Analysis, Design, and Software Architecture (BDSA)

Paolo Tell

Introduction to Software Engineering

IT University
of Copenhagen

Go to **www.menti.com** and use the code **5023 9124**

Which topics do you expect or wish to be covered in BDSA?

Outline

- Literature
  - [OOSE] ch. 1
  - Optional reading [SE10] ch. 1+2
  - [Article] F. P. J. Brooks, "No Silver Bullet Essence and Accidents of Software Engineering," in Computer, vol. 20, no. 4, pp. 10-19, April 1987.

- Introduction to Software Engineering
  - History
  - FAQ about SE

- What is SE?

- Process activities
  - Software specification
  - Software development
  - Software validation
  - [Software evolution]

Introduction to Software Engineering

IT University
of Copenhagen

Why Software Engineering?

- More and more systems are software controlled.
- Nowadays, the expenditure on software represents a significant fraction of gross national product (GNP) in all developed countries.
- Individual approaches were unable to scale up to larger and more complex systems.
- The “software crisis” (Naur and Randell, 1969), the birth of “software engineering” in 1968.
  - http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF
- Between the 1970s and 1980s, a variety of software engineering techniques and methods were developed together with tools and standardized notations…
- … and this trend is still continuing.
- … however, after more than 50 years, developing software is still a challenging endeavour that can fail.

Ariane 5 Flight 501

https://www.youtube.com/watch?v=gp_D8r-2hwk

- 4th June 1996. Approximately 37 seconds after a successful lift-off, the Ariane 5 launcher lost control (a 370 kk dollar firework).
- The crash report identified a software bug as the direct cause (integer overflow).
- Incorrect control signals were sent to the engines and these swivelled so that unsustainable stresses were imposed on the rocket.
- It started to break up and was destroyed by ground controllers.
- The system failure was a direct result of a software failure. However, it was symptomatic of a more general systems validation failure.
  - http://en.wikipedia.org/wiki/Ariane_5_Flight_501
  - Inertial reference platform from the Ariane 4

FAQs about software engineering

- What is software?
- What is software engineering?
- What is the difference between software engineering and computer science?
- Why is software development difficult?

What is software?

- Computer programs and associated documentation [1, p.20], e.g., requirements, design models, and user manuals.

- New software can be created by developing new programs, configuring generic software systems, or reusing existing software.

- Software products may be
  - Generic – developed to be sold to a general market, e.g., Excel or Word.
  - Custom – developed for a single customer according to their specification.

[1] Sommerville, Ian. "Software Engineering, 10th edition". Pearson Education Limited, 2016.

What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production from initial conception to operation and maintenance [1, p.20].
- "Programming is fun, but developing quality software is hard." [2, p.xv]
- Software engineers should adopt a systematic approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints, and the resources available.

[1] Sommerville, Ian. "Software Engineering, 10th edition". Pearson Education Limited, 2016.
[2] Larman, Craig. "Applying UML and patterns". Pearson Education, 2012.

What is the difference between SE and computer science?

- Computer science focuses on theory and fundamentals [1, p.20].
- Software engineering is concerned with the practicalities of developing and delivering useful software [1, p.20].
- Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike, e.g., physics and electrical engineering).

[1] Sommerville, Ian. "Software Engineering, 10th edition". Pearson Education Limited, 2016.

Why is software development difficult?

- The problem domain (also called application domain) is difficult.
- The solution domain is difficult.
- The development process is difficult to manage.
- Software offers extreme flexibility.

- Software is a discrete system
  - Continuous systems have no hidden surprises
  - Discrete systems can have hidden surprises! (Parnas)

- No silver bullet [1]:
  - Complexity
  - Conformity
  - Changeability
  - Invisibility

David Lorge Parnas is an early pioneer in software engineering who developed the concepts of modularity and information hiding in systems which are the foundation of object oriented methodologies.

[1] FP Jr, Brooks. "No Silver Bullet Essence and Accidents of Software Engineering." Computer 4 (1987).

- Modelling activity
- Problem solving activity
- Knowledge acquisition activity
- Rationale management activity

What is Software Engineering?

Software engineering is a modelling activity

- Application domain.
- Solution domain.
- Object-oriented methods combine the application domain and the solution domain modelling activities into one.

Software engineering is a problem solving activity

Problem-solving activity

1. Formulate the problem
2. Analyse the problem
3. Search for solutions
4. Decide on the appropriate solution
5. Specify the solution

OOSD

1. Requirement elicitation
2. Analysis
3. System design
4. Object design
5. Implementation
6. Testing

Software engineering is a knowledge acquisition activity

- Knowledge acquisition is not linear

Software engineering is a rational management activity

- Application domain eventually stabilises.
- Solution domain is constantly changing.
- How can we reason about a decision taken in the past?

Techniques, Methodologies, and Tools

- Techniques (Methods):
  - Formal procedures for producing results using some well-defined notation (e.g., algorithms, software processes)

- Methodology:
  - Collection of techniques applied across software development and unified by a philosophical approach (e.g., Agile methods)

- Tools:
  - Instruments or automated systems to accomplish a technique (e.g., Visual Studio, continuous integration)

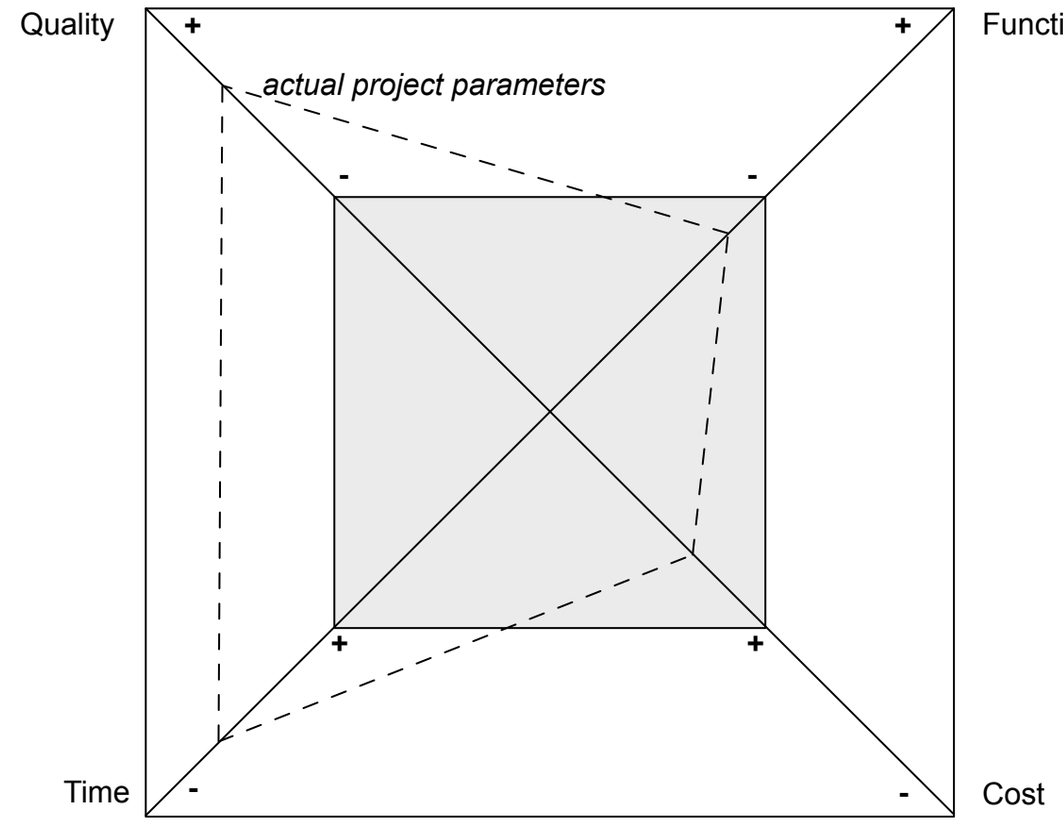
Software Engineering: A Working Definition

Software Engineering is a collection of <u>techniques</u>, <u>methodologies,</u> and <u>tools</u> that help with the production of *a high quality* software system developed with a given *budget* before a given *deadline* while *change* occurs

Challenge: Dealing with complexity and change

Software Project Management

***A high quality software* system developed with a given *budget* before a given *deadline* while *change* occurs**



Challenge: Dealing with complexity and change

- Software specification
- Software development
- Software validation
- Software evolution

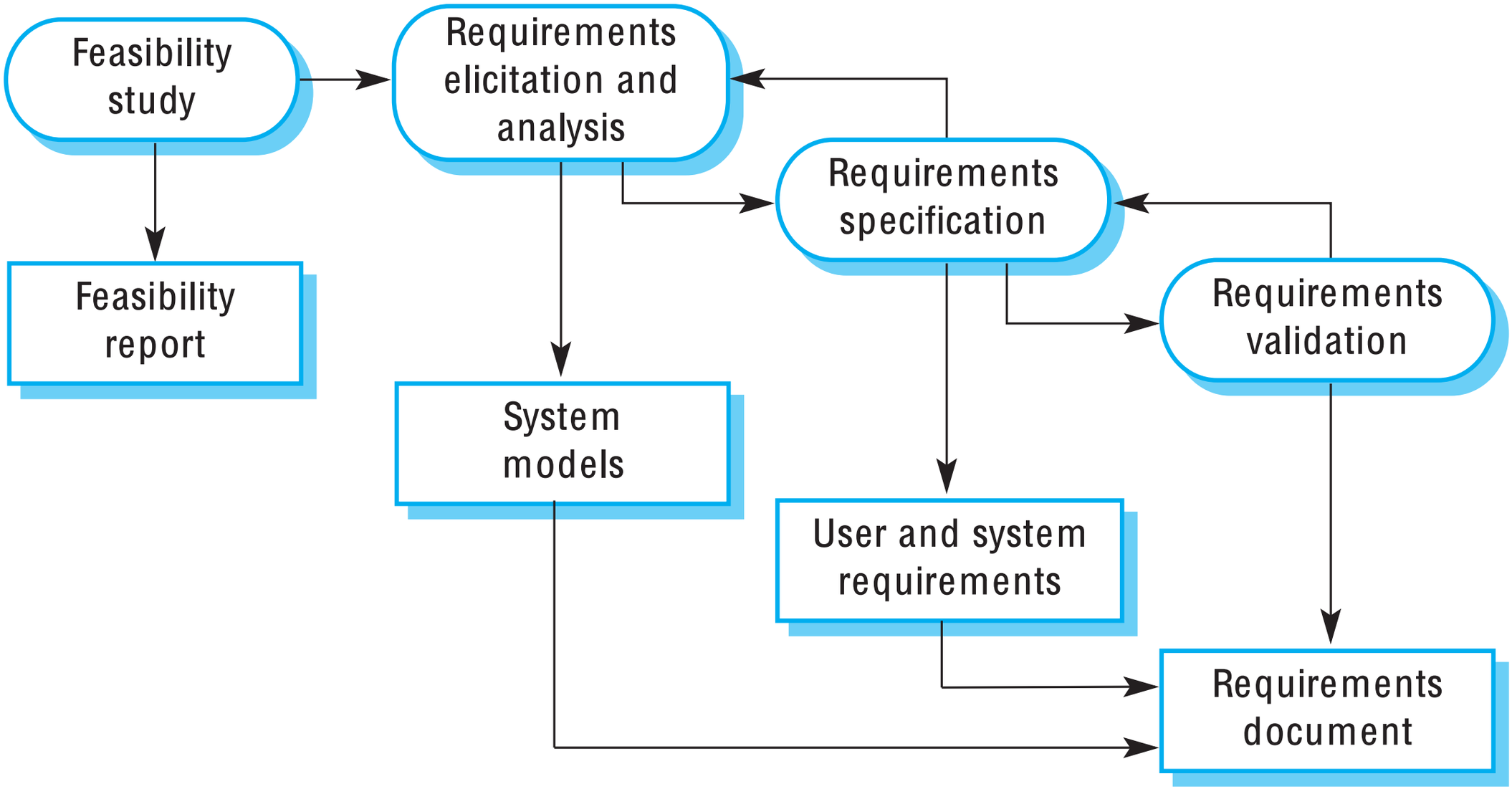
Process Activities

Software specification [or requirement engineering]

- The process of establishing:
  - what services are required (functional requirements) and
  - the constraints on the system's operation and development (non-functional requirements).

- Requirements engineering process
  - Feasibility study
    - Is it technically and financially feasible to build the system?
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
  - Requirements specification
    - Defining the requirements in detail
  - Requirements validation
    - Checking the validity of the requirements (realism, consistency, and completeness)

The requirements engineering process



[1] Sommerville, Ian. “Software Engineering, 10th edition”. Pearson Education Limited, 2016.

Software development [design and implementation]

- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realizes the specification;
- Implementation
  - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be inter-leaved.

Main design activities

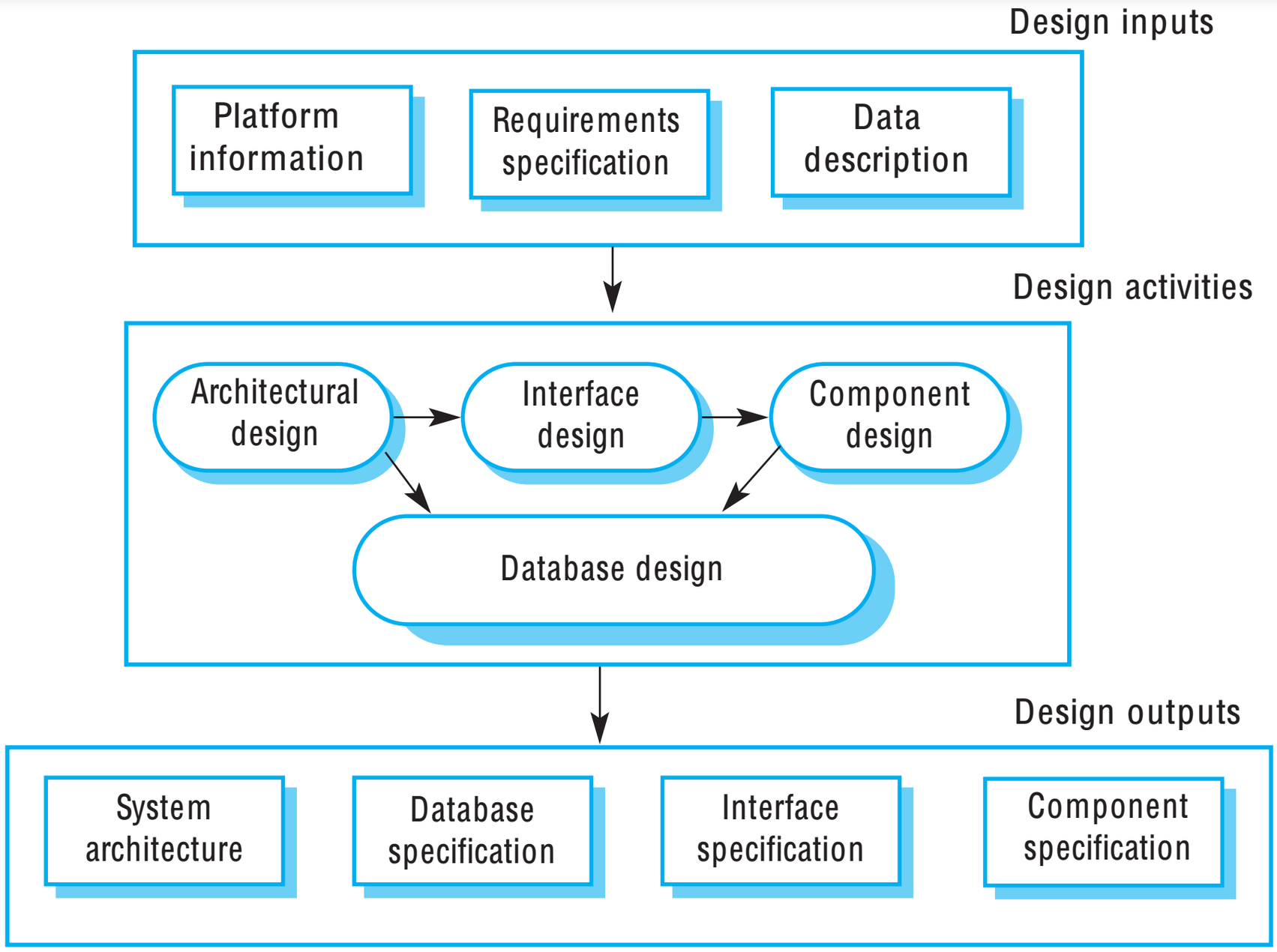
- Architectural design
  - Defining the overall structure of the system, the principal components (sub-systems, modules), their relationships and how they are distributed.

- Interface design
  - Defining the interfaces between system components.

- Component design
  - Defining each system component and design how it will operate.

- Database design
  - Designing the system data structures and how these are to be represented in a database.

A general model of the design process



[1] Sommerville, Ian. "Software Engineering, 10th edition". Pearson Education Limited, 2016.

Software Validation [verification and validation]

- Verification and validation (V&V) is intended to show that a system conforms to its specification and meets the requirements of the customers.
  - Verification: "Are we building the product right?"
  - Validation: "Are we building the right product?"

- Involves checking and review processes and system testing.

- Testing is the most commonly used V&V activity.

Main Testing Stages

- Development or component testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.

- System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important. Test cases can be derived from the specification of the real data to be processed by the system.

- Acceptance testing
  - Testing with customer data to check that the system meets the customer's needs.

Software evolution [and maintenance]

DevOps, Software Evolution and Software Maintenance, BSc

Spring 2022 | BSc in Software Development

The course "DevOps, Software Evolution and Software Maintenance" is a BSc elective.

In this course, the students will discover all the software engineering activities that take place after an initial software product is delivered or after a legacy system is taken over from a theoretical and practical perspective. Students (in groups) will take over such a system that is live and serving users, they will refactor and migrate it to the languages and technologies of their liking. All subsequent DevOps, software evolution and software maintenance activities will be performed directly on the systems of the students.

ECTS 7.5

Semester Spring 2022

Language English

Exam D: Submission of written work with following oral, Internal (7-point scale)

Teacher

Programme BSc in Software Development

This course is offered as a single subject

This course is offered to guest students

Read more

The software process

- A structured set of activities that leads to the production of a software system.
- Activities common to all software processes:
  - Software specification;
  - Software development;
  - Software validation;
  - Software evolution.

- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

Examples of software process models

- The waterfall model
  - Plan-driven model.
  - Separate and distinct phases of specification and development.

- Incremental development
  - Specification, development, and validation are interleaved.

- Reuse-oriented software engineering
  - The system is assembled from existing components. Maybe using plan-driven or iterative/agile.

- Scrum
  - Agile framework.
  - The system is developed during a set of "sprints" where input from a customer proxy is implemented.

Summing up

Key Points I

- Software is key to most technical aspects of modern life
  - pervasive, complex, costly, inevitable, evolving, ...

- Software engineering is an engineering discipline that is concerned with <u>all</u> aspects of software production.

- Software engineering is a <u>systematic</u> approach, that
  - depending on the problem and resource constraints
  - uses appropriate methodologies, techniques, and tools.

Key Points II

- Requirements engineering
  - is the process of developing a software specification.

- Design and implementation processes
  - concerned with transforming a requirements specification into an executable software system.

- Software validation
  - is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system.

- Software evolution
  - takes place when you change existing software systems to meet new requirements. The software must evolve to remain useful.

Key Points III

- Software processes are structured sets of activities involved in producing a software system. Common activities are:
  - specification, development, validation, and evolution.
- Software process models are an abstract description of how the process activities are organized.
- General process models (… more on these next semester in BSUP)
  - Examples of software process models

This Lecture

- Literature
  - [OOSE] ch. 1
  - Optional reading [SE9] ch. 1+2
  - [Article] F. P. J. Brooks, "No Silver Bullet Essence and Accidents of Software Engineering," in Computer, vol. 20, no. 4, pp. 10-19, April 1987.

- Introduction to Software Engineering
  - History
  - FAQ about SE

- What is SE?

- Process activities
  - Software specification
  - Software development
  - Software validation
  - [Software evolution]