Systeme Hoher Sicherheit und Qualität
Universität Bremen WS 2015/2016

Lecture 14 (01.02.2016)

Concluding Remarks

Christoph Lüth Jan Peleska Dieter Hutter

Where are we?

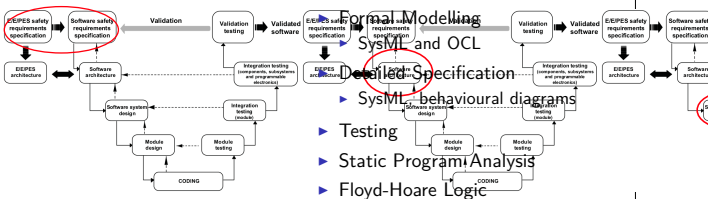
- 01: Concepts of Quality
- 02: Legal Requirements: Norms and Standards
- 03: The Software Development Process
- 04: Hazard Analysis
- 05: High-Level Design with SysML
- 06: Formal Modelling with SysML and OCL
- 07: Detailed Specification with SysML
- 08: Testing
- 09: Program Analysis
- 10: Foundations of Software Verification
- 11: Verification Condition Generation
- 12: Semantics of Programming Languages
- 13: Model-Checking
- 14: Conclusions and Outlook

Introductive Summary

- This lecture series was about developing systems of high quality and high safety.
- Quality is measured by quality criteria, which guide improvement of the development process. It is basically an economic criterion.
- Safety is "freedom from unacceptable risks". It is a technical criterion.
- Both high quality and safety can be achieved by the means described in this lecture series.
- Moreover, there is the legal situation: the machinery directive and other laws require (indirectly) you use these techniques where appropriate. This is why these lectures are so important: disregarding this state of the art may make you personally liable.

Quality in the Software Development Process

- Hazard analysis
- High-level design
  - SysML, structural diagrams
- Formal Modelling
  - SysML and OCL
- Detailed Specification
  - SysML, behavioural diagrams
- Testing
- Static Program Analysis
- Floyd-Hoare Logic
- Semantics of Programming Languages
- Model-Checking



Examples of Formal Methods in Practice

- Hardware verification:
  - Intel: formal verification of microprocessors
  - Infineon: equivalence checks
- Software verification (research projects):
  - Verisoft — Microsoft Hyper-V (VCC)
  - L4.verified — NICTA, Australia (Isabelle)
- Tools used in industry (excerpt):
  - AbsInt tools: aiT, Astrée, CompCert (C)
  - SPARK tools (Ada)
  - SCADE (MatLab/Simulink)
  - UPAALL, Spin, FDR2, other model checkers

SSQ at University of Bremen

- AG BS (Prof. Jan Peleska): Testing, abstract interpretation.
  - Strong industrial links to aerospace and railway industry, spin-off (Verified Systems)
- DFKI CPS and AG RA (Profs. Rolf Drechsler, Dieter Hutter, Christoph Lüth):
  - Strong industrial links: Infineon, Intel, NXP
  - Hardware and system verification
  - Software verification
  - Security
  - Further application areas: robotics and AAL
- SyDe Graduate College (University of Bremen, DFKI, DLR)
  - Includes more application areas: Space, robotics, real-time image processing

Questions

Lecture 01: Concepts of quality

- What is quality? What are quality criteria?
- What could be useful quality criteria?
- What is the conceptual difference between ISO 9001 and CMM?

Lecture 02: Concepts of Safety and Security

- What is safety?
- Norms and Standards:
  - Legal situation
  - What is the machinery directive?
  - Norm landscape: First, second, third-tier norms
  - Important norms: IEC 61508, ISO 26262, DIN EN 50128, DO-178B, ISO 15408
- Risk analysis:
  - What is a SIL? Target SIL?
  - How do we obtain a SIL? What does it mean for the development?

Lecture 03: Quality of the Software Development Process

- Which software development models did we encounter?
- Waterfall, spiral, agile, MDD, V-model:
  - How does it work?
  - What are the advantages and disadvantages?
- Which models are appropriate for safety-critical developments?
- What are the typical artefacts (and where do they occur)?
- Formal software development:
  - What is it, and how does it work?
  - How can we define properties, what kind of properties are there, how are they defined?
  - Development structure: horizontal vs. vertical, layers and views

Lecture 04: Hazard Analysis

- What is hazard analysis?
- Where (in the development process) is it used?
- Basic approaches: bottom-up vs. top-down, and what do they mean?
- Which methods did we encounter?
- FMEA, FTA, Event traces — how do they work, advantages/disadvantages?
- What are the prime verification techniques?

Lecture 05: High-level Design

- High-level specification and modelling:
  - What is it, where in the development process does it take place, what formalisms are useful?
- What is SysML? How does it relate to UML?
- Basic elements of SysML used for high-level design:
  - Structural diagrams:
  - Package diagram
  - Block definition diagram (describes classes, class diagram)
  - Internal block diagrams (describes instances of blocks, flow specifications)
  - Parametric diagram (equational modelling)

Lecture 06: Formal Modelling with SysML and OCL

- What is OCL?
  - A specification language for UML/SysML models
  - Characteristics: pure and typed
- What can we use it for?
  - Invariants on classes and types
  - Pre- and postconditions on operations and methods
- OCL types:
  - Basic types: Boolean, Integer, Real, String; OclAny, OclType, OclVoid
  - Collection types: Sequence, Bag,OrderdedSet, Set
  - Model types
- Logic: three-valued Kleene logic

Lecture 07: Detailed Specification

- What is detailed specification?
  - Specification of single modules — „last“ level before code
- What elements are used in specification?
- SysML behavioural diagrams:
  - State diagrams (hierarchical finite state machines)
  - Activity diagrams (flow charts)
  - Sequence diagrams (message sequence charts)
  - Use-case diagrams

Lecture 08: Testing

- What is testing, and what are the aims? What can it achieve, what not?
- What are test elevels?
- What is a black-box test? How are test cases chosen?
- What is a white-box test?
- What is the control-flow graph of a program?
- What kind of coverages are there, and how are they defined?

Lecture 09: Static Program Analysis

- Is what? Where in the development process is it used? What is the difference to testing?
- What is the basic problem, and how is circumvented?
- What does it mean when we say an analysis is sound, or safe?
- What are false positives?
- Did we consider inter- or intraprocedural analysis?
- What examples for forward/backward analysis did we encounter?

Lecture 10: Verification with Floyd-Hoare Logic

- What is Floyd-Hoare logic, what does it do (and what not), and where is used in the development process?
- How does it work?
- What is the difference between $\models \{P\}\, p\, \{q\}$ and $\vdash \{P\}\, p\, \{q\}$?
- What do the notations $\{P\}\, p\, \{Q\}$ and $[P]\, p\, [Q]$ mean?
- What rules does the Floyd-Hoare logic have?
- How are they used?
- Which properties does it have?

Lecture 11: Verification Condition Generation

- What does VCG do?
- How is it related to Floyd-Hoare logic?
- What is a weakest precondition, and how do we calculate it?
- What are program annotations? Why do we need them? How are they used?
- What does $\mathrm{vc}(c, P)$ and $\mathrm{pre}(c, P)$ mean, and how do we calcuate them?
- Which tools do VCG?

Lecture 12: Semantics

- What is semantics? What do we need it for?
- What are the three kinds of semantics, and how to they work?
  - Operational semantics specifies how the program is executed, often as a relation $\langle c, \sigma \rangle \to \sigma$.
  - Denotational semantics models the program as a mathematical entity, often as a partial function $\Sigma \to \Sigma$ using complete partial orders (cpos). Cpos provide mathematical means to handle partiality and fixpoints (iteration).
  - Axiomatic semantics gives proof rules for programs, such as the Floyd-Hoare rules.
  - We can show equivalence of semantics (correctness).
- When do we use which?
  - Operational semantics: implementing the language
  - Denotational semantics: high-level reasoning
  - Axiomatic semantics: reasoning about programs

Lecture 13: Model-Checking with LTL and CTL

- What is model-checking, and how is it used? How does it compare with Floyd-Hoare logic?
- What is the basic question? $\mathcal{M} \models \phi$
  - What do we use for $\mathcal{M}$, ϕ, and do we prove it?
- What is a finite state machine, and what is temporal logic?
- LTL, CTL:
  - What are the basic operators, when does a formula hold, and what kind of properties can we formulate?
  - Which one is more powerful?
  - Which one is decidable, and with which complexity?
- What is the basic problem (and limitation) of model-checking?
- Which tools did we see to model-check LTL/CTL?

Module Exams (Modulprüfungen)

- We have the following five areas:
  - Lectures 1 – 4: Quality, Norms and Standards, Development Processes, Requirements Analysis
  - Lecture 5 – 7: SysML
  - Lecture 8 – 9: Testing and Static Program Analysis
  - Lecture 10 – 12: Semantics, Floyd-Hoare Logic and Verification Conditions
  - Lecture 13: Model-Checking with LTL and CTL
- You may choose two areas (except for the first). You need to tell us before the exam starts.
- Questions may come from all lectures, but we will concentrate on the first and your chosen areas.

Final Remark

- Please remember the evaluation (see stud.ip)!

Thank you, and good bye.