in real life

PA14[13]5
Software Design
Introduction

Mikael Svahnberg[1]

2016-03-08

[1] Mikael.Svahnberg@bth.se

About Me: Mikael Svahnberg

- Associate Professor, PhD in Software Engineering
- mailto:Mikael.Svahnberg@bth.se
- https://sites.google.com/site/mikaelsvahnberg/
- Interests:
    - Software Architectures, Software Architecture Evaluation, Software Architecture Evolution, Requirements Engineering, Large Scale Requirements Engineering, Market-Driven Requirements Engineering, Software Product Lines, Software Reuse, Empirical Research Methodology, Software Engineering Decision Support, Static Code Analysis, Software Architecture Reconstruction

Course Charter: PA1415

Efter genomförd kurs skall studenten:

- på en grundläggande nivå i grupp kunna ta fram krav på en programvara och uttrycka dem i en kravspecifikation
- i grupp producera en översiktlig utvecklingsprojektplan baserat på en kravspecifikation
- i grupp kunna skapa en detaljerad objektorienterad design för ett mjukvaruprogram
- i grupp kunna implementera ett mjukvaruprogram inom rimlig tid, baserat på en kravspecifikation och en objektorienterad design
- på en grundläggande nivå i grupp kunna planera och genomföra testning av producerad programvara, baserat på en kravspecifikation
- skapa och analysera objektorienterade artefakter uttryckta i UML
- kunna motivera och använda designmönster i utvecklingen av mjukvarusystem

Course Charter: PA1435

Kunskap och förståelse Efter genomförd kurs ska studenten:

- kunna visa förståelse för grundläggande principer i objektorienterad programvaruutveckling.
- kunna visa förståelse för UML som modelleringsspråk.
- kunna visa kunskap om grundläggande designprinciper.
- kunna visa kunskap om grundläggande designmönster.

Färdigheter och förmåga Efter genomförd kurs ska studenten:

- kunna uttrycka strukturen och beteendet hos ett system i termer av objektorienterade koncept.
- kunna korrekt använda UML för att uttrycka struktur och beteende hos ett system.
- kunna korrekt transformera en objektorienterad design till källkod.
- kunna tillämpa designprinciper och designmönster i allmänhet och inom en särskild domän.

Värderingsförmåga och förhållningssätt Efter genomförd kurs ska studenten:

- kunna analysera källkod för eventuella förbättringar.
- kunna analysera och kritiskt diskutera en design för eventuella förbättringar.

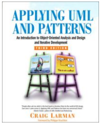
Course Structure

Already covered by @LKU, but:

- Lectures
- Assignments
    - Startup Seminars
    - Work/Submission
    - Feedback Meetings
        - For first Assignment, there is also a midway discussion meeting

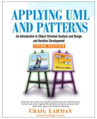
See course homepage on It's for deadlines etc.

Literature

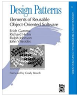

APPLYING UML
AND PATTERNS
An Introduction to Object-Oriented Analysis and Design
and Iterative Development
THIRD EDITION
CRAIG LARMAN

- C. Larman, *Applying UML and Patterns*, Prentice Hall, 3rd Edition
- Try to find an older edition!

Literature


APPLYING UML
AND PATTERNS
An Introduction to Object-Oriented Analysis and Design and Iterative Development
THIRD EDITION
CRAIG LARMAN

- C. Larman, *Applying UML and Patterns*, Prentice Hall, 3rd Edition
- Try to find an older edition!



- Gamma, Helm, Johnson, Vlissides, *Design Patterns, Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional

- R. Nystrom, *Game Programming Patterns*, Genever Benning, 2014.
- Also Available at: `http://gameprogrammingpatterns.com/contents.html`

Tools

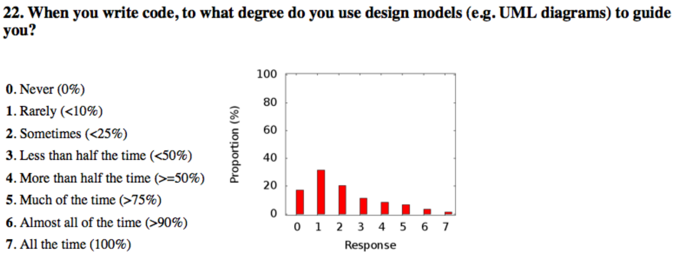
Any UML Tool will work, except pen and paper.

- `http://staruml.io/`
- `https://www.visual-paradigm.com/`
- `http://www.eclipse.org/papyrus/`
- `http://argouml.tigris.org/`
- `https://marketplace.eclipse.org/content/uml-designer`
- ...

Why Bother About Modelling

T. Gorschek, E. Tempero, L. Angelis, *On the use of software design models in software development practice: An empirical investigation*, in Journal of Systems and Software 95(2014):176–193.

- TL;DR: Nearly 4000 industry practitioners were asked “Do you model?”. Answers ranged from “no” to “hell no!”.



22. When you write code, to what degree do you use design models (e.g. UML diagrams) to guide you?

0. Never (0%)

1. Rarely (<10%)

2. Sometimes (<25%)

3. Less than half the time (<50%)

4. More than half the time (>=50%)

5. Much of the time (>75%)

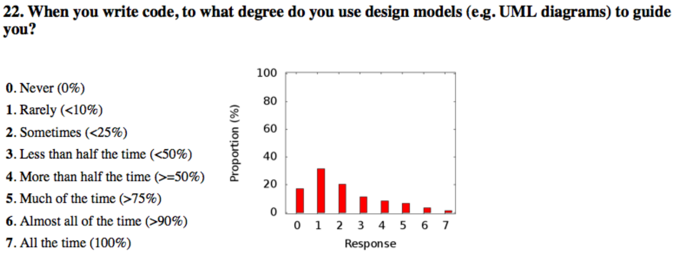
6. Almost all of the time (>90%)

7. All the time (100%)

Why Bother About Modelling

T. Gorschek, E. Tempero, L. Angelis, *On the use of software design models in software development practice: An empirical investigation*, in Journal of Systems and Software 95(2014):176–193.

- TL;DR: Nearly 4000 industry practitioners were asked “Do you model?”. Answers ranged from “no” to “hell no!”.
- . . . There is, of course, more to this story.



22. When you write code, to what degree do you use design models (e.g. UML diagrams) to guide you?

0. Never (0%)
1. Rarely (<10%)
2. Sometimes (<25%)
3. Less than half the time (<50%)
4. More than half the time (>=50%)
5. Much of the time (>75%)
6. Almost all of the time (>90%)
7. All the time (100%)

Why Bother About Modelling

- In the freetext answers a different story emerges:
    - They do use sketches, informal models, casual diagrams, etc, but not formal UML.
- Common explanations:
    - “Only for very complex designs, sometimes”
    - “Only use initially then start coding (diagrams not kept/updated)"
    - “Enables visualisation of the big picture/high level”
    - “Other types of models but not UML”
    - “Use models to communicate and coordinate with other developers”
- ∑ Models are not used as researchers expect. Instead they are used for conceptual analysis and exploration, problem solving, visualisation, and communication.

So, why bother?

- conceptual analysis and exploration
- problem solving
- visualisation
- communication

Also:

- This course trains you in a particular mindset, where you begin to analyse a problem in terms of its *objects* and their *interactions*.
    - This problem solving mindset is difficult to reach when bogged down with all the implementation details.
- This is the only place where you are expected to use an all-out thermonuclear UML approach to analysis and design.
    - Later on, you will cherry-pick models in order to understand/visualise/communicate a particular problem area better.
- Bear in mind that you throw out a few good things with the bath water too.

Development Phases

- Requirements
  - Problem formulation
  - Quality constraints of the system
  - Planning and estimations
- Analysis / Domain Analysis
  - Real World abstractions, mechanisms, relationships
- Design
  - Convert domain analysis into a technical solution
  - design patterns etc.
- Implementation
  - “Execution” of the design
- Testing
- Maintenance

Object Oriented Analysis and Design

- Object Orientation
  - Objects
  - Attributes
  - Relationships
  - Collaborations
  - Responsibilities
- OO Analysis
  - Problem domain and requirements
  - *Objects* in the problem domain
- OO Design
  - Logical Software Objects (with attributes and methods, plus collaborations)
- OO Construction/Implementation

OO Modelling

- A traceable chain from requirements to code/test.
  - Each model is transformed to a [more detailed] model that is closer to the end-product.
  - Do this fully, and you have *Model-Driven Development*
  - The overall idea is that
    - models are cheaper than code.
    - models are abstractions of code.
    - models are more rigorous than code :barf.png:
  - UML is *one* set of models.
  - RUP is the process used to transform the system through the UML graphs from requirements to code.

RUP/UML

- Rational Unified Process
- Unified Modelling Language

Process:

1. Use Case Diagrams / Use Cases
2. Conceptual Models / Domain Models
3. System Sequence Diagram
4. Class Diagrams
5. Sequence Diagrams / Interaction Diagrams
6. Goto (4)

Design Patterns

- Design patterns are reusable solutions to known problems
  - With known consequences
- There is nothing that *requires* you to use design patterns; they are a convenience.
- Design patterns focus primarily on structure (class view), and interaction (sequence diagrams).
  - Thus, we will come back to them later in the course.

Excercise

Discussion Forum

Design a Conceptual Model of a Discussion forum with categories, topics, posts, users, user profiles, and private messages. The system consists of a server park (including the database), a web client, and an android client.