Software Design, Modelling and Analysis in UML

Lecture 1: Introduction

2011-10-25

Prof. Dr. Andreas Podelski, 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.
  • be able to decide whether you want to stay with us...

• Content:
  • Analogy: Model-based/-driven development by construction engineers.
  • Software engineers: “me too” — Model-based/-driven Software Engineering.
  • UML Mode of the Lecture: Blueprint.
  • Contents of the course

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 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?...

Approach: Floorplan

1. Requirements

2. Design

3. System

Observation: Floorplan abstracts from, e.g.,

• kind, number, and placement of bricks,
• subsystem details (e.g., window style),
• water pipes, wiring, and...
Approach: Floorplan

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.
   • Cost 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 $A$ has window if and only if representation in $\gamma(F)$ 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 $\alpha/\gamma$ preserve 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.
  • Cost 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 x 15m and the coach is 1m x 1m?

Good for Anything Else? Documentation.

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

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. [Eisen, 2008, 62]
A model is a concrete or mental image (Vorbild) of something or a concrete or mental archetype (Vorlage) for something. Three properties are constituent:
1. the image property (Abbildungsmerkmal), i.e., there is an entity (called original) whose image or archetype the model is.
2. the reduction property (Verkürzungsmerkmal), i.e., only those attributes of the original that are relevant in the modelling context are represented.
3. the pragmatic property, i.e., the model is built in a specific context for a specific purpose.
Model-Driven Software Engineering

Software System (Very Abstract View)

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 $L$-labelled) transition relation $\rightarrow \subseteq S \times L \times S$.

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

The software engineering problem:

- Given: informal requirements $\phi$.
- Desired: correct software, i.e., software $M$ such that $M$ satisfies $\phi$.

Two prominent obstacles:

- Getting $\phi$ formal in order to reason about $\phi$ and $M$, e.g., prove $M$ correct.
- $M$ typically too large to "write it down" at once.
Consequences of the Pragmatic Attribute

Recall [Gömez, 2008, 425]:

 [...] (4) the pragmatic attribute, i.e. the model is built in a specific context for a specific purpose.

Examples for context (purpose):

- Floorplan as sketch
- Floorplan as blueprint
- Floorplan as program

With UML it’s the Same [http://martinfowler.com/bliki]

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.

- UMLAsProgrammingLanguage, and
- UMLAsBlueprint, and
- UMLAsSketch

(...E. Mok are 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 see a different UMLModel to you.”

Claims:

- 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/bliki]

Needed: A Modelling Language for SW-Engineering

• What would be a “from-scratch” approach?
  (1) Define a formal language to define requirements and designs.
  (2) Equate it with a formal semantics.
  (3) Define consistency/transformation in terms of semantics.

• The approach in this course:
  (1) Introduce a common semantical domain — what is a very abstract mathematical characterization of object based transition systems?
  Why? Because in the end SW-Engineering is about the creation of (object based) transition systems and Modeling is about describing them.
  (2) Take (a fragment of) the visual formal language UML, as syntax.
  (3) Introduce an abstract mathematical representation of diagrams.
  Why? Because it is easier to handle that “pictures”, it abstracts from details such as graphical layout (which don’t contribute to the semantics — note: it doesn’t know it does).
  (4) Study the UML and design documents for the internal semantics.
  (5) Define a mapping from (abstract representations of) diagrams to the semantical domain, assign meaning to diagrams.
  (6) Define (in terms of the meaning) when a diagram is, e.g., consistent.
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, 2007b] as far as possible.

AsBlueprint: The “mode” fitting the lecture best is AsBlueprint.

- 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 content/purpose (maybe obtaining a Domain Specific Language).
- Lecture could be worked out into mode AsProgrammingLanguage.

Motivation and Overview (VL 01)
- Semantical Domain (VL 02)
- OCL (VL 03)
- Object Diagrams (VL 04)
- Modeling Structure: Class Diagrams (VL 05-06)
- Modeling Behaviour: State Machines (VL 09-16)
- Live Sequence Charts (VL 17-19)
- Inheritance (VL 20-21)
- Meta-Meeting (VL 22)
- Putting it all together: MDA, MDSE (VL 23)
Table of Non-Contents

- 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, 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: Event

- Lecturer: Dr. Bernd Westphal
- Support: Evis Plaku
- Homepage:
  http://swt.informatik.uni-freiburg.de/teaching/winter-term-2011-2012/sdmauml/sdmauml
- Questions:
  - "online":
    (i) ask immediately or in the break
  - "offline":
    (i) try to solve yourself
    (ii) discuss with colleagues
  - Exercises: constant tutor by mail (cf. homepage)
  - Rest: contact lecturer by mail (cf. homepage) or just drop by: Building 52, Room 00-020

Formalia: Dates/Times, Break

- Location:
  - Tuesday, Wednesday: here (Bldg. 108, room 00-007)

- Schedule:
  - Week X, Wednesday, 12–14 lecture
  - Week X+1, Tuesday, 12-14 lecture
  - Week X+1, Wednesday, 12–14 lecture
  - Week X+2, Monday, 9:00 (exercises X early submission)
  - Week X+2, Tuesday, 12:30 (exercises X late submission)
  - 12–14 tutorial

  With a prefix of lectures, see homepage for details.

- Break:
  - Unless a majority objects now:
    - we’ll have a 10 min. break in the middle of each event from now on.

Formalia: Lectures

- Course language: English
  (slides/writing, presentation, questions/discussions)
- Presentation:
  half slide/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:
  - silence often required but it takes two, so please ask/comment immediately
Formalia: Exercises and Tutorials

- **Schedule/Submission:**
  - **Hand-out on Wednesday after lecture.**
  - Early turn-in on following Monday by 9:00 local time.
  - Regular turn-in on following Tuesday by 12:00 local time.
  - Submit working groups of approx. 3.
  - Clearly give names on submission.
  - Paper submissions are tolerated.

- **Rating system:**
  - Admissions points (good-will rating, upper bound).
  - Exam-like points (evil ratings, lower bound).
  - 10% bonus for early submission.

- **Tutorial:**
  - Plenary.
  - Together develop one good proposal, starting from discussion of the early submissions (anonymous).

Formalia: Exam

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

Formalia: 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 favor 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.

Formalia: Literature

- **UML:**

- **Modelling:**

- **Textbooks:**

- **http://www.springerlink.com/content/0170-6012**
References


