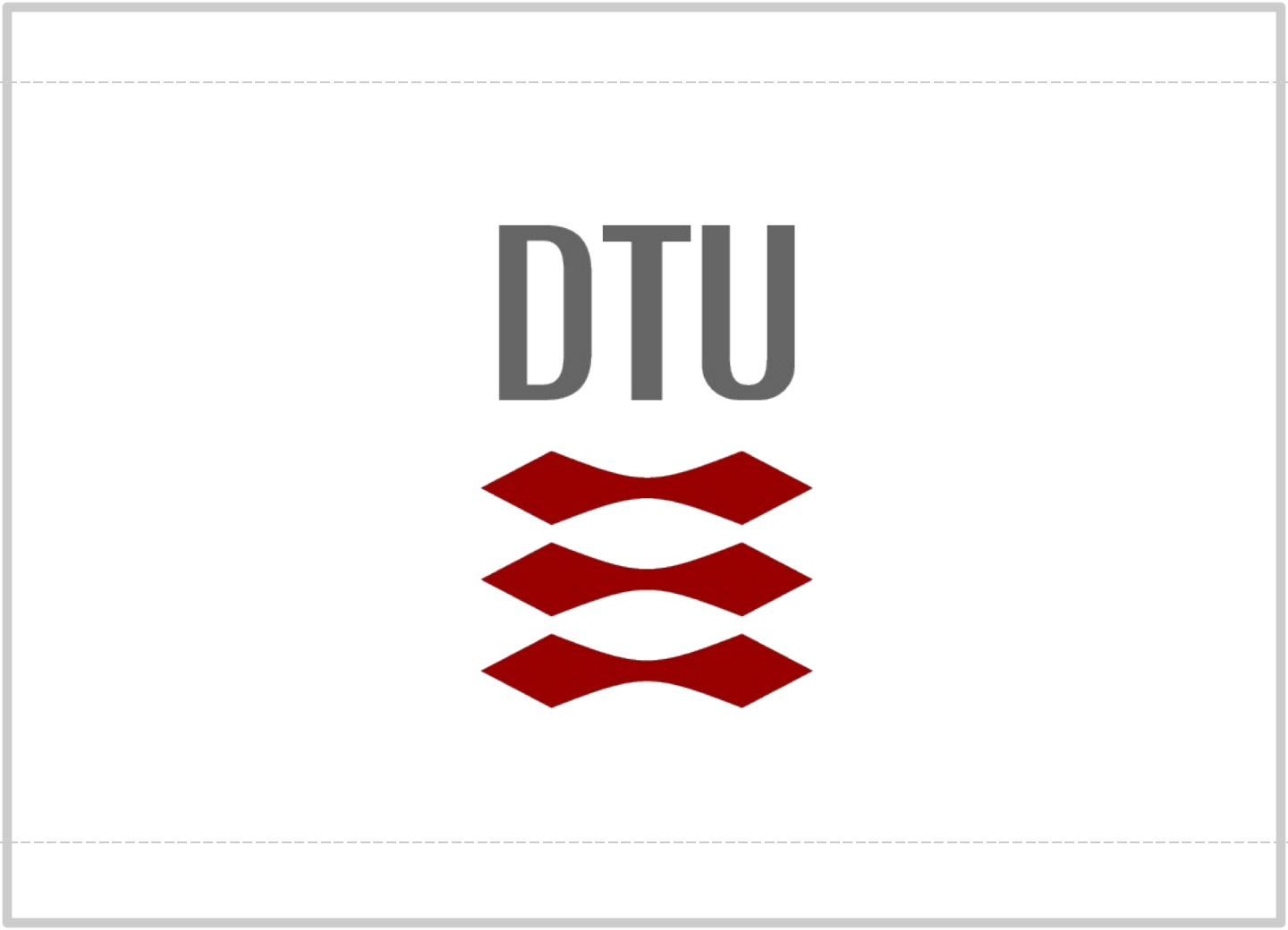

DTU

Designing a Flexible Kernel Providing VDM++ Support for Eclipse

M. Sc. Thesis Project at Technical University of Denmark

Title: Development of an Overture/VDM++ Tool Set for Eclipse

by

Jacob Porsborg Nielsen and Jens Kielsgaard Hansen

Supervisors: Anne Haxthausen and Hans Bruun

External Supervisor: Peter Gorm Larsen

January 25th to August 15th 2005

Technical University of Denmark

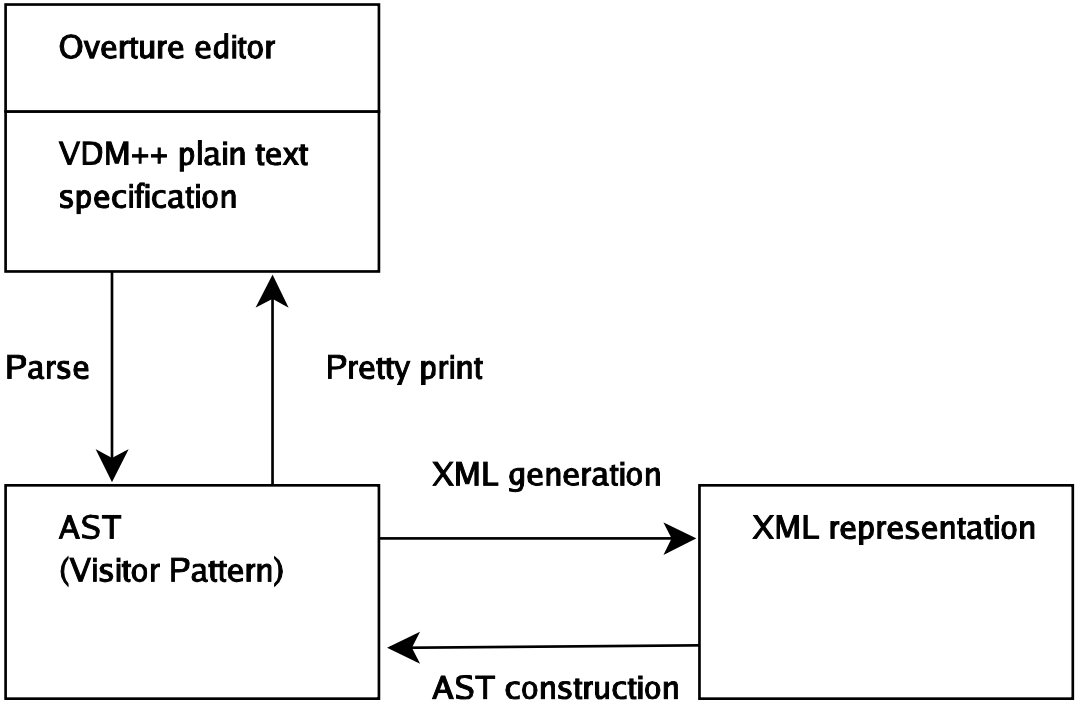
Technical Overview

Role of the Kernel

- Support development in the language OML/VDM++
- Integrate with Eclipse (IDE)
- Parse OML specifications and build Abstract Syntax Tree (AST)
  - Kernel is to do syntax check, not check of static semantics
- Export and Import AST to/from XML
- Provide extension points such that new functionality can be added as independent plug-ins (e.g. static semantics check or code generation)

Technical Overview

Role of the kernel - illustration



DTU

Structure of the Kernel

Tools used in the different parts

- In general: Environment Eclipse, Java 1.5.0
- Parser: Generated using ANTLR (parser is recursive decent)
- AST-classes: Constructed by hand
- XML-Schema: Created with XML Spy
- Export to XML: JDOM
- Import from XML: JDOM
- Export to textual form from AST: Visitor (constructed by hand)
- Outline view in Eclipse: Visitor (constructed by hand)

Eclipse

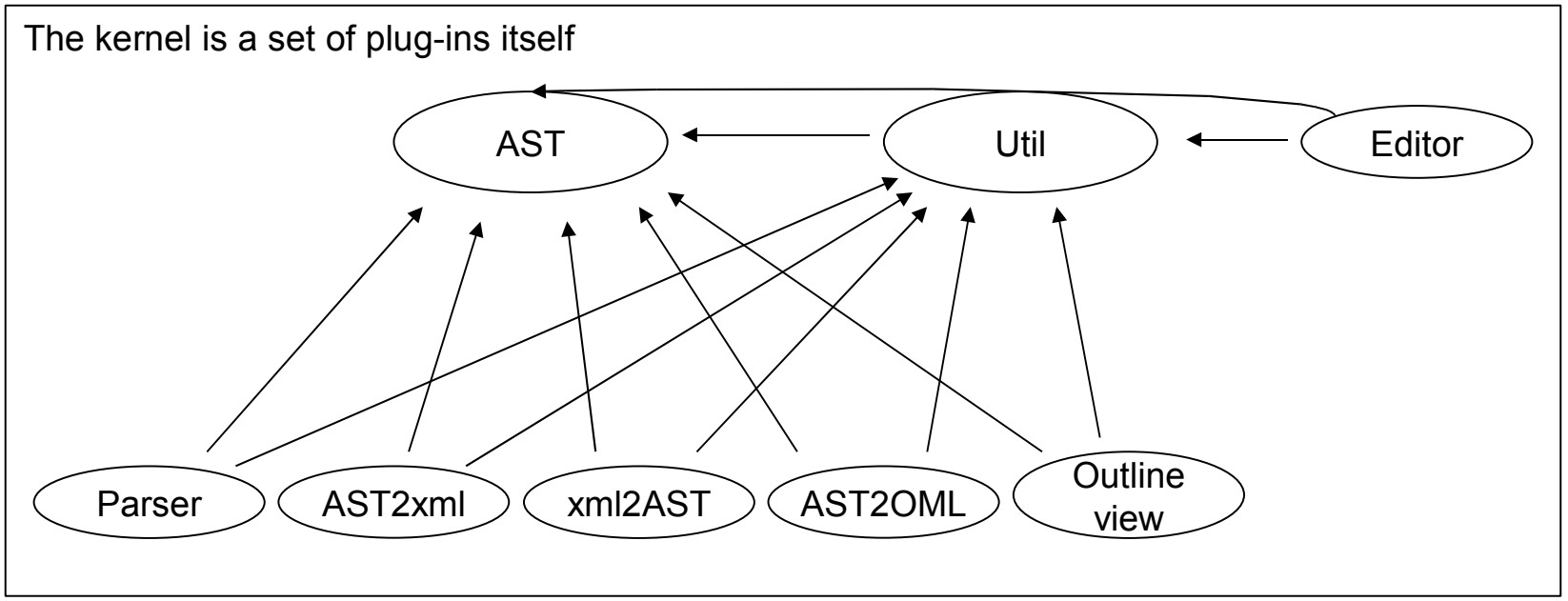
Benefits of using the Eclipse framework

- Eclipse is:
  - A general framework for tools
  - Open source
  - Well documented

- Useful features of Eclipse for the kernel
  - Facilities to build editors, views and perspectives
  - Advanced features (e.g. CVS integration)
  - Concept of plug-ins and extension points

  - Powerful Java IDE used in the kernel development
  - Concept of releases – easy to build, package, and distribute a release

Plug-in Structure of the Kernel



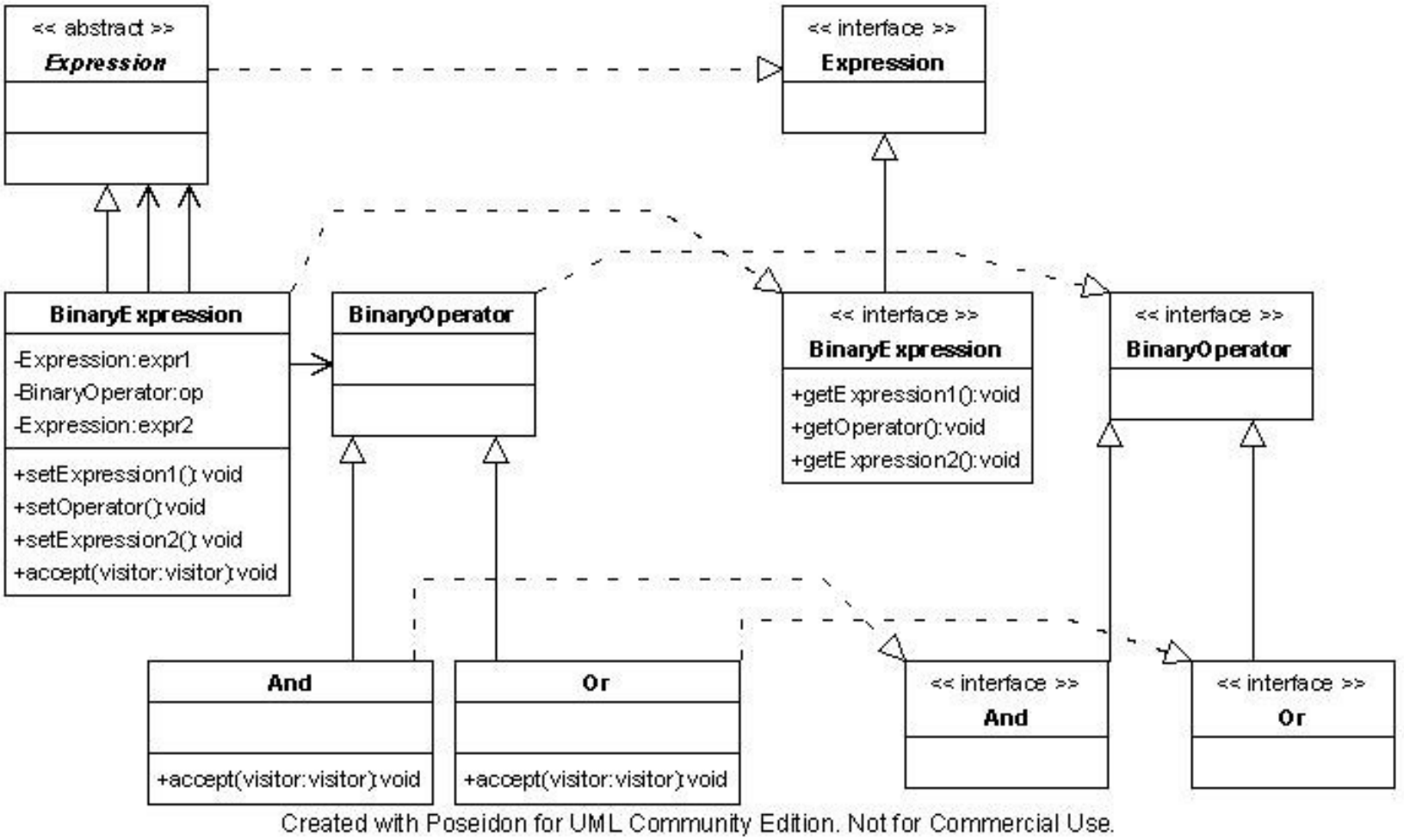
- Many plug-ins use the *ast* or the *internal.ast* packages offered by the AST plug-in
- A plug-in can add functionality by extending an extension point defined by another plug-in

Extension Points

Nature of extension points

- Editor defines extension point to plug-ins to be added. The available extension point is defined in an xml-file
- Util defines interface, that the extending plug-in should implement
- A plug-in extends an extension point, by implementing the interface. It shows it's existence to Eclipse by defining the extension in an xml-file.

Abstract Syntax Tree (AST) Structure



Abstract Syntax Tree (AST) Structure

Summary

- Plug-ins can operate on either the classes or the interfaces depending on whether they need write access to the AST nodes or not
- Positions and comments are stored in the AST nodes
- The AST interfaces (and classes) support visitor design pattern
- Plug-ins can store information (e.g. used for type checking) in the AST nodes.

Extending the Kernel with new Plug-ins

Case: Adding a plug-in providing type checking

- Obtain the M.Sc.thesis – it describes the design and implementation of the kernel. A chapter is dedicated to describe how to extend the kernel
- Create a new Eclipse plug-in project, make it depend on the AST plug-in and find a suitable extension point to extend.
- Analyse whether the plug-in will have to modify or add information to the AST nodes. A type checker will need this, - therefore it will need to operate on the AST classes (*"internal.ast"* package)
- Create a class to represent the information to store. It will have to implement the interface *"ASTAdditionalInformation"*
- Find a suitable visitor or a visitor interface to extend/implement
- Create the action code. The plug-in can traverse the AST as it pleases.

Status of the Kernel Development

What has this project achieved?

- Basic Eclipse based editor created. With syntax highlighting of OML/VDM++ code.
- Extension points that enables additional plug-ins to provide new facilities and operate on the AST
- Parser implemented, builds AST's (no static semantics/type checking)
- Test cases written, that reaches all branches of the parser
- XML: xml-schema created, xml export, and import from xml implemented. Xml-files are validated with the xml-schema
- Eclipse outline view/visitor implemented
- AST can be written to a file/an editor as plain OML code.

Distribution

How to obtain the kernel?

- Final release from this M. Sc. project (from August 15th) at:
  http://www.overturetool.org
- Source code will be available (from August 15th) at
  http://sourceforge.net/projects/overture/
- Current release (last beta-release) available now, see:
  http://www.student.dtu.dk/~s001842/overturetool/install.txt

Technical University of Denmark