Software Design, Modelling and Analysis in UML

Lecture 1: Introduction

2012-10-23

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.
    • thus 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.

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 thing.

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...?
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 $R$ has window in $H$ if and only if $R$-representation in $\alpha(H)$ has a 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.
  • 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 10 m wide and the couch is 11 m × 11 m?

Good for Anything Else? Documentation.

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

Distinguish:

- Sometimes the plan $F$ and the realisation $H \in \gamma(F)$ come first.
- Sometimes the realisation $H$ and the floorplan $F = \alpha(H)$ come 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.
(vi) Define (in terms of meaning) when a diagram is, e.g.,

Implementation

Abort/terminate

tone /enable speech

do/ play ringing

caller hangs up

callee answers

dial digit(n)

(vi) Define a mapping from (abstract representation of) diagrams to the

standard documents for the UML

UML

(iv) Study the

informal semantics

of(object based) transitions systems and Modeling is about the

mathematical characterisation of

desire: in floor plans (it does).

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

the mathematical nature of objects.

(iii) Introduce an abstract mathematical

representation of diagrams

Diagram

State

• Model-Driven Software Engineering with UML

Software System (Very Abstract View)
The purpose of the lecture's formal semantics is:

- to be precise to the programmer who puts it like this: Martin Fowler
- to be consistent with the (informal semantics) from the standard
- to allow formal analysis of consistency/implication
- to be developed by a designer whose job is to build a blueprint
- to be sufficiently complete that all designers can contribute
- to be detailed enough, and provide sufficient sketches for system construction
- to be more sophisticated than tools used for sketching

The last slide is inspired by S. Mellor independently came up with the same classifications.

Blueprints require much more effort than diagrams you draw and programming languages.

Their emphasis is on sketches in order to handle more sophisticated tools than Forward Engineering tools.

Actually, the last slide is inspired by S. Mellor independently came up with the same classifications.

A promising result is that blueprints are more successful and provide better code than current programming languages.

UML is a higher level language and thus more productive than standard 

Blueprints require much more thought than diagrams you draw and programming languages.

The promise of this is that forward engineering tools can take the UML and develop it into executable code.

The idea is that blueprints are more successful than forward engineering tools because it's graphical.

The tools used for sketching are light-weight drawing tools and often people aren't too good at drawing them.

Sketches are also useful in communicating rather than according to the pragmatism of the UML.

To you.

And this not only applies to UML modeling but at least as well to individual UML models.

Because it's graphical, forward engineering tools can take the UML and develop it into executable code just as a language would.

UmlAsBlueprint, UmlAsProgrammingLanguage, and UmlAsSketch are examples for context or purpose.

The purpose of the lecture's formal semantics is: find errors early, and provide pragmatic attribute.

In this UmlMode developers develop the designer the idea is that blueprints are more successful than forward engineering tools because it's graphical.

UML as Blueprint

Floorplan as blueprint

Floorplan as sketch

Floorplan as program

Examples for context or purpose

Model Instances

Recall: In this scenario, the model is built in a specific context for a specific purpose, i.e. the model is built in a specific context for a specific purpose.
Meta-Modeling (VL21)

• Inheritance (VL19–20)

• N, E, f

σ = (π \text{cons}, \ldots, ε, \text{cons})

Reflective: = refine

Constructive: | = 

• System model

State Machines

Live Sequence Charts (VL16–18)

Model Instances

• Domain Specific Language

Object Diagrams (VL04)

• Diagram

Domain Semantics (VL02)

Course Overview

The purpose of the lecture's formal semantics:

• Standard

While being consistent with the (informal semantics) from the standard, the design level to allow formal analysis of consistency/implication to avoid misunderstandings.

Course Path: OverMap

UML-Mode of the Lecture: As Blueprint

The "mode" fitting the lecture best is

• AsProgrammingLanguage

AsBlueprint

As Sketch

Plus: Standard

while being consistent with the (informal semantics) from the standard.
Table of 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.

Formalia:

• Lecturer: Dr. Bernd Westphal
• Support: Evis Plaku
• Homepage: http://swt.informatik.uni-freiburg.de/teaching/WS2012-13/sdmauml
• Questions:
  • "online": (i) ask immediately or in the break
  • "offline": (i) try to solve yourself
    (ii) discuss with colleagues
• Exercises: contact tutor by mail (cf. homepage)
• Rest: contact lecturer by mail (cf. homepage) or just drop by: Building 52, Room 00-020

Formalia:

• 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 of lectures portal with max. 1 week delay (link on homepage)
• Interaction: absence often moaned but it takes two, so please ask/comment immediately.

Formalia:

• Schedule/Submission:
  • hand-out on Wednesday after lecture, early turn in on following Monday by 12:00 local time
  • regular turn in on following Tuesday by 10:00 local time
  • should working group of approx. 3, clearly give names on submission
  • please submit electronically by Mail to E. Plaku and B. Westphal (cf. homepage); paper submissions are tolerated
• Ratingsystem:
  • 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:

• Location:
  • Tuesday, Wednesday: here (bldg. 51, room 03-026)
• Schedule:
  • Week N, Wednesday, 10–12 lecture (exercisesheet K online)
  • Week N + 1, Tuesday, 10–12 lecture
  • Wednesday, 10–12 lecture
  • Week N + 2, Monday, 12:00 (exercises early submission)
  • Tuesday, 10:00 (exercises late submission)
  • 10–12 tutorial
  • With a prefix of lectures, see homepage for details.
• Break:
  • We’ll have a 15 min break in the middle of each event from now on, unless a majority objects now.
Formalia: Exam

• Exam Admission: Achieving 50% of the regular admission points in total is sufficient for admission to the 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 early, you may want to do a favour and tell us the reasons – by participating in the mid-term evaluation (will be announced on the 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. hyperlink on course homepage)


Literature: Modeling


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


Questions?
References


