VerCors v4 Tutorial for the Developers

INRIA Sophia-Antipolis

September 2016

oleksandra.kulankhina@gmail.com
eric.madelaine@inria.fr

VCE version 4.1.1

Contents

1 Brief overview

This document helps the developers of the VerCors platform in understanding the project architecture and provides simple steps to configure their local workspace.

Here we provide a very brief overview of the used technologies. References to all the necessary resources are listed in Section 11. The VerCors platform represents a set of plug-ins for the Eclipse IDE. It includes graphical designers for the architecture and behaviour of GCM components. All the designers are based on the Sirius (Obeo designer) technology which is, in its turn, based on EMF and GMF. VerCors includes the source code of the Obeo UML designer for the management of UML elements. In addition, VerCors includes plug-ins generating Java code of the modelled application and plug-ins generating an input for the CADP model-checker. The source code of VerCors is stored in an SVN repository, the examples are stored on GitHub. VerCors is a Maven project with several modules where each module is one of the plug-ins. The project is continuously built on a Jenkins server.

2 Configuring the workspace

1. Download and install Obeo Designer community version: `http://www.obeodesigner.com/download`. Obeo designer is based on Eclipse IDE, hence, it includes all standard functionalities of Eclipse. If you are not familiar with Eclipse IDE, the following url will help you to get the basic information: `https://www.eclipse.org/users/`

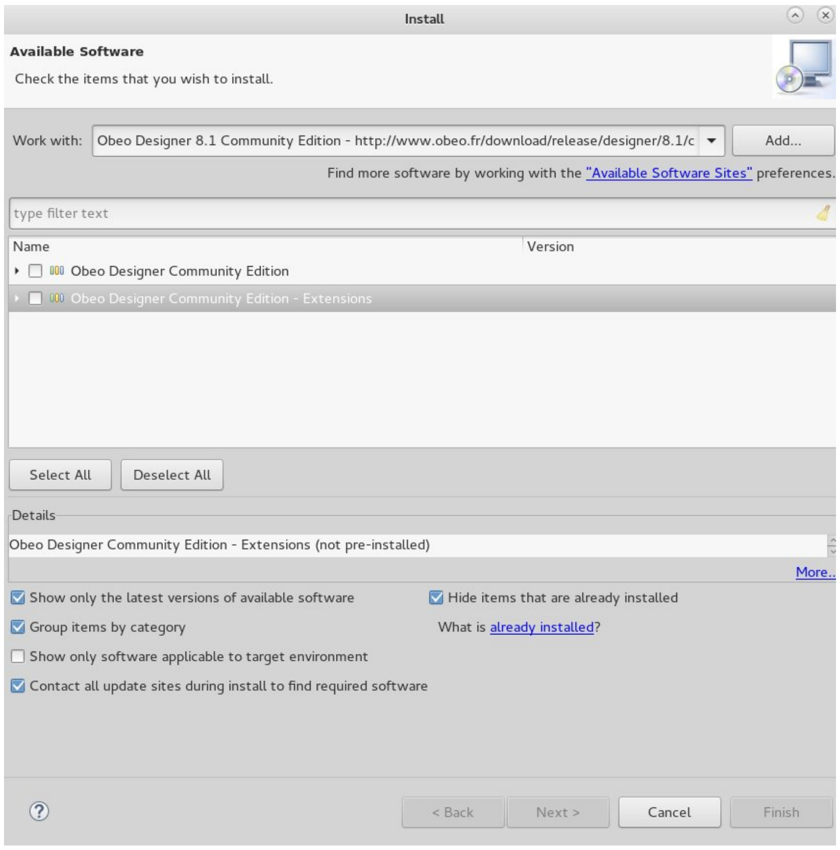
2. Use Eclipse Marketplace in order to install the following plug-ins on top of Obeo Designer:

   (a) Subversive SVN team provider: `http://marketplace.eclipse.org/content/subversive-svn-team-provider`

   (b) Maven integration for Eclipse: `http://marketplace.eclipse.org/content/maven-integration-eclipse-juno-and-newer`

3. Install the Eclipse plug-ins for the UML meta-model. In the toolbar on the top of Eclipse select `Help -> Install new Software`. This will open a wizard. In the `Work with` textfield select `Obeo Designer Community Edition` as illustrated in Figure 1. A list of software to be installed should appear. Select `Obeo Designer Community Edition - Extentions -> UML 2 Extender SDK` and finalise the procedure.

Important! If you already have an Obeo Designer with VerCors installed, please, use another installation of Obeo Designer for working with the source code.



Obeo Designer 8.1 Community Edition - http://www.obeo.fr/download/release/designer/8.1/c

Find more software by working with the "Available Software Sites" preferences.

Name	Version
☐ Obeo Designer Community Edition	
☐ Obeo Designer Community Edition - Extensions	

Obeo Designer Community Edition - Extensions (not pre-installed)

- [✓] Show only the latest versions of available software
- [✓] Hide items that are already installed
- [✓] Group items by category
- What is already installed?
- [ ] Show only software applicable to target environment
- [✓] Contact all update sites during install to find required software

Figure 1: Installation of UML meta-model plug-ins

3 Working with SVN

3.1 Checkout

All the source files are stored at INRIA GForge. At `https://gforge.inria.fr/scm/?group_id=193` the developer can find all the information required to successfully check-out the project.

Access. If you checkout the project anonymously, you will not be able to commit modifications. Moreover, in order to be able to commit in the repository, the developer should have such a right provided by one of the administrators of the repository. Hence, if you are a developer, use either a **developer access** method or access the repository via ssh.

Eclipse Subversive. In order to keep things coherent, we suggest using Eclipse Subversive for all operations with SVN. The following web-page provides the references to the documentation: `https://eclipse.org/subversive/documentation.php`. If you have issues with checking-out the project, make sure you have an SVn connector installed or trying changing it to another one. More about SVN connectors: `https://polarion.plm.automation.siemens.com/products/svn/subversive`.

The current version of VerCors includes several branches and the most important ones are:

- `/branches/Vercors/VCEv3Sirius/vceSirius1.0` - the main branch used for the build
- `/branches/Vercors/VCEv4-OpenPNets` - the branch for working with open pNets

3.2 Commit

Before committing, please, choose carefully the commit message and increase the version of VerCors as explained in 4. If the committed changes do not affect the business logic of the application (e.g. the changes to the documentation), then the version should not be incremented.

Important! Please, do NOT commit the `bin` folders and all the files related to the local build.

4 Versioning

VerCors fallows `x.y.z` versioning standard, where:

x is always number '4'. It can be incremented if the underlying technology is completely changed.

y is incremented only for major change

z is incremented for bug fix

5 Running VerCors from the source code

During the development, the programmer often needs to run his application from the source code. In order to run VerCors from the source code, you should follow a standard procedure for the development Eclipse/Sirius-based as it is described in the Sirius tutorial and here: `http://www.vogella.com/tutorials/EclipsePlugin/article.html#starting-the-eclipse-ide-from-eclipse`.

Important! Only the plug-ins imported in the root of the workspace can be run from the source code. Hence, if you have imported VerCors in your workspace together with the `VCEv3` folder, you should additionally import the projects included in the folder into your workspace. For this, use Eclipse Existing Project Import wizard and do NOT copy the plug-ins. In this case you will see each plug-in in your workspace twice: once in the root and once in the `VCEv3` folder. It is not an issue because if you did not copy the plug-ins, there is in fact only one instance of each plug-in. During the development, you should work on the plug-ins displayed in the root.

6 Maven build

VerCors is implemented as a Maven project with several modules. We refer to `https://maven.apache.org/` for more information about maven. The parent `pom.xml` of VerCors is located in the `VCEv3` folder.

In order to build the project from Eclipse, right click on the root pom.xml and select `Maven -> maven install`.

7 Architecture of the VerCors platform

VerCors is implemented as a collection of Eclipse plug-ins. Their names are prefixed by "fr.inria.oasis.vercors.". The plug-ins together with their functionalities are listed below:

- Plug-ins for working with component architecture
  - **vce** contains an EMF meta-model for GCM component architecture;
  - **vce.edit** is an EMF .edit plug-in for GCM component architecture, it was generated automatically from the meta-model;
  - **vce.editor** is an EMF-based editor for GCM component architecture, it was generated automatically from the meta-model;

  - **vce.design** is a Sirius-based graphical designer for GCM component architecture;

- Plug-ins for ADL and Java generation:
  - **vce.adl** translates component architecture into an ADL file; it is based on the JAXB technology
  - **vce.adl.module** translates component architecture and behaviour into Java code; it is based on the Acceleo technology

- Plug-ins for the project build and integration:
  - **vce.feature** assembles all VerCors plug-ins into one feature;
  - **vce.p2** is a p2 repository module, it generates p2 repository for VerCors;
  - **vce.wizard** is a wizard for the creation of VCE pro jects;

- Plug-ins for working with pNets:
  - **vce.pnets** generates pNets, Fiacre v2, EXP, and SVL; please, note that the plug-in generates Fiacre v2 but not v3
  - **vce.pnets.model** contains an EMF meta-model of pNets;

- Plug-ins for working with UML elements:
  - **vce.smextension** contains an EMF meta-model for the declaration of local variables of state machines;
  - **vce.smextension.edit** is an automatically generated EMF .edit project for the declaration of local variables of UML state machines;
  - **vce.smextension.editor** is an automatically generated EMF editor for the declaration of local variables of UML state machines;
  - **vce.smtransitions.parsed** is a parser of the labels of the transitions of UML state machines; it is based on the JFlex/Java-Cup technologies;
  - **vce.uml** provides services for processing UML models in the context of GCM-based applications;
  - **org.obeonetwork.dsl.uml2.design** is a clone of the UML Designer plug-in. It contains the Sirius-based editors for the UML diagrams;
  - **org.obeonetwork.dsl.uml2.properties** is a clone of the UML Designer plug-in for working with the properties of UML elements;
  - **org.obeonetwork.gmf.runtime.diagram.extension** is a clone of the UML Designer plug-containing several services for working with UML diagrams;

- Plug-ins for working with VerCors types:
  - **vce.vcetypes** contains an EMF meta-model for the VerCors Type diagram;

  - **vce.vcetypes.edit** is an automatically generated EMF .edit project for the VerCors types;
  - **vce.vcetypes.editor** is an automatically generated EMF editor for the VerCors types;
  - **vce.vcetypes.design** is a Sirius-based graphical designer for the VerCors Type diagrams;

- Other plug-ins:
  - **vce.xtext.model** includes an EMF mate-model of the parsed labels of the transitions of UML state machines and the expressions used by the labels of pNets;

- Plug-ins for testing:
  - **vce.test** is an OLD project for testing static validation of component diagrams;

- Deprecated plug-ins (the following plug-ins were used in the project at some point):
  - **vce.fc2** translates pNets into fc2 format;
  - **vce.simpletypes** is an old plug-in for the VerCors types;
  - **vce.expressions2**, **vce.expressions2.edit**, and **vce.expressions2** are old plug-ins for the VerCors expressions;
  - **vce.xtext.model.edit** and **vce.xtext.model.editor** are the .edit and .editor plug-ins generated automatically by EMF for the meta-model of expressions

8 Continuous integration with Jenkins

Jenkins continuous integration server `http://jenkins-ci.org/` has been set to ease the development and to generate VCEv4 p2 repository from where the users are able to download and update the project. The Jenkins server is supported by the continuous integration platform of Inria. In order to access it, the developer should have an account at Inria Ci `https://ci.inria.fr/` and to be a member of the VerCors project `https://ci.inria.fr/project/vcev3/show`.

The Jenkins server manages three projects:

- **sirius vce** launches a Maven build for the latest version of the VerCors project;

- **p2 management** takes the latest version of the build and saves it to the p2 repository (i.e. to `fr.inria.oasis.vercors.vce.p2/target/repository`) from where the users can install it; the build of this module is often marked with a ”failure” but anyway it works;

- **VCEv3** launches a Maven build for the old version of VerCors which was based on the Obeo Designer with an academic license.

All the builds can be triggered on demand. An automatic build can be configured in the settings of Jenkins.

9 Adding a new plug-in

If you have implemented a new plug-in and you would like to integrate it in the VerCors platform, you have to do the following:

1. Make sure your that project is a Maven project. If it is not the case, right-click on your project and select `Configue -> Convert to Maven Project`;
2. Move your project inside the `VCEv3` folder;
3. Include your project as a module of the VCEv3 Maven build. For this, open the root `pom.xml` file and add the name of your project in the `modules` section;
4. Configure the Maven properties of the build of your project: set its parent `pom.xml` file to the one of VerCors and set the packaging type to `eclipse-plug-in` (see the other plug-ins inside VerCors for an example);
5. Include your project in the list of the plug-ins assembled by the `vce.features` project. For this, include your project in the `feature.xml` file of the `vce.features` project (see how this is done for the other VerCors plug-ins for an example).

10 Other information

The examples and the documentation for VerCors are located at: `https://github.com/Scale-VerCors/VCEv4`. The documentation is written in Latex.

11 Resources

- The core underlying technologies:
  - **Eclipse IDE**: `https://eclipse.org/`
  - **Eclipse plug-in development**: `http://www.vogella.com/tutorials/EclipsePlugin/article.html`
  - **Sirius and Obeo Designer**: `https://eclipse.org/sirius/`
  - **EMF and GMF**: `https://eclipse.org/modeling/emf/` and `http://www.eclipse.org/modeling/gmp/`
  - **Obeo UML Designer**: the official web-site: `http://www.umldesigner.org/`, the sources: `https://github.com/ObeoNetwork/UML-Designer`
  - **OCL**: `http://www.omg.org/spec/OCL/`
- The technologies for parsing and code generation:

  - **Acceleo**: `https://eclipse.org/acceleo/`
  - **JAXB**: `http://www.oracle.com/technetwork/articles/javase/index-140168.html`
  - **Jflex/jcup**: `http://jflex.de/manual.html` and `http://www2.cs.tum.edu/projects/cup/`

- The technologies for the maintenance:
  - **Maven**: `https://maven.apache.org/`
  - **Eclipse Maven integration**: `http://marketplace.eclipse.org/content/maven-integration-eclipse-juno-and-newer`
  - **Tycho**: `https://eclipse.org/tycho/`
  - **Jenkins**: `https://jenkins.io/index.html`
  - **SVN**: `http://svnbook.red-bean.com/en/1.7/`
  - **Eclipse subversive**: `http://marketplace.eclipse.org/content/subversive-svn-team-`
  - **git**: `https://git-scm.com/`

- Inria platforms:
  - **Gforge**: `https://gforge.inria.fr/`
  - **CI**: `https://ci.inria.fr/`

- Forums:
  - **Eclipse forum**: `https://www.eclipse.org/forums/?`
  - **Obeo designer forums**: `http://www.obeonetwork.com/group/obeo-designer`

- VerCors resources:
  - **Home page**: `https://team.inria.fr/scale/software/vercors/`
  - **Installation repository**: `http://vercors.gforge.inria.fr/repository/`
    .
  - **Source code**: `https://gforge.inria.fr/projects/vercors/`
  - **Tutorials and examples**: `https://github.com/Scale-VerCors/VCEv4`
  - **VerCors on Inria CI**: `https://ci.inria.fr/project/vcev3/show`

- Other resources:
  - **The ProActive research branch**: `https://github.com/scale-proactive/scale-proactive`
  - **CADP**: `http://cadp.inria.fr/`
  - **Fiacre v2**: `http://projects.laas.fr/fiacre/fiacreV2.html`