Overture – Open-source Tools for Formal Modelling TR-2010-04
February 2010

2010 Release Planning for Overture and DESTECS

by

Peter Gorm Larsen
Kenneth Lausdahl
Augusto Ribeiro

Contents

Chapter 1

Introduction

This document is intended to provide an overview of the release planning for the open source project called Overture as well as the European research project DESTECS. Efforts will be made to ensure that stable releases of both the executables for Overture and DESTECS in a systematic fashion. It is recommended that official public releases of these executables are synchronised with each other. The planning presented here is aggressive and probably on the optimistic side but naturally this also depends upon the time the different stakeholders are able to deliver to the projects during 2010.

Official major public releases will be made yearly and will be made available through the Overture and DESTECS official websites respectively. One month before the major release, a Release Candidate will be made for the purpose of testing and locate bugs. In the meantime, snapshot builds will be provided via SourceForge. In this period only bug fixes needed for the features planning in that release will be checked in. Minor official public releases will be released approximately once every three months and for each of these two weeks of release candidate periods will be used. A full release procedure will be followed for each of the public releases such that the users can expect all public releases to be stable. In between these public releases ad-hoc releases will be produced on a needs basis but it is suggested that at least a weekly build that contains the latest source code will be put in system. For such ad-hoc releases the documentation cannot be guaranteed to be updated and it will not be tested at the same level of detail as for official releases so no guarantee is provided for all ad-hoc releases. All the builds will consist of stand alone branded version for the Windows, Mac and Linux platforms.

This document start off with providing an overview of the release numbering scheme in Section 2. This is followed by Section 3 which provides the current plan for the releases of Overture executables during 2010. Finally Section 4 provides the current plan for the additional features released in the DESTECS executable releases in 2010. Later on the document will be updated with plans for subsequent years.

Chapter 2

Release Build Numbering

The Overture and DESTECS project deliverables (both executables as well as associated documentation) build numbering will be made with the following format:

$$\boxed{\textbf{X.Y.Z}}$$

The **X** indicates which is the major public release version of the tool. During the DESTECS project there will be three major releases of the DESTECS Tool (month 12, 24 and 36, i.e. December 2010, 2011 and 2012 respectively). It is suggested that these builds are numbered 1.0.0, 2.0.0 and 3.0.0 respectively.

An increase of **Y** indicates a minor public release including significant update since last version, for example, a new feature which have been through a proper release procedure. Thus the users can expect it to be in a stable state.

An increase of **Z** indicates an ad-hoc release with minor update since the last version, for example, bug fixes. No guarantee for stability is provided for these ad-hoc releases because they have not been going through the release procedure used for public releases.

When release candidates are made available at the X and Y level there shall be a list of checks that needs to be performed by different named stakeholders before the release candidate is lifted up to before an official X or Y release. This check list shall include:

1. All examples for all dialects must be up to date with the software from the release.

2. The tutorials for all dialects must be updated to fit the newest screen dumps and the features in the new release.

3. Checks to be performed on the executable at each OS platform (Mac, Linux and Windows). The explicit list of what to check for the executables must be created.

Chapter 3

Contents of Overture releases in 2010

This section presents a plan for the contents of the 2010 releases of the Overture executable.

Release 0.2.0 (March 2010): The aim for this release is to include the following features:

1. Analyse whether we can go for a debugger without DLTK. AugustoRibeiro ,**Dependencies: None**
2. Follow standard way to use pictures in Eclipse and document this for all developers. AugustoRibeiro. **Dependencies: None**
3. The combinatorial testing feature will be updated in the IDE with the features for reducing the size of the test suites in intelligent ways made available in VDMJ for all VDM dialects. KennethLausdahl, *Done(Prototype): 2. feb 2010 - Kenneth* Unit test, user manual and tutorial update still outstanding.
4. In the edit perspective automatic type checking will only be performed in case the syntax is correct for all definitions such that type errors does not get mixed with syntax errors. KennethLausdahl, *Done: 1. feb 2010 - Kenneth* **Remarks:** missing doc
5. When the Overture tool is first started up and the user kills the welcome window the VDM++ edit perspective will be started automatically. AugustoRibeiro, *Done: 1. feb 2010 - Kenneth*
6. Syntax and type check the arguments used in the debug configuration. KennethLausdahl, **Dependencies: None Remarks:** missing doc
7. Remove all communication exceptions from VDMJ which causes the console to change (e.g. the existing problem with the debugger in the IDE whenever an invariant for instance invariants are broken). NickBattle, KennethLausdahl, **Dependencies: None**
8. Documentation of the Overture IDE in order to enable more developers to make updates for the IDE and for the DESTECS development to follow similar principles. This shall document the implementation of the editor, the parser, the builder, the outline and the AST access. **Status:** Ongoing AugustoRibeiro, **Dependencies: 3, 1**

9. Specify the top level scheduling strategy for the interpreter dealing with multiple threads in VDM++ and VDM-RT (to subsequently be implemented in the interpreter). KennethLausdahl,++

10. Make the pretty printer with test coverage usable for all VDM dialects. KennethLausdahl, NickBattle
Remarks: missing doc

Release 0.3.0 (June 2010): The aim for this release is to include the following features:

1. Provide help (F1) inside Eclipse; AugustoRibeiro
2. provide goto definition and completion; AugustoRibeiro
3. Complete the new ASTGen utility and add all sources for it to the tools utility in the Overture SF SVN. NickBattle **Dependencies: None**,
4. Change to the AST generated from the new version of ASTGen all over the Overture sources (i.e. this will affect the development of UML, JML and POtrans components). NickBattle, KennethLausdahl, **Dependencies: 3**
5. Replace DLTK (For lower level description of the goals please refer to appendix A). AugustoRibeiro, **Dependencies: 2**
6. Launch configuration setup:
   (a) When creating a new debug configuration it shall be named after the name of the project (and not just "New configuration").
   (b) The configuration should be linked to the project so it is available from *Debug As*
   (c) General plugin.xml problem.

   AugustoRibeiro, **Dependencies: R020:1**
7. Making sure that the run and debug buttons in the IDE work appropriately and does not crash whenever the user tries to start them. Ideally these should simply change to the debug perspective if no debug configuration is present and be ready to type an expression into a quick console where simple expressions can be typed and executed (and up and down in the list of commands given is supported). AugustoRibeiro, **Dependencies: 6**
8. A button for moving a test case from the combinatorial testing perspective into the debugger such that it will stop at the right place for the run-time error reported for the test case selected. KennethLausdahl,ChristianThillerman, **Dependencies: 6**
9. Stop all threads when a breakpoint or a run-time error has occured in one of the threads. NickBattle,++, **Dependencies: R020:9**
10. In the debug perspective it shall be possible to look into the contents of objects in the variables view. ??, **Dependencies: R020:1,3**
11. Include all definitions in the Outline view for all VDM dialects. AugustoRibeiro, **Dependencies: R020:3**

12. When creating new projects (by wizard) in the IDE include the possibility for including standard libraries (i.e. (VDM headers) IO and Math) in the project. These are then to be included in a special "lib" directory. ??, **Dependencies: None**
13. Make sure that proof obligations for standard libraries are ignorred. NickBattle, **Dependencies: None**
14. Make it possible for users to provide static implementations to "is not yet specified" functions and operations by adding a Jar file to a lib folder in the project. KennethLaudahl,NickBattle *Prototype done*
15. Make it possible for users to add a java implementation of a VDM class by adding a jar to the lib folder. The implementation should be done so one instance in java corresponds to one instance in VDM. No static. KennethLausdahl,NickBattle, **Dependencies: 14**
16. Make use of **`measure`**'s inside the the interpreter so it would be checked in the same way as invariants, pre and post conditions. NickBattle

Release 0.4.0 (September 2010): The aim for this release is to include the following features:

1. First stable bi-directional connection from Overture/VDM++ to/from JML. CarlosVilhena
2. New version of the interpreter for scheduling of multiple threads in VDM++ and VDM-RT prepared for co-simulkation also able to break after a fixed amount of time. KennethLausdahl,++
3. User documentation in pdf available as a part of the Overture executable so the user can search on-line in these sources (including a VDM-10 language manual). PeterGormLarsen,++

Release 1.0.0 (December 2010): The aim for this release is to include the following features:

1. Quick fix incorporated in the editor view of the IDE for all VDM dialects. ??
2. Goto definition in the edit view of the IDE across all files in a project for all VDM dialects. ??
3. First version of refactoring support for all VDM dialects. ??
4. Possibly first release of proof support feature for proving a subset of the POs generated. NickBattle
5. Support in the Proof Obligation Explorer view for new icons indicating whether the user have proved a PO or have manually inspected it and approved it. This also means that the icon with the red cross shall only be used for those where proof have failed and a less intemidating iconj shall be used for the POs that one have not yet dealt with. ??
6. Include VDMDoc (inspired by JavaDoc) for all VDM dialects. ??
7. Stable version of the UML mapper enabling proper round-trip engineering between VDM and UML class diagrams and enable generation of traces from a subset of UML sequence diagrams. KennethLausdahl,PeterGormLarsen

8. First release of code generator from VDM to a programming language. $\boxed{\text{MarcelVerhoef}}$
9. Increased LaTeX support in the editor such that parts outside the `vdm_al` environments are highlighed as TeXClipse would do it. $\boxed{??}$

Release X (?): The aim for this release is to include the following features:

1. VDM Tools
   (a) Start VDM Tools command line prompt from a Overture project. Where the prompt is a console inside Eclipse.
   (b) Create a VDM Tools project file from a Overture project and launch the VDM Tools GUI.
   (c) Create a number of buttons for VDM Tools code generation through the VDM Tools comman dline interface, with output in the console.
   (d) Make a VDM Tools corba DBGPReader to enable a VDM Tools debugger to connect to the Overture IDE and use the basic debug facility.
   (e) Preferences for VDM Tools
2. Make Java code generator
3. Make C code generator
4. Make C++ code generator
5. Make VDM Doc parser
6. Make VDM ast pretty printer for math syntax
7. Make VDM source format
8. Coverage Editor
9. DLTK replacement
   (a) Recreate Debugger. New debug protocol separation from VDMJC. Launch, Debug runner,
   (b) Recreate the parser
   (c) Recreate the builder
   (d) Recreate the Editor
   (e) Recreate the outline
   (f) Recreate the package explorer
10. Ant scripts to build a product
11. Tests of the plug-ins
12. Test program which can parse/type/interpret all examples and compare results with expected.
13. Extract VDM is not specified classes from Java implementation of external interpreter modules.
14. Make a VDM model of Multi-core CPU distribution/scheduling.

Chapter 4

Contents of DESTECS releases in 2010

This section presents a plan for the contents of the 2010 releases of the DESTECS executable.

Release 0.1.0 (March 2010): The aim for this release is to include the following features:

1. Develop development environment for DESTECS enabling different plug-ins to be added and docuenting this. $\boxed{\text{KennethLausdahl}}$
2. Stand-alone DESTECS executable with the DESTECS icon $\boxed{\text{KennethLausdahl}}$
3. Develop wizard for creating a new DESTECS project. This shall create 3 different directories in the project for “Discrete Event”, “Continuous Time” and “Fault Modelling” respectively. $\boxed{\text{??}}$
4. Ability to invoke the 20-sim editor for 20-sim models in a DESTECS project. $\boxed{\text{ControlLabs}}$

Release 0.2.0 (June 2010): The aim for this release is to include the following features:

1. TBD

Release 0.3.0 (September 2010): The aim for this release is to include the following features:

1. TBD

Release 1.0.0 (December 2010): The aim for this release is to include the following features:

1. Co-simulation between Overture VDM models and 20-sim models is fully enabled. $\boxed{\text{KennethLausdahl,++}}$
2. Synchronising the log viewer from Overture with the plot of variables from 20-sim in an Eclipse setting. $\boxed{\text{ControlLabs,++}}$

*

Appendix A

Low level goals for IDE release plan.

Low level goals for IDE release plan.

1. Core:
   - Utilities (VdmProject)
   - AST
   - VDMJ (export)
   - Core Parser
   - Core Builder
   - Commands
2. Core.Vdmpp
   - Nature
   - Content Type
3. VDMJ Builder/Parser Update
4. UI:
   - Abstract Editor
     - Syntax Highlight
     - Navigation
     - Auto Completion
     - ...
   - Outline
   - Navigator
   - Properties (Language Version, etc)

- Abstract Wizards (New Project/File)
- Perspective?

5. UI.Vdmpp
   - Editor
   - Wizards
   - Templates
   - Perspective?
6. Debug
   - Launch Core
   - Debug Model Core
   - Launch UI - (Tabs, shortcuts)
   - Debug UI - (source highlight, source lookup)
7. Debug.Vdmpp - (Launch tabs)?

A.1 Component status

A.1.1 Core

ast Scheduled for replacement

astgenerator Scheduled for replacement

jmltrans Carlos has taken an action to redevelop the initial JML transformation.

parser Scheduled for replacement

potrans Unknown

proofsupport Unknown

showtrace Stable, besides that unknown

stdlib Has beem split up and now only a few VDM libraries it still located here.

traces Stable contains CT for all dialects including filtering.

umltrans Stable for VDMPP and VDMRT. Only from VDM to UML. The other direction is incomplete do to upgrades for VDM to UML.

vdmj Stable. New check for history counters, Delegation to Java added including a RemoteControl feature.

vdmjc Unknown

vdmtools Unknown

vdmunit Unknown

A.1.2 IDE

ast Stable, scheduled to be moved to *org.overutre.ide.core*

builders/core

builders/vdmj

debug/core

debug/launching Stable, VM arguments updated to a set of arguments can be parsed to the debuggibg VM.

parsers/vdmj

platform

plugins/idedebughelper

plugins/latex Stable. Contrins support for all dialects. Assums that pdflatex is avaliable

plugins/poviewer Stable supports all dialects.

plugins/proofsupport

plugins/showtrace Stable

plugins/traces Stable

plugins/umltrans Stable

ui

utility Stable, scheduled to be moved to *org.overutre.ide.core*

vdmpp/core

vdmpp/ui

vdmpp/debug/core

vdmpp/debug/ui

vdmrt/core

vdmrt/ui

vdmrt/debug/core

vdmrt/debug/ui

vdmsl/core

vdmsl/ui

vdmsl/debug/core

vdmsl/debug/ui