

HOL-OCL

Tool Demonstration

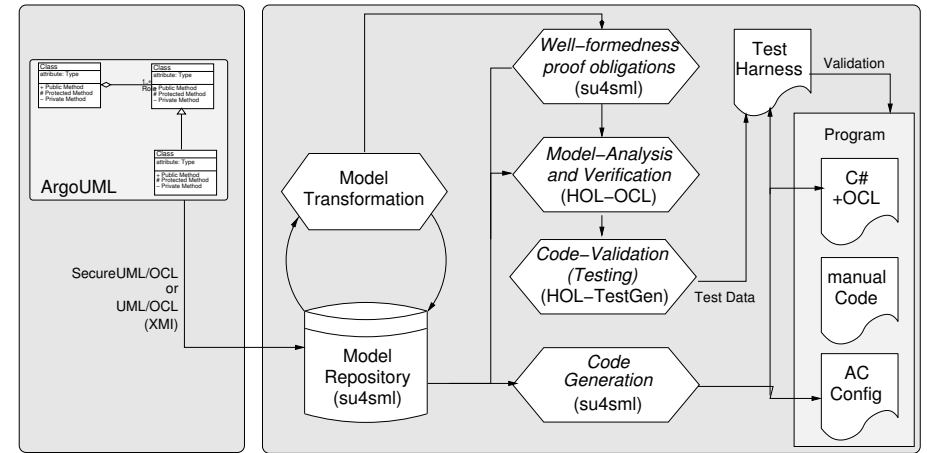
Achim D. Brucker

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

OCL Workshop at the UML/ModelS Conferences 2008
 Toulouse, 30th September 2008

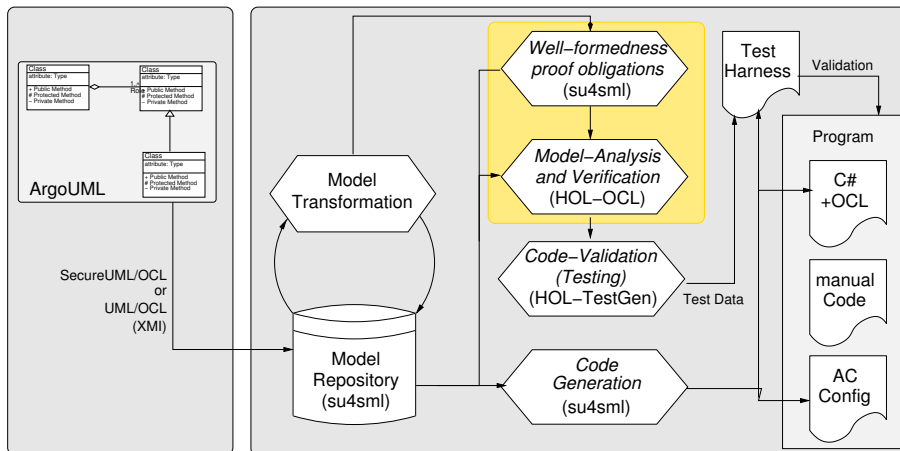
The HOL-OCL Vision:

A Tool Supported Formal Model-driven Engineering Process with Tool-support



The HOL-OCL Vision:

A Tool Supported Formal Model-driven Engineering Process with Tool-support



Tool Demo

Tool-Demo

```

3 emacs@nakagawa.inf.ethz.ch
File Edit Options Buffers Tools Preview LaTeX Command X-Symbol Help
State Context Goal Retract Undo Next Use Goto Q.E.D. Find Command Stop Restart Info Help
\begin{small}
\listinputlist[style=ocl][company.ocl]
\end{small}

\begin{figure}
\centering
\includegraphics[scale=.6]{company}
\caption{A company Class Diagramm\label{fig:company_classdiag}}
\end{figure}

*)
load_xml "company_ocl.xml"

\thm Company.Person.inv.inv_19_def

lemma "h Company.Person.inv self -> Company.Person.inv.inv_19 self"
apply(simp add: Company.Person.inv_def)
apply(auto)

-1- ** company.thy 80% (45.14) SVN-27978 (Isar script [PDFLaTeX/F] MM XS:holocl/s Scripting)----6:35 2.39
\<sync>\thm Company.Person.inv.inv_19_def; \</sync>
Person.inv.inv_19 =
\self. \forall p2 \in OclAllInstances
self \cdot (\forall p1 \in OclAllInstances
self \cdot ((p1 \cdot \cdot p2) ->
(Company.Person.lastName p1 \cdot \cdot Company.Person.lastName p2)))

-1- **response All (6,101) (response)----6:35 2.39 Mail
    
```

Proof Obligations: Liskov's Substitution Principle

Liskov substitution principle

Let $q(x)$ be a property provable about objects x of type T . Then $q(y)$ should be true for objects y of type S where S is a subtype of T .

For constraint languages, like oo, this boils down to:

- *pre-conditions* of overridden methods must be *weaker*.
- *post-conditions* of overridden methods must be *stronger*.

Which can formally expressed as implication:

- Weakening the pre-condition:

$$op_{pre} \rightarrow op_{pre}^{sub}$$

- Weakening the pre-condition:

$$op_{post}^{sub} \rightarrow op_{post}$$

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 November 2008.
- HOL-OCL is integrated into a toolchain providing:
 - code generators,
 - a transformation framework (including PO generation),
 - support for SecureUML via model transformations.

Methodology

A tool-supported methodology should

- integrate into existing toolchains and processes,
- provide a unified approach, integrating ,
 - syntactic requirements (well-formedness checks),
 - generation of semantics requirements (proof obligations),
 - means for **verification** (proving) or **validation**, and of course
- all phases should be supported by tools.

Example

A package-based object-oriented refinement methodology.

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 natively.
 -

The next Challenge for OCL Tools

- **State of the art:**
 - There are a lot of good OCL tools, which work in isolation.
 - There is no “one sizes fits all” OCL tool.
 - There is no (integrated) development process supporting.
- **Observation:** Successful specification languages comprise:
 - tools that work together.
 - one or more development processes that are well supported by tools.
- **Conclusion:** We, as the OCL Community, should
 - combining the strengths of different OCL tools.
 - provide methodologies (development processes) on top of OCL.