

Software Design, Modelling and Analysis in UML

Lecture 1: Introduction

2013-10-21

Prof. Dr. Andreas Podolski, Dr. Bernd Westphal
Albert-Ludwigs-Universität Freiburg, Germany

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: "the tool" – Model-based/-Driven Software Engineering.
- UML: Mode of the Lecture: Blueprint.
- Contents of the course
- Formalia

Modelling

3/49

Disclaimer

- The following slides may raise thoughts such as:
 - "everybody knows this"
 - "completely obvious"
 - "trivial"
 - "duh"
 - "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.

4/49

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 furniture shall fit into the living room.
- The bathroom shall have a window.
- The cost shall be in budget.

Wanted: a house which satisfies the requirements.

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

- Consists of a huge number of bricks.
- Consists of subsystems, such as windows.
- Water pipes 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...?

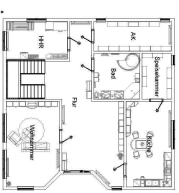
5/49

Approach: Floorplan

1. Requirements

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

2. Design



3. System



Observation: Floorplan abstracts from, e.g., ...

- kind, number, and placement of bricks.
- water pipes/wiring, and subsystem details (e.g., window style).

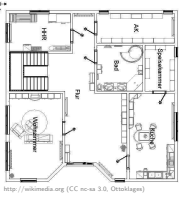
6/49

Approach: Floorplan

1. Requirements

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

2. Design



3. System



Observation: Floorplan preserves, e.g., ...

- house and room extensions (to scale)
- presence/absence of windows and doors,
- placement of subsystems (such as windows).

6/9

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

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

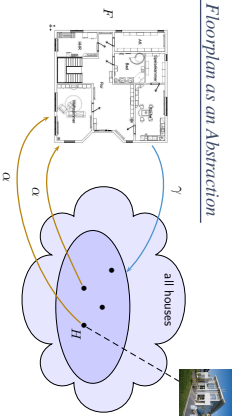
Example:

- 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?

9/9

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

7/9

Good for Anything Else? Documentation.

- **Given:** a house.
- **Want:** a concise description for potential buyers.
- **Approach:** draw a floorplan.



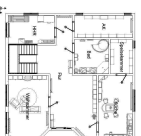
Distinguish:

- 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**.

10/9

What is it good for? Build by Plan.

- As said before, the floorplan abstraction α **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 ϕ , and **if** α/γ preserve this property, then H' has (or will have) property ϕ .
- 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.
 - Cost shall be in budget.
- And: it's typically easier (and cheaper) to correct errors in the plan, rather than in the finished house.



8/9

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. [Ginz, 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:

- the **image attribute** (Abbildungsmerkmal), i.e. there is an entity (called **original**) whose image or archetype the model is;
- the **reduction attribute** (Verfälschungsmerkmal), i.e. only those attributes of the original that are relevant in the modeling context are represented;
- the **pragmatic attribute**, i.e. the model is built in a specific context for a specific purpose.

11/9

- Needed: A Modelling Language for SW-Engineering
- What would be a "from scratch" approach?
- (i) Define a **formal language** to define requirements and designs.
 - (ii) Equip it with a **formal semantics**.
 - (iii) Define consistency/satisfaction relation in terms of semantics.
- The approach in this course:
- (i) Introduce a common **semantical domain** — what is a very abstract mathematical characterisation of **object-based transitions systems**? *Why? Because in the end SW-Engineering is about the creation of (object based) transitions systems and modeling is about describing them.*
 - (ii) Take (a fragment of) the visual formal language **UML** as syntax.
 - (iii) Introduce an abstract mathematical representation of diagrams. *Why? Because it is easier to handle than pictures. It abstracts from the visual representation of diagrams and from the visual representation of the semantics — more: in four steps (if does)*
 - (iv) Study the **UML** standard documents for the **informal semantics**.
 - (v) Define a mapping from (abstract representations of) diagrams to the **semantical domain**: **assign meaning to diagrams**.
 - (vi) Define (in terms of the meaning) when a diagram is, e.g., **consistent**.

- We see **software M** as a **transition system**.
- It has a (possibly infinite) set of states S .
 - an initial state s_0 , and
 - a (possibly Labelled) transition relation
- $$— S \times L \times S$$
- (behaviour)

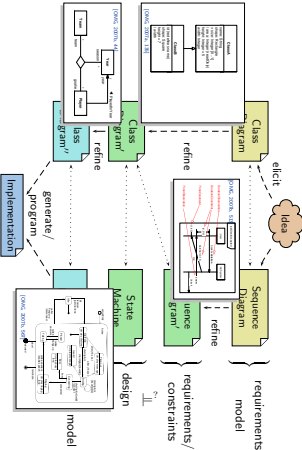
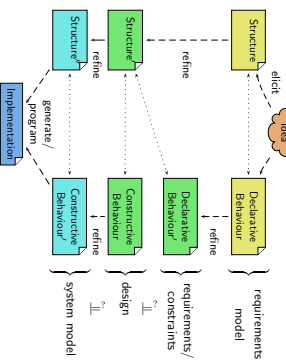
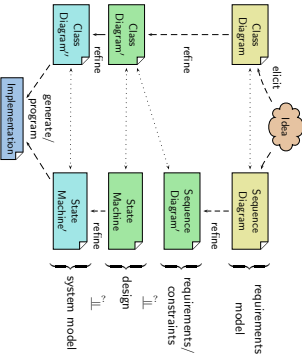
Software may have infinite and finite runs, i.e. sequences of consecutive states.

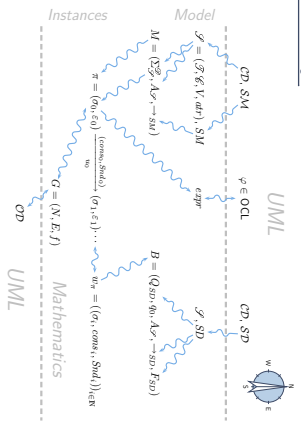
The **software engineering problem**:

- **Given:** Informal requirements φ .
- **Desired:** correct software, i.e. software M such that M satisfies φ .

Two prominent obstacles:

- Getting φ formal in order to reason about φ and M , e.g. prove M correct.
- M typically too large to "write it down" at once.



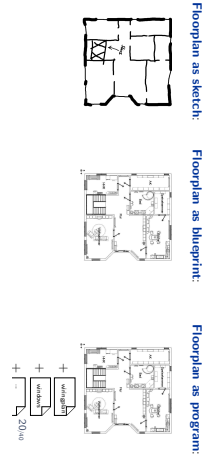


UML Mode

Consequences of the Pragmatic Attribute

Recall [Ginz, 2008, 425]:
 [...] (iii) the pragmatic attribute, i.e. the model is built in a specific context for a specific purpose.

Examples for context/purpose:



With UML it's the Same [http://martinfowler.com/33x3]

Actually, the last slide is inspired by **Martin Fowler**, who puts it like this:
 " [...] people differ about what should be in the UML because there are differing fundamental views about what the UML should be."

I came up with three primary classifications for thinking about the UML: **UML as Sketch**, **UML as Blueprint**, and **UML as Programming Language**.
 [...] S. Maber independently came up with the same classifications.)

So when someone else's view of the UML seems rather different to yours, it may be because they use a different **UML mode** to you."

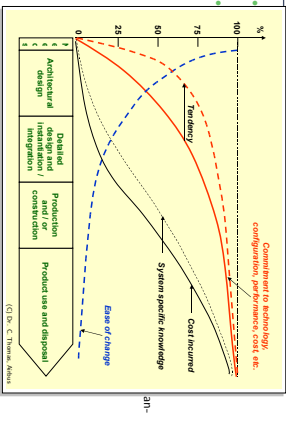
- Claim:
- And this not only applies to UML as a language (what should be in it?)
 - but at least as well to individual UML models.

With UML it's the Same [http://martinfowler.com/33x3]

A	Sketch	Blueprint	Programming Language
C	In this UML mode developers communicate some aspects of a system. [...] Sketches are also useful in documents, in which case the designer should be clear rather than comprehensive. [...] The tool used for sketching are lightweight drawing tools and often people aren't too concerned about the quality of the diagram. Most UML diagrams shown in books, such as mine, are particular about looking to represent a specific aspect of a system. Most UML diagrams shown in books, such as mine, are particular about looking to represent a specific aspect of a system. Most UML diagrams shown in books, such as mine, are particular about looking to represent a specific aspect of a system.	[...] In forward engineering, sketches are developed by a designer whose job is to build a defined design for a system. That design should be sufficiently complete that all design decisions are laid out in a fairly straightforward activity that requires little thought. [...] The details required for the sketch are more sophisticated tools than sketches in order to handle the details required for the sketch.	If you can draw the UML, you can draw the system. If you need to use a software, you can draw the UML. If you can draw the UML, you can draw the system. [...] The premise of this is that the question of course is not whether you can draw the UML, but whether you can draw the system. I don't believe that graphical programming will succeed just because it's graphical. [...] to hold the information. [...]

UML-Mode of the Lecture: As Blueprint

- The "node" fitting the lecture best is **AsBlueprint**.
- The purpose of the lecture's formal semantics is:
 - to be precise to avoid misunderstandings
 - to allow formal analysis of consistency/implication on the design level — find errors early
- while being consistent with the (informal semantics) from the standard [OMG, 2007a, OMG, 2007b] as far as possible.



- The 'mode' fitting the lecture best is **AsBlueprint**.
- The purpose of the lecture's formal semantics is:
 - to be precise to avoid **misunderstandings**
 - to allow formal analysis of consistency/implication on the **design level** — find errors early, while being consistent with the (informal semantics) from the standard [OMG, 2007a, OMG, 2007b] as far as possible.

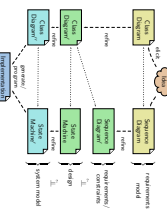
Plus:

- Being precise also helps for mode **AsSketch**: it should be easier to "fill in" missing parts or resolve inconsistencies.
- Lecture serves as a starting point to define your semantics for your context/purpose (maybe obtaining a **Domain Specific Language**).
- Lecture could be worked out into mode **AsProgrammingLanguage**.

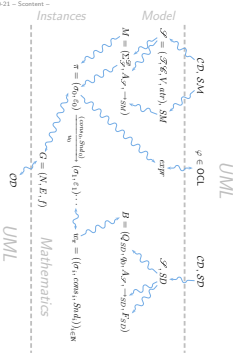
COURSE OVERVIEW

Table of Contents

- Motivation and Overview (VL 01)
- Semantical Domain (VL 02)
- OCL (VL 03)
- Object Diagrams (VL 04)
- Modeling Structure: (VL 05-09)
- Class Diagrams (VL 10-15)
- Modeling Behaviour (VL 16-18)
- State Machines (VL 19-20)
- Constructive: (VL 21)
- State Machines (VL 22)
- Reflective: (VL 22)
- Live Sequence Charts (VL 22)
- Inheritance (VL 16-18)
- Meta-Modeling (VL 19-20)
- Putting it all together: (VL 22)
- MDA, MDSE



Course Path: Over Map



Course Path: Over Time

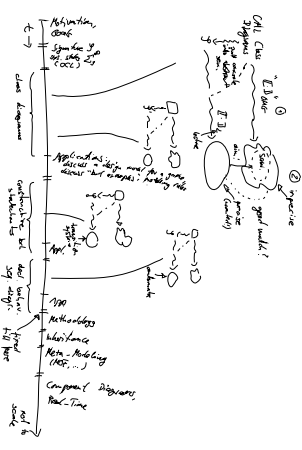


Table of Non-Contents

- **Everything else, including**
- **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

27th

Formalia: Lectures

- **Course language: English**
(slides/writing, presentation, questions/discussions)
- **Presentation:**
half slides/half on-screen **hand-writing** — for reasons
- **Script/Media:**
 - slides with annotations on **homepage**, 2-up for printing, typically soon **after** the lecture
 - recording on **lectures portal** with max. 1 week delay (link on **homepage**)
- **Interaction:**
absence often moaned but **it takes two**,
so please ask/comment immediately

30th

Formalia

— 1 - 2013-10-21 - Sternia -

28th

Formalia: Dates/Times

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

Week N,	Monday,	10-12 lecture A2
Week N + 1,	Monday,	10:00 10-12 lecture B1 (exercise A early submission)
	Wednesday,	10-12 lecture B2 (exercise sheet B online)
	Wednesday,	10-12 lecture B3 (exercise A late submission)
Week N + 2,	Monday,	10-12 tutorial A
	Wednesday,	10-12 lecture B3 (exercise B early submission)
Week N + 3,	Monday,	10:00 10-12 lecture C1 (exercise sheet C online)
	Wednesday,	10:00 10-12 tutorial B (exercise B late submission)

With a prefix of lectures, see homepage for details.

31st

Formalia: Event

- **Lecturer:** Dr. Bernd Westphal
- **Support:** Rebecca Albrecht
- **Homepage:**
<http://se.inf.uni-leipzig.de/~frenchling/se2013-14/semester1>
- **Questions:**
 - **"online":**
 - (i) ask immediately or in the break
 - **"offline":**
 - (i) try to solve yourself
 - (ii) discuss with colleagues
 - (iii) Exercises: contact tutor by mail (cf. homepage)
 - Rest: contact lecturer by mail (cf. homepage) or just drop by: Building 52, Room 00-020

— 1 - 2013-10-21 - Sternia -

29th

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 20:00 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), **paper submissions** are **tolerated**
- **Rating system:** "most complicated rating system 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)

— 1 - 2013-10-21 - Sternia -

32nd

Formula: Break

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

33/0

- 1 - 2013-10-21 - Sternella -

Formula: Exam

- **Exam Admission:**
 - Achieving 50% of the regular **admission points** in total is **sufficient** for admission to exam.
 - Typically, 20 regular admission points p per exercise sheet.
- **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.

33/0

- 1 - 2013-10-21 - Sternella -

Formula: Evaluation

- **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 earlier you may want to do us 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.**

33/0

- 1 - 2013-10-21 - Sternella -

Literature

36/0

- 1 - 2013-10-21 - main -

Literature: UML

- 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. hyperlinks on course homepage)
- A. Kleppe, J. Warner: The Object Constraint Language, Second Edition, Addison-Wesley, 2003
- D. Harel, E. Gery: Executable Object Modeling with Statecharts, IEEE Computer, 30(7):31-42, 1997.
- B. P. Douglass: Doing Hard Time, Addison-Wesley, 1999
- B. P. Douglass: ROSES: Rapid Object-Oriented Process for Embedded Systems, i-Logix Inc., Whitepaper, 1999.
- B. Osterriedt: Analyse und Design mit UML 2.1, 8. Auflage, Oldenbourg, 2006.
- H. Stoerle: UML 2 für Studenten, Pearson Studium Verlag, 2005.

36/0

- 1 - 2013-10-21 - SS -

Literature: Modelling



- W. Hesse, H. C. Mayer: Modellierung in der Softwareentwicklung, eine Betrachtung in der Informatik Spektrum, 31(5):377-393, 2008.
- O. Pastor, S. España, J. I. Panach, N. Aquino: Model-Driven Development, Informatik Spektrum, 31(5):394-407, 2008.
- M. Ginz: Modellierung in der Lehre an Hochschulen: Thesen und Erfahrungen, Informatik Spektrum, 31(5):408-424, 2008.

<http://www.springerLink.com/content/0170-6012>

- U. Kastens, H. Kleine Büning: Modellierung – Grundlagen und Formale Methoden, 2. Auflage, Hanser-Verlag, 2008.

36/0

- 1 - 2013-10-21 - SS -

38/0

Questions?

39/40

References

- [Dobing and Parsons, 2006] Dobing, B. and Parsons, J. (2006). How UML is used. *Communications of the ACM*, 49(5):1109–114.
- [Ginz, 2008] Ginz, M. (2008). Modellierung in der Lehre an Hochschulen: Theorien und Erfahrungen. *Methodik Spektrum*, 31(3):425–434.
- [OMG, 2007a] OMG (2007a). Unified modeling language: Infrastructure, version 2.1.2. Technical Report formal/07-11-04.
- [OMG, 2007b] OMG (2007b). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.

40/40