Analysis, Design, and Software Architecture (BDSA)

Paolo Tell

Introduction to Object-Oriented Software Engineering (OOSE)

IT University of Copenhagen

Outline

- Literature
  - [OOSE] ch. 1
  - [SE9] ch. 1+2
- Introduction to Software Engineering
  - History
  - FAQ about SE
- What is SE?
- Software Processes
  - Software Process Models
  - Process Activities

IT University of Copenhagen

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 methodologies were developed together with tools and standardized notations…
- … and this trend is still continuing.
- … however, after more than 40 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 identifies 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

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 such as 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.

What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- “Programming is Fun – Developing Quality Software is Hard” (Craig Larman)
- 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.

What is the difference between SE and computer science?

- <u>Computer science</u> is concerned with theory and fundamentals.
- <u>Software engineering</u> is concerned with the practicalities of developing and delivering useful software.
- Computer science theories are still insufficient to act as a complete underpinning for software engineering (unlike e.g. physics and electrical engineering).

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)

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.

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

What is Software Engineering?

Software engineering is a modeling activity

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

Software engineering is a problem solving activity

OOSD

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

Problem-solving activity

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

Software engineering is a knowledge acquisition activity

- Knowledge acquisition is not linear

IT University of Copenhagen

Software engineering is a rational management activity

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

Techniques, Methodologies, and Tools

- Techniques:
  - Formal procedures for producing results using some well-defined notation (e.g. algorithms)
- Methodologies:
  - Collection of techniques applied across software development and unified by a philosophical approach (e.g. SCRUM)
- Tools:
  - Instruments or automated systems to accomplish a technique (e.g. Visual Studio, unit testing)

Software Engineering: A Working Definition

Software Engineering is a collection of techniques, methodologies, and tools 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 specification or Requirement Engineering
- Design and Implementation
- Validation
- Evolution

Software Processes

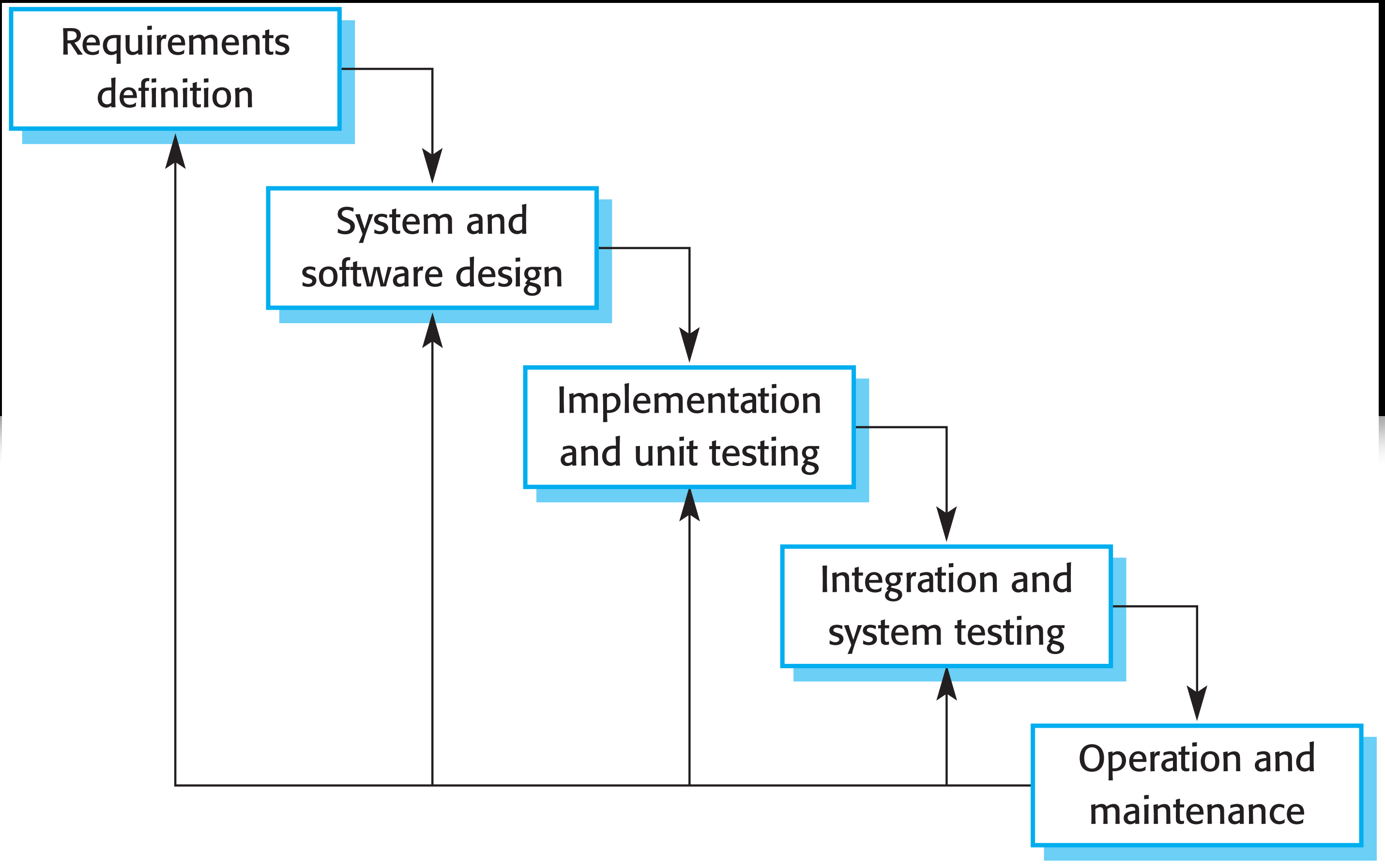
The software process

- A structured set of activities required to develop a software system.
- Activities common to all software processes:
  - Specification;
  - Design;
  - Validation;
  - Evolution.
- A software process model (or methodology) is an abstract representation of a process. It presents a description of a process from some particular perspective.

Plan-driven and agile processes

- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In iterative or incremental processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In agile processes, development of "shippable" software take precedence over planning and documentation
- In practice, most practical processes include elements of both plan-driven and iterative approaches.
- There are no right or wrong software processes.

Software process models


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Examples

- 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. May be plan-driven or iterative/agile.
- SCRUM
  - The system is developed during a set of "sprints" where customer input is implemented.

IT University of Copenhagen

Process activities

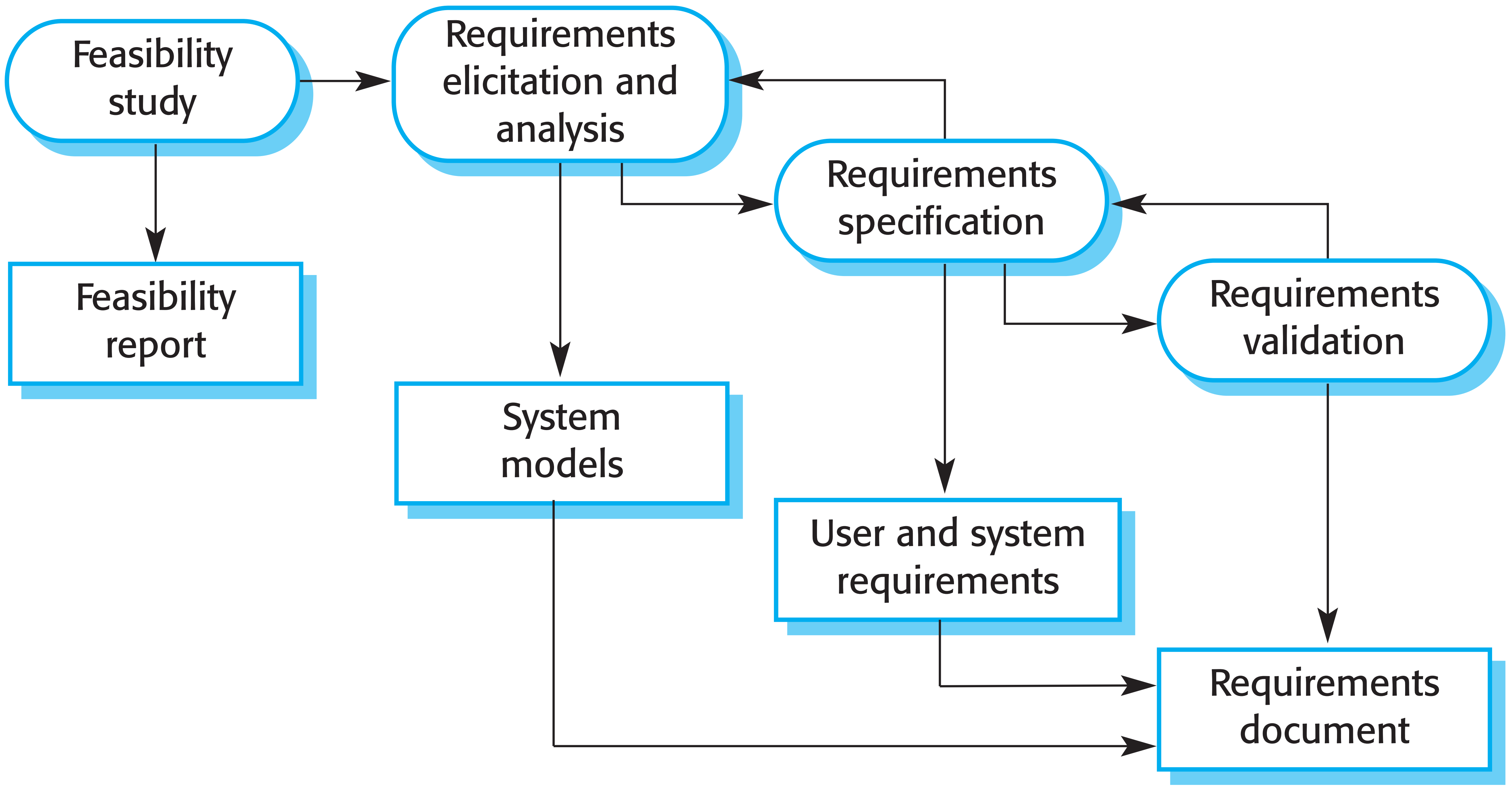
- Real software processes are inter-leaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing, and testing a software system.
- The four basic process activities of
  - specification,
  - development,
  - validation, and
  - evolution
- are organized differently in different development processes. In the *waterfall model*, they are organized in sequence, whereas in *incremental development* they are inter-leaved.

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)

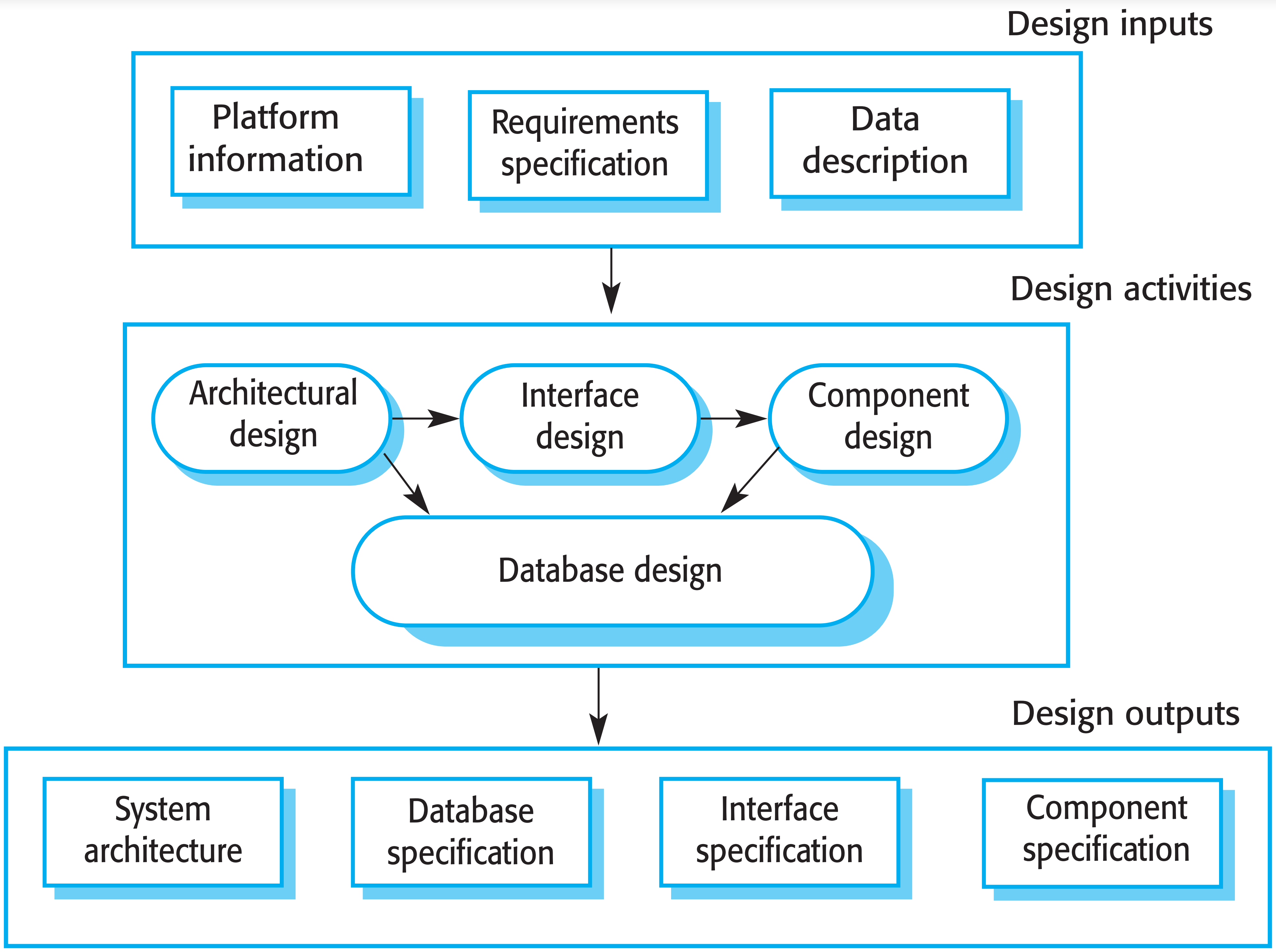
The requirements engineering process



Software design and implementation

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

A general model of the design process



IT University of Copenhagen

Main Design Activities

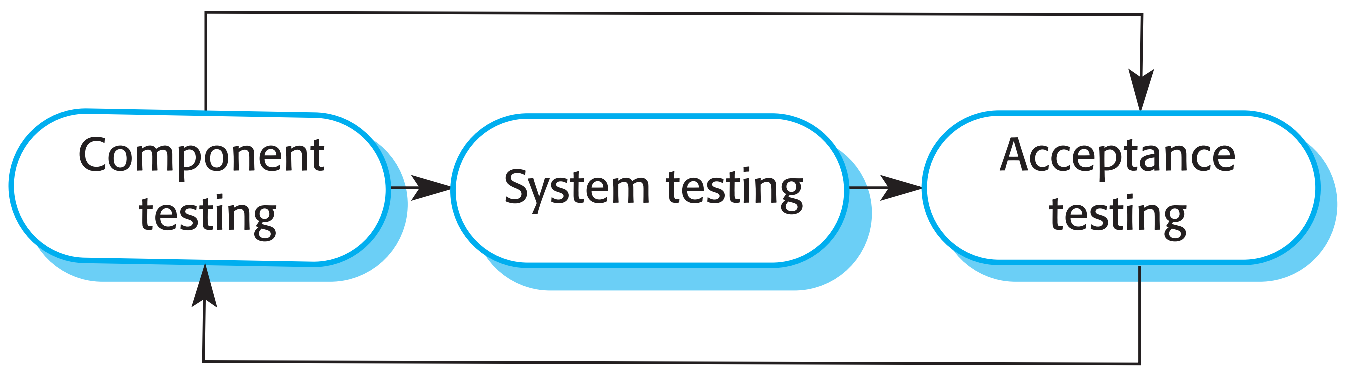
- Architectural design
  - defining the overall structure of the system, the principal components (sub-systems, modules), their relationships and how they are distributed.
- Database design
  - designing the system data structures and how these are to be represented in a database.
- Interface design
  - defining the interfaces between system components.
- Component design
  - defining each system component and design how it will operate.

Software Validation

- Verification and validation (V&V)
  - is intended to show that a system conforms to its specification and meets the requirements of the customers.
- Involves checking and review processes and system testing.
- System testing
  - executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- Testing is the most commonly used V&V activity.

IT University of Copenhagen

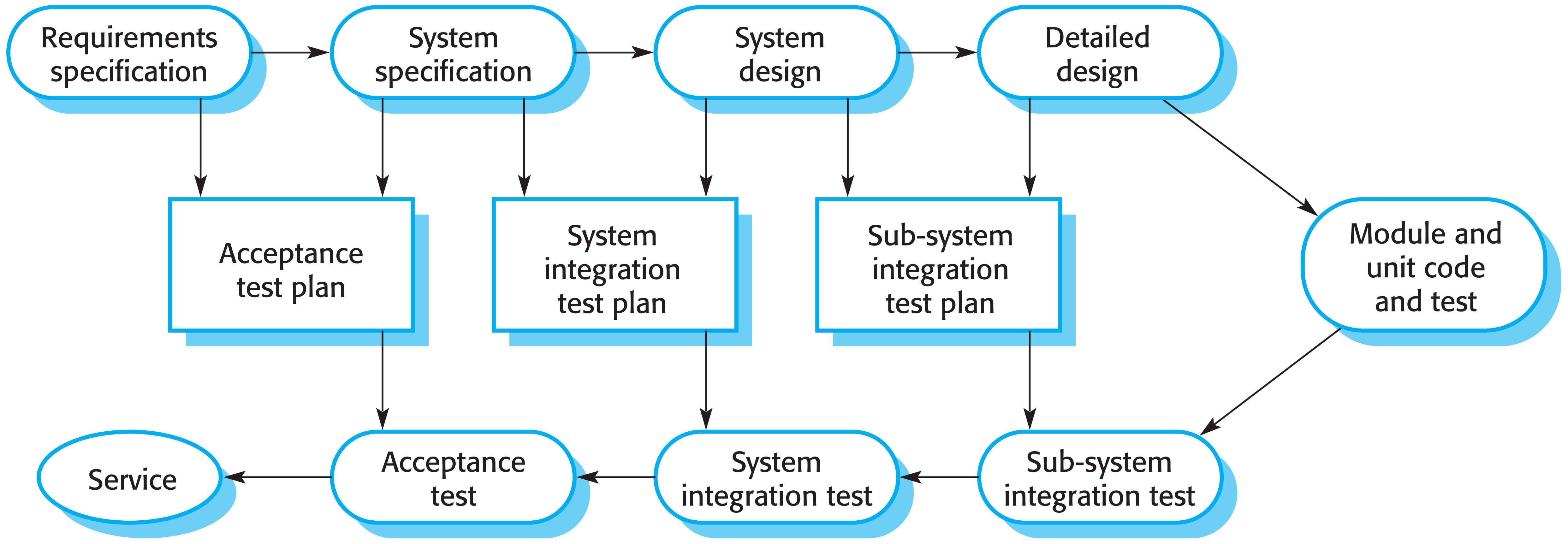
Stages of Testing



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

Testing phases in a plan-driven software process


Requirements specification
System specification
System design
Detailed design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and test
Service
Acceptance test
System integration test
Sub-system integration test

IT University of Copenhagen

Summing up

IT University of Copenhagen

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 <u>problem</u> and <u>resource</u> constraints
  - uses appropriate methodologies, techniques, and <u>tools</u>.

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 (… to be continued)
  - plan-driven
  - incremental or iterative
  - agile

- Examples of specific methodologies include (… to be continued)
  - the 'waterfall' model
  - scrum

This Lecture

- Literature
  - [OOSE] ch. 1
  - [SE9] ch. 1+2
- Introduction to Software Engineering
  - History
  - FAQ about SE
- What is SE?
- Software Processes
  - Software Process Models
  - Process Activities