# Lecture 1: Introduction

## Contents & Goals

This Lecture:

- **Educational Objectives:**
  - After this lecture you should
    - be able to explain the term *model*.
    - know the idea (and hopes and promises) of model-based SW development.
    - be able to explain how UML fits into this general picture.
    - know what we'll do in the course, and why.
    - thus be able to decide whether you want to stay with us...

- **Content:**
  - Analogy: Model-based/-driven development by construction engineers.
  - Software engineers: "metoo" – Model-based/-driven Software Engineering.
  - UML mode of the Lecture: Blueprint.

## Disclaimer

The following slides may raise thoughts such as:

- "everybody knows this",
- "completely obvious",
- "trivial",
- "clear",
- "irrelevant",
- "oversimplified" ...

Which is true, in some sense, but: "everybody" is a strong claim, and I want to be sure that this holds for the audience from now on. In other words: that we're talking about the same things.

## An Analogy: The House-Building Problem (Oversimplified)

**Given** a set of requirements, such as:

- The house shall fit on the given piece of land.
- Each room shall have a door, the doors shall open.
- The given furnitures shall fit into the living room.
- The bathroom shall have a window.
- The costs shall be in budget.

**Wanted**: a house which satisfies the requirements.

Now, strictly speaking, a house is a complex system:

- Consist of a huge number of bricks.
- Consist of subsystems, such as windows.
- Waterpipes and wirings have to be in place.
- Doors have to open consistently.
- Floors depend on each other (load-bearing walls).
- ...

How do construction engineers handle this complexity...?
1. Requirements

• Shall fit on given piece of land.
• Each room shall have a door.
• Furniture shall fit into living room.
• Bathrooms shall have a window.
• Costs shall be in budget.

2. Design

http://wikimedia.org (CC nc-sa 3.0, Ottoklages)

3. System

http://wikimedia.org (CC nc-sa 3.0, Bobthebuilder82)

Observation: Floorplan preserves, e.g.,
• house and room extensions (to scale),
• presence/absence of windows and doors,
• placement of subsystems (such as windows).

Floorplan as an Abstraction

• Floorplan $F$ denotes a set $\gamma(F)$ of houses (concretisations of $F$), which differ, e.g., in colour of bricks, or making of windows.

• Floorplan $F$ represents house $H$ according to abstraction $\alpha$.

By adding information to $F$ (such as making of windows), we can narrow down $\gamma(F)$.

What is it good for? Build by Plan.

• As said before, the floorplan abstraction $\alpha$ preserves some properties. For instance, we have:

- Room $R$ has window in $H$ if and only if $R$-representation in $\alpha(H)$ has window.

• And we have the general rule:

- If a house $H'$ is (or: will have been) built according to plan $F$, and if plan $F$ has property $\phi$, and if $\alpha/\gamma$ preserves this property, then $H'$ has (or: will have) property $\phi$.

- So we can answer some questions about $H$ before even building it, e.g.:

- Bathroom shall have a window.
- Shall fit on given piece of land.
- Each room shall have a door.
- Furniture shall fit into living room.
- Costs shall be in budget.

• And: it's typically easier (and cheaper) to correct errors in the plan, rather than in the finished house.

“Silver Bullet” or Can Anything Go Wrong...

• If the requirements are already contradictory (or inconsistent), then there is no sense in drawing a plan.

Example:

- The house shall fit on the given piece of land.
- The given furniture shall fit into the living room.
- What if the land is 10m narrow and the couch is 11m × 11m?

Good for Anything Else? Documentation.

• Given: a house.
• Wanted: a concise description for potential buyers.

Approach: draw a floorplan.

Distinguish:

• Sometimes the plan $F$ is first, and the realisation $H \in \gamma(F)$ comes later.

• Sometimes the realisation $H$ is first, and the "plan" $F = \alpha(H)$ comes later.

What's the Essence?

Definition.

[Folk] A model is an abstract, formal, mathematical representation or description of structure or behaviour of a (software) system.

Definition. [Glinz, 2008, 425]

A model is a concrete or mental image (Abbild) of something or a concrete or mental archetype (Vorbild) for something.

Three properties are constituent:

(i) the image attribute (Abbildungsmerkmal), i.e. there is an entity (called original) whose image or archetype the model is,

(ii) the reduction attribute (Verkürzungsmerkmal), i.e. only those attributes of the original that are relevant in the modelling context are represented,

(iii) the pragmatic attribute, i.e. the model is built in a specific context for a specific purpose.
after (15 sec.)

Invalid

system model

why?

Because it is easier to handle than "pictures"; it abstracts from

model

Model-Driven Software Engineering with UML

Needed: A Modelling Language for SW-Engineering

The approach in this course:

(i) Introduce a common semantical domain—what is a very abstract

formal

system model

Mathematical characterisation of diagrams

(ii) Equip it with a visual formal language
defining

Formal
description of transitions systems and modeling is about

(iii) Define consistency/satisfaction relations in terms of semantics.

Why? Because it is easier to handle than "pictures"; it abstracts from

model

Model-Driven Software Engineering with UML

Wrong

Software may have infinite and finite

requirements/

Correct software, i.e. software

correct

Formal

implies

Correct software, i.e. software

correct

Formal
Back to the graphical because it's graphical. [\textellipsis]

Support diagram drawing and programming will succeed just as a language. 

And this not only applies to UML as a language and thus more productive than current languages. 

Blueprints require much more sophisticated tools than what should be in it? 

Floorplan as a blueprint: 

\[
G = (N,E,f) = (\pi_0,\epsilon_0) \text{ SD, cons, } \sigma_1, \ldots, \sigma_n \text{ SM}, A, q, B \text{ wiring plan}
\]

Floorplan as a sketch: 

Examples for context/purpose: 

\[
\text{Context: } OCL \in \phi \text{ cb SD, CD, SM, BV, CC, BW, SM, A, q, B, wiring plan}
\]

Recall the question, of course, is whether this promise is true. 

Tool can take the UML and the programming should be done; 

Sketches are also useful in books, such as mine, are more lightweight drawing tools than any strict rule of the UML. 

Their emphasis is on selective sketches in order to handle the details required for the design decisions are laid out. 

Blueprints require much more thought. [\textellipsis]

Martin Fowler puts it like this: 

The "mode" fitting the lecture best is UML Mode of the Lecture: As Blueprint. 

With UML it's the same as far as possible. 

Analyzing consistency/implication is straightforward activity that sufficient completeness that all diagrams you draw and need in software, you can make the UML be your programming language. 

The idea is that blueprints are developed by a designer whose job is to build a detailed design for a task. 

If you can detail the UML sufficiently complete that all code can be written up. 

If you can detail, then you can analyze consistency/implication of the system. [\textellipsis] 

In forward engineering tools, take the UML and the programmer should be able to follow as a pretty straightforward activity that avoids misunderstandings.

Strictly speaking, possibly at least as well to individual UML models but at least as well to individual UML models.

The emphasis is on selecting particular about keeping to communication rather than completeness. [\textellipsis]

The irony is that the more people come up with different UML diagrams shown, the more UML models and thus more complete specification are created. 

Forward engineering tools back it up with a repository. [\textellipsis] 

If someone else's view of the UML seems rather different to yours, it's usually because they use a different UML. 

In this UML Mode of the Lecture, developers focus is on the communication rather than completeness. [\textellipsis] 

Some people differ about what should be in the UML because there are differing fundamental views about what the UML should be.

In books, such as mine, Sketch is used for sketching and programming rather than completeness. [\textellipsis] 

For the pragmatic attribute.

For the pragmatic attribute.

With the pragmatic attribute, the model is built based on a specific context for a specific purpose. 

Recall the pragmatic attribute, i.e. the model is built in a specific context for a specific purpose. 

Claim: the pragmatic attribute is the enemy of comprehensibility.
Real-Time UML Components

Putting it all together:

Mathematics Meta-Modeling

Implementation program

Inheritance

Machine Diagram

State Machine Diagram Sequence Diagram

Constraints Generated

Domain Specific Language

As Programming Language Lecture could be worked out into mode

Motivation and Overview

Semantical Domain

Course Path: OverMap

Course Overview

Model Instances

Live Sequence Charts

Reflective

Class Diagrams

Object Diagrams

OCL

Idea

Domain

Motivation

•

•

•

•

Idea

Domain

•

•

•

And

they are the ones who are the philosophers.

The mode of the lecture is As Blueprint

As Blueprint

As Blueprint

As Blueprint

As Blueprint

As Blueprint

As Blueprint

As Blueprint

As Blueprint

As Blueprint
Development Process

UML is only the language for artefacts. But: we’ll discuss exemplarily, where in an abstract development process which means could be used.

How to come up with a good design

UML is only the language to write down designs. But: we’ll have a couple of examples.

Requirements Management

Versioning, Traceability, Propagation of Changes.

Every little bit and piece of UML

Boring. Instead we learn how to read the standard.

Object-Oriented Programming

Interesting: inheritance is one of the last lectures.

– Formalia –

Courselanguage:

English

(slides/writing, presentation, questions/discussions)

Presentation:

halfslides/half on-screen

hand-writing

—for reasons

Script/Media:

slides with annotations on homepage, 2-up for printing, typically soon after the lecture

recording one Lectures portal with max. 1 week delay (link on homepage)

Interaction:

absence often moaned but it takes two, so please ask/comment immediately.

– Formalia –

Location:

Monday, Wednesday: here (building 51, room 01-034)

Schedule:

Week N, Monday, 10:00 (exercises A early submission)

10–12 lecture A2

Wednesday, 10–12 lecture A3

Week N +1, Monday, 10:00 (exercises B late submission)

10–12 lecture B1

(exercisesheet B online)

Wednesday, 10:00 (exercises A late submission)

10–12 tutorial A

Week N +2, Monday, 10–12 lecture B2

Wednesday, 10–12 lecture B3

Week N +3, Monday, 10:00 (exercises B early submission)

10–12 lecture C1

(exercisesheet C online)

Wednesday, 10:00 (exercises B late submission)

10–12 tutorial B

With a prefix of lectures, see homepage for details.

– Formalia –

Exercises and Tutorials

Schedule/Submission:

hand-out / online on Monday before lecture, early turn-in on following Monday by 10:00 local time

regular turn-in on following Wednesday by 10:00 local time

should work in groups of approx. 3, clearly give names on submission

please submit electronically by Mail to R. Albrecht and B. Westphal (cf. homepage); papersubmissions are tolerated

Ratingsystem:

“most complicated ratingsystem ever”

Admission points (good will rating, upper bound)

“reasonable proposal given student’s knowledge before tutorial”

Exam-like points (evil rating, lower bound)

“reasonable proposal given student’s knowledge after tutorial”

10% bonus for early submission.

Tutorial:

Plenary.

Together develop one good proposal, starting from discussion of the early submissions (anonymous).

– Formalia –
• Break:
  We'll have a 10 min. break in the middle of each event from now on, unless a majority objects now.

• Exam Admission:
  Achieving 50% of the regular admission points in total is sufficient for admission to exam. Typically, 20 regular admission points per exercisesheet.

• Exam Form:
  • oral for BSc and on special demand,
  • written for everybody else (if sufficiently many candidates remain).

Scores from the exercises do not contribute to the final grade.

• Mid-term Evaluation:
  We will have a mid-term evaluation (early December, roughly 1/3 of the course's time).

If you decide to leave the course early you may want to use a favour and tell us the reasons — by participating in the mid-term evaluation (will be announced on homepage).

Note: we're always interested in comments/hints/proposals/wishes/... concerning form or content. Feel free to approach us (tutors, me) in any form. We don't bite.

• Literature:
  • OMG: Unified Modeling Language Specification, Infrastructure, 2.1.2
  • OMG: Unified Modeling Language Specification, Superstructure, 2.1.2
  • OMG: Object Constraint Language Specification, 2.0
  • All three: http://www.omg.org (cf. hyperlinkson course homepage)

• Literature: Modelling
References


