Seminar: Specification and Verification of Object-oriented Software

# The KeY Tool

developed by: W. Ahrendt, T. Baar, B. Beckert, R. Bubel, M. Giese, R. Hähnle, W. Menzel, W. Mostowski, A. Roth, S. Schlager, P.H. Schmitt, and others

> Achim D. Brucker Information Security, ETH Zürich, Switzerland http://www.brucker.ch/ brucker@inf.ethz.ch

> > January 21, 2004 Wintersemester 2003/04

The KeY Tool

Motivation

# Why use Formal Methods in Software

#### Development

There are many reasons for using formal methods:

- ▶ safety critical applications, e.g. flight or railway control.
- ▶ security critical applications, e.g. access control.
- ▶ financial reasons (e.g. warranty), e.g. embedded devices.
- ▶ legal reasons, e.g. certifications.

Many successful applications of formal methods proof their success!

- Software systems are
  - getting more and more complex.
  - used in safety and security critical applications.
- We think:
  - Complex software systems require a precise specification of its architecture and components.
  - Semi-formal methods (like UML diagrams) are not strong enough.

Specification should be useful, i.e. not only documentation!

Achim D. Brucker

Seminar: Specification and Verification of Object-oriented Software

The KeY Tool

Motivation

#### 3

# Why Formal Methods are not widely accepted in software industry?

- Only a few formal methods address industrial needs:
  - support for object-oriented programming.
  - highly automatic (?).
  - integration in standard CASE tools and processes.
- Formal methods people and industrial software developer are often speaking different languages.
- The KeY tool tackles these challenges by using a industrial accepted specification languages (UML/OCL) and by providing a strong integration into standard CASE tools.

The KeY Tool Road Map 4	The KeY Tool Background 5
Road Map	UML
Motivation	diagrammatic OO modeling language       context Account:makeDeposit(amount:Real):Boolean pre: amount >= 0
► Foundations: UML/OCL and JavaCard	<ul> <li>many diagram types, e.g.</li> <li>class diagrams (static)</li> <li>Customer 1 Direction 1.99 Account</li> <li>Identification/String owner (Direction account) - balance: Real</li> </ul>
► The KeY Tool	<ul> <li>state charts (dynamic)</li> <li>getidentification():String</li> <li>getidentification():String</li> <li>setAddress():String</li> <li>setAddress():String</li> <li>setAddress(:String</li> <li>setAddress(:String</li></ul>
► Conclusion	semantics currently standardized by the OMG           CreditAccount           -creditReal
	<ul> <li>wide use in SE-Tools         <ul> <li>(ArgoUML, Rational Rose,</li> <li>Together,)</li> </ul> </li> </ul>
Achim D. Brucker       Seminar: Specification and Verification of Object-oriented Software         The KeY Tool       Background       6	Achim D. Brucker Seminar: Specification and Verification of Object-oriented Software The KeY Tool Background 7
Are UML diagrams enough to specify OO	OCL
systems formally?	designed for annotating UML di- agrams
The short answer:	based on logic and set theory context Account::makeDeposit(amount:Real):Boolean pre: amount >= 0 post: balance = balance@pre + amount
<ul> <li>UML diagrams are not powerful enough for supporting formal reasoning over specifications.</li> </ul>	<ul> <li>in the context of class-diagrams:</li> <li>preconditions</li> <li>1.99 Account accounts-balance:Real</li> </ul>
	<ul> <li>postconditions</li> <li><u>id:Integer</u></li> <li><u>id:Integer</u></li> <li><u>getBalance():Real</u></li> <li><u>hmakeDeposit(amount:Real):Boolean</u></li> </ul>
<ul> <li>The long answer:</li> <li>We want to be able to</li> </ul>	- invariants
- verify (proof) properties	will also be used for other dia- gram types
<ul> <li>refine specifications</li> </ul>	▶ part of the UML standard

context Account inv:

context Account inv:

pre: amount >= 0

OCL keywords

Achim D. Brucker

The KeY Tool

and Account.owner->size = 1

Meaning of the method makeDeposit():

post: balance = balance@pre + amount

OCL — A Simple Examples

Account.allInstances->forAll(a1,a2 | a1.id = a2.id implies a1 = a2)

▶ Properties of the class diagram can be described, e.g. multiplities:

"Uniqueness" constraint for the class Account:

context Account::makeDeposit(amount:Real):Boolean

8

#### 9

11

#### JavaCard

JavaCard is a proper subset of Java, excluding among other:

- threads
- dynamic class loading
- not all data types (e.g., float, double)
- restricted I/O (no GUI)

JavaCard supports basic object oriented features:

- state depends on local vars & attribute values of existing objects
- evaluation of expressions can have side effects
- ▶ int, short, arrays, reference types (aliasing)
- exceptions
- ▶ initialization of objects

Achim D. Brucker

Seminar: Specification and Verification of Object-oriented Software

The KeY Tool

The KeY Tool

#### **Excursion:** Formal Challenges

Only few formal methods are specialized for analyzing object oriented specifications.

- Problems and open questions:
  - object equality and aliasing
  - embedding of object structures into logics
  - referencing and dereferencing, including "null" references
  - dynamic binding
  - polymorphism
  - . . .
- Turning UML/OCL into a formal method:
  - semantics for OCL only given in a semi-formal way
  - OCL expressions are only meaningful together with the underlying UML model
  - no proof calculi for OCL
  - no refinement notions for OCL

- . . .

Achim D. Brucker

The KeY Tool

Seminar: Specification and Verification of Object-oriented Software

Syntax from UML model

## The KeY Tool — Overview

Combining class diagrams and OCL results in a data-oriented formal

specification language similar to Z, VDM, B,...

KeY is a CASE tool extension for formal specifications, supporting

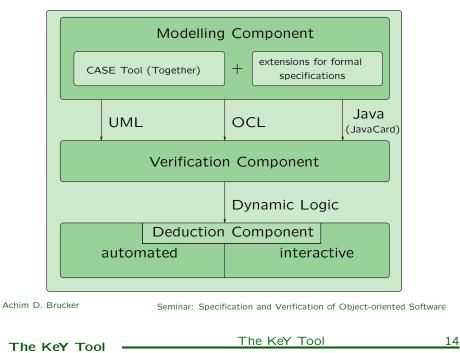
- ▶ the *creation* of constraints (design patters)
- the formal analysis of constraints
- ▶ the *verification* of implementations

These features are highly integrated into an commercial CASE tool, aiming for the

- development of industrial software without special needs for security.
- development of security critical software.
- use in education and training.

10

#### The Architecture of the KeY tool



#### The Verification Component

- Translates OCL into dynamic logic (first order logic with modal operators)
- Generation of proof goals in first order logic, e.g. the invariant of a subclass implies the invariant of the superclass (Liskov).
- Generation of proof goals in dynamic logic, e.g. the implementation honors a given invariant.

#### context Account inv:

Account.allInstances->forAll(a1,a2 | a1.id = a2.id implies a1 = a2)

Can be translated into typed first order logic (using OO syntax):

 $(\forall a1, a2 : Account)(a1.id \doteq a2.id \rightarrow a1 \doteq a2)$ 

### The Modelling Component

	C X Designer	(# II X
2 % CP 10 % 15 31	Ba   ++ + 12   EE   500	
A 14 DI 14	h4 88 <default></default>	
Y parker     Basedefault=     Basedefault=     Basedefault=     Basedefault=     Basedefault=     Basedefault=     Basedefault=	R Start PayCard ChargeUI	<b>A</b>
B D IssueCardU B D PapCard B D PapCardunior B D Rant	timit.int balance:int chargeUI +initGUI:void	
	2 + charc New Operation Intert	
	java.lang.Exception Rename F2 UI	
	CardException Search for Unages	
	+CardException	
inspector: charge 🌲	= x +char Clone Ctrl+Shift+C UI	
Javadoc HTMLdoc Requirements	Parte Onlyw id	
Properties Hyperlink Descrip	on Delate Delate	
Name Value	Editor	₩ D x
name tharge stereotype	▲ 면····································	
alian	25 * Spreconditions = Abdrace	
parameters # int amount	26 * @postcondition that	
associates	Filmer in Andre Terre	
return value void	E Cenerate Sequence Diagram	
throws 🕸	128 D public void charge( Hyperlink To ,	
visibility public	29 if (this.balar Tools +	
etatic 🗌	And the second s	MathodSpac
final 🗆	30 Enis.unsuc	vioral Subtyping
abstract 🔲	31 Jalsa J	investimatiant
synchronized		esPostCondition
native 🗌		espostCondition [P]
Press Ctrl+Alt+i to finish editing and close in	pettor [ 2 minute reprint a manual and a minute reprint a manual and a minute reprint a manual and a minute reprint a minute	

The KeY Tool

The KeY Tool

# Generated Proof Obligations

▶ Proof obligation (OCL):

context Account inv: Account.allInstances->forAll(a1,a2 | a1.id = a2.id implies a1 = a2)

is an invariant of the method getBalance()

Claim to be proven (dynamic logic):

 $(a1.id \doteq a2.id \rightarrow a1 \doteq a2) \vdash \langle getBalance(); \rangle (a1.id \doteq a2.id \rightarrow a1 \doteq a2)$ 

15

The KeY Tool —	The Key Tool	16	The KeY Tool	The KeY Tool
The	Deduction Component		The	e KeY Features
▶ based on a sequer	nce calculus for dynamic logic		► create formal requirem	ents specifications in OCL
▶ automatic and int	eractive proof support		▶ translate OCL requiren	nents into correctness assertions
► counter example g	generation		▶ render OCL into natura	al language
In the state of t	novice and a constraining of the constraint of t		► check correctness of a	specification
Apply Hewistics Proofabilgations	Cal Fack Cal Fack Current Cal		► check correctness of a	implementation
User Constrai Proof Ga Proof Tree imp_right	air (self.balance) pectolea (self.juniorLim pectolt mit)		▶ generates counter exan	nples for invalid assertions

Seminar: Specification and Verification of Object-oriented Software

The KeY Tool

relax ine

hide left case\_distin

cut\_direct\_l close\_poal\_ante to clipboard Create abbrevia

(self.balance)

Case 2

## **Case Studies**

Kgy Integrated Deductive Software Design: Ready

The KeY tool was successfully applied to several larger case studies, e.g.:

- **Refinement:** The Java Collection Framework (JFC) Abstract specification of standard data structures (e.g. lists, sets) are stepwise refined to a concrete implementation.
- **Security:** access control (PAM authentication with iButton) Analysis of the state diagram of a JavaCard application used as authentication token.
- Safety: computation of speed restrictions for Deutsche Bahn AG (railway)

A reference implementation computing a "speed book" was specified and analyzed (about 80 Java classes).

Achim D. Brucker

The KeY Tool

- ▶ Integration of other formal techniques, e.g. model checking
- ▶ Integrate other UML diagram types, e.g. state charts
- ▶ Further applications, e.g. test case generation
- Build a formal, object-oriented software development process

Achim D. Brucker

Seminar: Specification and Verification of Object-oriented Software

17

19

Conclusion The KeY Tool

### **Further Work**

▶ Support challenging Java Features (long term), e.g. threads, floating point arithmetic, dynamic class loading, GUI specification

in logic

22

#### Summary

► UML/OCL allows one to

The KeY Tool

- formally specify object-oriented data models in prec-/postcondition style.
- introduce formal methods in a lightweight way.
- use an industry accepted OMG standard. This will hopefully lead to more acceptance (and hence tool support).
- ▶ The KeY tool allows one to
  - write formal OCL/UML specifications.
  - proof properties on the specification level.
  - proof properties on the implementation level.
  - proof that a implementation fulfills its specification.

And all that providing an easy to use, first class integration into a widely accepted CASE tool!

The KeY tools fulfills its main goal, making formal methods usable by the object-oriented software industry.

Achim D. Brucker

The KeY Tool

Dynamic Logic

23

Seminar: Specification and Verification of Object-oriented Software

#### A Program Logic: Dynamic Logic

- Syntax
  - typed first order logic
  - program logic
  - modal operators [p] and  $\langle p \rangle$
- Semantics
  - operators are evaluated in the terminating state of  $\boldsymbol{p}$
  - -[p]F: if p terminates, then F holds (partial correctness)
  - $-\langle p \rangle F$ : p terminates and F holds (total correctness)

Critics and Limitations

Nevertheless, there are some weak points of the KeY architecture:

- ▶ tactlets (tactics) are not shown to be correct.
- the semantic definitions (for OCL and JavaCard) are done in a axiomatic way and hence there is no guarantee for the consistency of the system.
- the translation from OCL into dynamic logic is quite naïve and doesn't honor the OCL standard. As such, KeY uses a "Dynamic Logic with OCL Syntax" instead of standard compliant OCL to annotate UML class diagrams.
- there is no guarantee that the axiomatized JavaCard semantics is compliant to Java standard.

Seminar: Specification and Verification of Object-oriented Software

The KeY Tool

# Appendix

## **Slides for Answering Questions**

if-then

while

 $pre, b \doteq \texttt{false} \vdash \langle q \rangle F$ 

Calculus: "if-then" and "while" rule

(simplified)

 $pre \vdash \langle \text{if } b \text{ then } \{p\} \text{ else}\{q\} \rangle F$ 

 $pre \vdash \langle \text{if } b \text{ then } \{p\} \text{ while}(b)\{p\} \rangle F$ 

 $pre \vdash \langle while(b) \{p\} \} \rangle F$ 

24

27

#### **Related Work**

- Tools supporting OCL can be roughly divided into:
  - Runtime checking of OCL constraints, e.g. based on the Dresden OCL compiler [7].
  - Model simulation and validation, e.g. the USE tool [9].
  - Proof environments, namely HOL-OCL [5]; which is implemented as a shallow embedding of OCL into Isabelle/HOL. HOL-OCL tries to strictly follow the OCL 2.0 standard, e.g. including a three valued logic.

► Formalizing the Java semantics:

 $-\mu$ Java an embedding of a Java subset into Isabelle/HOL [8].

Achim D. Brucker	Seminar: Specification and Verification of Object-ori	ented Software
The KeY Tool -	References	26

#### References

[1] The KeY project, January 2004. http://www.key-project.org.

 $pre, b \doteq \texttt{true} \vdash \langle p \rangle F$ 

- [2] Wolfgang Ahrendt, Thomas Baar, Bernhard Beckert, Richard Bubel, Martin Giese, Reiner Hähnle, Wolfram Menzel, Wojciech Mostowski, Andreas Roth, Steffen Schlager, and Peter H. Schmitt. The key tool. Technical Report 2003-05, Chalmers, Göteborg University, 2003.
- [3] Wolfgang Ahrendt, Thomas Baar, Bernhard Beckert, Martin Giese, Elmar Habermalz, Reiner Hähnle, Wolfram Menzel, and Peter H. Schmitt. The KeY approach: Integrating object oriented design and formal verification. Technical Report 2000/4, University of Karlsruhe, Department of Computer Science, January 2000. http://i12www.ira.uka.de/~projekt/publicat.htm.
- [4] Achim D. Brucker and Burkhart Wolff. HOL-OCL: Experiences, consequences and design choices. In Jean-Marc Jézéquel, Heinrich Hussmann, and Stephen Cook, editors, UML 2002: Model Engineering, Concepts and Tools, number 2460 in Lecture Notes in Computer Science, pages 196–211. Springer-Verlag, Dresden, 2002. ISBN 3-540-44254-5.

http://www.brucker.ch/bibliography/abstract/brucker.ea-hol-ocl-2002.

[5] Achim D. Brucker and Burkhart Wolff. A proposal for a formal OCL semantics in Isabelle/HOL. In César Muñoz, Sophiène Tahar, and Víctor Carreño, editors, *Theorem Proving in Higher Order Logics*, number 2410 in Lecture Notes in Computer Science, pages 99–114. Springer-Verlag, Hampton, VA, USA, 2002.

Achim	D.	Brucker
-------	----	---------

#### Achim D. Brucker

Achim D. Brucker

The KeY Tool

ISBN 3-540-44039-9

Publishers, 2003.

Seminar: Specification and Verification of Object-oriented Software

References

[7] OCL Compiler Suite, 2001. http://dresden-ocl.sourceforge.net/.

http://www.brucker.ch/bibliography/abstract/brucker.ea-proposal-2002.

[8] Tobias Nipkow, David von Oheimb, and Cornelia Pusch. μJava: Embedding a programming language in a theorem prover. In Friedrich L. Bauer and Ralf Steinbrüggen, editors, Foundations of Secure Computation, volume 175 of NATO Science Series F: Computer and Systems Sciences, pages 117–144. IOS Press, 2000. http://isabelle.IN.TUM.de/Bali/papers/MOD99.html.

[6] Richard Bubel and Reiner Hähnle. Integration of informal and formal development

of object-oriented safty-critical software: A case study with the key system. In

Electronic Notes in Theoretical Computer Science, volume 80, Elsevier Science

[9] Mark Richters and Martin Gogolla. OCL - syntax, semantics and tools. In Tony Clark and Jos Warmer, editors, *Advances in Object Modelling with the OCL*, pages 43–69. Springer, Berlin, LNCS 2263, 2001.

The KeY Tool —	Table of Contents	28	The KeY Tool	Table of Contents	29
Why Formal Meth Road Map Road Map Background UML Are UML diagram OCL OCL — A Simple JavaCard The KeY Tool — Excursion: Forma The Architecture The Modelling Co Generated Proof O The Verification O Generated Proof O The Deduction Co The Deduction Co The KeY Features Case Studies Conclusion Further Work Critics and Limita Summary	Methods in Software Development hods are not widely accepted in software industry? hs enough to specify OO systems formally? e Examples • Overview al Challenges of the KeY tool omponent Component Obligations omponent s	$ \begin{array}{c} 1\\ 1\\ 2\\ 3\\ 4\\ 4\\ 5\\ 5\\ 6\\ 7\\ 8\\ 9\\ 10\\ 10\\ 11\\ 12\\ 13\\ 14\\ 15\\ 16\\ 17\\ 18\\ 19\\ 19\\ 20\\ 21\\ \end{array} $	Related Work References References Table of Contents Contents	nd "while" rule (simplified)	22 23 24 25 25 26 26 28 28
Achim D. Brucker	Seminar: Specification and Verification of Object-oriented S	Software	Achim D. Brucker	Seminar: Specification and Verification of Object-orie	nted Software