Overture – Open-source Tools for Formal Modelling YY–NN
3 July 2012

Overture Release Process & Feature List

by

Joey W. Coleman

Contents

Chapter 1

Release Process

This section is an attempt to explain a process and rationale for the release process of the Overture tool. Much of the information in this document should be obvious, however, stating it explicitly is useful to reduce ambiguity in discussions.

1.1 Version Numbering

Version numbers are of the generally form $x.y.z$, e.g. 1.1.2, and in the run-up to a non-trivial release, may include a release candidate suffix, e.g. 1.2.0-RC1. The meaning of each position is as follows:

Major, x._._ Indicates major changes to the tool, including replacement of a foundational subsystem (such as the AST), or the addition of a major new feature set (such as the possible future merge of the COMPASS tool).

This sort of change requires as much testing as we can manage and must use release candidates prior to release.

Minor, _.x._ Indicates either the addition of significant (but not foundational) new features or replacement of a minor subsystem. Adding a feature such as code completion is an example.

This feature requires thorough testing and should use release candidates.

Increment, _._.x Indicates an apparently regression-free bugfix release. The only changes relative to the prior version are bug fixes and very minor feature changes.

In order to allow these release to happen quickly, this sort of release should not use release candidates. However, it must pass the unit test suite and it must have had the manual GUI testing procedure checked.

Release Candidates, x.y._-RCn Indicates a version that, if no problems are found, could become the version indicated (1.2.0, in this case).

Any release candidate should pass the unit tests and manual GUI testing procedure, but this is not guaranteed. It must pass all tests before becoming a full release.

1.2 Development with Git

One of the primary features of Git is its ability to cheaply create, maintain, and re-integrate many branches of development within a repository. The actual semantic work of merging is no cheaper than other version control systems –nor is it likely that anything can make it cheaper– but the cost of tracking files and parents is minimized through technical support.

So, on that basis, every develop should commit changes to their own personal branch on sourceforge, and contact the Release Manager when the feature(s) they are working on are ready to be merged into the main 'master' branch. Around that point they should first pull from the master branch into their own branch and perform the integration before their changes are merged back into the master branch.

1.3 Testing

Testing, at present, is comprised of a few procedures:

1. an incomplete set of unit tests; and,
2. an informal procedure for exercising the tool's features using the GUI; and,
3. distributing the tool to many users to see if it works for them.

Clearly, we need to work on this.

1.4 Bug Tracking & Feature Requests

For Overture in general we use SourceForge to track both, and for DESTECS we use ChessForge. COMPASS will likely use SourceForge as well.

I am strongly considering using a Trac instance to track bugs, features, and releases, if only to have some automatic dependency tracking.

1.5 Documentation

Much more ought to be said about this.

Every release should be accompanied by release notes, the contents of which are yet to be fully determined.

We need a complete build guide that lists the development tools used, their role in the process (and any deviations from "standard" usage by us), and how to invoke the process in various contexts (release building, local building, etc). This is in preparation, and will end up on the wiki (at least).

This document itself is intended to be a living reference, as part of this it will be updated regularly. Furthermore, I will be investigating tools like Trac to produce a task list that records dependencies, release versions, and so on.

Chapter 2

Feature Plan

This section presents the list of planned features and an approximate timeline of when they are expected to be in an official release of the Overture/DESTECS/COMPASS Tool. The intention is to cover the development of all three tools, and the feature planning for Overture does, occasionally, rely on the context of DESTECS and (eventually) COMPASS development.

The feature plan is fairly sparse, but represents the things that are firmly expected.

It's worth noting that I hope to release minor versions of the various tools regularly, so those are not explicitly recorded unless it's relevant.

2.1 Actions Timeline

February 2012

- Overture 1.2.1 release, with minor bugfixes [ARI, KEL]
- DESTECS 1.3.0 release, based on Overture 1.2.1 [ARI, KEL]

March 2012

- DESTECS 1.3.1 release, based on Overture 1.2.1 [ARI, KEL]
- Overture v2 release, based on New AST [KEL, JWC]

April 2012

- DESTECS 1.3.2 release, based on Overture 1.2.1 [ARI, KEL]

May 2012

- DESTECS 1.3.3 release, based on Overture 1.2.1 ARI
- DESTECS development freeze

2.2 Language Board "execution" RMs

These features and changes come from the Overture Language Board and represent things that have change and are now a part of the VDM language family standards.

Language board Requests for Modification will be added to this list after they reach "execution" phase. As and when people pick up these tasks, they will be added to the action timeline in Section 2.1; they will be removed when they have been completed.

SF.net ID	Title	Who	When
3011828	Include the non-deterministic statement inside traces	Unknown	Unknown
3220182	Expressions in periodic thread definitions	Nick B	in Git
3220223	Values in duration / cycles statements	Nick B	in Git
3220437	Extend duration and cycles	Unknown	Unknown
3438625	Append Map Pattern	Unknown	Unknown

2.3 Other Features

This section pulls in the old release planning document and adds some that don't yet have specific work plans. Some may no longer be relevant.

1. Complete the new ASTGen utility and add all sources for it to the tools utility in the Overture SF SVN. NickBattle **Dependencies: None**
2. Include VDMDoc (inspired by JavaDoc) for all VDM dialects. / Allowing multi line comments in VDM to be represented in the AST ??
3. Make VDM ast pretty printer for math syntax
4. Documentation of the Overture IDE in order to enable more developers to make updates for the IDE 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:** 1
   (a) Parser
   (b) Builder; How to insure TC is completed
   (c) How to add debug with a new interpreter and corresponding launch configuration.

5. Provide help (F1) inside Eclipse
6. Provide goto definition and completion; AugustoRibeiro
7. 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 and POtrans components). NickBattle, KennethLausdahl, **Dependencies: 1**
8. Quick fix incorporated in the editor view of the IDE for all VDM dialects. ??
9. Goto definition in the edit view of the IDE across all files in a project for all VDM dialects. ??
10. First version of refactoring support for all VDM dialects. ??
11. 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
12. First release of code generator from VDM to a programming language. MarcelVerhoef
13. Increased LaTeX support in the editor such that parts outside the `vdm_al` environments are highlighed as TeXClipse would do it. ??
14. Make Java code generator
15. Make C code generator
16. Make C++ code generator
17. Make VDM Doc parser
18. Make VDM source format
19. Tests of the plug-ins
20. Make a VDM model of Multi-core CPU distribution/scheduling.
21. GUI unit testing within Eclipse