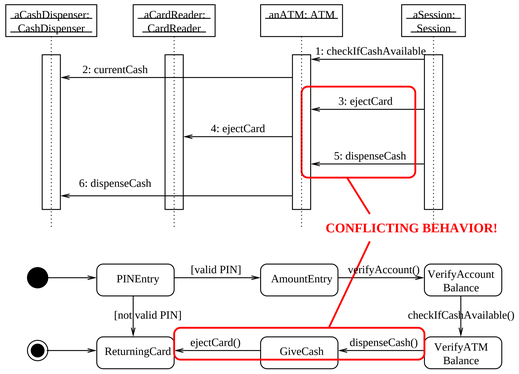
A Tool for Automatic UML Model Consistency Checking: MCC

Jocelyn Simmonds (University of Toronto/University of Chile) and M. Cecilia Bastarrica (University of Chile)

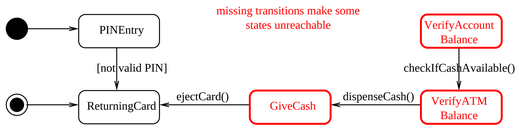
The need for automated verification for UML

- UML standard allows for inconsistencies between diagrams
- Manual verification - tedious and error-prone
- Goal: Automated Verification
- Must be scalable w.r.t.:
  - Model size
  - Number of UML elements considered
- Must be extensible:
  - add new UML elements
  - add new consistency checks

Motivational Examples



INCOMPLETE BEHAVIOR SPECIFICATION



Related Work

Related work w.r.t. to UML consistency

	Diagrams studied	Formalism used
Engels et al. (EHK01)	Protocol statecharts (w.r.t. inheritance)	CSP
Ehrig and Tsiolakis (ET00)	Class and sequence diagrams	Graphs
Schfer et al. (SKM01)	Collaboration and statecharts	PROMELA models

Related work w.r.t. to existing tools

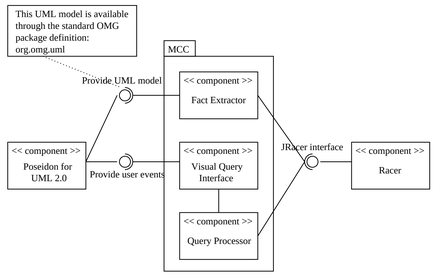
	Diagrams included	Formalism used
Rational Rose (IBM04)	Class diagrams	NA
Rose Model Checker (MM04)	Class diagrams	NA
Cortes et al. (CFd04)	Component diagrams	UML Profile

Our approach

- Formalise UML using Description Logics (DL)
- Offer a uniform verification for the widest possible range of consistency problems
- Usable framework - no interaction with the underlying formalism
- Extensible framework - must easily incorporate changes that originate from:
  - the UML specification
  - changing the subset of supported UML modeling elements
  - the addition of new consistency checks

Framework Architecture

MCC: Model Consistency Checker - our framework (`http://www.dcc.uchile.cl/~jsimmond`)
Poseidon SE 3.2: UML CASE tool (`http://www.gentleware.com`)
RacerPro v1.8: DL reasoning engine (`http://www.racer-systems.com`)



Consistency Checks

	Behavioral	Structural
Specification	invocable collaboration behavior consistency observable collaboration behavior inconsistency	dangling (type) reference inherited association inconsistency instance specification missing
Specification / Instance	**incompatible specification**	instance specification missing
Instance	invocable behavior conflict observable behavior conflict **incompatible behavior conflict**	disconnected model

Categories in **blue** have alredy been included into the framework

Why use Description Logics (DL)?

DL is a Knowledge Representation formalism. Concepts, Roles and Instances are used to represent knowledge.

- Theoretical Box (*Tbox*): set of axioms that defines concepts and roles
- Assertion Box (*Abox*): set of instances of concepts

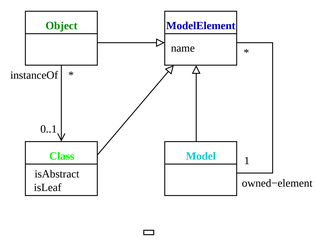
PROS

- decidable two-variable fragment of first order logic
- sufficiently expressive to express UML (CCDGL02)
- existence of optimised DL reasoning engines
- reasoning engines offer query languages

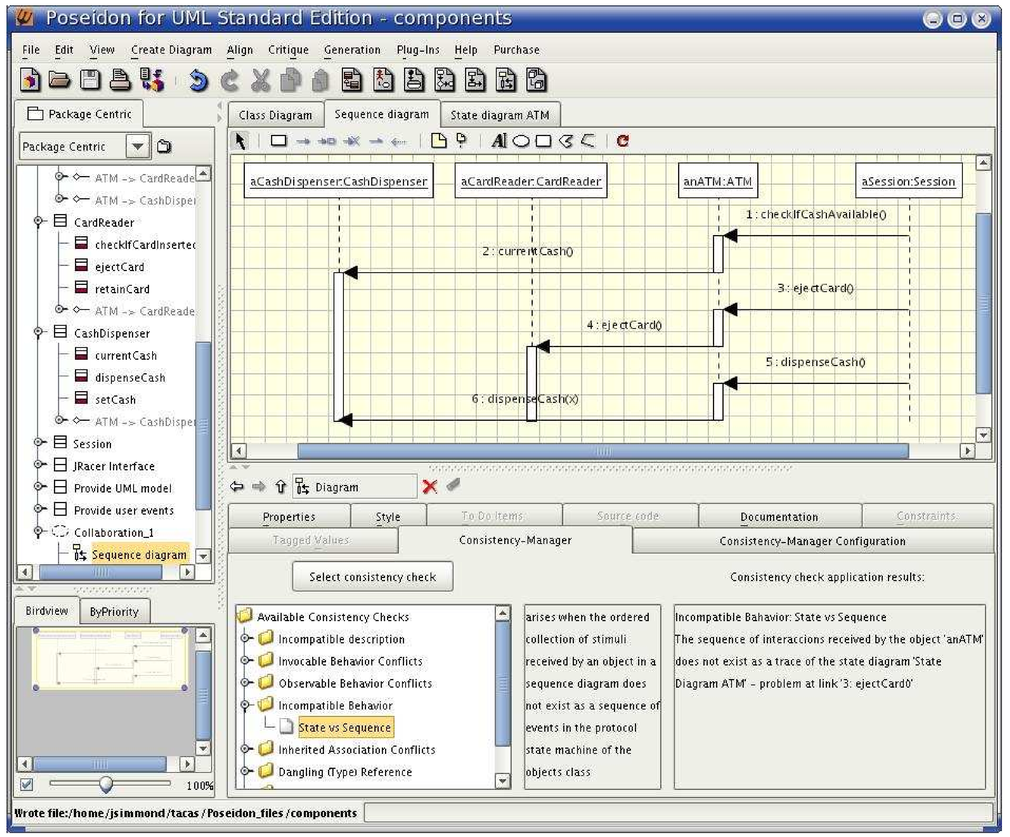
CONS

- reasoning on UML models is EXPTIME-hard (but the UML metamodel is bound to about 250 classes)

DL - UML mapping (*Tbox* example)



Motivational Example revisited



Adding New Checks

- Checks implemented as nRQL queries over the ABox
- New checks easy to add:
  - Implement the Check interface
  - Add to the XML check configuration file

UML standard changes

- Each UML element type has its own translation method
- Check and modify the TBox definition
- Check and modify the element's translation method

Adding new UML elements to the framework

- Add concepts and roles to the TBox
- Add translation methods

Limitations

- Can only reason w.r.t. horizontal evolution
- Both Poseidon and Racer are licensed products

Current Status

Aspect	Status
Checks	More on the way
UML 2.0	Sequence diagrams needs updating
Scalability	Currently testing larger models
Interface	updated to Poseidon 3.2
Reasoning Engine	updated to RecerPro 1.8

Available at: `http://www.dcc.uchile.cl/~jsimmond`

REFERENCES

[CCDGL02]Andrea Cali, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. A formal framework for reasoning on uml class diagrams. In *Proc. of the 13th Int. Sym. on Methodologies for Intelligent Systems (ISMIS 2002)*, volume 2366 of *Lecture Notes in Computer Science*, pages 503–513. Springer, 2002.

[CFd04]Mariela Cortés, Marcus Fontoura, and Carlos de Lucena. Using Refactoring and Unification Rules to Assist Framework Evolution. *UPGRADE*, 5(2):49–55, April 2004.

[EHK01]Gregor Engels, Reiko Heckel, and Jochen Malte Küster. Rule-based specification of behavioral consistency based on the UML meta-model. In Martin Gogolla and Cris Kobryn, editors, *Proc. Int'l Conf. UML 2001 - The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, number 2185 in Lecture Notes in Computer Science, pages 272–286. Springer-Verlag, October 2001. Toronto, Canada.

[ET00]H. Ehrig and A. Tsiolakis. Consistency analysis of UML class and sequence diagrams using attributed graph grammars. In H. Ehrig and G. Taentzer, editors, *ETAPS 2000 workshop on graph transformation systems*, pages 77–86, March 2000.

[IBM04]IBM. Rational Software, October 2004. `http://www-306.ibm.com/software/rational/`.

[MM04]Michael Moors. Rose Model Checker, October 2004. `http://www.rationalrose.com/modelchecker/index.htm`.

[SKM01]T. Schfer, A. Knapp, , and S. Merz. Model Checking UML State Machines and Collaborations. *In Electronic Notes in Theoretical Computer Science*, 47:1–13, 2001.

This poster was prepared with Brian Wolven's Poster LaTeX macros v2.1.