VCE v.3 Tutorial

Date:	June 2nd, 2014
Editor:	Oleksandra Kulankhina (INRIA, OASIS team) oleksandra.kulankhina@inria.fr Edited by: Nassim Jibai (INRIA, Scale Team) nassim.jibai@inria.fr
Version:	0.1

1. Introduction

This guide explains the basic functionality of VCE v.3. It provides the instructions for the implementation of a simple example of a Component model. It also explains how to validate the static semantics properties of the Component models.
VCE v.3 is based on Obeo Designer technology and possesses the functionality of the standard editors created with Obeo Designer. Hence, if one wants to get more information, it is recommended to read some of the Obeo Designer documentations in addition to this tutorial. In particular:

- Getting started for End-users: http://docs.obeonetwork.com/obeodesigner/6.1/Getting_Started_User.html
- How to create and manage Modeling Projects: http://docs.obeonetwork.com/obeodesigner/6.1/viewpoint/user/general/Modeling%20Project.html
- How to create and manage diagrams: http://www.obeonetwork.com/group/obeo-designer/page/obeo-designer-reference-document-6-1

1.1. Overview

All the models and diagrams are stored in a **VCE Modeling Project**. A **VCE** Modeling project can contain **VCE** and/or **UML models** and **one VTY model** which contains the types definition. It can also contain **VCE Components diagrams, UML diagrams** and a **VCE Types diagram** which illustrate the elements of the models. A VCE model can have links to UML models. In particular, a GCM interface can refer to a UML Interface, a GCM Primitive component refers to a UML Class. The UML Operations use VCE Types for the typing of the arguments and return values.

2. The first example

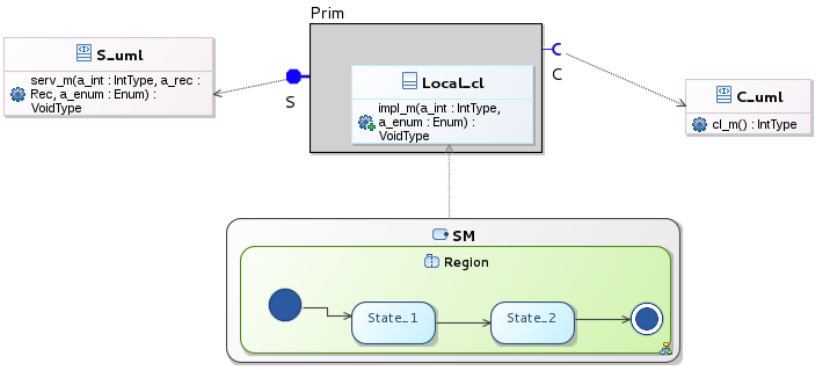
In this section we explain how to:

- create a VCE Project;
- edit GCM Component diagrams;
- specify types;
- create and edit UML classes, interfaces and operations;
- connect the operations of UML classes and interfaces;
- attach a UML interface to a GCM interface;
- attach a UML class to a GCM primitive component;
- create and edit a State Machine and attach it to a UML operation.

2.1. Example overview

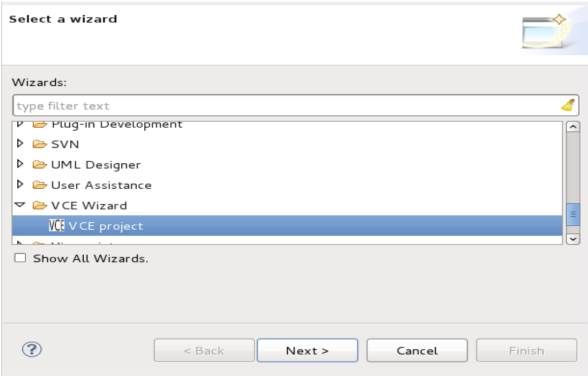
This section provides an example of a simple VCE project creation illustrated at a figure below. The VCE Component diagram has one primitive component **Prim** with one server and one client GCM interface (**S** and **C**). Each GCM interface is connected to a UML interface (**S_uml** and **C_uml** correspondingly). **S_uml** has an operation that can be served by the component, **C_uml** has the list of the operations that can be called by the component. A UML class **Local_cl** is attached to the component. It has one method implementing the operation of the server interface (**s_m).** The behavior of the method is

illustrated on a UML State Machine **SM**. The method **s_m** takes three arguments: **a_int** of type **IntType**, **a_rec** of type **Rec** and **a_enum** of type **Enum**. All the types are specified in VceTypes model.



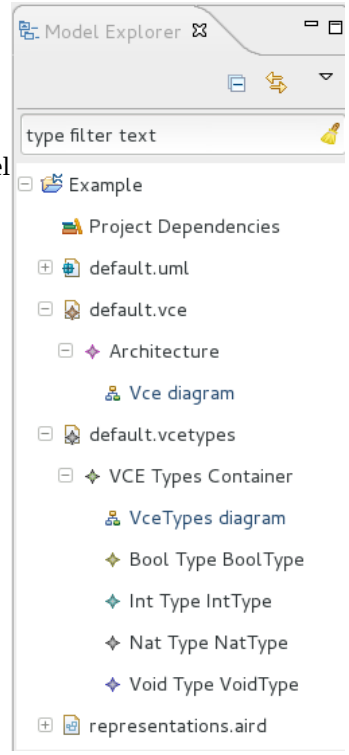
2.2. VCE Project creation

- Create a new VCE model. Right-click in the Model Explorer and select **"New->Other..."**
- Select "**VCE Wizard->VCE project"** and press on **"Next"**



- Give a name to the project
- Press on "**Finish**"

As a result, a new VCE project should be created. Its structure is illustrated at the figure below.

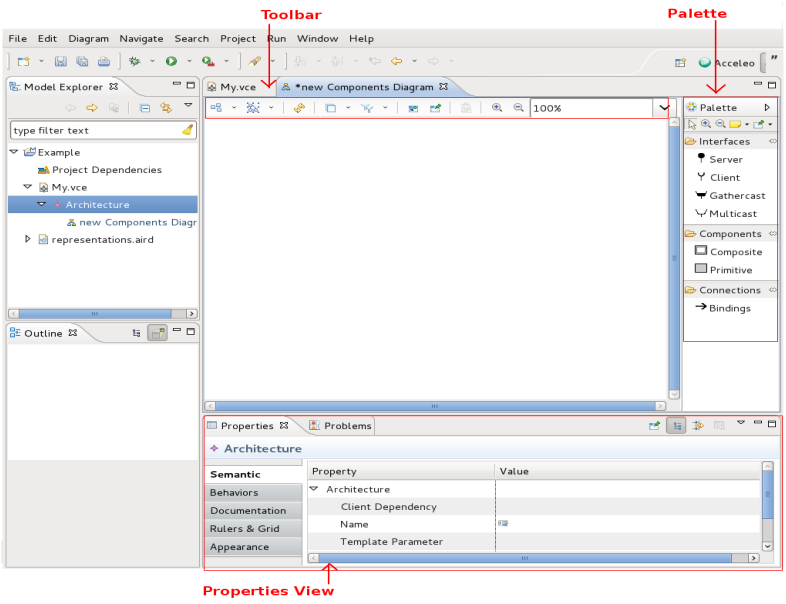


- A VCE project has the following elements:
- default.uml – a UML model;
- default.vce – a VCE model;
- VCE diagram – a VCE Component diagram illustraiting the elements of a VCE model specified in default.vce;
- default.vcetypes – a model containing the types specification;
- VceTypes diagram illustrates the types specified in default.vcetypes.

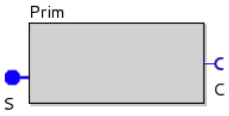
2.3. Build the component Architecture in VCE components diagrams

This section describes how to edit model in VCE components diagram.

- Open VCE components diagram by double-clicking on it. You can edit it using tools on the Palette. You can also use a toolbar on the top of a diagram editor. You can change the properties of your model elements using Properties View.



For our first example pick-up the tool called “Primitive” and draw a primitive component. Then, add to the component two interfaces using “Server” and “Client” tools. The diagram should look like this:



2.4. Types specification

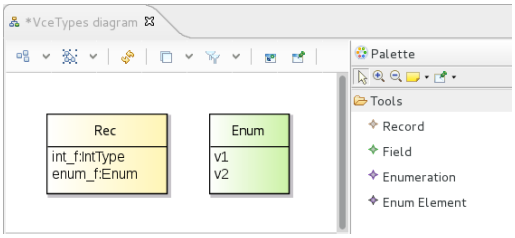
This section describes how to specify the types. The types specification, by default includes, four primitive types: **VoidType, BoolType, IntType** and **NatType**. However, one can construct its own types: **Enumerations** and **Records**. Our example has one Enumeration **Enum** with two values: **value1** and **value2**. It also has Record **Rec** with two fields: **int_f** of type IntType and **enum_f** of type Enum**.**

In order to specify the types, you need to:

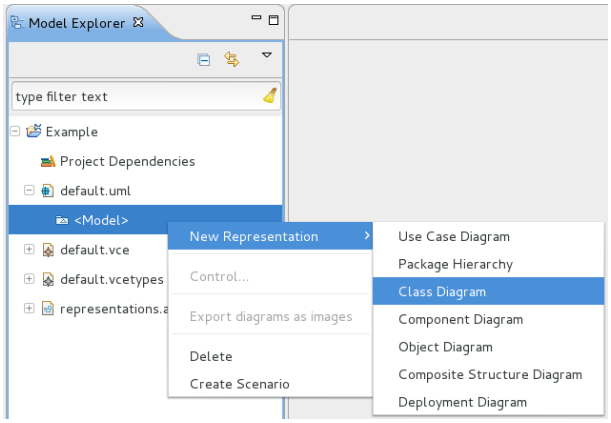
- open VceTypes diagram;
- create an Enumeration using “Enumeration” tool and give it a name using the properties view;
- add two values in the Enumeration using “Enum Element” tool and give them the names in the properties view;
- create a Record using “Record” tool and give it a name “using the properties view;
- add two fields in the Record using the tool “Field” and specify their types and names in the properties view;
- save the diagram.

2.4. UML Classes, Interfaces and Operations specification

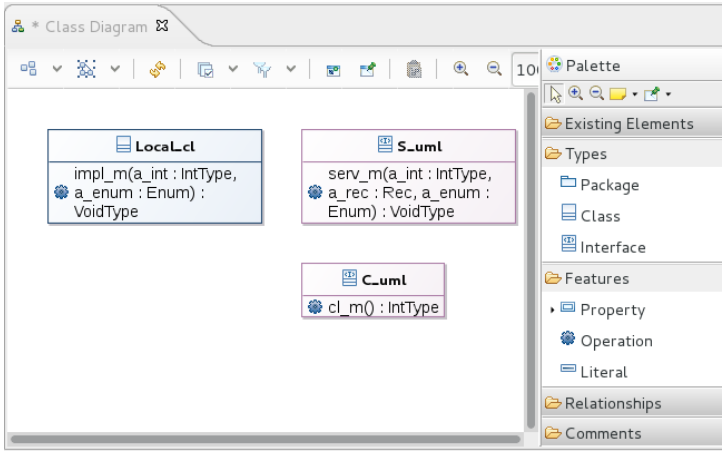
The UML Classes, Interfaces and Operations are specified on a UML Class diagram. In order to create it, right-click on



“**<Model>**” in the Model explorer and select “**New Representation -> Class diagram**”. The new diagram will be created and opened automatically.



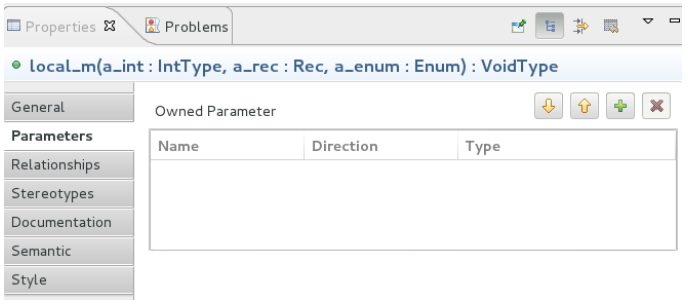
The Class diagram of our example is given below.



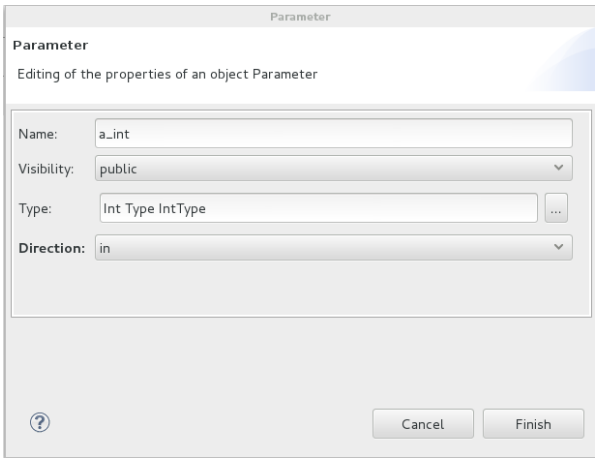
It has two UML Interfaces: **S_uml** and **C_uml**, and one UML Class **Local_cl**. In order to create them, **"Interface"** and **"Class"** tools are used.

The tool **"Operation"** is used in order to add an **Operation** to a Class or an Interface (pick-up the tool and click on the Class\Interface where you want to add the Operation). You can modify the Operation in its Properties view. The name can be set in **General** tab. The specification of the arguments and return type is a little bit more tricky. In order to modify them, you need to:

- go to the Parameters tab of the Operation Properties view;



- click on the "**green plus**" button in the top-right corner of the Properties view; this will create a parameter;
- you should see a window for the specification of the name, type and direction of the parameter. You can specify here either an argument of an operation, or its return type. Enter its **name** if you are specifying an argument; select the **type** among the ones declared before on VceTypes diagram; select the **direction** "**in**" for the arguments or "**return**" for the return type. The example of **a_int** argument of **local_m** operation is given below.

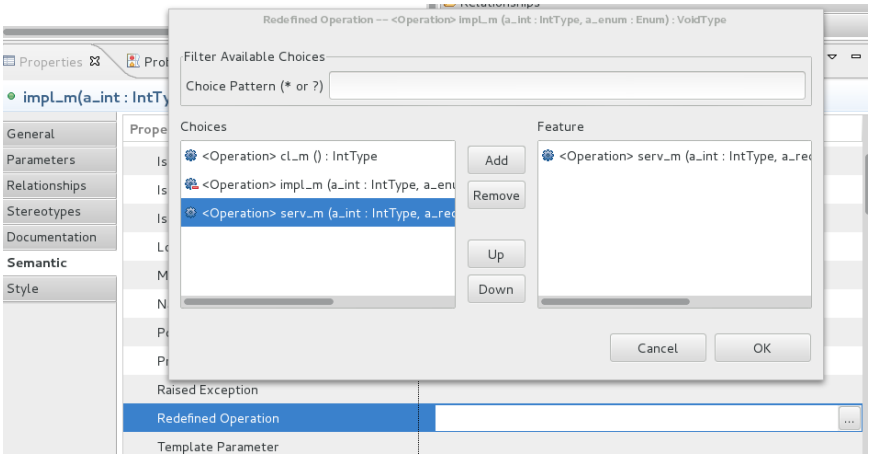


- finally, save the diagram;

2.5. Connect a Class Operation to an Interface Operation

Some methods of the UML Classes implement the ones declared on the UML Interfaces. In our example, **impl_m** of **Local_cl** implements **serv_m** of **S_uml**. order to establish this relationship you need to:

- go to the Properties view of **impl_m** operation;
- go to the **Semantic** tab;
- for the **Redefined Operation** parameter select **serv_m;**
- save the diagram;



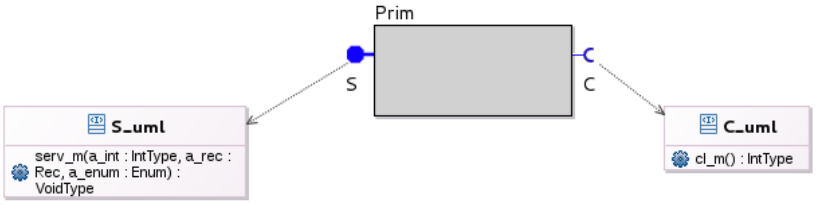
2.6. Attach a UML Interface to a GCM Interface

In our example, a UML Interface **S_uml** is attached to a GCM Interface **S** and **C_uml** is attached to **C.** In order to attach a UML Interface to a GCM Interface you need to:

- open the VCE Components Diagram (Vce diagram in our example);
- click on the GCM interface to which you want to attach a UML Interface (Interface S in our example);

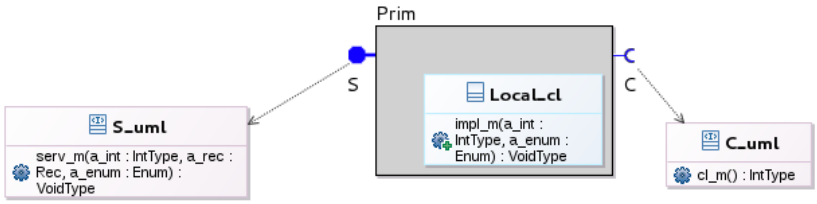
- go to its Properties view and set its ReferencedInterface to the proper one (S_uml in our example)

If you specify everything correctly, you should get the following diagram for our example:



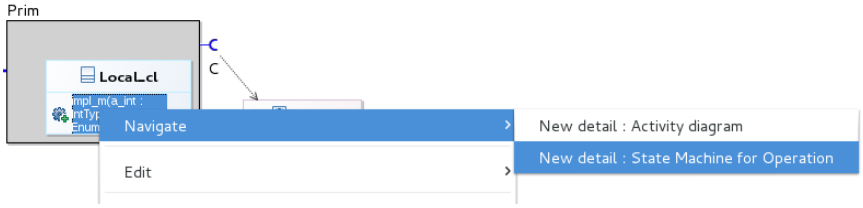
2.7. Attach a UML Class to a GCM Component

The UML Classes illustrate the lists of the operations that can be processed by the GCM Primitive Components. In our example, **Local_cl** contains the methods of a component **Prim**. In order to specify this relation, go to the Properties view of Prim on a Vce diagram and change its ReferencedClass to Local_cl. The result is illustrated below:

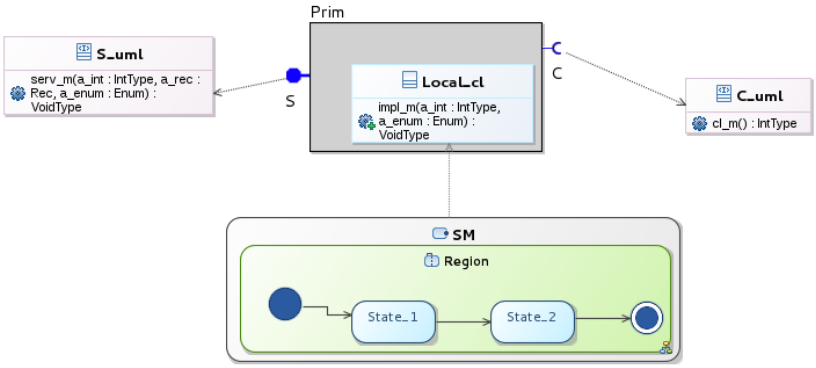


2.8. Create and attach a State Machine to a UML Operation

In order to create and attach a State Machine that specifies the behavior of an operation, right-click on a UML operation of a UML Class and select "**Navigate -> New detail: State Machine diagram for operatio**n". You can do it either on a UML class representation at a VCE Components Diagram, or at a UML Class Diagram.

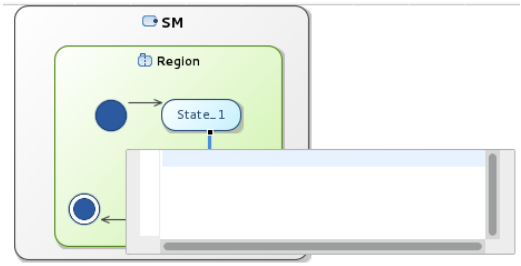


The new State Machine will be created and its diagrma will be opened automatically. You can edit the diagram using the tools on the Palette panel. You should save the diagram. After this, you will be able to see the State Machine on the VCE Components Diagram (Vce diagram in our example).



2.9. Edit the State Machine transitions labels

Xtext embedded editor should be used for the construction of the State Machine labels. In order to open the editor, go to the State Machine diagram, right-click on the transition and select "**OpenEmbeddedXtextEditor** ". You can use auto-complete (Ctrl+Space) in order to construct the labels.



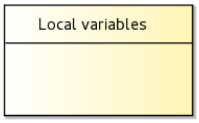
For the construction of the labels the following grammar should be used:

Non terminals are **Capitalized**, terminals are **UPPER-CASE**
Transition_label = "[" Guard_Expr "]" "/" Stats "." | "[" Guard_Expr "]" "." | "/" Stats "."
Stats = Stat | Stat ";" Stats
Stat = MCall | Assign | Return
Assign = Variable := Expr | Variable := MCall
Expr = Variable | Constant | Expr Bop Expr | "("Expr")"
Mcall = "call" Interface "->" Operation "( )" |
"call" Interface "->" Operation "(" Args")" |
"call" Operation "( )" |
"call" Operation "(" Args")"
Interface = ID"."ID
Operation = ID"."ID
Args = Constant | Variable | Constant "," Args | Variable "," Args
Constant = NUM | Boolean_constant
Bop = && | || | + | - | * | / | == | >= | <=|!=
Boolean_constant = "true"|"false"
Guard_Expr = Expr
Return = "return" Variable | "return" Constant | "return"

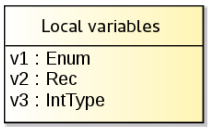
Examples of labels:
/call Prim.C -> C_uml.cl_m(); x:=11;.
[a == 15]/ x:=x+(y*5); call Local_cl.Local_m(a, true, 5);.
[z != 0].

2.10. Edit the Local variables of a State Machine

If you want to use some local variables on your State Machine labels, you should declare them in a specific section on a State Machine Diagram called “Local variables”.

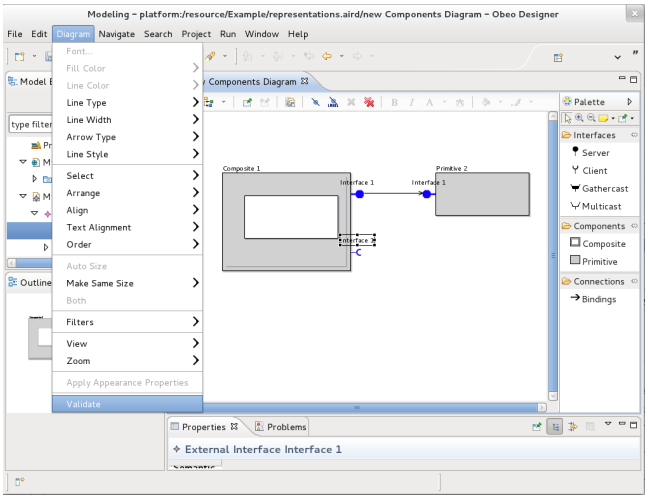


In order to add a new variable, click on the tool “**Variable**” (Palette/Variables) and then click on the box “Local variables” illustrated above. You can change the name and the type of a variable using its Properties view. For the typing you can use any type defined in the .vcetypes model. An example of three variables declaration is given below.



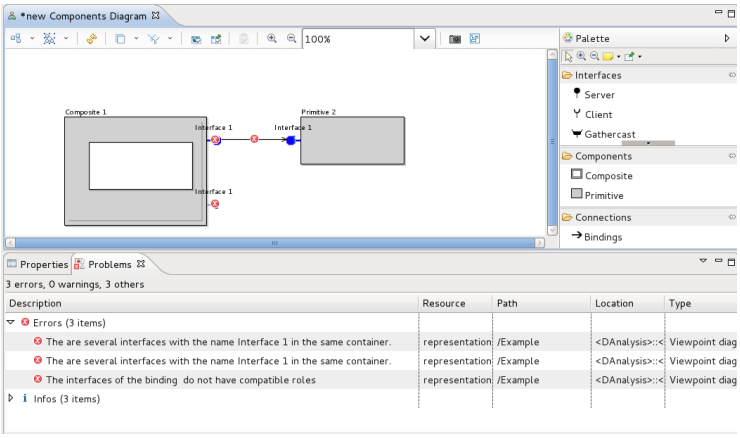
3. Diagrams Validation

Before using the diagrams, and in particular before generating the ADL files, it is mandatory to check the structural coherency of the semantics.. In order to do this open a VCE Components Diagram and select **“Diagram -> Validate”** in the menu.



The elements of the diagram which did not pass the validation should be marked with red singes. You can also see the validation results in the “**Problems**” tab.

In our example we got the following result:



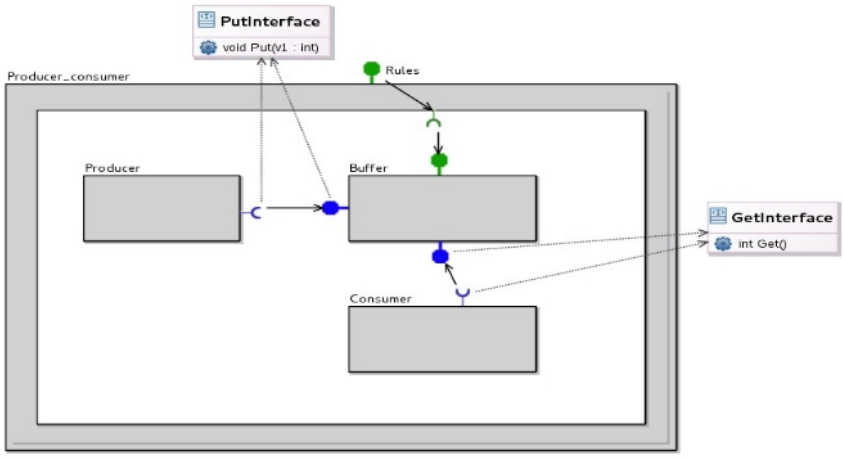
The interfaces of Composite1 did not pass the validation because they have the same names. The binding did not pass the validation because it goes from a server interface to a server interface.

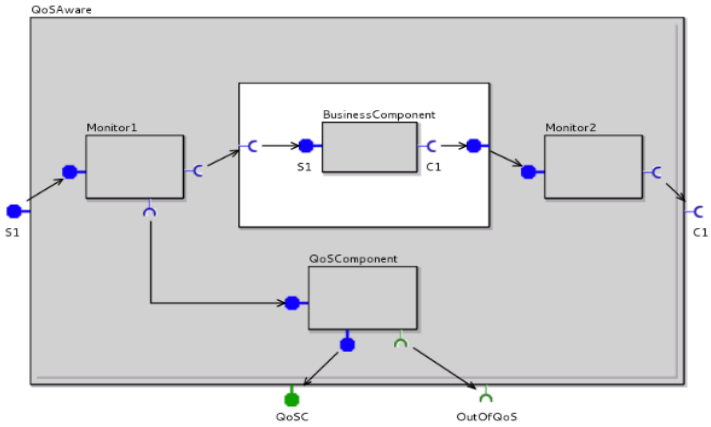
The following constraints are checked in the current version:

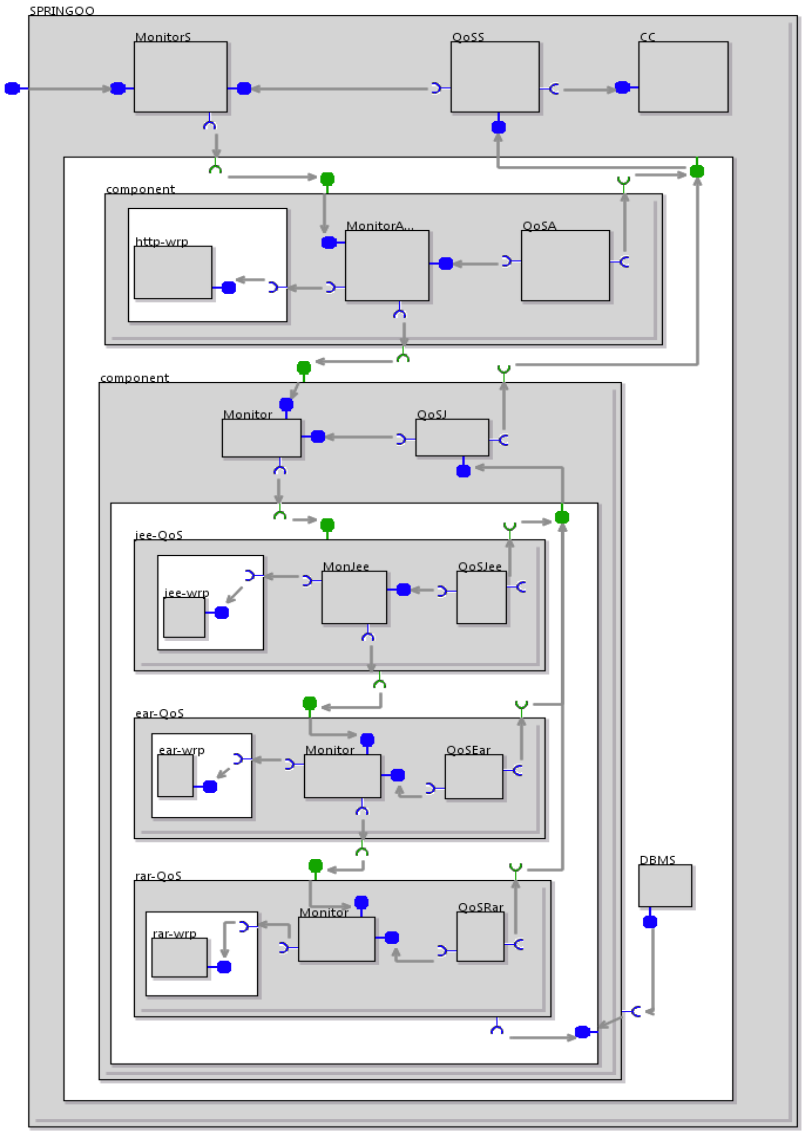
- Binding does not cross the border;
- The interfaces connected by a binding have compatible types;
- The interfaces connected by a binding have compatible roles;
- The interfaces connected by a binding have compatible natures;
- All the interfaces in the same container have different names;
- All the components in the same container have different names;

4. Examples of diagrams created with VCE v.3

The examples of simple diagrams created with VCE v.3 are given below. The projects containing the examples can be found here: www-sop.inria.fr/oasis/Vercors/software/VCE-v3/VCE-v3.Tutorial.zip




QoSAware
Monitor1
BusinessComponent
S1
C1
Monitor2
S1
C1
QoSComponent
QoSC
OutOfQoS


SPRINGQO
MonitorS
QoSS
CC
component
http-wrp
MonitorA...
QoSA
component
Monitor
QoSJ
jee-QoS
jee-wrp
MonJee
QoSJee
ear-QoS
ear-wrp
Monitor
QoSEar
rar-QoS
rar-wrp
Monitor
QoSRar
DBMS