Formal Methods for Software Development

Till Mossakowski, Lutz Schröder

20.10.2004

Overview of this Lecture

- MMISS
- Software Development and Formal Specification
- Overview of the course
- “Scheinkriterien”

MMISS

- MMISS = multimedia instructions in safe and secure systems
- aim: multimedia Internet-based adaptive educational system
- repository of lectures (in english)
- glossary with central notions translated to German
- See `www.mmiss.de`

Software Development and Formal Specification

Software Development and Formal Specification

Goals

- to understand the basic principles of software development
- to understand the role of formal methods in software development
- to understand the basic principles of functional programs and their formal specification

Software Engineering

deals with the technical and organisational aspects of the development and maintenance of large software systems.

State of the Art

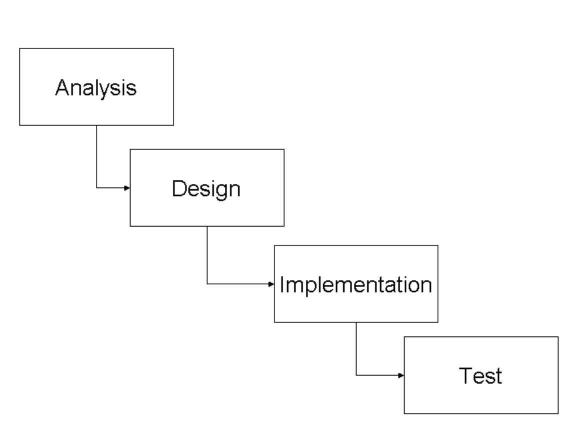
- natural language documentations
- diagrammatic modeling languages (e.g. UML)
- CASE Tools (Computer Aided Software Engineering)
- formal methods (for safety critical systems)

Formal Methods

- based on mathematics (set theory, algebra, math. logics)
- advantages:
  - unambiguous interpretation of syntactic constructs
  - verification of properties (of specifications, models, programs)
  - verification of the correctness of development steps
- difficulties:
  - knowledge of formal notations and their meaning
  - additional development costs

Process Models

- waterfall model



- iterative model, V-model, spiral model, XP/agile.......

Formal software development uses additionally *formal specifications* in the different phases. Verification of the correctness of a development step is only possible on the basis of formal specifications.

Proving the Correctness of the Implementation

a) Verification of the implementation against the requirements specification (post mortem) **OR**

b) Verification of each realization step (verification conditions)

Remarks

- Testing can only show the existence of errors. Verification can show the absence of errors.
- The adequateness of a (formal) specification w.r.t. the desires of the user can not be verified.

Verification Success Stories

- complete formal verification of Pentium 4 arithmetic
- NASA uses formal specification of physical units
- verification of the Java bytecode verifier
- found 12 deadlocks in occam code for international space station

Haskell

- is a purely functional programming language
- therefore it is well-suited for application of formal methods
- side-effects are encapsulated via monads
- Haskell specification logic P-logic
- specifications can be used for both testing and verification

Sorting in Haskell

```
insert :: Ord a => (a,[a]) -> [a]
insert(x,[]) = [x]
insert(x,y:l) = if x <= y then x:y:l
                   else y:insert(x,l)

insert_sort :: Ord a => [a] -> [a]
insert_sort([]) = []
insert_sort(x:l) = insert(x,insert_sort(l))
```

Test Cases

```
testSorting
  = TestCase
     (do let list = [7,2,6,3,5]
             sortedList = [2,3,5,6,7]
         assertBool "insert_sort is faulty"
           (insert_sort list == sortedList)
     )
```

Test Case Generation

```
propSorted [] = True
propSorted [x] = True
propSorted (x:y:xs) =
    x <= y && propSorted (y:xs)

instance Arbitrary [Int] where
  arbitrary =
    do len <- choose (0,20)
       l <- mapM (\x -> choose (0,20))
                 [1..len]
       return l
```

Specification

```
{-# AXIOMS
         "isSorted" forall l l1 l2 x y ->
             insert_sort l == l1++[x,y]++l2 =
#-}
```

This Course

Overview of the course

- Testing with user-defined test cases (HUnit)
- Testing with automated test-case generation (QuickCheck)
- Testing monadic programs
- P-logic specification logic
- Isabelle/HOL: verification of simple functional programs
- ISabelle/HOLCF: verification of general functional programs
- From P-logic to HOLCF

Scheinkriterien

- two big exercises (one in December, one at the end of the lecture)
- successful solution of the exercises and presentation in the course
- in groups of at most three students

Dates + Rooms

- Mon. 13-15h: MZH 7230
- Wed. 15-17h: MZH 7250
- Lectures and, from time to time, exercises (please bring your laptops)
- Web: `http://www.tzi.de/agbkb/lehre/ws04-05/fmsd`