University of Zurich[UZH]

Software Construction

Bertrand Meyer

University of Zurich
September-December 2017

Lecture 1: Introduction

Teaching team

Lectures: Bertrand Meyer

Head assistant: Jürgen Cito

Assistants:

- Livio Sgier
- Raphael Matile

About me

In industry until 2001
(President then CTO of Eiffel Software, Santa Barbara, California)

2001-2016: Professor of Software Engineering at ETH Zurich

Currently: Professor at Politecnico di Milano and head of Software Engineering Laboratory at Innopolis University (Kazan, Russia), plus Eiffel Software

Research areas: programming methodology, programming languages, formal methods, concurrent programming, software process (agile methods)

3

How to reach us

Bertrand Meyer, BIN 2.D.20, bertrand.meyer@inf.ethz.ch

Jürgen Cito, BIN 2.D.05, cito@ifi.uzh.ch

Raphael Matile, raphael.matile@uzh.ch

Livio Sgier, livio.sgier@gmail.com,

4

Slots

- Tuesdays
  - 8-9:45
- Wednesdays:
  - 8-9:45
  - 14-18 (in lecture weeks only)

Lecture weeks:

- September 19-20
- October 10-11
- November 7-8
- December 12-13

5

Course material

Course page:

www.ifi.uzh.ch/en/seal/teaching/courses/software-construction-17.html

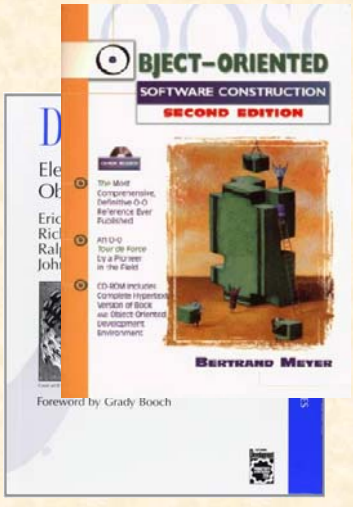
→ Check it regularly

Lecture material:

- Lecture slides
- Recommended textbooks:

  B. Meyer: *Object-Oriented Software Construction, 2nd edition*
  Prentice Hall, 1997

  E. Gamma et al.: *Design Patterns*
  Addison-Wesley, 1995



6

Supplementary recommended books

A good software engineering textbook (see precise references on course page):

- Ghezzi / Jazayeri / Mandrioli
  (broadest scope)
- Pfleeger / Atlee
  (the most recent)
- Pressman
  (emphasis on practitioners' needs)

On patterns: Karine Arnout's ETH PhD thesis

Goal of the course

(From the course description in the UZH page, abridged)

Knowing how to program does not make you a software engineer. The next step is to learn professional software development. This course is an introduction to both:

- *Software architecture:* methods for designing the structure of software systems, small or large, that will stand the test of time.
- *Software engineering:* methods and tools, including non-programming aspects (project management, requirements analysis, human factors, metrics, software processes including agile methods) necessary for producing successful systems.

Four blocks

1. *Key software engineering practices*:
   - Requirements analysis
   - Software testing and verification
   - Basics of configuration management
2. *Object-oriented design*:
   - Abstract data types
   - Information hiding,
   - The class concept
   - Inheritance
   - Principles of OO design (open-closed, command-query separation,...)
3. *Software architecture*:
   - Design by Contract
   - Classical design patterns
   - Architectural styles.
4. *Software process*:
   - Software process models
   - CMMI
   - Agile methods.

9

Grading

50% project, 50% end-of-semester exam

To pass, you must get ≥ 4.0 on both the project and the exam

About the exam:

- When: 9 January 2018, 8-10 AM
- What: all topics of semester
- How: no material allowed ("closed-book")

10

About the project

The project is an integral part of the course

Goal:

- Apply software architecture techniques
- Practice group work in software engineering
- Go through main phases of a realistic software project: requirements, design of both program and test plan, implementation, testing

More details about the project tomorrow (Wed 20/09, morning session)

11

Programming langage: Eiffel

First version 1985, constantly refined and improved since

Focus: software quality, especially reliability, extendibility, reusability

Emphasizes simplicity

Used for mission-critical projects in industry

Based on concepts of "Design by Contract".

International standard (ISO)

12

Some Eiffel-based projects

Axa Rosenberg
Investment management: from $2 billion to >$100 billion
2 million lines
Chicago Board of Trade
Price reporting system
Eiffel + CORBA +
Solaris + Windows + ...
Boeing
Large-scale simulations
of missile defense
HP printers

Swedish social security: accident reporting & management
etc.

13

More about Eiffel

- Method, language and environment
- Focus on software quality
- Applications: finance, aerospace, communication...
- Also used for teaching programming, "Object-Oriented Software Construction" and "Touch of Class" textbooks

- Pure OO approach
- Strong principles of OO design
- Design by Contract
- Seamless development

14

For the remainder of today

- What is software engineering?
- What is software architecture?
- An OO software architecture example

What is software engineering?

A definition of software engineering

From SWEBOK, the Software Engineering Body of Knowledge:

- **Software engineering** is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

17

A simpler definition

"The application of engineering to software"

Engineering (Wikipedia): "the discipline, art and profession of acquiring and applying technical, scientific, and mathematical knowledge to design and implement materials, structures, machines, devices, systems, and processes that safely realize a desired objective or invention"

A simpler definition of engineering: the application of scientific principles to the construction of artifacts

18

Parnas's view

(Cited in Ghezzi et al.)

"The multi-person construction of multiversion software"

19

For this course

The application of engineering principles and techniques, based on mathematics, to the development and operation of possibly large software systems satisfying defined standards of quality

20

"Large" software systems

What may be large: any or all of

- Source size (lines of code, LoC)
- Binary size
- Number of users
- Number of developers
- Life of the project (decades…)
- Number of changes, of versions

(Remember Parnas's definition)

Process and product

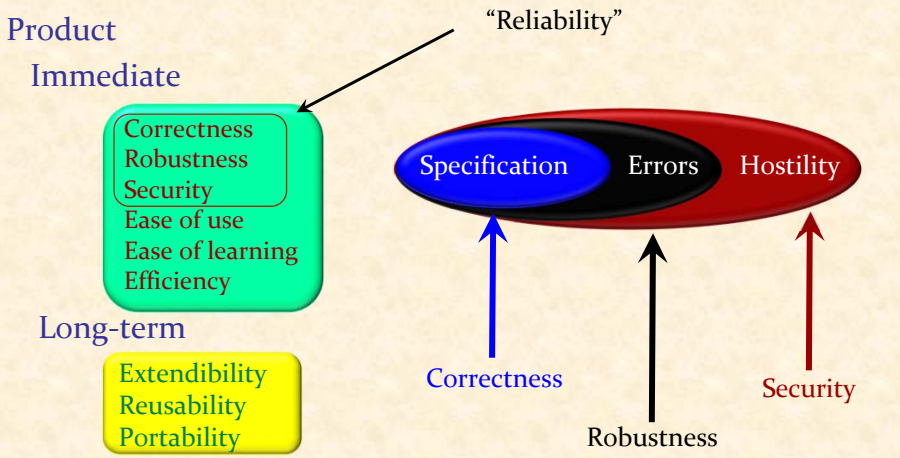
Software engineering affects both:

- Software products
- The processes used to obtain and operate them

Products are not limited to code. Other examples include requirements, design, documentation, test plans, test results, bug reports

Processes exists whether they are formalized or not

Software quality factors*



*Sometimes called the "ilities" list

23

Software engineering today

Three cultures:

- Process

- Agile

- Object

The first two are usually seen as exclusive, but all have major contributions to make.

24

Process

Emphasize:

- Plans
- Schedules
- Documents
- Requirements
- Specifications
- Order of tasks
- Commitments

Examples: Rational Unified Process, CMMI, Waterfall...

Agile

Emphasize:

- Short iterations
- Emphasis on working code; de-emphasis of plans and documents
- Emphasis on testing; de-emphasis of specifications and design . "Test-Driven Development"
- Constant customer involvement
- Refusal to commit to both functionality and deadlines
- Specific practices, e.g. Pair Programming

Examples: Extreme Programming (XP), Scrum

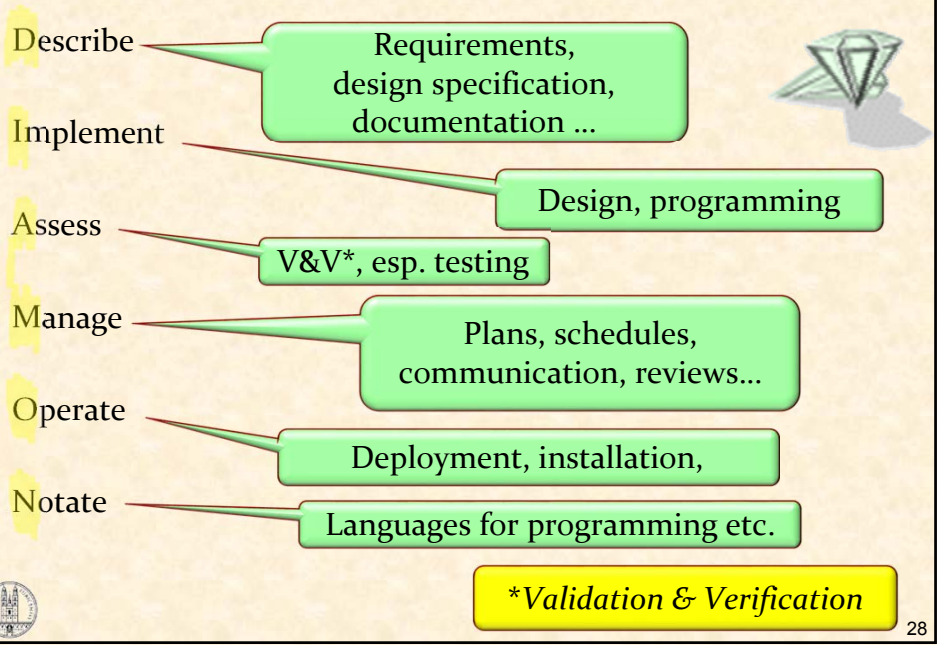
Object-oriented culture

Emphasizes:

- Seamless development
- Reversibility
- Single Product Principle
- Design by Contract

27

Six task groups of software engineering



28

What is software architecture?

Software architecture

We define software architecture as
*The decomposition of software systems into modules**

Primary criteria: extendibility and reusability

Examples of software architecture techniques & principles:

- Abstract data types (as the underlying theory)
- Object-oriented techniques: the notion of class, inheritance, dynamic binding
- Object-oriented principles: uniform access, single-choice, open-closed principle…
- Design patterns
- Classification of software architecture styles, e.g. pipes and filters

* From the title of an article by Parnas, 1972

Software architecture: a few milestones

1968: *The inner and outer syntax of a programming language* (Maurice Wilkes)

1968-1972: Structured programming (Edsger Dijkstra); industrial applications (Harlan Mills & others)

1971: *Program Development by Stepwise Refinement* (Niklaus Wirth)

1972: David Parnas's articles on information hiding

1974: Liskov and Zilles's paper on abstract data types

1975: *Programming-in-the-large vs Programming-in-the-small* (Frank DeRemer & Hans Kron)

1987: *Object-Oriented Software Construction*, 1st edition

1994: *An introduction to Software Architecture* (David Garlan and Mary Shaw)

1995: *Design Patterns* (Erich Gamma et al.)

1997: UML 1.0

2000: REST (Roy Fielding)

What we have seen

Basic definitions and concepts of software engineering

Basic definitions and concepts of software architecture