Formal Methods in Software Design

Markus Roggenbach

October 2001
Formal Methods

“Use of mathematics in software development”

main activities:

- writing formal specifications
Formal Methods

“Use of mathematics in software development”

main activities:
- writing formal specifications
- proving properties about formal specifications
Formal Methods

“Use of mathematics in software development”

main activities:
- **writing** formal specifications
- **proving** properties about formal specifications
- **constructing** a program by mathematical manipulating a formal specification
Formal Methods

“Use of mathematics in software development”

main activities:
- writing formal specifications
- proving properties about formal specifications
- constructing a program by mathematical manipulation of a formal specification
- verifying a program by mathematical argument
Non Formal, Semi Formal, Formal

“It has been widely accepted that syntax can be mathematically defined for quite some time, but there has been more resistance to the mathematical definition of semantics.”

(quoted freely from [1])
non formal:

in natural language

(open to arbitrary new symbols)

formal:

in a (fixed) language with

mathematically defined Syntax and Semantics
semi formal:
in a language with

- **Syntax** definition by mathematical methods
- **Semantics** definition in natural language or by tool
Specifications

Specification: “description by properties”

Main question on specifications:
“What happens if . . .”

Specifications should be
Specifications

Specification: “description by properties”

Main question on specifications:
“What happens if . . . ”

Specifications should be
• complete
Specifications

Specification: “description by properties”

Main question on specifications:
“What happens if . . .”

Specifications should be

- complete
- precise
Specifications

Specification: “description by properties”

Main question on specifications:
“What happens if . . .”

Specifications should be
• complete
• precise
• consistent (no contradictions)
Specifications

Specification: “description by properties”

Main question on specifications:
“What happens if . . .”

Specifications should be

- complete
- precise
- consistent (no contradictions)
Why formal Specifications?

- formal specifications are *precise*
  (non formal and sometimes even semi formal specifications are open to re-interpretation)
Why formal Specifications?

• formal specifications are precise (non formal and sometimes even semi formal specifications are open to re-interpretation)

• syntactical and semantical correctness independent of tools
Why formal Specifications?

• formal specifications are precise
  (non formal and sometimes even semi formal
  specifications are open to re-interpretation)

• syntactical and semantical correctness
  independent of tools

• mathematical methods
  (consistency, completeness)
Limitations of Formal Methods

“The world is not a formal system.”

I. Modelling means Abstraction
(only “essentials” are considered)

II. Errors within Formalisms.
III. Behaviour of a Program depends on
- Compiler
- Operating System
- Computer Hardware
- Embedding in a Technical Process
- Human Operator
Waterfall Modell

Requirement Elicitation and Analysis
↑ ↓

↑ ↓

↑ ↓

Progr. Lang. Implementation
↑ ↓

Test
↑ ↓

Maintenance

Validation

“Inv. & Verify” or Transformation

Specification Languages

“No single technique is adequate to address all issues of complex system development.”

Classification of Specification Languages:

- Model-oriented: Z, VDM
- Property-oriented: Larch, OBJ, CASL
- Process algebras: CCS, CSP, π-calculus