Evolution of the Overture Tool Platform

Joey W. Coleman, Anders Kaels Malmos,
Claus Ballegaard Nielsen Peter Gorm Larsen

Department of Engineering, Aarhus University, Denmark

28 August 2012 / 10th International Workshop Overture/VDM

COMPASS
www.compass-research.eu

SEVENTH FRAMEWORK PROGRAMME

The Goal of the Overture Open Source Project

- Develop an open platform to supply tool support for the VDM dialects

The Goal of the Overture Open Source Project

- Develop an open platform to supply tool support for the VDM dialects
- Our vision is to evolve the Overture platform into a more general platform

The Goal of the Overture Open Source Project

- Develop an open platform to supply tool support for the VDM dialects
- Our vision is to evolve the Overture platform into a more general platform
- Thus making it practical to support other formal languages

The Purpose of this Presentation

To show

- how the overture platform can be extended beyond the original goal

The Purpose of this Presentation

To show

- how the overture platform can be extended beyond the original goal
- how extensions can reuse the existing platform

The Purpose of this Presentation

To show

- how the overture platform can be extended beyond the original goal
- how extensions can reuse the existing platform
- how extensions can contribute their development effort back into the existing platform

Introduction

The Overture family

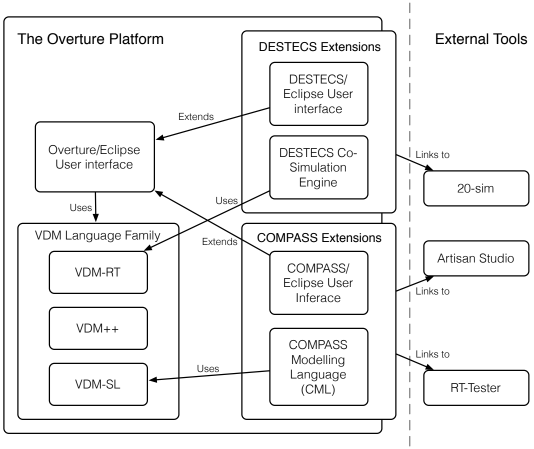
Extensibility of The Overture platform

Envisioned Reuse in COMPASS

Potential Contribution from COMPASS

Concluding remarks

The Overture Family Tree



The Overture Family Tree

The Overture AST

- An important part of Overture is the AST (Abstract Syntax Tree)

The Overture Family Tree

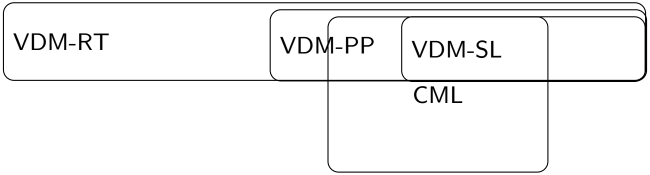
The Overture AST

- An important part of Overture is the AST (Abstract Syntax Tree)
- This is the key enabler of reuse between the languages

The Overture Family Tree

The Overture AST

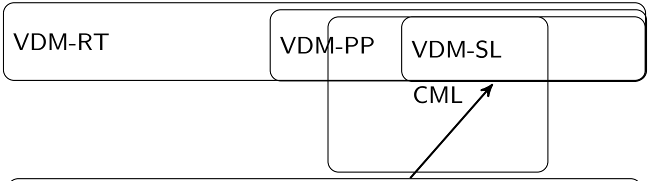
- An important part of Overture is the AST (Abstract Syntax Tree)
- This is the key enabler of reuse between the languages
- The languages have common parts of the AST



The Overture Family Tree

The Overture AST

- An important part of Overture is the AST (Abstract Syntax Tree)
- This is the key enabler of reuse between the languages
- The languages have common parts of the AST



- CML expressions $\subset$ VDM-SL expressions
- Some VDM-PP structural components like classes
- However, the semantics have changed!

Introduction

The Overture family

Extensibility of The Overture platform

Envisioned Reuse in COMPASS

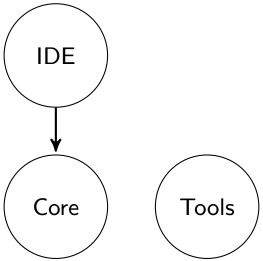
Potential Contribution from COMPASS

Concluding remarks

Basic Structure of Overture

The Parts

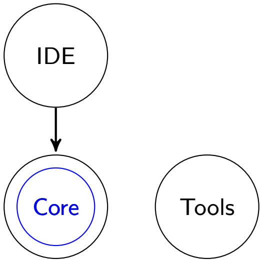
- Overture is comprised of three parts
- File layout has these in three separate folders
- Plug-ins for Overture should also conform to this structure



Basic Structure of Overture

The Parts

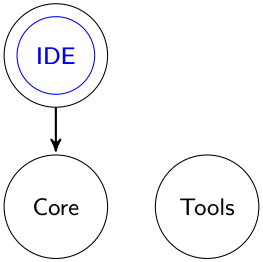
- Implements the VDM language family
- parser, typechecker, interpreter ...
- Together with associated language tools
- Independent of the IDE part



Basic Structure of Overture

The Parts

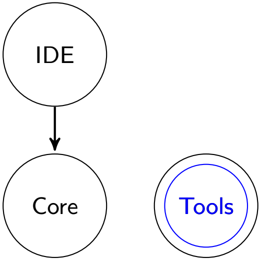
- Contains all of the IDE-related code
- Integrates the core library into Eclipse
- Giving all the basic functionally of a modern IDE



Basic Structure of Overture

The Parts

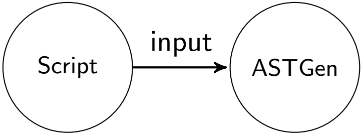
- Tools to assist the build process
- Contains the ASTGen (Abstract Syntax Tree Generator) tool
- This automates the generation of the AST based on a script



Basic Structure of Overture

The ASTGen Tool

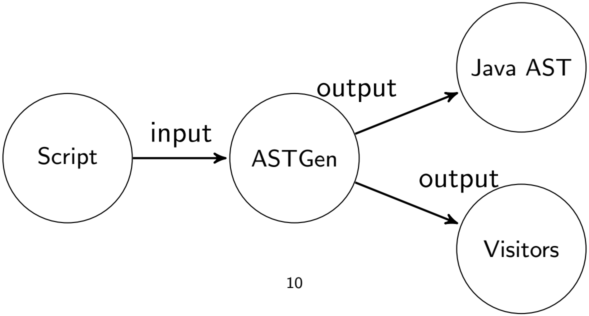
- ASTGen takes as input a script describing the structure of the AST



Basic Structure of Overture

The ASTGen Tool

- ASTGen takes as input a script describing the structure of the AST
- It outputs a Java class hierarchy representing the AST
- And also generates visitor classes



Basic Structure of Overture

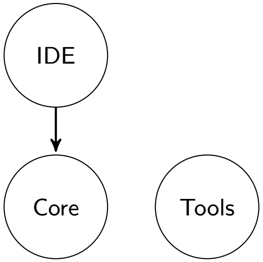
The structure of a Plugin for Overture

- All the plugins for Overture has the same basic structure
- The core part
- An optional IDE and tool part
- E.g. a model checker plug-in will have all the core logic implemented in the core part; and
- All the GUI related code in the IDE part

Envisioned Reuse in COMPASS

COMPASS Structure

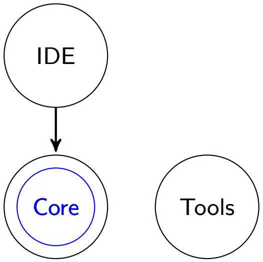
- COMPASS is structured in the same basic way as Overture
- This makes reuse of existing components easy to identify
- The core and IDE part of COMPASS has each has reuse



Envisioned Reuse in COMPASS

The Core Part

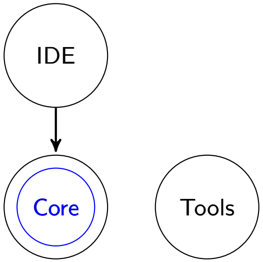
- Implements all the core CML language components
- Parser, typechecker, interpreter etc.



Envisioned Reuse in COMPASS

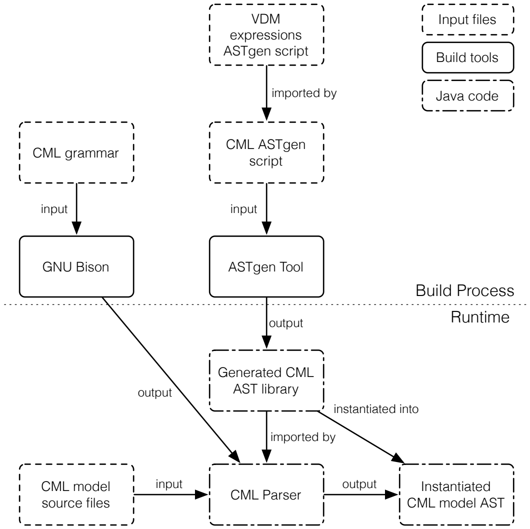
The Key Enablers of Reuse in the Core

- The expression language of CML is a proper subset of VDM
- The semantics of the CML expressions is identical to VDM expressions
- The common AST and visitors facilitates by the build process



Envisioned reuse in COMPASS

Overview of the Parser and AST Generation Process

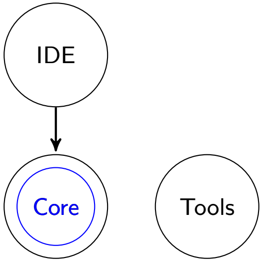


Envisioned Reuse in COMPASS

Reuse in the Core Part

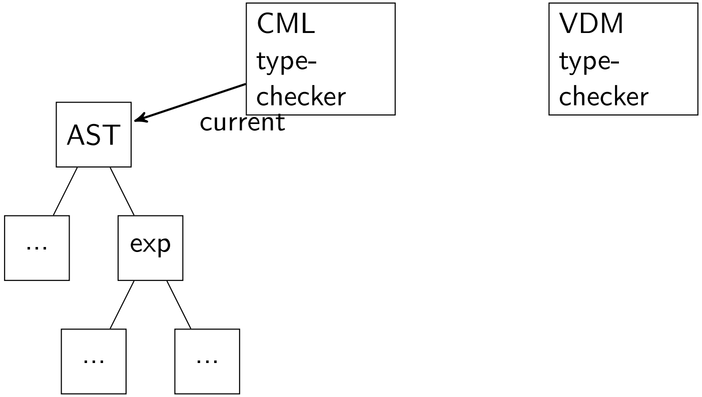
What can be reused in COMPASS?

- The VDM typechecker for CML expressions
- The VDM interpreter for CML expressions
- This gives COMPASS the typechecker and interpreter for expressions for free
- Furthermore, defects, errors and improvements will benefit several languages



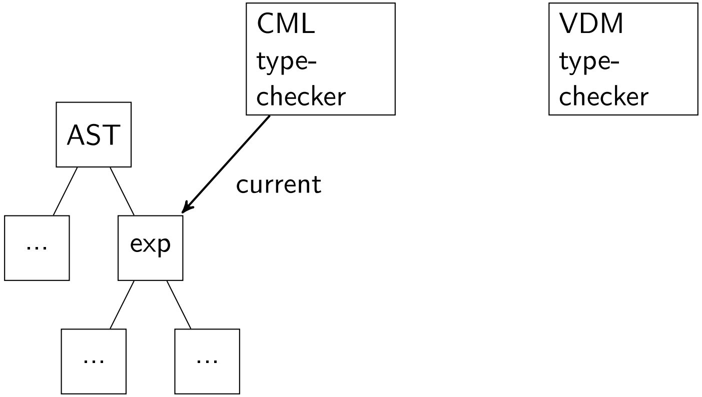
Envisioned Reuse in COMPASS

Reuse in the Core Part



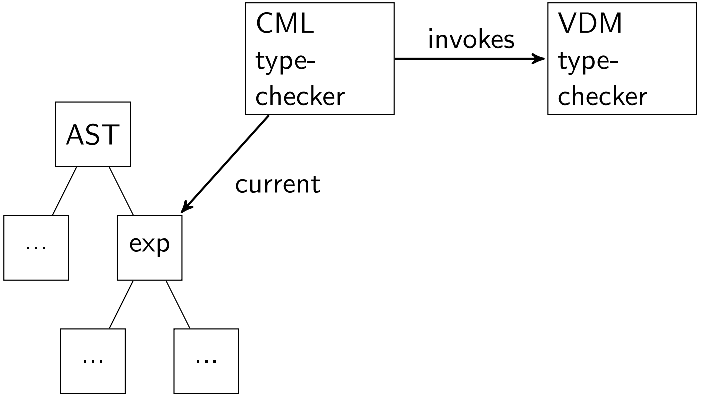
Envisioned Reuse in COMPASS

Reuse in the Core Part



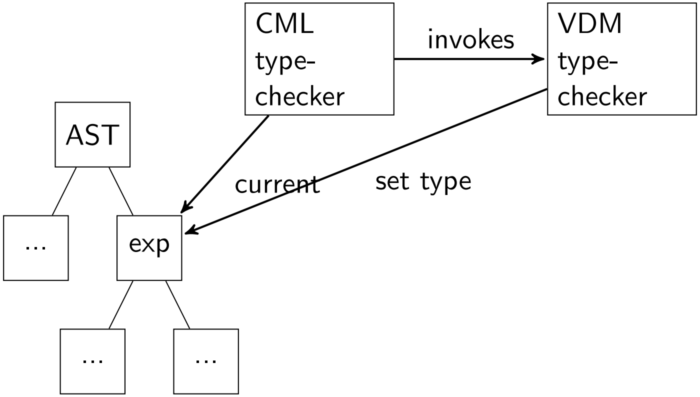
Envisioned Reuse in COMPASS

Reuse in the Core Part



Envisioned Reuse in COMPASS

Reuse in the Core Part

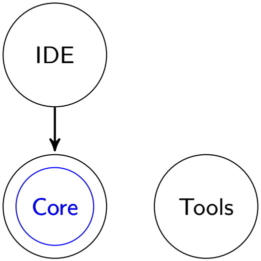


Envisioned Reuse in COMPASS

Reuse in the Core Part

What can't be reused in COMPASS?

- The VDM Parser
- The semantics of CML statements are substantially different from VDM
- Therefore reuse of statement components are not possible
- Even though they have common AST parts

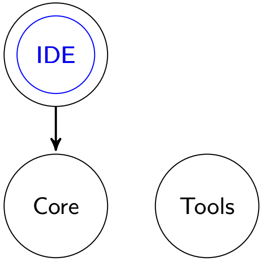


Envisioned Reuse in COMPASS

Reuse in the IDE

Basic structure

- The COMPASS IDE is Eclipse based
- It will have it's own perspective with standard features
- The plug-ins will be also integrated into the IDE

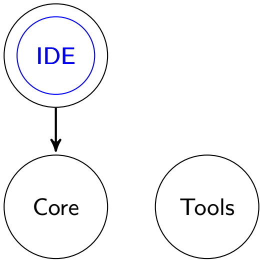


Envisioned Reuse in COMPASS

Reuse in the IDE Part

What are the key enablers of reuse?

- Eclipse is used as IDE
- The structure will be very similar to the general Overture VDM IDE

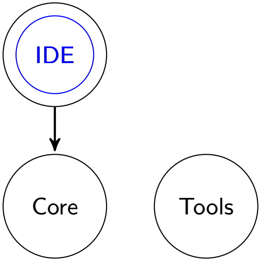


Envisioned Reuse in COMPASS

Reuse in the IDE Part

What can be reused?

- Basic IDE features
  - Editor
  - Debugger
  - outlining
  - etc.
- The COMPASS plug-ins will not be standard views, but still there might be parts of reuse



Potential Contribution from COMPASS

COMPASS plug-ins

COMPASS will develop a range of plug-ins which potentially could contribute to the entire Overture platform

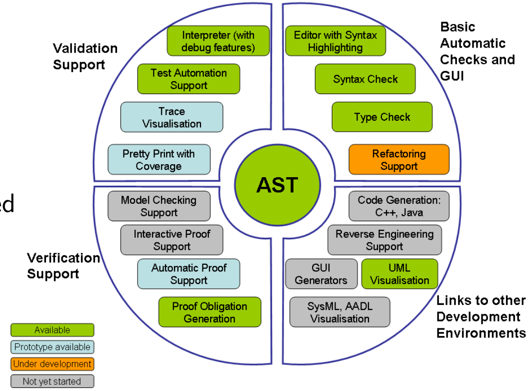
- Proof Obligation (PO) generator
- Theorem prover
- Model checker
- Refinement checker
- Static fault analyser
- Link to RT-Tester tool
- Link to Artisan Studio

Potential Contribution from COMPASS

COMPASS plug-ins

Proof Obligation (PO) generator

- The generated POs will be made available to the COMPASS theorem prover
- This link might be utilised by the VDM POG

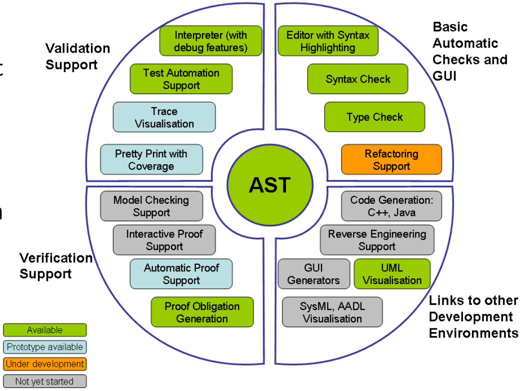

Validation Support
Interpreter (with debug features)
Test Automation Support
Trace Visualisation
Pretty Print with Coverage
Basic Automatic Checks and GUI
Editor with Syntax Highlighting
Syntax Check
Type Check
Refactoring Support
AST
Verification Support
Model Checking Support
Interactive Proof Support
Automatic Proof Support
Proof Obligation Generation
Links to other Development Environments
Code Generation: C++, Java
Reverse Engineering Support
GUI Generators
UML Visualisation
SysML, AADL Visualisation
Available
Prototype available
Under development
Not yet started

Potential Contribution from COMPASS

COMPASS plug-ins

Theorem prover

- The CML will be transformed into a format that can be processed by a theorem prover
- This work could also benefit the VDM theorem prover

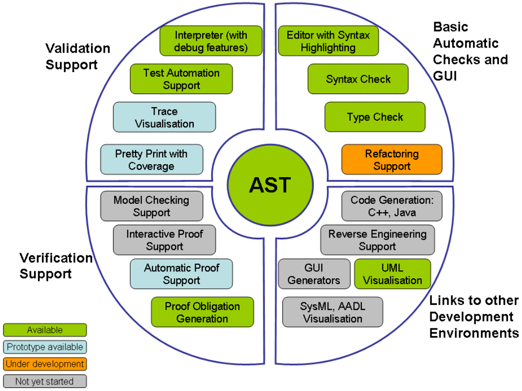

Validation Support
Interpreter (with debug features)
Test Automation Support
Trace Visualisation
Pretty Print with Coverage
Basic Automatic Checks and GUI
Editor with Syntax Highlighting
Syntax Check
Type Check
Refactoring Support
AST
Verification Support
Model Checking Support
Interactive Proof Support
Automatic Proof Support
Proof Obligation Generation
Links to other Development Environments
Code Generation: C++, Java
Reverse Engineering Support
GUI Generators
UML Visualisation
SysML, AADL Visualisation
Available
Prototype available
Under development
Not yet started

Potential Contribution from COMPASS

COMPASS plug-ins

Model checker

- Transforms CML into a format analysable by a model checker
- This will not be directly reusable, but the general architecture could be reused

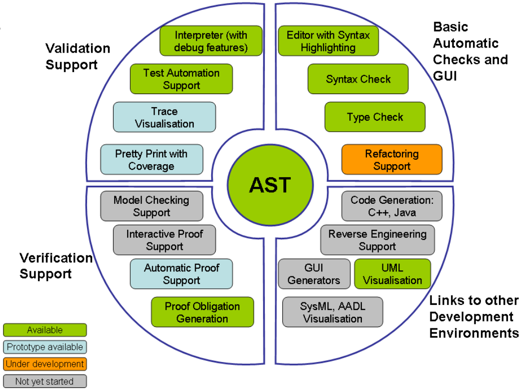

Validation Support
Interpreter (with debug features)
Test Automation Support
Trace Visualisation
Pretty Print with Coverage
Editor with Syntax Highlighting
Syntax Check
Type Check
Refactoring Support
Basic Automatic Checks and GUI
AST
Model Checking Support
Interactive Proof Support
Automatic Proof Support
Proof Obligation Generation
Verification Support
Code Generation: C++, Java
Reverse Engineering Support
GUI Generators
UML Visualisation
SysML, AADL Visualisation
Links to other Development Environments
Available
Prototype available
Under development
Not yet started

Potential Contribution from COMPASS

COMPASS plug-ins

Refinement checker

- Support for refinement of CML models will be developed
- The structure of this could be reused as a starting point for a VDM refinement checker

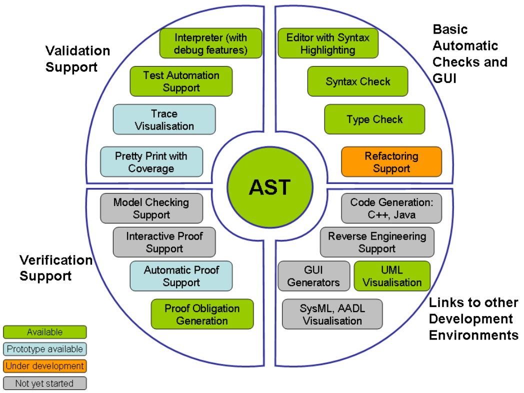

Validation Support
Interpreter (with debug features)
Test Automation Support
Trace Visualisation
Pretty Print with Coverage
Basic Automatic Checks and GUI
Editor with Syntax Highlighting
Syntax Check
Type Check
Refactoring Support
AST
Verification Support
Model Checking Support
Interactive Proof Support
Automatic Proof Support
Proof Obligation Generation
Links to other Development Environments
Code Generation: C++, Java
Reverse Engineering Support
GUI Generators
UML Visualisation
SysML, AADL Visualisation
Available
Prototype available
Under development
Not yet started

Potential Contribution from COMPASS

COMPASS plug-ins

Static fault analyser

- Brings fault injection capabilities when simulating a CML model
- This could be adaptable to the VDM interpreter

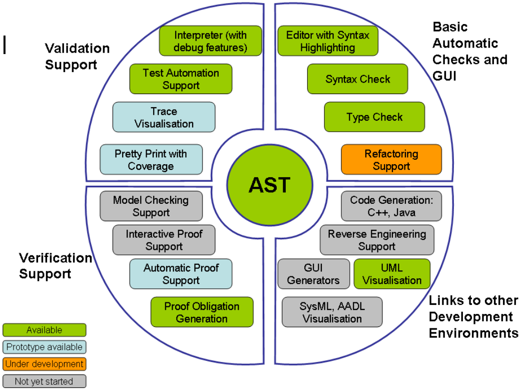

Validation Support
Interpreter (with debug features)
Test Automation Support
Trace Visualisation
Pretty Print with Coverage
Basic Automatic Checks and GUI
Editor with Syntax Highlighting
Syntax Check
Type Check
Refactoring Support
AST
Verification Support
Model Checking Support
Interactive Proof Support
Automatic Proof Support
Proof Obligation Generation
Links to other Development Environments
Code Generation: C++, Java
Reverse Engineering Support
GUI Generators
UML Visualisation
SysML, AADL Visualisation
Available
Prototype available
Under development
Not yet started

Potential Contribution from COMPASS

COMPASS plug-ins

Link to Artisan Studio for SysML/CML integration

- A mapping between CML constructs and SysML will be made
- A link to Artisan Studio will be made, enabling round-tripping of CML/SysML
- This could potentially be extended to VDM

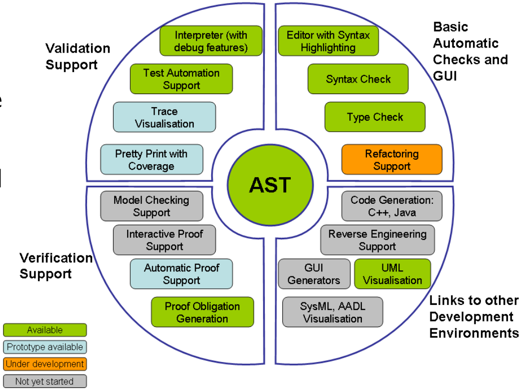

Validation Support
Interpreter (with debug features)
Test Automation Support
Trace Visualisation
Pretty Print with Coverage
Basic Automatic Checks and GUI
Editor with Syntax Highlighting
Syntax Check
Type Check
Refactoring Support
AST
Verification Support
Model Checking Support
Interactive Proof Support
Automatic Proof Support
Proof Obligation Generation
Links to other Development Environments
Code Generation: C++, Java
Reverse Engineering Support
GUI Generators
UML Visualisation
SysML, AADL Visualisation
Available
Prototype available
Under development
Not yet started

Potential Contribution from COMPASS

COMPASS plug-ins

Link to RT-Tester tool for test automation

- For dynamic analysis a link will be created to the RT-Tester tool
- This link could be utilised to support dynamic analysis of VDM models



Validation Support
Interpreter (with debug features)
Test Automation Support
Trace Visualisation
Pretty Print with Coverage
Basic Automatic Checks and GUI
Editor with Syntax Highlighting
Syntax Check
Type Check
Refactoring Support
AST
Verification Support
Model Checking Support
Interactive Proof Support
Automatic Proof Support
Proof Obligation Generation
Links to other Development Environments
Code Generation: C++, Java
Reverse Engineering Support
GUI Generators
UML Visualisation
SysML, AADL Visualisation
Available
Prototype available
Under development
Not yet started

Concluding remarks

- The Overture platform is capable of extending beyond the original goal
- Reuse of existing components can be achieved
- New development potentially contributes back to the existing platform
- The key enabling entity for reuse in Overture is the common AST and visitors