Software, formally

**Definition.** Software is a finite description \( S \) of a (possibly infinite) set \( \llbracket S \rrbracket \) of (finite or infinite) computation paths of the form 
\[
\sigma_0 \xrightarrow{\alpha_1} \sigma_1 \xrightarrow{\alpha_2} \sigma_2 \cdots
\]
where
- \( \sigma_i \in \Sigma \), \( i \in \mathbb{N}_0 \), is called state (or configuration), and
- \( \alpha_i \in A \), \( i \in \mathbb{N}_0 \), is called action (or event).

The (possibly partial) function \( \llbracket \cdot \rrbracket : S \mapsto \llbracket S \rrbracket \) is called interpretation of \( S \).

**Examples:**
- 'Hallo' (from Lect. 2): Can be seen as having one computation path.
- Quicksort implementation: Can be seen as having as many computation paths as possible inputs.
- Pedestrians Crossing controller: Usually has infinitely many computation paths (each sequence of pedestrians pressing button at particular times defines a different computation path).

**Note:** one software \( S \) may have different interpretations, ranging from 'only final result' (coarse; if well-defined) to 'register transfer level' (fine), with or without time-stamps, etc..

**Software Specification:** An Ideal Partitioning

**Software Specification:** Perceived Practice
Risks Implied by Bad Requirements Specifications

- Negotiation (with customer, marketing department, or...
- Design and implementation, without specification, programmers may just "ask around" when in doubt, possibly yielding different interpretations → difficult integration
- Documentation, e.g., the user's manual, without specification, the user's manual author can only describe what the system does, not what it should do ("every observation is a feature")
- Preparation of tests, without a description of allowed outcomes, tests are randomly searching for generic errors (like crashes) → systematic testing hardly possible
- Acceptance by customer, resolving later objections or claims. Without specification, it is unclear at delivery time whether behaviour is an error (developer needs to fix) or correct (customer needs to accept and pay) → nasty disputes
- Re-use, without specification, re-use needs to be based on re-reading the code → risk of unexpected changes later re-implementations.
- The new software may need to adhere to requirements of the old software; if not properly specified, the new software needs to be a 1:1 re-implementation of the old → additional effort

Discovering Fundamental Errors Late Can Be Expensive

- Average relative error costs over latency according to investigations at IBM, etc.

By (Boehm, 1979); Visualisation: Ludewig and Lichter (2013).
The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements. No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later. (F. P. Brooks, 1995)

**Topic Area Requirements Engineering: Content**

- **Introduction**
- **Requirements Specification**
- **Desired Properties**
- **Kinds of Requirements**
- **Analysis Techniques**
- **Documents**
- **Dictionary, Specification**
- **Specification Languages**
  - Natural Language
  - Decision Tables
  - Syntax, Semantics
  - Completeness, Consistency, ...
- **Vocabulary Techniques**
  - Informal
  - Semi-formal
  - Formal
  - Scenarios
  - User Stories, Use Cases
  - Live Sequence Charts

**Requirements on Requirements Specifications**

A requirements specification should be:

- Correct — it correctly represents the wishes/needs of the customer,
- Complete — all requirements (existing in somebody’s head, or a document, or ...) should be present,
- Relevant — things which are not relevant to the project should not be constrained,
- Consistent, free of contradictions — each requirement is compatible with all other requirements; otherwise the requirements are not realisable,
- Neutral, abstract — a requirements specification does not constrain the realisation more than necessary,
- Traceable, comprehensible — the sources of requirements are documented, requirements are uniquely identifiable,
- Testable, objective — the final product can objectively be checked for satisfying a requirement.

Correctness and completeness are defined relative to something which is usually only in the customer’s head. It is difficult (if at all possible) to be sure of correctness and completeness.
Kinds of Requirements: Functional and Non-Functional

Examples:

- Software “compute shipping costs”:
  - Examples:
    - Shipping parameters: shipping costs
    - Initial state: initial state
    - 0: initial state, 0: initial state

- Software “traffic lights controller”:
  - Examples:
    - 1: pedestrian presses button, 2: button pushed again
    - 1: button pushed again, 2: pedestrian presses button
    - i: button pushed again, o: pedestrian presses button

- A car entertainment system which produces “noise” (due to limited bus bandwidth or CPU power)
  - Examples:
    - 1: engine is running, 2: engine is not running
    - 1: engine is running, 2: engine is not running
    - 1: engine is running, 0: engine is not running

Kinds of Requirements: Hard and Soft Requirements

Examples:

- “Cancellation of a cheque”:
  - Examples:
    - 1: cheque is cleared, 2: cheque is not cleared
    - 1: cheque is cleared, 2: cheque is not cleared
    - 1: cheque is cleared, 0: cheque is not cleared

- “A vending machine dispenses the selected item within 1 s”:
  - Examples:
    - 1: vending machine dispenses item, 2: vending machine does not dispense item
    - 1: vending machine dispenses item, 2: vending machine does not dispense item
    - 1: vending machine dispenses item, 0: vending machine does not dispense item

- “A car’s engine is running”:
  - Examples:
    - 1: engine is running, 2: engine is not running
    - 1: engine is running, 2: engine is not running
    - 1: engine is running, 0: engine is not running

Requirements Specification Languages

Consider the following examples:

- “the list of participants should be sorted by immatriculation number, lowest number first”:
  - Examples:
    - Sort by immatriculation number numerically
    - Compare immatriculation numbers numerically
    - Sort by immatriculation number numerically

- “the list of participants should be sorted by immatriculation number, lowest number first”:
  - Examples:
    - Sort by immatriculation number numerically
    - Compare immatriculation numbers numerically
    - Sort by immatriculation number numerically

Pitfall: Vagueness vs. Abstraction

Requirements Specification

Minimizing assumptions and external dependencies.

Vocabulary: Requirements (Analysis)

- Analysis technique
- Kinds of requirements
- Functional requirement
- Non-functional requirement
- Soft requirement
- Hard requirement

Importance of Requirements Specifications

- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.

Requirements on Requirements Specification Documents

- The requirements specification should be easy to understand and should not create new unclarities or rooms for interpretation.
- The requirements specification should be easy to understand and should not create new unclarities or rooms for interpretation.
- The requirements specification should be easy to understand and should not create new unclarities or rooms for interpretation.

- Once again, it’s about compromises.
- Once again, it’s about compromises.
- Once again, it’s about compromises.

Content

Requirements on Requirements Specification Documents

Minimizing assumptions and external dependencies.

Vocabulary: Requirements (Analysis)

- Analysis technique
- Kinds of requirements
- Functional requirement
- Non-functional requirement
- Soft requirement
- Hard requirement

Importance of Requirements Specifications

- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.

- Once again, it’s about compromises.
- Once again, it’s about compromises.
- Once again, it’s about compromises.

Requirements on Requirements Specification Documents

Minimizing assumptions and external dependencies.

Vocabulary: Requirements (Analysis)

- Analysis technique
- Kinds of requirements
- Functional requirement
- Non-functional requirement
- Soft requirement
- Hard requirement

Importance of Requirements Specifications

- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.

- Once again, it’s about compromises.
- Once again, it’s about compromises.
- Once again, it’s about compromises.

Requirements on Requirements Specification Documents

Minimizing assumptions and external dependencies.

Vocabulary: Requirements (Analysis)

- Analysis technique
- Kinds of requirements
- Functional requirement
- Non-functional requirement
- Soft requirement
- Hard requirement

Importance of Requirements Specifications

- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.

- Once again, it’s about compromises.
- Once again, it’s about compromises.
- Once again, it’s about compromises.

Requirements on Requirements Specification Documents

Minimizing assumptions and external dependencies.

Vocabulary: Requirements (Analysis)

- Analysis technique
- Kinds of requirements
- Functional requirement
- Non-functional requirement
- Soft requirement
- Hard requirement

Importance of Requirements Specifications

- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.

- Once again, it’s about compromises.
- Once again, it’s about compromises.
- Once again, it’s about compromises.

Requirements on Requirements Specification Documents

Minimizing assumptions and external dependencies.

Vocabulary: Requirements (Analysis)

- Analysis technique
- Kinds of requirements
- Functional requirement
- Non-functional requirement
- Soft requirement
- Hard requirement

Importance of Requirements Specifications

- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.

- Once again, it’s about compromises.
- Once again, it’s about compromises.
- Once again, it’s about compromises.

Requirements on Requirements Specification Documents

Minimizing assumptions and external dependencies.

Vocabulary: Requirements (Analysis)

- Analysis technique
- Kinds of requirements
- Functional requirement
- Non-functional requirement
- Soft requirement
- Hard requirement

Importance of Requirements Specifications

- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.

- Once again, it’s about compromises.
- Once again, it’s about compromises.
- Once again, it’s about compromises.

Requirements on Requirements Specification Documents

Minimizing assumptions and external dependencies.

Vocabulary: Requirements (Analysis)

- Analysis technique
- Kinds of requirements
- Functional requirement
- Non-functional requirement
- Soft requirement
- Hard requirement

Importance of Requirements Specifications

- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.
- It is not trivial to have both, low maintenance effort and low accessibility.

- Once again, it’s about compromises.
- Once again, it’s about compromises.
- Once again, it’s about compromises.
Kinds of Requirements: Open and Tacit

- **Open**: Customer is aware of and able to explicitly communicate the requirement.
- **(semi-)Tacit**: Customer not aware of something being a requirement (obvious to the customer but not considered relevant by the customer, not known to be relevant).

Examples:
- Buttons and screen of a mobile phone should be on the same side.
- Important web-shop items should be on the right hand side because the main users are socialised with right-to-left reading direction.
- The ECU (embedded control unit) may only be allowed use a certain amount of bus capacity.

Analyst knows domain new to domain
Customer/Client explicit requirements discovered requirements discoverable requirements
tacit requirements discoverable with difficulties

Content

- Introduction
- Vocabulary: Requirements (Analysis)
- Importance of Requirements Specifications
- Requirements Specifications
- Requirements Analysis
- Goal Properties
- Kinds of Requirements
- Analysis Techniques
- Documents
- Dictionary
- Specification
- Requirements Specification Languages
- Natural Language

Requirements Analysis Techniques

Requirements Engineers See the World Differently

- The human brain is great at seeing information (even if there isn't so much);
- Requirements Engineering is about seeing the absence of information.

Example: Wireless Fire Alarm System

- Observation:
  - Customers cannot be assumed to be trained in stating/communicating requirements.
  - It is the task of the analyst to:
    - elicit requirements,
    - elicit perceptions,
    - elicit problems.

- Elicitation:
  - BRUT: Mandate/mandate the requirements.
  - Analysis:
    -BRUT: Expose potential discrepancies.
    - Continue.

Example:

- Wireless Fire Alarm System:
  - The loss of the ability of the system to transmit a signal from a component to the central unit is detected in less than 300 seconds and displayed at the central unit within 100 seconds thereafter.
Abstract:
The content and qualities of a good software requirements specification (SRS) are de-
INTRODUCTION

1.1 Purpose

1.2 Acronyms and Definitions

1.3 References

1.4 User Characteristics

FUNCTIONAL REQUIREMENTS

2.1 Function Set 1

2.2 etc.

REQUIREMENTS TO EXTERNAL INTERFACES

3.1 User Interfaces

3.2 Interfaces to Hardware

3.3 Interfaces to Software Products / Software / Firmware

3.4 Communication Interfaces

REQUIREMENTS REGARDING TECHNICAL DATA

4.1 Volume Requirements

4.2 Performance

4.3 etc.

GENERAL CONSTRAINTS AND REQUIREMENTS

5.1 Standards and Regulations

5.2 Strategic Constraints

5.3 Hardware

5.4 Software

5.5 Compatibility

5.6 Cost Constraints

5.7 Time Constraints

5.8 etc.

PRODUCT QUALITY REQUIREMENTS

6.1 Availability, Reliability, Robustness

6.2 Security

6.3 Maintainability

6.4 Portability

6.5 etc.

FURTHER REQUIREMENTS

7.1 System Operation

7.2 Customisation

7.3 Requirements of Internal Users