SEVENTH FRAMEWORK PROGRAMME

Grant Agreement: 287829

Comprehensive Modelling for Advanced Systems of Systems

COMPASS

Initial release of the COMPASS Tool User Manual

Technical Note Number: D31.1a

Version: 1.0

Date: September 2012

Public Document

http://www.compass-research.eu

Contributors:

Peter Gorm Larsen, AU
Joey Coleman, AU
Anders Kaels Malmos, AU
Rasmus Lauritzen, AU
Stefan Hallerstede, AU

Editors:

Joey Coleman, AU
Stefan Hallerstede, AU

Reviewers:

Alexander Romanovsky, NCL
Margherita Forcolin, Insiel
Simon Foster, York

Document History

Ver	Date	Author	Description
0.1	18-07-2012	Peter Gorm Larsen	Initial document version
0.2	30-07-2012	Stefan Hallerstede	Edited; assigned tasks
0.3	13-08-2012	Joey Coleman	Edited; wrote section on commandline; commented section on simulation/debug (now in commandline)
0.4	06-09-2012	Joey Coleman	Editing; cleanup of most remaining draftnotes
0.5	07-09-2012	Joey Coleman	Last little cleanups; ready for internal draft review
0.6	25-09-2012	Joey Coleman	Incorporate internal draft comments
1.0	28-09-2012	Peter Gorm Larsen	Ready for 1st year project review

Contents

1 Introduction

This document is a user manual for the COMPASS tool, an open source tool supporting systematic engineering of System of Systems using the COMPASS Modelling Language (CML). The ultimate target is a tool that is built on top of the Eclipse platform, that integrates both with the RT-Tester tool, and that also integrates with Artisan Studio. In addition the tool functionality is available from a command-line interface which also is explained. This document is targeted at users with limited experience working with Eclipse based tools. Directions are given as to where to obtain the software.

This user manual does not provide detail regarding the underlying CML formalism. Thus if you are not familiar with this, we suggest the tutorial for CML before proceeding with this user manual [WCF+12, BGW12].

This version of the document supports version 0.0.1 of the COMPASS tool suite. It is intended to introduce readers to how this version of the tool can be used with CML models. The connection to the Artisan Studio tool and the RT-Tester tool is not yet available and, hence, is not described further in this deliverable.

At present, the delivered software is split into several pieces. The core functionality includes the ability to read CML models and perform basic typechecking on them. On top of this we have built two proof-of-concept tools: the first is the initial COMPASS IDE, which wraps the core functionality and provides editing abilities; the second is the command-line tool, which also wraps the core functionality, but provides the ability to launch the initial version of the CML simulator.

The next version of the tool, for the D31.2 deliverable, will integrate the two proof-of-concept tools we have here into a single IDE tool that incorporates all the features. We will, however, retain the command-line tool for development purposes.

Section 2 describes how to get hold of the software and get it installed on your own computer. Section 3 describes the command-line interface to the COMPASS tool. Afterwards Section 4 explains the different views in the COMPASS Eclipse perspective. This is followed by Section 5 which explains how to manage different projects in the COMPASS tool.

2 Obtaining the Software

This section explains how to obtain the COMPASS IDE and COMPASS command-line tool, described in this user manual.

The COMPASS suite is an open source tool, developed by universities and industrial partners involved in the COMPASS EU-FP7 project [FLW12]. The tool is developed on top of the Eclipse platform [Car05].

The source code and pre-built releases for the COMPASS CML tool are hosted on SourceForge.net, and this has been selected as our primary mechanism for supporting the community of users of CML and the developers building tools for the COMPASS platform. It has facilities for file distribution, source code hosting, and bug reporting.

The simplest way to run the COMPASS Tool is to download it from the Sourceforge.net project files download page at

`https://sourceforge.net/projects/compassresearch/files/`

This download is a specially-built version of the Eclipse platform that only includes the components that are neccessary to run the COMPASS Tool — it does not include the Java development tools usually associated with the Eclipse platform.

Also available from that page is the command-line tool, with which it is possible to use the initial CML simulator.

Note that the COMPASS tools require the Java SE Runtime Environment version 6 or later. On Windows environments, either the 32-bit or 64-bit versions may be used, on Mac OS X and Linux, the 64-bit version is required.

3 The Command-line Interface

The command-line interface to the COMPASS tool was conceived as a tool for developers to quickly allow them to access and test the core libraries. This allows developers of the tool to quickly test new functionality for correctness without having to create the GUI elements that will control the functionality in the integrated IDE. A beneficial side-effect of having this tool is that general users are not required to load the IDE to test CML programs, but instead may invoke them via the command-line.

3.1 Available Functionality

The command-line tool presently has access to the following features in the core libraries:

- CML parser
- CML typechecker
- CML AST to DOT graph generation
- CML interpreter
- Example core plugins

The CML parser is the primary element of the command-line tool, as nothing can happen without using it. Generally, the tool will read in a (sequence of) CML file(s) and then perform a typecheck on the abstract syntax tree (AST). At this point, the data is ready to be used by the rest of the core libraries and plugins. It is possible to run the core libraries on an AST that has not been typechecked, but doing so is not recommended except to test error reporting or if the user only wishes to generate a DOT graph of the AST.

The DOT graph generator will output a representation of the AST generated from the input CML files in the DOT language. The output is suitable for use in the Graphviz suite of graph visualization utilities.[1] The output is useful for producing a visual representation of the data used internally by the COMPASS tool to represent the static structure of a model of a system of systems. This allows a developer to quickly verify whether the input CML files result in the expected internal data structures.

[1]Found at `http://www.graphviz.org`.

The Proof Obligation Generator (POG) can be invoked by the command-line tool and doing so will cause it to produce the internal representation of the consistency and validation checks that the input CML files require. The theorem proving and model checking plugins planned for future releases will be able to use these proof obligations to verify the consistency and correctness of the input CML model.

The CML interpreter is only accessible from the command-line tool in the M12 release of the COMPASS tools. Invoking the CML interpreter on a set of input CML files will result in the model being executed in a simulation run. The results of the simulation will be printed to the console during the run. Interfaces to graphical components are not yet available.

3.2 Basic Invocation

After obtaining the commandline tool package, decompress it into a folder. In that folder will be –among others– the files `cmlc` and `cmlc.bat`. Invocation of the `cmlc` (Linux, Mac OS X) or `cmlc.bat` (Windows) script with no parameters will produce the following output:

```
COMPASS command line CML Checker - CML 0

Usage: cmlc [switches] <file1>, ...,<fileN>
Switches:
    -coe    -   Continue on Exception
    -dotg   -   DOT graph generation,
                -dotg=<out> write output to <out>
    -dwa    -   Run the Div Warn Analysis example
    -e      -   Simulation,
                -e=<processId> simulate the process identified
                   by <processId>
    -empty  -   Empty analysis, run the empty analysis
    -i      -   Interactive mode
    -notc   -   Omit type checking phase
    -po     -   Parse Only, stop analysis after the parsing
    -soe    -   Silence on Exception
    -tco    -   Type Check Only
```

Assuming some CML model in a file, `example.cml`, loading it into the command-line interface is accomplished by typing `cmlc example.cml`. If run in this manner, the output will be:

```
COMPASS command line CML Checker - CML 0
Parsing file: example.cml
1 file(s) successfully parsed. Starting analysis:
 Running The CML Type Checker on example.cml
```

Note that, by default, the interpreter is not invoked on input; see Section 3.3.

It is also possible to input CML directly into the command-line tool when invoked with the `-i` option. This is useful for quickly cutting and pasting small bits of CML, for example.

To generate a DOT-language graph representation of a parsed CML model, we use the `-dotg=<file>` option. The invocation `cmlc -dotg=example.gv example.cml` will produce console output:

```
COMPASS command line CML Checker - CML 0
Parsing file: example.cml
1 file(s) successfully parsed. Starting analysis:
 Running eu.compassresearch.ast.preview.DotGraphVisitor on
     example.cml
 Running The CML Type Checker on example.cml
```

And it will also write out the file `example.gv` in the process. This file can then be processed with a DOT language processor (such as Graphviz) into many other formats, including PDF, SVG, PNG, and JPEG.

3.3 CML Simulation

Please note: this section serves as the documentation requirements of D32.1.

The commandline tool enables simulation of CML models when invoked with the `-e` option. Since the CML model may have more than one process defined, the `-e=<processId>` option must be supplied, where `<processId>` is the name of the process that is to be simulated.

As an example of how this works, consider the following CML model in a file called `example.cml`:

```
channels
  init, a, b

process A = begin
  @ init -> a -> Skip
end

process B = begin
  @ init -> b -> Skip
end

process C = A;B
```

The following command will simulate the process identified by `C`:

```
cmlc -e=C example.cml
```

This results in the following output being printed to the console:

```
COMPASS command line CML Checker - CML 0
Parsing file: example.cml
1 file(s) successfully parsed. Starting analysis:
 Running The CML Type Checker on example.cml
 Running The CML Interpreter on example.cml

--------begin step---------
Offered Events:
<init>
Current interpretation state:
C = (A = (init->a->Skip);B)
Trace after step:
<init>

--------begin step---------
Offered Events:
<a>
Current interpretation state:
C = (A = (a->Skip);B)
Trace after step:
<init><a>

--------begin step---------
```