Table of contents

Prerequisites:	1
Exercise 1:	1
a) Start of the Overture Editor	1
b) Debugging a VDM Model using the Overture Editor	2
b1) Setting up a project	2
b2) Debugging a VDM Model	6
b3) DBGP	7
Exercise 2:	8
a) Syntax highlight the keyword	8
b) Wizard	8
b1) wizard	8
b2) action	10
TIP: Debugging in the Overture Editor	11
Exercise 3: Calling code from VDMTools	11
Appendix	12
SortExample.vpp	12
Solution to selectionSorter	14
OvertureGenerateCppWizard	14
GenerateCppCode	16
OvertureGenerateCodeCppWizardPage	16

Prerequisites:

To be able to make these exercises a few things is needed:

- An Eclipse IDE with the Plug-in Development Environment (PDE)
- Checked all of the Overture project out using the guide Kenneth made
- Installed the plug-in DLTK

Exercise 1:

a) Start of the Overture Editor

You should at least have all the plug-in projects in you workspace, the following figure shows the projects:

 org.overturetool.eclipse.plugins.dbgp.core [trur

 org.overturetool.eclipse.plugins.debug.ui [trunk/o

 org.overturetool.eclipse.plugins.debug.ui [trunk

 org.overturetool.eclipse.plugins.editor.core [tru

 org.overturetool.eclipse.plugins.editor.overture

 org.overturetool.eclipse.plugins.editor.overture

 org.overturetool.eclipse.plugins.editor.ui [trunk

 org.overturetool.eclipse.plugins.launching [trur

 org.overturetool.eclipse.plugins.showtrace.core

 org.overturetool.eclipse.plugins.stdlib [trunk/or

 org.overturetool.eclipse.plugins.stdlib [trunk/or

 org.overturetool.eclipse.plugins.stdlib [trunk/or

 org.overturetool.eclipse.plugins.traces [trunk/o

 org.overturetool.eclipse.plugins.traces.core [tru

 org.overturetool.eclipse.plugins.traces.core [tru

 org.overturetool.eclipse.plugins.umltrans [trunl

 org.overturetool.eclipse.plugins.umltrans [trunl

Choose the menu "Run" \rightarrow "Debug Configurations..." you should now see the debug configuration dialog, where you should create a new EclipseApplication (Which is done by double clicking in the left menu). Choose a location for the workspace for the new Eclipse instance and click on the debug button.

	Name: EclipseApplicatio	n			
type filter text	Catala Annun ant				
 Apache Tomcat Eclipse Application 	<u>Main</u> (** Arguments) ** Plug-ins Configuration Tracing Tracing Tracing Workspace Data				
EclipseApplicatior		e/overture/DebugecripseApp			
Eclipse Data Tools	Clear: <u>w</u> orkspace	e O log <u>W</u> orkspace File System Varia <u>b</u> les			
Generic Server	Ask for confirmation	a before clearing			
🗄 Generic Server(Exter	Program to Run				
HTTP Preview I2EE Preview	• Run a product:	org.eclipse.platform.ide			
🖻 Java Applet	O Run an <u>application</u> :	org.eclipse.ui.ide.workbench			
Java Application	Java Runtime Environment				
Filter matched 18 of 18 item		Apply Revert			

b) Debugging a VDM Model using the Overture Editor

b1) Setting up a project

Create a new Overture Project by clicking on the and select the menu File \rightarrow New \rightarrow other...

Select a wizard				
Wizards: Java Emitter Ier JavaScript JavaScript Maven CovertureTool wi Overture File Overture Proj Pilg-in Develop Pig-in Develop Pig-in Develop Pig-in Develop Pig-in Develop	npiates zards lect ment agging Explorer			
۲	< <u>B</u> ack	Next >	Einish	Cancel

The first time the user creates a Overture project the default interpreter needs to be set up. This is done by clicking on configure interpreters... and choose the VDMJ interpreter.

Project name:						
Contents						
• Create new project in <u>w</u> orkspace						
 Create project from existing source 						
Directory: /home/kedde/Overture/DebugEclipseApp	Browse,					
Host: Localhost 🗘						
Execution						
Target Host	\$					
• Use default interpreter (Currently 'VDMJ Interpreter')	re interpreters					
○ Use a project specific interpreter: VDMJ Interpreter ≎						
Dialect						
Select dialect OML \$						

Name the new Project "Sort Example" and select finish. If everything is done properly the new project is created and is visible to the left. Add a new Overture File sort.vpp (File \rightarrow new \rightarrow Overture File) and copy the content from the file in the appendix to the newly created file. The editor should now look like:



Small exercise: Double click on the error in the problem view and fix it.

b2) Debugging a VDM Model

Debugging a VDM model is similar to debugging a Java application in Eclipse. There is one addition that you need to select a "void" operation/function the interpreter going to evaluate.

	<u> </u>		
(A			
Main Sinter	preter 🖾 <u>C</u> ommon		
Project:			
Sort Example		Browse	
Launch script:			
sort.vpp	sort.vpp		
Operation:			
Class	SelectionSort	Browse	
Class.		Diowse	
Operation:	startSort		
Debug Options			
Break on first l	ine		
Enable DBGP I	aging		
	299119		
Debug From Co	onsole		
v			
	 Project: Sort Example Launch script: sort.vpp Operation: Class: Operation: Debug Options Break on first li Enable DBGP Ic Debug From Cc 	Project: Sort Example Launch script: sort.vpp Operation: Class: SelectionSort Operation: startSort Debug Options Break on first line Image: Debug From Console Image: Debug From Console	



At a breakpoint you can use the either the Variables view values of variables in a given context (upper right) and you can use the interactive console to evaluate expression("Window" \rightarrow "show view" \rightarrow "Interactive console" lower right on the above picture). To step through the model it is possible to use the menu "run" \rightarrow "step".

Exercise:

Set a breakpoint at line 95 (SelectionSorter(1)) by right-click in the left margin at line 95 and select Toggle Breakpoint, a blue dot will appear in the left margin.

Debug the model with the options shown above. Now observe the variable "data" before and after the operation call.

b3) DBGP

If you are interested in the communication between the IDE and the debugger server it is possible to view this, by first selecting "enable dbgp logging" in the debug configuration dialog and afterwards show the "script debug log" view (window \rightarrow show view \rightarrow other... \rightarrow Script debug log)

🖳 Console 🙆 Tasks 🗖 Script Debug Log 🛛 🧧 🗖					
					Copy Clear 🗢
Date	Time	Туре	Session	Message context_get -c 1 -d	0 -i 152
2009-05-06	20:43:41.838	Event		SUSPEND from org.eclipse.dltk.internal.debug.core.model.ScriptThread	
2009-05-06	20:43:43.872	>>	7	context_names -d 0 -i 150	
2009-05-06	20:43:43.872	<<	7	<response command="context_names" transaction_id="150"><context context_get"="" name="</td><td></td></tr><tr><td>2009-05-06</td><td>20:43:43.875</td><td>>></td><td>7</td><td>context_get -c 0 -d 0 -i 151</td><td></td></tr><tr><td>2009-05-06</td><td>20:43:43.913</td><td><<</td><td>7</td><td><response command=" transaction_id="151"><property context_get"="" name="sel</td><td></td></tr><tr><td>2009-05-06</td><td>20:43:43.916</td><td>>></td><td></td><td>context_get -c 1 -d 0 -i 152</td><td></td></tr><tr><td>2009-05-06</td><td>20:43:43.958</td><td><<</td><td>7</td><td><response command=" transaction_id="152"><property <="" name="da" td=""><td></td></property></property></context></response>	
2009-05-06	20:43:43.962	>>	7	context_get -c 2 -d 0 -i 153	
2009-05-06	20:43:44.002	<<	7	<response command="context_get" transaction_id="153"></response>	
2009-05-06	20:44:29.153	>>	7	eval -i 154 U2VsZWN0aW9uU29ydGBkYXRh	
2009-05-06	20:44:29.154	<<	7	<response command="eval" success="1" transaction_id="154"><property name<="" td=""><td>~</td></property></response>	~

Exercise 2:

a) Syntax highlight the keyword

Adding a new keyword is quite simple, open the class OvertureCodeScanner located in the project org.overturetool.eclipse.plugins.editor.ui. And add the wanted keyword.

b) Wizard

Generate C++ code					
Please choose	the wnated optio	ns			
Select project:	Sort Example	~			
			\$		
0		<u>Finish</u>	Cancel		

b1) wizard

To create a wizard page similar to the above we first need to create the content of the dialog, which is done by extending WizardPage. Use the following code snippet and insert the label and the combobox.

The new class should be located in org.overturetool.eclipse.plugins.editor.internal.ui.wizards in the project called org.overturetool.eclipse.plugins.editor.ui

```
package org.overturetool.eclipse.plugins.editor.internal.ui.wizards;
import org.eclipse.jface.wizard.WizardPage;
import org.eclipse.swt.widgets.Combo;
public class OvertureGenerateCodeCppWizardPage extends WizardPage{
     Combo combo;
     IProject[] iprojects;
     protected OvertureGenerateCodeCppWizardPage(String pageName) {
           super (pageName);
           setTitle("Generate C++ code");
            setDescription("Please choose the wnated options");
      }
     public void createControl(Composite parent) {
           IWorkspaceRoot iworkspaceRoot =
                 ResourcesPlugin.getWorkspace().getRoot();
            iprojects = iworkspaceRoot.getProjects();
      }
```

The wizard can now be created by extending Wizard. Again you can use a code snippet, but needs to insert the rest:

```
package org.overturetool.eclipse.plugins.editor.internal.ui.wizards;
import java.util.ArrayList;
import jp.co.csk.vdm.toolbox.api.corba.ToolboxAPI.ToolType;
import org.eclipse.core.resources.IContainer;
import org.eclipse.core.resources.IFile;
import org.eclipse.core.resources.IFolder;
import org.eclipse.core.resources.IProject;
import org.eclipse.core.resources.IResource;
import org.eclipse.core.runtime.CoreException;
import org.eclipse.jface.wizard.Wizard;
import org.overturetool.vdmtools.VDMToolsProject;
public class OvertureGenerateCppWizard extends Wizard {
      OvertureGenerateCodeCppWizardPage cppWizard;
      private String[] exts = new String[] { "vpp", "tex", "vdm" };
       public void addPages() {
             cppWizard = new OvertureGenerateCodeCppWizardPage("C++ page");
             addPage(cppWizard);
       }
private ArrayList<IFile> getAllMemberFiles(IContainer dir, String[] exts) {
            ArrayList<IFile> list = new ArrayList<IFile>();
            IResource[] arr = null;
            try {
                  arr = dir.members();
            } catch (CoreException e) {
            }
            for (int i = 0; arr != null && i < arr.length; i++) {</pre>
                  if (arr[i].getType() == IResource.FOLDER) {
                        list.addAll(getAllMemberFiles((IFolder) arr[i], exts));
                  }
                  else {
                        for (int j = 0; j < exts.length; j++) {</pre>
                              if
(exts[j].equalsIgnoreCase(arr[i].getFileExtension())) {
                                    list.add((IFile) arr[i]);
                                    break;
                              }
                        }
                  }
            }
            return list;
      }
      @Override
      public boolean performFinish() {
      try
      {
```

}

```
ArrayList<IFile> fileNameList = getAllMemberFiles(project, exts);
return true;
}
catch (Exception e) {
return true;
}
}
```

There is missing some code in the method performFinish() to determine which project member files should be code generated.

b2) action

To test the wizard, there have already been created an extension point to the menu Overture menu in the plugin.xml in the project org.overturetool.eclipse.plugins.editor.ui.



You should create an action class that extends IWorkbenchWindowActionDelegate placed in org.overturetool.eclipse.plugins.editor.ui.actions in the run method insert the following snippet.

```
OvertureGenerateCppWizard wizard = new OvertureGenerateCppWizard();
WizardDialog dialog = new WizardDialog(null, wizard);
dialog.create();
dialog.open();
```

If you didn't name action class GenerateCppCode then you will have to rename the class attribute in the plugin.xml.

TIP: Debugging in the Overture Editor

If you didn't get the wizard right the first time, it is possible to debug the Overture Editor, just by

putting a breakpoint e.g. in the run method in the action class.

Another way to detect errors is to look in the error log in the overture editor(window \rightarrow show view \rightarrow other \rightarrow error log)

Exercise 3: Calling code from VDMTools

To access the VDMTools Api ther in our wizard, we make use of a binary plug-in project, which is already set up for you in the manifest file.

To finish the wizard these two snippet might be helpful.

```
vdmProject.addFilesToProject(filenamesString);
vdmProject.codeGenerateCPP(false);
```

Appendix

SortExample.vpp

```
-- class Sort
___
-- Consist
___
class Sort
 instance variables
   protected data : seq of int := [0];
   protected temp : bool := false;
   protected fibonacciReturn : nat := 0;
 operations
   public
   data init : seq of int ==> ()
   data init (1) ==
     data := 1;
   sort ascending : () ==> ()
   sort ascending () == is subclass responsibility;
    sort descending : () ==> ()
   sort descending () == is subclass responsibility
end Sort
/*
     MergeSort
*/
class MergeSort is subclass of Sort
operations
 public
 sort_ascending : () ==> ()
 sort ascending() ==
    data := MergeSorter(data)
functions
 MergeSorter: seq of real -> seq of real
 MergeSorter(l) ==
   cases l:
              -> l,
     []
              -> l,
       [e]
___
     others \rightarrow let 11^12 in set {1} be st abs (len 11 - len 12) < 2
                 in
                   let l_l = MergeSorter(l1),
                       l_r = MergeSorter(12) in
                    Merge(l_l, l_r)
    end;
 Merge: seq of int * seq of int -> seq of int
 Merge(11,12) ==
    cases mk (11,12):
     mk ([],1),mk (1,[]) -> 1,
```

```
-> if hd 11 <= hd 12 then
      others
                               [hd 11] ^ Merge(tl 11, 12)
                             else
                                [hd 12] ^ Merge(11, tl 12)
    end
 pre forall i in set inds 11 \& 11(i) >= 0 and
      forall i in set inds 12 & 12(i) >= 0
end MergeSort
/*
     SelectionSort
* /
class SelectionSort is subclass of Sort
 functions
   min index : seq1 of nat -> nat
   min_index(l) ==
      if len 1 = 1
      then 1
      else let mi = min_index(tl l)
           in if 1(mi+1) < hd 1
              then mi+1
              else 1
 operations
    /*
    * StartSort testing :
   */
     public
    startSort : () ==> ()
    startSort () ==
    (
     data init([2,4,67,70,3,60]);
           SelectionSorter(1);
           temp := print();
     fibonacciReturn := fibonacci(3);
     temp := print();
    );
     private print : () ==> bool
     print () ==
      (
           return false;
            --return new IO().echo("Hello world!\n");
     );
     private
    sort ascending : () ==> ()
    sort ascending () == SelectionSorter(1);
     /*
            SelectionSorter
     */
     private
    SelectionSorter : nat ==> ()
    SelectionSorter (i) ==
      if i < len data</pre>
      then (
                  dcl temp1: nat;
            dcl mi : nat := min index(data(len data,...,len data)) + i - 1;
```

```
temp1 := data(mi);
data(mi) := data(i);
data(i) := temp1;
SelectionSorter(i+1)
)
functions
public fibonacci: nat -> nat
fibonacci(x) ==
(
if x = 1
then 1
else x * fibonacci(x-1)
);
end SelectionSort
```

Solution to selectionSorter

```
private
SelectionSorter : nat ==> ()
SelectionSorter (i) ==
    if i < len data
    then (dcl temp1: nat;
        dcl mi : nat := min_index(data(i,...,len data)) + i - 1;
        temp1 := data(mi);
        data(mi) := data(i);
        data(i) := temp1;
        SelectionSorter(i+1)
        )
correct is
min_index(data(i,...,len data)) + i - 1;
```

OvertureGenerateCppWizard

```
package org.overturetool.eclipse.plugins.editor.internal.ui.wizards;
import java.util.ArrayList;
import jp.co.csk.vdm.toolbox.api.corba.ToolboxAPI.ToolType;
import org.eclipse.core.resources.IContainer;
import org.eclipse.core.resources.IFile;
import org.eclipse.core.resources.IFolder;
import org.eclipse.core.resources.IProject;
import org.eclipse.core.resources.IResource;
import org.eclipse.core.runtime.CoreException;
import org.eclipse.jface.wizard.Wizard;
import org.overturetool.vdmtools.VDMToolsProject;
public class OvertureGenerateCppWizard extends Wizard {
    OvertureGenerateCodeCppWizardPage cppWizard;
    private String[] exts = new String[] { "vpp", "tex", "vdm" };
    public void addPages() {
```

```
cppWizard = new OvertureGenerateCodeCppWizardPage("C++ page");
             addPage(cppWizard);
       }
       /**
       * This method returns a list of files under the given directory or its
       * <u>subdirectories</u>. The directories themselves are not returned.
       * Oparam dir
                    a directory
       * @return list of IResource objects representing the files under the
given
                 directory and its subdirectories
       * /
     private ArrayList<IFile> getAllMemberFiles(IContainer dir, String[] exts)
{
            ArrayList<IFile> list = new ArrayList<IFile>();
            IResource[] arr = null;
            try {
                  arr = dir.members();
            } catch (CoreException e) {
            }
            for (int i = 0; arr != null && i < arr.length; i++) {</pre>
                  if (arr[i].getType() == IResource.FOLDER) {
                        list.addAll(getAllMemberFiles((IFolder) arr[i], exts));
                  }
                  else {
                        for (int j = 0; j < exts.length; j++) {</pre>
                              if
(exts[j].equalsIgnoreCase(arr[i].getFileExtension())) {
                                     list.add((IFile) arr[i]);
                                    break:
                               }
                        }
                  }
            }
            return list;
      }
      @Override
      public boolean performFinish() {
            try
            {
                  VDMToolsProject vdmProject = VDMToolsProject.getInstance();
                  vdmProject.init("/home/kedde/Programs/vice/bin/vicegde",
ToolType. PP TOOLBOX);
                  IProject proj = null;
                  for (IProject project : cppWizard.iprojects) {
                        if
(cppWizard.combo.getText().equals(project.getName())) {
                              proj = project;
                        }
                  ļ
                  ArrayList<IFile> fileNameList = getAllMemberFiles(proj,
exts);
                  ArrayList<String> filenamesString = new ArrayList<String>();
                  for (IFile file : fileNameList) {
                        filenamesString.add(file.getLocationURI().getPath());
                  }
                  vdmProject.addFilesToProject(filenamesString);
                  vdmProject.codeGenerateCPP(false);
                  return true;
```

```
}
catch (Exception e) {
    return true;
}
}
```

GenerateCppCode

```
package org.overturetool.eclipse.plugins.editor.ui.actions;
import org.eclipse.jface.action.IAction;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.wizard.WizardDialog;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.IWorkbenchWindowActionDelegate;
import
org.overturetool.eclipse.plugins.editor.internal.ui.wizards.OvertureGenerateCpp
Wizard;
public class GenerateCppCode implements IWorkbenchWindowActionDelegate {
     public void dispose() {
     }
     public void init(IWorkbenchWindow arg0) {
     }
     public void run(IAction arg0) {
           OvertureGenerateCppWizard wizard = new OvertureGenerateCppWizard();
       WizardDialog dialog = new WizardDialog(null, wizard);
       dialog.create();
        dialog.open();
     }
     public void selectionChanged(IAction arg0, ISelection arg1) {
      }
}
```

OvertureGenerateCodeCppWizardPage

```
package org.overturetool.eclipse.plugins.editor.internal.ui.wizards;
import org.eclipse.core.resources.IProject;
import org.eclipse.core.resources.IWorkspaceRoot;
import org.eclipse.core.resources.ResourcesPlugin;
import org.eclipse.jface.wizard.WizardPage;
import org.eclipse.swt.SWT;
```

```
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Combo;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Label;
public class OvertureGenerateCodeCppWizardPage extends WizardPage{
     Combo combo;
     IProject[] iprojects;
     protected OvertureGenerateCodeCppWizardPage(String pageName) {
            super(pageName);
            setTitle("Generate C++ code");
        setDescription("Please choose the wnated options");
      }
     public void createControl(Composite parent) {
            Composite composite = new Composite(parent, SWT.NONE);
        GridLayout layout = new GridLayout();
        layout.numColumns = 2;
        composite.setLayout(layout);
        setControl(composite);
        new Label(composite,SWT.NONE).setText("Select project: ");
        combo = new Combo(composite, SWT.NONE);
        IWorkspaceRoot iworkspaceRoot =
ResourcesPlugin.getWorkspace().getRoot();
            iprojects = iworkspaceRoot.getProjects();
           for (IProject project : iprojects) {
                  combo.add(project.getName());
            }
           combo.select(0);
      }
}
```