Vercors Component Environment

Mikolaj Baranowski email: mikolaj.baranowski@gmail.com

October 23, 2008

Contents

1	Intr	oduction	1
2	Inst	allation	2
	2.1	Dependencies	2
		2.1.1 Environment.	2
		2.1.2 Plugins	2
		2.1.3 Vercors Component Environment	2
3	Cre	ating diagrams	2
	3.1	Creation of diagram.	3
	3.2	Creation of model.	6
	3.3	Creation of diagram on existing model	7
	3.4	Creation of more than one diagram for the same model	8
	3.5	Changing signatures and content classes	8
4	AD	L.	10
	4.1	Importing	10
		4.1.1 Importing with coordinates	12
	4.2	Exporting	12
		4.2.1 Exporting with coordinates	14
	4.3	Translation of internal interfaces	15
	4.4	Implementation	16
		4.4.1 Importing from ADL	16
		4.4.2 Exporting to ADL	16
		4.4.3 Coordinates	17
	4.5	Problems	18
5	TO	DO	18

1 Introduction

This document explains usage of Vercors Component Environment and also implementation of its import/export feature.

2 Installation

2.1 Dependencies

2.1.1 Environment.

To run VCE on your machine, you need JAVA 1.5 or higher and Eclipse 3.3 (minimum).

2.1.2 Plugins.

Uncompress fallowing archives to your eclipse directory.

- $\bullet~{\rm emf}\mbox{-sdo-xsd-SDK-2.3.1.zip}$
- mdt-ocl-SDK-1.1.1.zip
- $\bullet\,$ emf-query-SDK-1.1.zip
- $\bullet\,$ emf-transaction-SDK-1.1.1.zip
- emf-validation-SDK-1.1.1.zip
- GEF-ALL-3.3.1.zip
- GMF-sdk-2.0.1.zip
- mdt-uml2-SDK-2.1.1.zip
- org.topcased.sdk-R-1.2.0-200712131010.zip

2.1.3 Vercors Component Environment.

Download fallowing jars to your plugin directory.

- fr.inria.oasis.vercors.vce_2.0.1.0631.jar
- fr.inria.oasis.vercors.vce.adl_2.0.1.0631.jar
- fr.inria.oasis.vercors.vce.diagrams_2.0.1.0631.jar
- fr.inria.oasis.vercors.vce.model_2.0.1.0631.jar
- fr.inria.oasis.vercors.vce.model.edit_2.0.1.0631.jar
- fr.inria.oasis.vercors.vce.model.editor_2.0.1.0631.jar

3 Creating diagrams

Fallowing figure sequences present typical model/diagram creations.

Creation of diagram. 3.1



Figure 1: To create new, empty diagram, select $New \rightarrow Other$

	0320	
Select a wizard	1	
<u>W</u> izards:		
type filter text		
👂 🗁 Plug-in [Development	
🖻 🗁 SVN		
D 🗁 Topcase	2d	
👂 🗁 User As	sistance	
🗢 🗁 Vercors	Component Environment	
Sec. Com	ponents Diagram	
😽 Com	ponents Model	
Differ 🔁 🕑		
	25	-

Figure 2: Vercors Component Environment \rightarrow Component Diagram.

Figure 3: Then you can specify directory, model name and template.

	Create new components diagrams	
omponents D	iagrams	
You must sel	lect a Template.	
Oreate a ne	w Model	
Directory :	/MyProject2/model	
Model name :	DefaultName	
Template :		\$
Select :		
Select : Root Diagram	;	\$
Select : Root Diagram ☑ Initialize th	:	\$
Select : Root Diagram	: ee diagram with existing model objects	•

Create a new Model Orectory : /////Project2/model Model name : DefaultName Template : VCE Components Diagram Create from an existing Model Model : Select : Root Diagram : Initialize the diagram with existing model objects	I Iguit	Treate new Components diagrams
Components Diagrams Define the model diagram informations. Image: Create a new Model Directory : [MyProject2/model] Model name : DefaultName Template : [VCE Components Diagram] Create from an existing Model Model : Select : Root Diagram : Initialize the diagram with existing model objects		create new components magnants
Oefine the model diagram informations. O Create a new Model Directory : Model name : DefaultName Template : VCE Components Diagram O Create from an existing Model Model : Select : Select : Root Diagram : C Initialize the diagram with existing model objects	omponents Di	agrams
Create a new Model Directory : ///YProject2/model Model name : DefaultName Template : //CE Components Diagram Create from an existing Model Model : Select : Root Diagram :	Define the mode	I diagram informations.
Directory : ////Project2/model	Oreate a new	v Model
Model name : DefaultName Template : VCE Components Diagram C create from an existing Model Model :	Directory :	/MyProject2/model
Template : VCE Components Diagram (*) Create from an existing Model (*) (*) Model : (*) (*) Select : (*) (*) Root Diagram : (*) (*) Initialize the diagram with existing model objects (*)	Model name :	DefaultName
Create from an existing Model Model : Select : Root Diagram : Initialize the diagram with existing model objects	Template :	VCE Components Diagram
Root Diagram :	Select :	
☑ Initialize the diagram with existing model objects	Root Diagram	
	🗹 Initialize th	e diagram with existing model objects
Rack Nexts Finish Cancel	0	Park Next Sinish Carrel

ComponentsDiagram :	null	/ MyModel

Figure 5: New diagram.

C Resource Set				
🔗 platform:/resource/MyProject2/model/MyModel.components				
✓ ♦ Architecture				
♦ Membrane				
♦ Content				

Figure 6: New model.

3.2 Creation of model.

VCE gives you a possibility to create empty model without associated diagram.

Figure 7: Select: New \rightarrow Other \rightarrow Vercors Component Environment \rightarrow Component Model.

<i>.</i>	New	2
Select a wizar Create a new	d Components model	
<u>W</u> izards:		
type filter tex	6	
👂 🗁 Plug-in	Development	
👂 🗁 SVN		
D 🗁 Topcas	ed	
👂 🗁 User A	ssistance	
	Component Environment	
💙 Com	nponents Diagram	
\delta Com	aponents Model	
👂 🗁 Other		
	les	

Figure 8	8: Then,	specify	model	name.
)		New		
Components Model Create a new Compo	onents model			
Enter or select the pa	arent folder:			
MyProject2/model				
10 Q D				
MyProject1				
✓ ⊌ MyProject2				
Þ 🗁 bin				
Eile name: Mu com	onestr			
Advanced >>)
0	< <u>B</u> ack	<u>N</u> ext >	Einish	Cancel

Figure 9: Select model root object (by default it is **Architecture**) and click finish to create a <u>new model</u>.

		New		×
Components Model Select a model obje	ct to create			
<u>M</u> odel Object				
Architecture				-
XML Encoding				
UTF-8				-
0	< <u>B</u> ack	Next >	Einish	Cancel

3.3 Creation of diagram on existing model.

To create diagram file based on existing model, select: New \rightarrow Other \rightarrow Vercors Component Environment \rightarrow Component Diagram.

	Create new Components diagrams	
Components D	iagrams s file already exists. It will be overwritten !	
O Create a ne	w Model	
Directory :	(/MyProject2/model	
Model name :	DefaultName	
Template :		
	Protection and the state of the	
Select :		
Select : Root Diagram	: ComponentsDiagram	
Select : Root Diagram	: ComponentsDiagram e diagram with existing model objects	
Select : Root Diagram	: ComponentsDiagram ee diagram with existing model objects	
Select : Root Diagram	: ComponentsDiagram re diagram with existing model objects	

Figure 10: Then you need to select *Create from existing Model*, path to Model and *Root Diagram*.

3.4 Creation of more than one diagram for the same model.

Unfortunately, for now, you can not specify diagram file name in diagram creation wizard and if there is already one it will be overwritten. But, you can go around this problem by changing file name before doing steps from section 3.3.

3.5 Changing signatures and content classes.



Figure 11: You want to create your own interface ...



Figure 12: ... and set primitive content class, in context menu select Change Model Properties \rightarrow Change Primitive Content Class

Figure 13: Or, to set interface signature, select Change Model Properties \rightarrow Change Interface Signature.



Select entries:	•
Fo	
Matching items:	
O FocusListener	
9 FontAttribute	
FontPeer	
FontSupport	
🛈 Foo - fr.inria.oasis	
FormatString	
Formattable	
🖶 fr.inria.oasis - MyProject2/src	
0	OK Cancel

Figure 14: You can make a choice between all classes in class path.

4 ADL.

The Fractal Architecture Description Language[2] is a XML-based language used to define component architectures. VCE allow us to translate models between ADL and VCE internal representation.

4.1 Importing

	Ne <u>w</u>
	Go <u>I</u> nto
	Open in <u>N</u> ew Window
Shift+Alt+W	Sho <u>w</u> In
Ctrl+C	<u>С</u> ору
	Copy Qualified Name
Ctrl+V	🖹 <u>P</u> aste
Delete	📕 <u>D</u> elete
3	<u>B</u> uild Path
Shift+Alt+T	Refac <u>t</u> or
	2 m
	🖾 Exp <u>o</u> rt
F5	🕰 Export 🖗 Re <u>f</u> resh
F5	⊴ Exp <u>o</u> rt
F5	 A Export Refresh Assign Working Sets Run As
FS	Setyport Refresh Assign Working Sets Run As Debug As
F5	Export Refresh <u>A</u> ssign Working Sets <u>R</u> un As <u>D</u> ebug As Tgam
FS	Set Export Refresh Assign Working Sets Run As Debug As Tgam Compare With
FS	Sexport Refresh Assign Working Sets Run As Debug As Tgam Compare With Restore from Local History
FS	Sexport Refresh Assign Working Sets Run As Debug As Team Compare With Restore from Local History Topcased

Figure 15: From context menu select **Import**

type filter text	Source.	
👂 🗁 General		
CVS		
👂 🗁 Model		
👂 🗁 Plug-in De	velopment	
Þ ⊜svn		
👂 🗁 Team		
▽ 🗁 VCE Impo	rt Wizards	
ADL Im	port Wizard	
👂 🗁 Other		

Figure 16: From VCE Import Wizards group select ADL Import Wizard

Figure 17: Select ADL files to import and specify target directory. For now, you can not set model file name explicate. It is composed from ADL file name (sequence *.fractal* is changed for *.components*).

ıpty.	
	Browse
elect All	
:2/model	Browse
older structure	
	pty.

	VCE ADL Importer
From directory: //auto/sea/u/sea/0,	/user/mbaranow/runtime-EclipseApplication/I 👻 Browse
🗏 🖕 examples	🗆 🐼 01. components
	🗆 📄 01.componentsdi
	🖬 🗎 01.fractal
	🗆 🖗 02.components
	🗆 🗎 02.componentsdi
	🗆 🗎 02.fractal
	🗆 🖗 03.components
Filter <u>Types</u> <u>Select All</u> nto fo <u>l</u> der: MyProject2/model	Deselect All Browse
Dptions Qverwrite existing resources w <u>C</u> reate complete folder structu Create s <u>e</u> lected folders only	vithout warning re
Ø	< Back Next > Finish Cancel

Figure 18: Click finish to create imported models and diagrams.

▽ 🗁 m c	odel
	01.components
Đ	

Figure 19: New files.

4.1.1 Importing with coordinates

By default, diagram elements are initialized with coordinates from ADL file.

4.2Exporting

MyModel componentsdi	Ne <u>w</u>	•
in the second	<u>O</u> pen	F3
	Open Wit <u>h</u>	,
	Sho <u>w</u> In	Shift+Alt+W
	<u>С</u> ору	Ctrl+C
	Copy Qualified Name	
	<u> P</u> aste	Ctrl+V
1	X <u>D</u> elete	Delete
	<u>B</u> uild Path	,
	Refac <u>t</u> or	Shift+Alt+T
ŝ	🔄 Import	
	🖾 Export	
ŝ	🔗 Re <u>f</u> resh	F5
	Assign Working Sets	

Figure 20: From context menu select **Export**

	Export	
elect		Z
<u>S</u> elect an export de	itination:	
Ceneral]
Java		
Plug-in Devel	opment	
Þ ⊜Team		
VCE Export	Vizards	
ADL Expo	rt Wizard	
v 🖾 Otner		
0	< Back	Einish Cancel

Figure 21: Select item ADL Export Wizard from ADL Exports Wizards

Figure 22:	Select models w	which you	want to	o export	to AD	ıL.

VCE/	ADL Exporter	
Select model to export Choose the Component Models to Export		1
<pre>> Idoption Section Sectio</pre>	MyModel.components MyModel.components MyModel.componentsdi wexample.components wexample.componentsdi	
Select Iypes) Select All Deselect A		
0	Back	Cancel

Figure 23:	Specify.	ADL fil	e name.	
5	VCE ADL Exp	onter		×
select output file Choose the destination main file				
Enter or select the parent folder:				
MyProject2/model				
MyProject1				
✓ ➡ MyProject2				
Þ 🗁 bin				
-ile na <u>m</u> e: [my-model.fractal]				
Options				
Export Components in different files				
Advanced >>				
0	< <u>B</u> ack	Next >	<u>Einish</u>	Cancel

Figure 24: To export each component in a different file - select proper option.

	a second of and and anticipated dated	
Select output file		
Choose the destination main file		
Enter or select the parent folder:		
MyProject2/model		
MyProject1		
✓ ⊌MyProject2		
Þ 🗁 bin		
╘ model		
File name: my-model.fractal		
Export Components in different files)	
Adversed	/	
Auranout		
0	< Back Next >	Einish Cancel

4.2.1 Exporting with coordinates

VCE provides functionality which lets you to export model with diagram coordinates. It means that you can keep sizes and positions of diagram figures in ADL file and restore them in different tool or with import feature.

Figure 25: To export diagram coordinates you need to make export from diagram – not like before – from model file.

\$	VCE ADL Exporter	×
Select model to export Choose the Component Model:	s to Export	
 ▷ I I I I I I I I I I I I I I I I I I I	MyModel.components MyModel.components Avantation of the second of the sec	
Select Types) Select A	NI Deselect All	Carel

4.3 Translation of internal interfaces

ADL definition doesn't keep information about internal interfaces. It means, with exporting you are loosing all information: name, signature and cardinality of internal interface.

Figure 26: Exporting collective internal interface. From left to right: before exporting, visualization of ADL representation, after importing from ADL. Because internal interface has only one binding, it is restored as singleton interface.



Figure 27: From left to right: before exporting, visualization of ADL representation, after importing from ADL. In that case, internal interface has two bindings and it is restored as collective interface then.

4.4 Implementation

4.4.1 Importing from ADL

The main import class is fr.inria.oasis.vercors.vce.adl.wizards.ADL-ImportWizard.

ADLImportWizardSelectionPage (figure 18) is used to provide basic graphical interface which lets user to select ADL files to import and specify target directory.

Translating process VCE import feature is developed using objectweb fractal loader[1]. The key to understand translating process is method performFinish in class ADLImportWizard. It uses component loader defined in fr.inria.oasis.vercors.vce.adl.VCELoader.

VCELoader extends default loader provided from objectweb org.objectweb.fractal.adl.BasicLoader which uses classloader to load every signature and content class used in diagram. The alternative to changing classloader is to remove it from loading process.

After all, we don't want to instantiate these classes but only get their names. VCELoader limits this functionality by using classes XMLLoader, VCETypeLoader, VCETypeBindingLoader, VCEImplementationLoader from fr.inria.oasis.vercors.vce.adl package.

Each ADL module (such as component, interface, coordinate) has proper analyze* function. For example: method analyzeComponent which has two arguments: component (instance of org.objectweb.fractal.adl.components.-Component and componentDefinition (instance of fr.inria.oasis.vercors.vce.model.components.ComponentDefinition) gets information from component (which contains data from ADL file) and puts them to componentDefinition (which represents component in VCE model).

Importing internal interfaces This process starts in analyzeBinding method in ADLImportWizard class. When function finds binding between interfaces of the same type, method makeServerClient or makeClientServer is invoked (they are named after palette item in diagram editor which creates these figures).

In the easiest case, mediator interface is created and two bindings, one to each external interface.

Situation is more complicated for collective internal interfaces. This case is recognized in condition: if (clientServerMap.containsKey(sourceInterface)) or its equivalent in other method. It means that there was already one connection from sourceInterface. Mediator interface which already exists is changed for new collective interface.

4.4.2 Exporting to ADL

VCE export feature is implemented using JAXB. In performFinish method, after validation, model is passed to ADLModelTranslator. There, every part of model has a proper method.

Translating process TODO

Exporting internal interfaces Actually, I should name this process "binding translation" – I need to keep connections between external interfaces (look at figures 26 and 27) without using internal interfaces.

Binding translating process is based in caseInterface and caseBinding methods in ADLModelTranslator class. In first one, all processed interfaces are stored in interfaceSet variable.

In fact, this method analyzes only external interfaces and this is my intention. caseBinding uses this variable to distinguish between internal and external interfaces.

In second method, every interfaces of binding from server interface to client interface are stored in interfaceInterfaceTranslator map. Internal interface becomes a key and external interface - value. This information lets me to make a binding between external interfaces in the next step.

4.4.3 Coordinates

Coordinates are represented by 6 attributes:

- $\bullet\,$ x0 which keeps horizontal coordinate of left top corner of figure
- x1 which keeps horizontal coordinate of right bottom corner of figure
- y0 which keeps vertical coordinate of left top corner
- y1 which keeps vertical coordinate of right bottom corner
- name which keeps name of figure (coordinates are distinguished by names)
- color not obligatory attribute

Coordinates are normalized. For example, if x0 and y0 equal 0, left top corner of diagram element is based in left top corner of available space and if x1 and y1 equal 1, right bottom corner of diagram element is based in right bottom corner of available space.

Importing coordinates Coordinates importing process is placed in ADLImportWizard class. Coordinates from ADL file are stored in coordinatesContainer field and analyzing process begins in saveModel method where diagram file is initialized. After initialization, initializeContent method is invoked. Arguments are:

- 1. Elist<org.topcased.modeler.di.model.DiagramElement> elements this variable keeps diagram elements from same layer
- 2. EList<fr.inria.oasis.vercors.vce.model.components.Component> components this variable keeps components from same layer
- 3. Coordinates[] coordinatesContainer this variable keeps coordinates from same layer
- 4. double xSize
- 5. double ySize

Variables elements, components, coordinatesContainer represent trees. They are parsing at the same time. What is most important is that each layers of each tree has a correspondent layer in other trees. I mean - one layer of DiagramElement tree corresponds only to one layer of Component tree and one layer of Coordinates tree and vice versa.

Variables **xSize** and **ySize** keep height and width of parent diagram element (basically, it's size of parent component content). VCE keeps sizes explicate in pixels but, as I mentioned before, sizes in ADL are normalized. Then, position of every element is calculated by multiplying coordinates from ADL by one of these variables.

Condition if (! matched) is explained in a 4.5.

Exporting coordinates Coordinates exporting process takes a place when user is exporting model from diagram file. 'If'' statement with this condition is placed in performFinish method in ADLExportWizard class. If exporting element is identified as DiagramsImpl, analyzeDiagramCoorinates method is invoked.

analyzeDiagramCoorinates analyzes diagram element (there should be only one) and invokes analyzeGraphNodeCoordinates with diagram element as argument.

In analyzeGraphNodeCoordinates, if GraphNode represents Component-Definition, coordinates from VCE diagram are translated to ADL format and stored in variable parentCoordinates. Then, analyzeGraphNodeCoordinates is invoked for each content of diagram element.

4.5 Problems

There is a problem when component extends other component. Loader looks for parent component in classpath and, if classloader can not find it, it rises exception.

if(! matched)

5 TODO

- Importing setting size of diagram.
- Exporting coorinates relative to non-relative
- Importing internal interfaces remove this mapx

References

- Fractal ADL Documentation. http://fractal.objectweb.org/current/doc/javadoc/fractaladl.
- [2] Fractal ADL Tutorial. http://fractal.objectweb.org/tutorials/adl/index.html.