Overture Maven Core Components Guide

Kenneth Lausdahl

April 2009

Introduction

The Overture Tool project is sub divided into the following modules:

Core components : The core components is the components which it either specified directly in Java or VDM-SL / VDM++. Example of components is:

- **AST:** The abstract syntax of VDM used in this project. Specified in VDM-SL
- **VDMJ:** A tool with functionalities such as Type check, Interpretor, PO among more for VDM-SL and VDM++.
- **Parser:** A parser for VDM++ which can parse a vpp file into the AST.

Eclipse integration : The eclipse integration is primarily plug-ins wrapping the core components in an easy to use packing so they can be used from eclipse. Additional to this an editor with corresponding perspectives are also present i this module.

Tools : This is Maven tools used in the development process. This includes tools such as integration of the code generator from VDM Tools into the Maven build cycle.

1 Check out

To check out the overture core components the use a SVN client:

```
svn co https://overture.svn.sourceforge.net/svnroot/overture/trunk/core overture
```

Now the core components should be checked out to a folder called overture.

2 Build

The core components can be build using Maven. For more information about Maven and how to install Maven go to the Overture Wiki: `http://www.overturetool.org/twiki/bin/view/Main/DevFaQ` Step 2 describe how to install Maven. When Maven is installed go to the newly check out source in the overture folder and execute the following command:

```
mvn package
```

Now Maven should start building all the packages. After the build finishes a build status like the on in listing **??** should be shown. If it displays "BUILD SUCCESSFUL" then all components have been build successfully.

```
[INFO]
------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
------------------------------------------------------------------------
[INFO] Top-level POM for OvertureTool .................. SUCCESS [4.700s]
[INFO] The VDM++ Standard Library ..................... SUCCESS [3.928s]
[INFO] The Overture Abstract Syntax ................... SUCCESS [6.282s]
[INFO] The VDMunit Support Library .................... SUCCESS [2.012s]
[INFO] The VDM++ parser ............................... SUCCESS [5.792s]
[INFO] The Overture Eclipse plugins ................... SUCCESS [0.003s]
[INFO] Bi-directional OML to UML translator ........... SUCCESS [3.532s]
[INFO] Interface for VDM Tools ........................ SUCCESS [0.003s]
[INFO] API for VDM Tools - wrapper for corba .......... SUCCESS [1.391s]
[INFO] The VDMJ Interpreter ........................... SUCCESS [13.692s]
[INFO] Combinatorial Testing of VDM++ models .......... SUCCESS [5.518s]
[INFO] showtrace ...................................... SUCCESS [0.857s]
[INFO]
------------------------------------------------------------------------
[INFO]
------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO]
------------------------------------------------------------------------
```

Listing 1: Core components build result.

2.1 Using Eclipse with Overture Tool development

The overture tool project supports Eclipse as a development tool. To prepare the overture core components to support Eclipse the build step above must be finished with success. The first step is to install Eclipse 3.4.2 with the Maven plug-in. A guide ecplaining this can be found on the overture Twiki `http://www.overturetool.org/twiki/bin/view/Main/DevFaQ` step 3 to 5. After the installation the following Maven command must be executed in a terminal in the overture folder which is the root of the checked out source.

```
mvn eclipse:eclipse
```

Now Maven have created Eclipse project files for all components and they are ready for import in Eclipse.

Open Eclipse and select a workspace of choice. The do as followed (fig.**??**):

Select Import → General→ Maven Projects

Select the directory where the overture artifacts is located as shown in fig. **??**

Now you should see: fig. **??**.

Do not select the top level POM named core as shown in fig. **??**.

Click finish

Now eclipse should show no errors like in fig. **??**

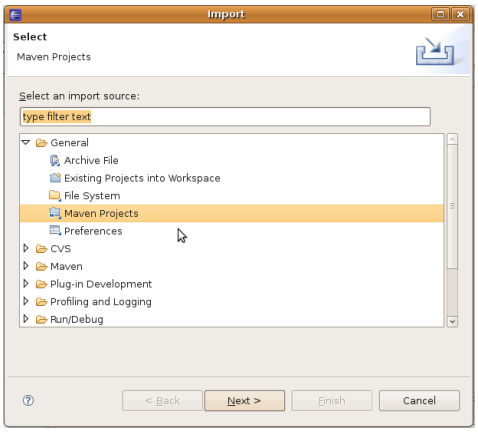

Figure 1: Import Maven Projects in Eclipse.

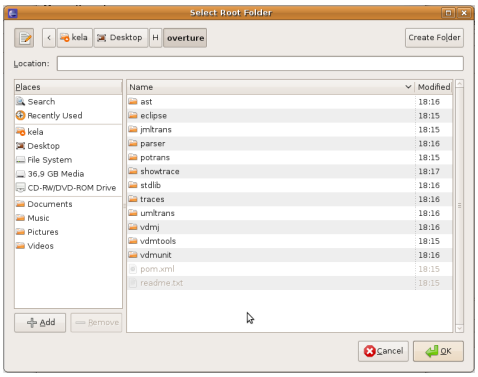

Figure 2: Select artifacts folder (core).

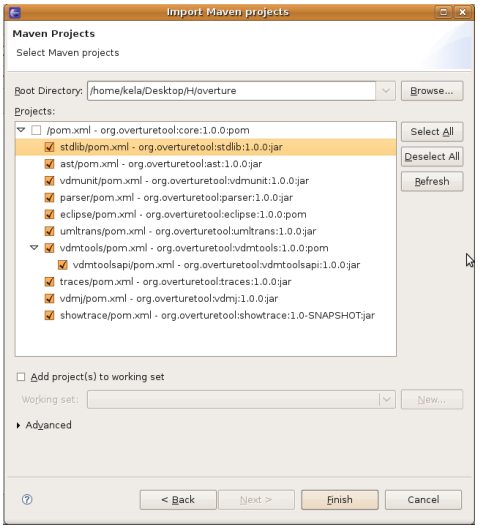

Figure 3: Select the artifacts to import.

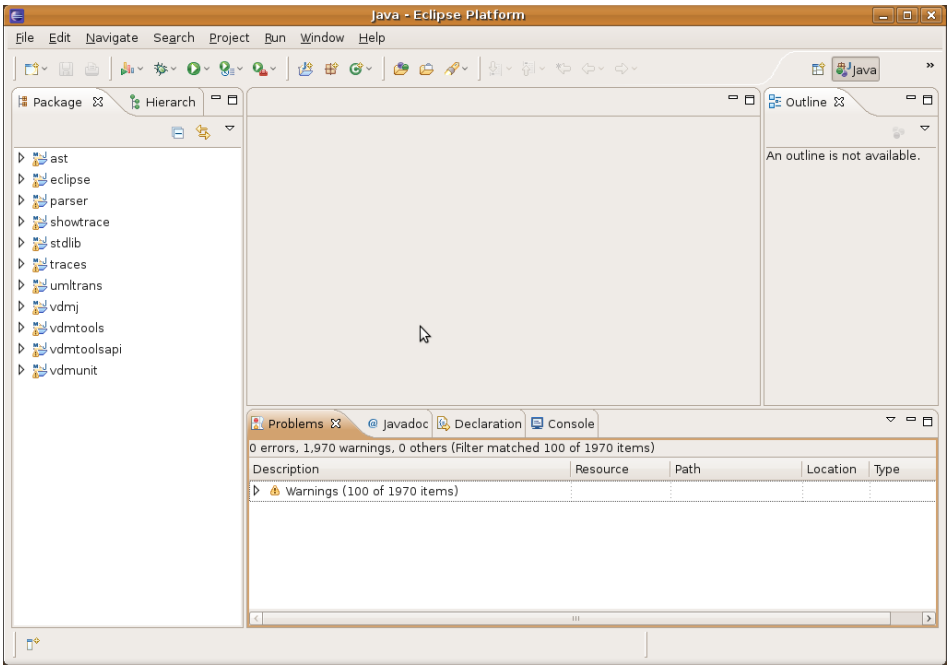

Figure 4: Select the artifacts to import.

2.2 Known issues

The Overture Maven repository has been updated which may result in errors after a check out / update.

All eclipse generated project files has been removed from the repository.
What to do:

Either the procidure for check out descriped in this document can be used or the alternative check out procidure descriped at: `http://www.overturetool.org/twiki/bin/view/Main/DevFaQ` under the "Connecting to the Subversion repository and import into Eclipse"

When the check out is complete open a terminal and go to the core artifact folder.

- Execute: mvn eclipse:clean
- Execute: mvn eclipse:eclipse
- Execute: mvn clean

Now the projects has been cleaned and all eclipse project file regenerated.

- Refresh all project in eclipse.

On the projects that give build errors right click and select Maven → Update Project Configuration

- Do a Project clean in eclipse.

Now all projects should build with no errors.

3 Adding a new component (maven artifact)

In the below example a new artifact is created in the "core" module. A new artifact named <artifact-Name>is created in the org.overturetool group by execution the command shown in listing **??** in the directory trunk/core which in this guide is checked out as overture.

```
mvn archetype:create
  -DgroupId=org.overturetool
  -DartifactId=org.overturetool.<artifactName>
```

Listing 2: Command for creation of a new Maven artifact.

3.1 SVN:Ignore

When a new artifact is added a SVN:Ignore file should be added to the new artifact folder:

```
target*
.settings
.classpath
.project
```

Listing 3: SVN:Ignore file.

4 Enabling VDM Tools integration

First step is to enter a property with the path to VDM Tools in the maven settings.xml file. The property is added to a profile which is set to default. The Maven settings is located in the .m2 repository location.

${user.home}/.m2/settings.xml

```
<profile>
  <id>default</id>
  <activation>
  <activeByDefault>true</activeByDefault>
  </activation>
  <properties>
    <user.vdmtoolscmdpath>
      c:\Program Files\The VDM++ Toolbox v8.2b\bin\vppde.exe
    </user.vdmtoolscmdpath>
  </properties>
</profile>
```

Listing 4: Set VDM Tools path in the profile property.

Since only the core components of overture tool has been checked out the location of the VDMT Maven plug-in must be specified to Maven can fetch the last build of the plug-in. To do so add this XML tag to the current profile:

```
<pluginRepositories>
  <pluginRepository>
    <id>sf.net</id>
    <name>Lausdahl Snapshots</name>
    <releases>
      <enabled>true</enabled>
      <updatePolicy>daily</updatePolicy>
      <checksumPolicy>ignore</checksumPolicy>
```

```
    </releases>
    <snapshots>
      <enabled>true</enabled>
      <updatePolicy>never</updatePolicy>
      <checksumPolicy>ignore</checksumPolicy>
    </snapshots>
    <url>http://overturetraces.svn.sourceforge.net/viewvc/overturetraces</url>
    <layout>default</layout>
  </pluginRepository>
</pluginRepositories>
```

A complete settings file can be found in section **??**

The Maven VDM Tools plugin can be used in a maven artifact. The plugin contains two goals:

type : Run VDM Tools type check on the project

code : Run VDM Tools Java Code gen

Before the plugin can be used in the artifacts build cycle it has to be attatched to artifact. This is done as shown in listing **??**.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.overturetool.tools</groupId>
      <artifactId>vdmt</artifactId>
      <configuration>

        <excludePackages>
          <param>org.overturetool.traces.test</param>
          <param>org.overturetool.traces.VDMUnit</param>
        </excludePackages>

        <excludeClasses>
          <param>SomeClassToExclude</param>
        </excludeClasses>

        <importPackages>
          <param>org.overturetool.ast.itf</param>
          <param>org.overturetool.ast.imp</param>
        </importPackages>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Listing 5: Enabling VDMT plug-in for a Maven artifact.

The configuration tag in listing **??** is optional. It includes features to limit the code generation:

excludePackages Here VDM packages that should be excluded from the code generation can be specified.

excludeClasses Here VDM classes that should be excluded from the code generation

importPackages Here additional Java Imports can be listed. The code generator will include those in the import section of all the generated Java files.

All parameters from above should be put in a param tag. They will be treaded like a list so multiple param's can be added.

4.1 VDM Tools plugin package structure

The VDM Tools plugin for maven uses the vpp files places in the project under:

/src/main/vpp

This folder is not created automatically so it have to be created and all VDM files with the extension .vpp must be placed inside this folder for the VDMT plug-in to work properly. The same applies to all the depended project of the artifact. The vpp files from this projects will also be included. The code generator uses the folders stored as sub folders to vpp as VDM packages like in Java. This means if the class A below is code gennerated it will be placed in a java package called: org.overture.testProject

```
org
  overture
    testProject
      A
```

Listing 6: VDM package structure.

4.2 Executing VDMT goals

In order to execute the goals the complete plug-in name must be used:

- org.overturetool.tools:vdmt:type

```
[INFO]
------------------------------------------------------------------------
[INFO] Building org.overturetool.testProject
[INFO]    task-segment: [org.overturetool.tools:vdmt:type]
[INFO]
------------------------------------------------------------------------
[INFO] [vdmt:type]
[INFO] VDM Tools cmd path located at:/usr/vdmpp/bin/vppde
[INFO] File added:
/home/overture/core/org.overturetool.testProject/src/main/vpp/org/overturetool/A.vpp
[INFO]
------------------------------------------------------------------------
[INFO] VDM Type check SUCCESSFUL
[INFO]
------------------------------------------------------------------------
[INFO]
------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO]
------------------------------------------------------------------------
```

Listing 7: VDMT type build result.

- org.overturetool.tools:vdmt:code

```
[INFO]
------------------------------------------------------------------------
[INFO] Building org.overturetool.testProject
[INFO]    task-segment: [org.overturetool.tools:vdmt:code]
[INFO]
------------------------------------------------------------------------
[INFO] [vdmt:code]
[INFO] VDM Tools cmd path located at:/usr/vdmpp/bin/vppde
[INFO] File added:
```

```
/home/overture/core/org.overturetool.testProject/src/main/vpp/org/overturetool/A.vpp
[INFO] Class: A Package: org.overturetool Java:
/home/overture/core/org.overturetool.testProject/src/main/java/org/overturetool/A.java
[INFO]
------------------------------------------------------------------------
[INFO] VDM Code generation SUCCESSFUL
[INFO]
------------------------------------------------------------------------
[INFO] Updating imports
[INFO]
------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO]
------------------------------------------------------------------------
```

Listing 8: VDMT Code build result.

4.3 Importing the newly created VDMT enabled project in Eclipse

To import the newly created project in Eclipse the Eclipse project files must be created like in section **??**. To create the project file execute the following in a terminal placed in the root of the project. Folder overture:

```
mvn eclipse:eclipse
```

Follow the same procidure as in section **??**. After selecting the import option in Eclipse the fig. **??** will be displayed. Select the new artifact and click finish then the new artifact fill be imported in the Eclipse workspace.

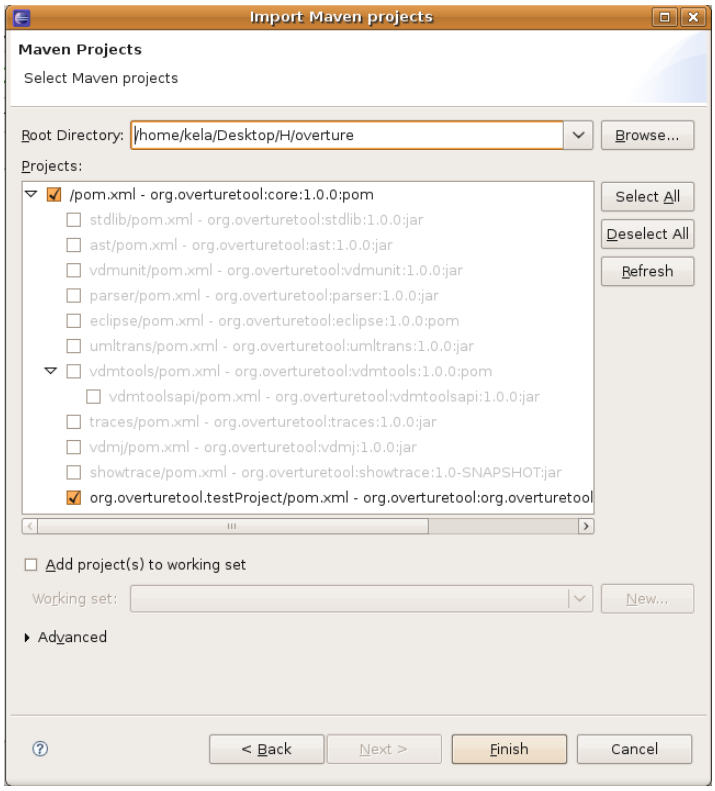

Figure 5: Select artifacts folder (core).

Appendix

4.4 VDMT Maven settings.xml

```
<settings>
  <activeProfiles>
    <activeProfile>default</activeProfile>
  </activeProfiles>
  <profiles>
    <profile>
      <id>default</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <properties>
        <user.vdmtoolscmdpath>
          c:\Program Files\The VDM++ Toolbox v8.2b\bin\vppde.exe
        </user.vdmtoolscmdpath>
      </properties>
      <pluginRepositories>
        <pluginRepository>
          <id>sf.net</id>
          <name>Lausdahl Snapshots</name>
          <releases>
            <enabled>true</enabled>
            <updatePolicy>daily</updatePolicy>
            <checksumPolicy>ignore</checksumPolicy>
          </releases>
          <snapshots>
```

```
            <enabled>true</enabled>
            <updatePolicy>never</updatePolicy>
            <checksumPolicy>ignore</checksumPolicy>
          </snapshots>
          <url>http://overturetraces.svn.sourceforge.net/viewvc/overturetraces</url>
          <layout>default</layout>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
</settings>
```