Formal Specification of Software

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU

Summer Term 2004

Based on a lecture held by
Peter H. Schmitt at the
University of Karlsruhe

Web Page

All information relevant to this lecture can be found on the web page

www.uni-koblenz.de/˜beckert/Lehre/Spezifikation

Contents

Mathematical and logical basis

- Set Theory
- Predicate logic
- Modal logic

Specification techniques

- UML class diagrams (by example)
- Object Constraint Language, OCL (by example)
- Formal semantics of OCL (and UML class diagrams)
- Abstract State Machines, ASMs
- Abstract Data Types
  Common Abstract Specification Language, CASL
- State Charts
- The specification language Z

Why Formal Methods?

Quality: Important for . . .

- **Safety-critical applications** (railway switches)
- **Security-critical applications** (access control, electronic banking)
- **Financial reasons** (phone cards)
- **Legal reasons** (electronic signature, EAL6/7 in Common Criteria)

Productivity: Important for . . .

Obvious reasons

Why Formal Methods?

Quality through . . .

- Better and more precise understanding of model and implementation
- Better written software (modularisation, information hiding, . . .)
- Error detection with runtime checks
- Test case generation
- Static analysis
- Deductive verification

Why Formal Methods?

Productivity through

- **Error detection in early stages of development**
- **Re-use of components** **(requires specification and validation)**
- **Better documentation, maintenance**
- **Knowledge about formal methods leads to better software development**

Favourable Development

Design and specification

- **Unified Modeling Language – UML**

  Graphical language for object-oriented modelling
  Standard of Object Management Group (OMG)

- **Object Constraint Language – OCL**

  Formal textual assertion language
  UML Substandard

- **Consolidation and documentation of design knowledge**

  Patterns, idioms, architectures, frameworks, etc.

Industrial implementation languages

- **Java, C#**

Types of Requirements

Types of Requirements

- **functional requirements**
- **communication, protocols**
- **real-time requirements**
- **memory use**
- **security**
- **etc.**

Different Formal Methodsx

- **deductive verification**
- **model checking**
- **static analysis**
- **run-time checks**
  (of formel specification)

Limitations of Formal Methods

Possible reasons for errors

- Program is not correct (does not satisfy the specification)
  Formal verification proves absence of this kind of error
- Program is not adequate (error in specification)
  Formal specification/verification avoid/find this kind of error
- Error in operating system, compiler, hardware
  Not avoided (unless compiler etc. specified/verified)

No full specification/verification

In general, it is neither useful nor feasable to fully specify and verify large software systems. Then, formal methods are restricted to:

- Important parts/modules
- Important properties/requirements