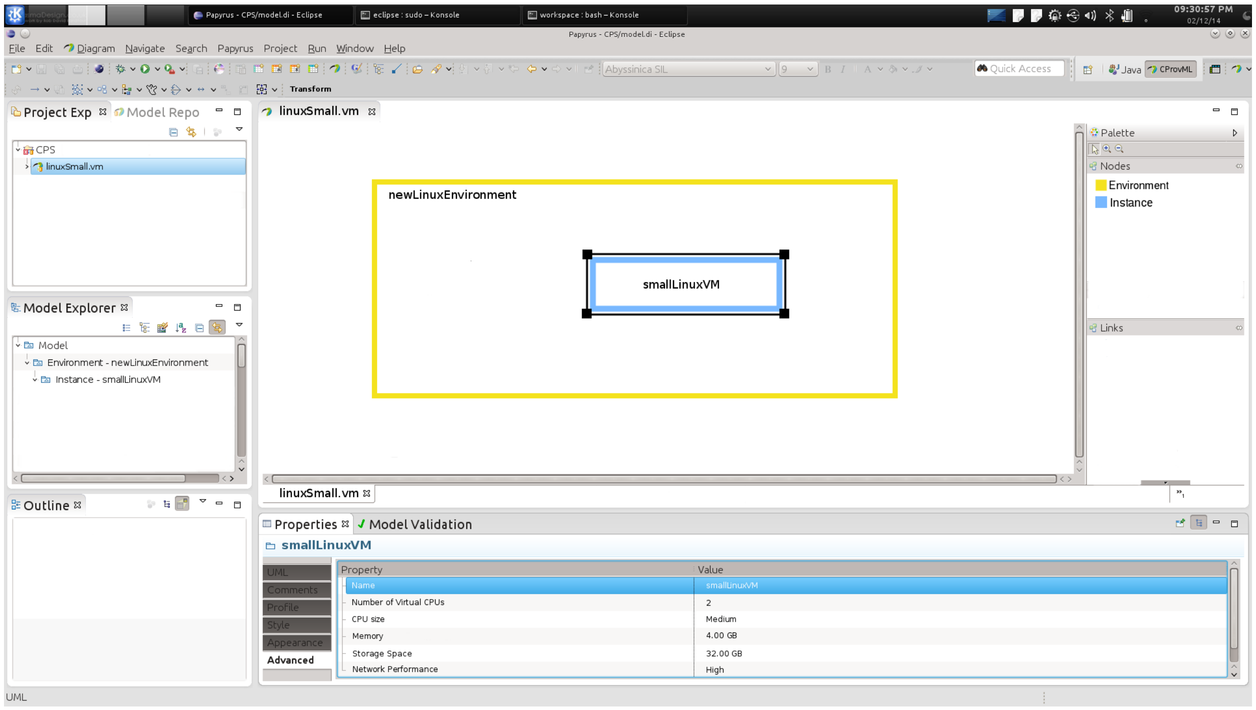
Constraints

1. Usability:
    a. Not enough knowledge of a Graphical Visual Environment.
    b. On average an user should take 1 second for a user to add an element to a Model.
2. Reliability:
    a. 2% failures for every twenty four hours of operation is acceptable.
3. Performance:
    a. Request should be saved within 5 secs.
4. Supportability:
    a. The application will need to be supported across all main platforms (PC, Mac, etc).

Modification History:
Owner: Team 1
Initiation date: 02/12/2014
Date last modified: 02/12/2014

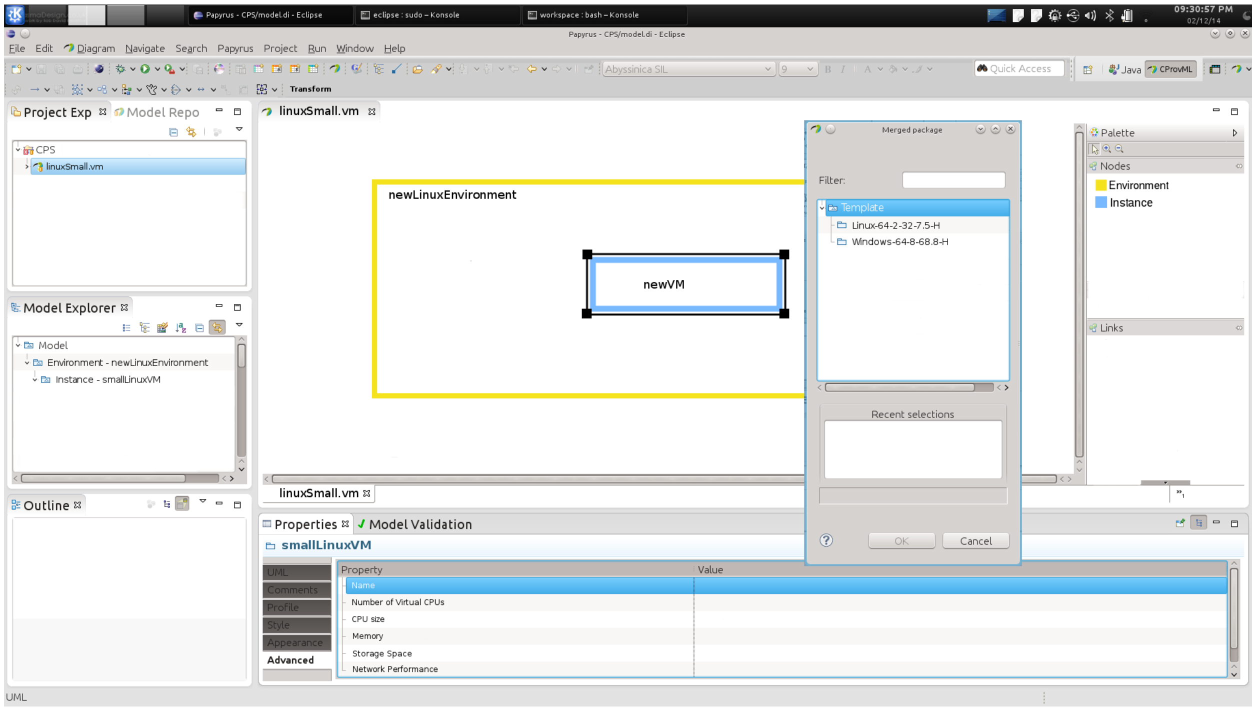
7.3 Appendix C - User interface designs

Create VM Mockup



Property	Value
Name	smallLinuxVM
Number of Virtual CPUs	2
CPU size	Medium
Memory	4.00 GB
Storage Space	32.00 GB
Network Performance	High

Create VM From Template Mockup



Property	Value
Name	
Number of Virtual CPUs	
CPU size	
Memory	
Storage Space	
Network Performance	

7.4 Appendix D - Diary of meetings and tasks

Date/time	Location	Team Members Present	Team Members Late	Summary of Meeting	Tasks Assigned
01/17/2014	Grad Lab (ECS 252)	Rachel Chavez Roly Vicaria Javier Luque Rudy Padron		We worked on developing the feature model.	Roly was tasked with creating a graphical representation of the feature model.
02/01/2014	Grad Lab (ECS 252)	Roly Vicaria Javier Luque Rudy Padron		We identified the 18 uses cases. Created the use case diagram. Wrote up the use case description for 5 of the 6 cloud provisioning use cases. Began working on the SRAD.	None
02/02/2014	Grad Lab (ECS 252)	Roly Vicaria Javier Luque Rudy Padron		Met to assign sections of the SRAD to each team member.	Rudy to work on Section 1 of the SRAD. Roly to work on Section 2 of the SRAD. Javier to work on Section 3 of the SRAD.
02/08/2014	Grad Lab (ECS 252)	Roly Vicaria Javier Luque Rachel Chavez Rudy Padron	Rachel Chavez Rudy Padron	We met to discuss what remains and assign tasks.	Roly to work on class diagram. Javier and Rudy to work on writing use cases and scenarios. Rachel to work on sequence diagrams.
02/09/2014	Grad Lab (ECS 252)	Roly Vicaria Javier Luque Rudy Padron		We met to discuss what remains and assign tasks.	Roly to work on object diagrams. Javier and Rudy to work on activity diagrams.
02/10/2014	JCCL (ECS 241)	Roly Vicaria Javier Luque Rudy Padron		We discussed the progress made on the previously assigned tasks and assigned new tasks.	Javier and Rudy to work on state machine diagrams. Rachel was assigned to work on Sequence diagrams, Section 4 & abstract of the

					document and presentation 1. Roly to work on the UI mockups.
02/12/2014	JCCL (ECS 241)	Roly Vicaria Javier Luque Rachel Chavez Rudy Padron		We discussed the progress made on the previously assigned tasks.	Rachel was assigned to work on the modeling-only use cases. All others continued with their prior assignments
02/15/2014	Grad Lab (ECS 252)	Roly Vicaria Javier Luque Rudy Padron			Rudy was tasked with working on the abstract. Rachel was tasked with putting together the presentation.
02/16/2014	Grad Lab (ECS 252)	Roly Vicaria Javier Luque Rudy Padron		We discussed the progress made on the previously assigned tasks.	Rudy was tasked with reworking the use case diagram. Javier was tasked with developing the glossary.
02/17/2014	Grad Lab (ECS 252)	Roly Vicaria Javier Luque Rudy Padron		We discussed the progress made on the previously assigned tasks.	