HOL-OCL
A Formal Proof Environment for UML/OCL

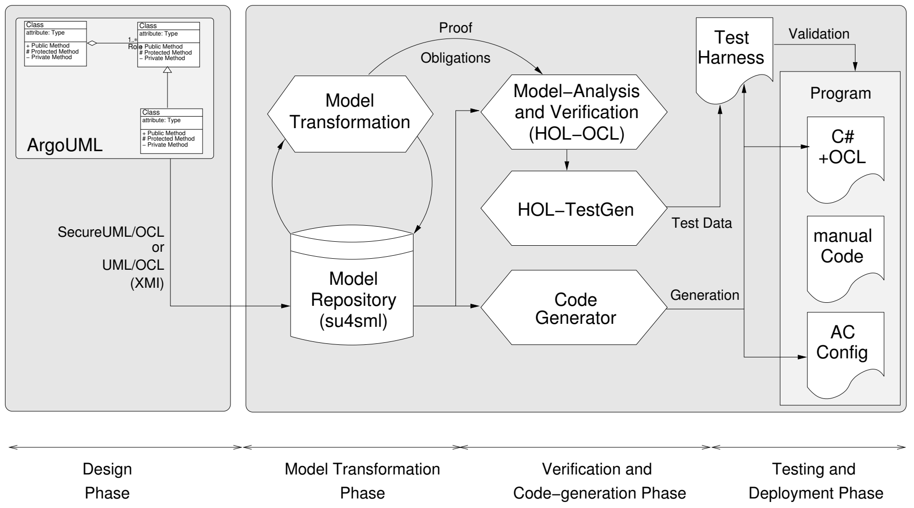
Achim D. Brucker[1] Burkhart Wolff[2]

[1]SAP Research, Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany
achim.brucker@sap.com

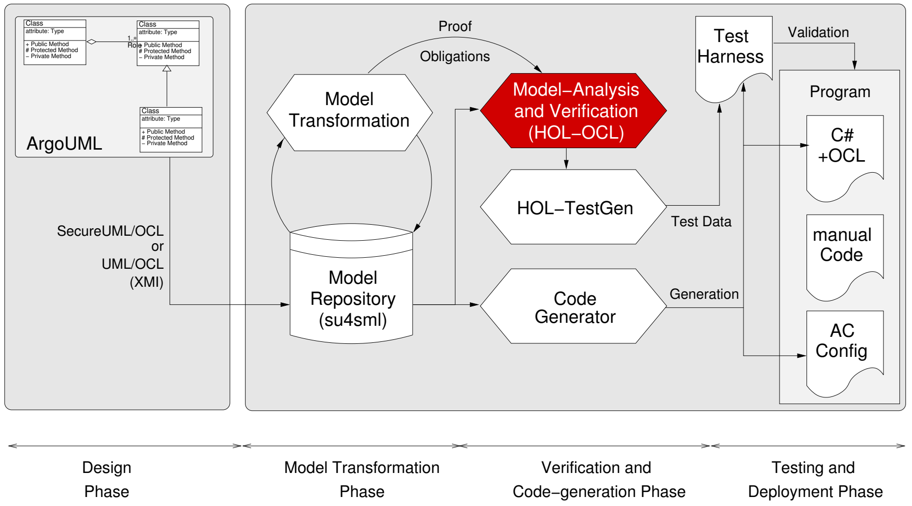
[2]Universität des Saarlandes, 66041 Saarbrücken, Germany
wolff@wjpserver.cs.uni-sb.de

ETAPS 2008
Budapest, 31st March 2008

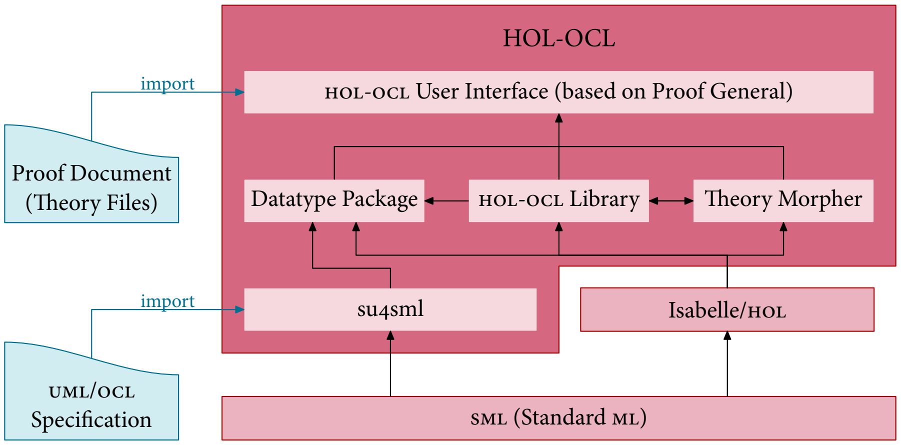
The HOL-OCL Vision:
Tool Supported Formal Methods for (Model-driven) Software Development


ArgoUML
SecureUML/OCL or UML/OCL (XMI)
Proof Obligations
Model Transformation
Model Repository (su4sml)
Model–Analysis and Verification (HOL–OCL)
HOL–TestGen
Code Generator
Test Harness
Validation
Test Data
Generation
Program
C# +OCL
manual Code
AC Config
Design Phase
Model Transformation Phase
Verification and Code–generation Phase
Testing and Deployment Phase

The HOL-OCL Vision:
Tool Supported Formal Methods for (Model-driven) Software Development


ArgoUML
SecureUML/OCL or UML/OCL (XMI)
Proof Obligations
Model Transformation
Model Repository (su4sml)
Model–Analysis and Verification (HOL–OCL)
HOL–TestGen
Code Generator
Test Harness
Validation
Test Data
Generation
Program
C# +OCL
manual Code
AC Config
Design Phase
Model Transformation Phase
Verification and Code–generation Phase
Testing and Deployment Phase

The HOL-OCL Architecture


HOL-OCL
import
HOL-OCL User Interface (based on Proof General)
Proof Document (Theory Files)
Datatype Package
HOL-OCL Library
Theory Morpher
import
su4sml
Isabelle/HOL
UML/OCL Specification
SML (Standard ML)

Conclusion

- We presented HOL-OCL providing:
  - a formal, machine-checked semantics for OO specifications,
  - an interactive proof environment for OO specifications,
  - publicly available: `http://www.brucker.ch/projects/hol-ocl/`,
  - next (major) release planned in July 2008.
- HOL-OCL is integrated into a toolchain providing:
  - code generators,
  - a transformation framework (including PO generation),
  - support for SecureUML.

Ongoing and Future Work

- Ongoing work includes the development of support for:
  - well-formedness-checking,
  - proof-obligation generation (Liskov, Refinement,),
  - consistency checking,
  - Hoare-style program verification,
  - better proof automation.
- Future works could include the development for
  - integrating OCL validation tools, e.g., USE,
  - test-case generation (i.e., integrating HOL-TestGen),
  - supporting SecureUML.
  - ….