Course 02263
Formal Aspects of Software Engineering
Introduction

Anne E. Haxthausen

DTU Informatics (IMM)
Technical University of Denmark

Personal Background

Associate Professor Anne Haxthausen:

- Research interests: formal aspects of software engineering, in particular design of specification languages and verification techniques.
- M.Sc.E., DTU 1985, Ph.D. in Comp. Science, DTU 1989
- Software engineer at DDC and CRI 1988-1994,
- Employed at DTU (ID, IT, IMM) since 1995
- Guest professor at Japanese research lab 1993 and at a university in Paris 2005/06.

Agenda

- Course Introduction
  - Course aim and contents
  - Practical informations

Course Aim

to introduce you to *formal aspects* of software engineering, and in particular to give you skills in

- reading and writing formal specifications
- abstraction and modelling

What is Formal Methods?

In software engineering, *formal methods* are mathematically based techniques for the synthesis (construction) and analysis of software systems.

Why using Formal Methods?

To produce high quality software. In particular to reduce the number of software bugs.

Software is a Cornerstone in our Society

Software is used everywhere:

- in finans
- in health care
- in transportation
- in defense
- in telecommunication
- ...

and is often rather complex.

The *correctness* and efficiency of such software is critical for us.

Software Development

- Software is often full of bugs, and
- software projects fail due to delays, cost overrun, usability problems, etc.

Economic Costs of Software Bugs

COMPUTERWORLD, 2002:

Software bugs are costing the U.S. economy an estimated 59.5 billion USD each year.

Example of a Critical Software Bug

Ariane 5 rocket explosion 1996 due to data conversion of a too large number.

Development costed 7 billion USD, cargo valued at 500 million USD.

Human Costs of Software Bugs

Human costs range from inconvenience to death.
Examples:

- Death resulted from a bug in the London Ambulance Service software.
- Several 1985-7 deaths of cancer patients were due to overdoses of radiation resulting from a race condition between concurrent tasks in the Therac-25 software.
- ...

Use of Mathematics in Engineering

In other engineering disciplines (civil, mechanical, electrical, ...) the engineers use mathematics to model and analyse objects to be created.

Example:
Before building a bridge, the civil engineer will create a mathematical model in order to analyse wheather it is safe (will not crash).

Use of Formal Methods in Software Engineering

Also in software engineering we can benefit from mathematics alias formal methods.

Especially relevant for the development of safety/business/mission critical software.

Some Standards Advocate/Require the use of Formal Methods

- The Common Criteria (ISO/IEC 15408) for computer security, Evaluation Assurance Levels:
  1. Functionally Tested
  2. Structurally Tested
  3. Methodically Tested and Checked
  4. Methodically Designed, Tested and Reviewed
  5. Semiformally Designed and Tested
  6. Semiformally Verified Design and Tested
  7. Formally Verified Design and Tested
- CENELEC European Standard EN50128 about software for railway control.
- ...

Typical Use of Formal Methods in Software Engineering

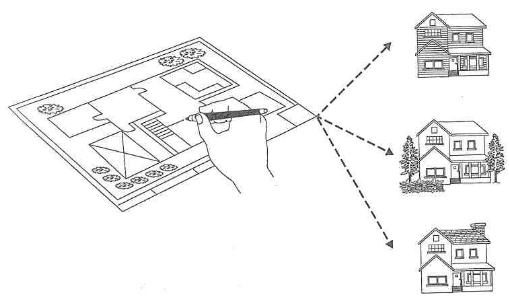
- Formal specifications are used for describing requirements and designs.
- Formal verification is used to prove correctness properties (e.g. that a design meets requirements).
- Formal specifications may be used as base for code generation.

Note: formal methods *complement* traditional methods.

What is a Specification?

A *specification* is a description of some properties of a product (either to be produced or existing).

Architects use *drawings* to describe buildings:



Software engineers use *specifications* to describe software.

Formal Specification

A *formal specification* is a sentence in a formal specification language.
A *formal specification language* is characterized by having:

- a precise syntax
- a mathematical meaning (semantics) of each sentence of the language
- a logic proof system

Many formal specification languages exist:

- VDM, Z, RSL, CASL, Petri Nets, DC, OCL, JML, EML, ...

Examples of mathematical objects used in specification languages:

- numbers, sets, lists, functions
- algebras
- logical conditions

Why Formal Specification?

Formal specifications

- are unambiguous
- enforces the specifier to think deeply about the problem
- abstract away from irrelevant details (are thereby simpler)
- allows formal verification

=>

- fewer errors
- errors found earlier than by testing

Formal and Semi-formal Verification

(Semi-)Formal Verification is the act of using math to establish correctness properties.

Main techniques:

- *Formal proof*: proofs are objects that are constructed using proof rules of a mathematical logical system.
- *Semi-formal proof* (often used in text books): a sketch of a formal proof made as a mixture of natural language and formulas.
- *Model checking*: basically exhaustive testing of an executable model.

Course Contents

Focus on specification language issues:

- basic concepts of specification languages
- various specification styles
- the development paradigm of stepwise refinement and verification
- mandatory exercise(s)

We will use the **RAISE** Specification Language and Method to teach you this.

Practical Informations: Course Home Page

http://www2.imm.dtu.dk/courses/02263
Inspect the course home pages regularly!

Practical Informations: Course Activities

A typical week:

	Tuesday	Friday
8:15-10:00		lecture
10:00-12:00		problem session
13:00-15:00	(databar)	(databar)

- **Lectures:** here (aud. 13 in bldg. 308).
- **Problem sessions and project advise:** in room 033 in bldg. 322
- **Tool sessions:** on your own at home or in E databar.
  Reservations in the E databar:
  - Tuesdays 13.00-15.00
  - Fridays 10.00-12.00 and 13.00-15.00
- **Homework**

Detailed plans and time schedules for course activities will be published at the course WWW pages.

Practical Informations: Course Material

- Textbook: *The RAISE Specification Language*:
  can be bought at the DTU Informatics Bookshop, bld. 321 for 300 kr.
- Notes + Exercises + Solutions + Foils + ... :
  will be published during the semester at the course WWW pages.

DTU Informatics Bookshop: opening hours

- 1.-2. week: Monday-Friday 9-14.
- 3. week - ...: Monday 9-12.

Practical Informations: Homework for Next Week

- Buy the *The RAISE Specification Language* book at DTU Informatics Bookshop.
- Download from the WWW homepages for the course:
  - The DTU tools guides.
  - Notes: *Changes to RSL* and ... *Comments and typos*.
  - The exercise collection.
  - Copies of overheads OH0, OH1, OH2, OH3, OH4, OH5.
- Read sections 1-6, 7.1–7.11 and 8 in the book.
- Install the RAISE tools on own PC or set them correctly up in the E databar, and check that they work.