

Theoretical Foundations of the UML

Lecture 1: Introduction

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

<http://moves.rwth-aachen.de/i2/uml09100/>

16. Oktober 2012

Target audience

You are studying:

- Master Computer Science, or
- Master Systems Software Engineering, or
- Bachelor Computer Science

Usage as:

- elective course Theoretical Computer Science
- not a Wahlpflicht course for bachelor students
- specialization **MOVES** (Modeling and Verification of Software)
- complementary to **Model-based Software Development** (Rumpe)

Target audience (contd.)

In general:

- interest in system software engineering
- interest in formal methods for software
- interest in semantics and verification
- application of mathematical reasoning

Prerequisites:

- mathematical logic
- formal language and automata theory
- algorithms and data structures
- computability and complexity theory

Organization

Schedule:

Type	Day	Time	Lecture hall
Lecture	Mon	13:15 - 14:45	AH4
	Tue	10:00 - 11:30	5056
Exercises	Wed	15:45- 17:15	Seminarroom i2

about 19 lectures in total; Keep track of website for precise dates!

People involved:

Type	Lecturer	EMail
Lecture	Joost-Pieter Katoen	katoen@cs.rwth-aachen.de
Exercises	Sabrina von Styp	sabrina.von-styp@cs.rwth-aachen.de
	Falak Sher	falak.sher.vira@cs.rwth-aachen.de

Assignments:

- (almost) weekly assignments
- available from course web-site
- first assignment: **Wednesday October 16**
- hand in solution at next exercise class
- groups of maximally two students

Examination: (6 ECTS credit points)

- written exam: February 6, 2013 (afternoon)
- re-exam: March 22, 2013 (morning)

Admission:

- at least 50% of exercise points

Motivation

Scope:

- **Goal:** formal description + analysis of (concurr.) software systems
- **Focus:** the Unified Modeling Language

More specifically:

- Sequence Diagrams (used for requirements analysis), PDL
- Communicating Finite State Automata
- Hierarchical State Machines (behavioral description of systems)
- The Object Constraint Language (OCL)

Aims:

- clarify and make precise the semantics of treated UML fragments
- formal reasoning about basic properties of UML models
- algorithms to verify such properties

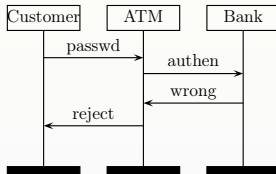
What this course is **NOT** about:

What is it ****not**** about?

- the use of the UML in the software development cycle
 - see the complementary course by Prof. Rumpe
- other notations of the UML (e.g., class diagrams, activity diagrams)
- what is precisely in the UML, and what is not
 - liberal interpretation of which constructs belong to the UML
- applying the UML to concrete SW development case studies
- empirical results on the usage of UML
- drawing pictures
- ...

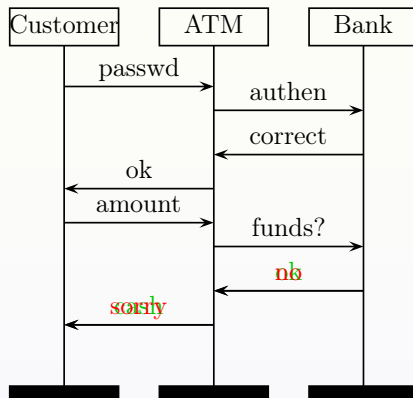
Sequence Diagrams

- origin: telecommunications: “Message Sequence Charts” (MSCs)
- describe **interactions** between processes (or objects)
- attractive **visual** formalism



- describes a possible **scenario**
- **standardized** by the ITU (Z. 120)
- adopted by the OMG for **UML**

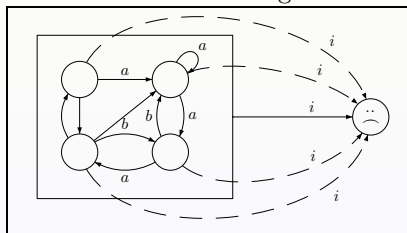
Another example MSC



- MSCs
(syntax, semantics, linearizations, races)
- Message sequence graphs
(composition, expressiveness, compositional MSCs)
- Realizability
(communicating finite-state machines, reachability in CFSMs, MSCs vs. CFSMs, boundedness)
- Regularity
(regular MSCs and MSGs, realizability)
- Verification
(positive + negative model checking, complexity results, basic properties: **MSCan**)
- PDL
(Propositional Dynamic Logic for checking MSC properties)

Hierarchical State Machines

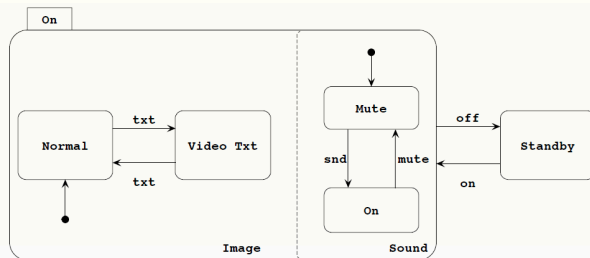
- finite state machines
 - no strategy for top-down or bottom-up development (“states have no structure”)
 - no natural notion of hierarchy
 - uneconomical concerning transitions (e.g., high-level interrupt)



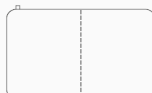
- uneconomical wrt. parallel composition (exponential growth in # states)

Statecharts = Mealy machines
+ depth
+ orthogonality [Harel'86]
+ broadcast
+ data

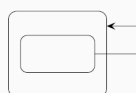
Statecharts (contd.)



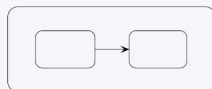
hierarchical
state



AND state



inter-level
transition



intra-level
transition



inter-level
transition

- Harel's Statecharts
(basic features, syntax, state hierarchy, orthogonality, intra- and inter-level transitions)
- Semantics
(main issues, formal semantics, flattening, succinctness)
- Verification
(expressiveness, reachability, LTL model checking)

- allows specification of basic properties on objects:

Example

context Room invariant

guest \rightarrow size \leq numOfBeds

context Hotel::checkIn (g:Guest)

pre not guests \rightarrow includes(g)

post guests \rightarrow size = (guests@pre \rightarrow size)+1

and guests \rightarrow includes(g)

- not related to particular diagram of UML
- often: annotations to different types of UML diagrams (e.g. class diagrams, activity diagrams, statecharts, ...)

Topics:

- OCL basics
(types, operations, navigation, class diagrams)
- semantics of the OCL
(operational model, logic, types + values)
- Embedding into temporal logic
(LTL/CTL, from OCL to temporal)