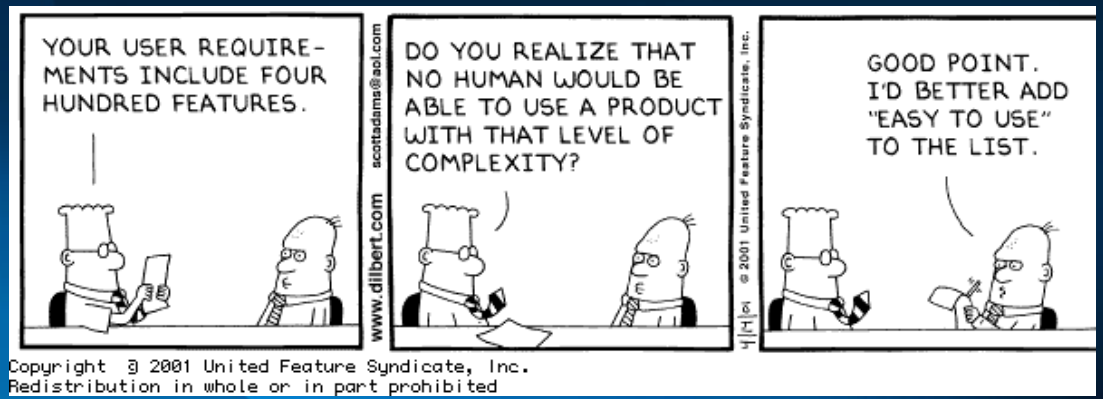
D0020E PROJECT IN COMPUTER SCIENCE 2023/2024 LECTURE 5.2: REQUIREMENTS

Ulf Bodin

LTU 07.11.2023



LULEÅ UNIVERSITY OF TECHNOLOGY

”Failing gracefully”

- Experience is won through an equal amounts of **successes** and **failures**
- One way to become truly successful is to know how to ”**fail gracefully**”!
- Start small and simple, then evolve in small steps!
  - A failure does not mean that too much is lost (should it be small!)
- Manage risks by:
  - Identifying risks early, then weigh value against risk to prioritize work.
  - Doing the parts of the system with least value/risks ratio last!
  - Starting with studying critical risks! (The hardest parts)

LULEÅ UNIVERSITY OF TECHNOLOGY

What now?

- Requirements
  - Why specify requirements?
  - How to specify requirements?
  - Black Box / Customer
  - Functional / Detailed
  - Non-Fuzzy, Non-ambigous, Consistent and Complete
  - Prioritized, Testable and Traceable

LULEÅ UNIVERSITY OF TECHNOLOGY

Complexity in specification

- The common way to express requirements is huge volumes of text, which are:
  - difficult to comprehend,
  - open for varying interpretations, and
  - contains design rather than essential requirements.

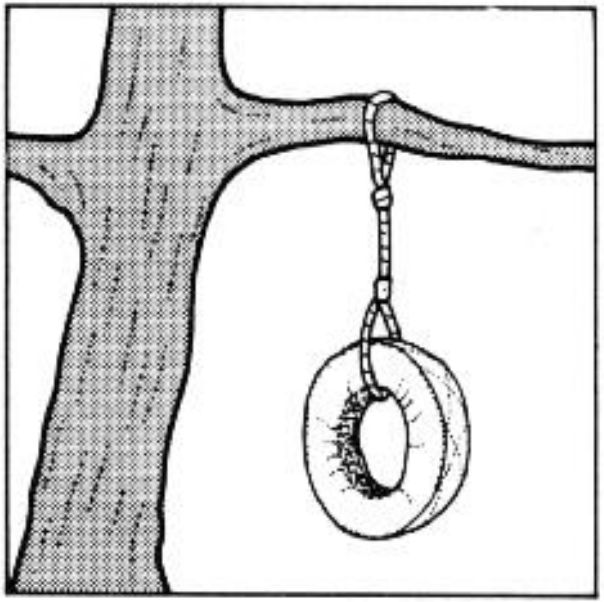
- **The requirements change during the development.**

LULEÅ UNIVERSITY OF TECHNOLOGY

On a cold railway track outside Boden...

LULEÅ UNIVERSITY OF TECHNOLOGY

Why requirements?

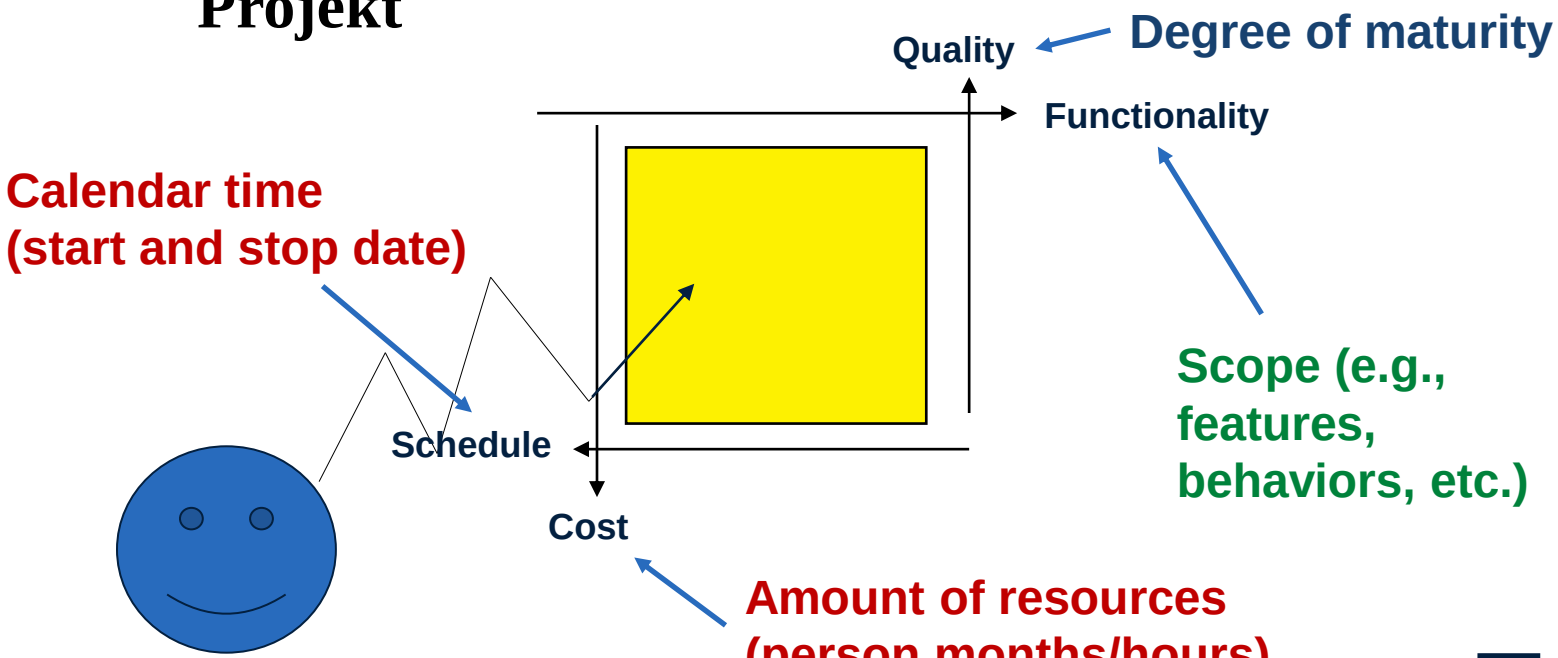


The **shared** system view!

LULEÅ UNIVERSITY OF TECHNOLOGY

The Window of Opportunity

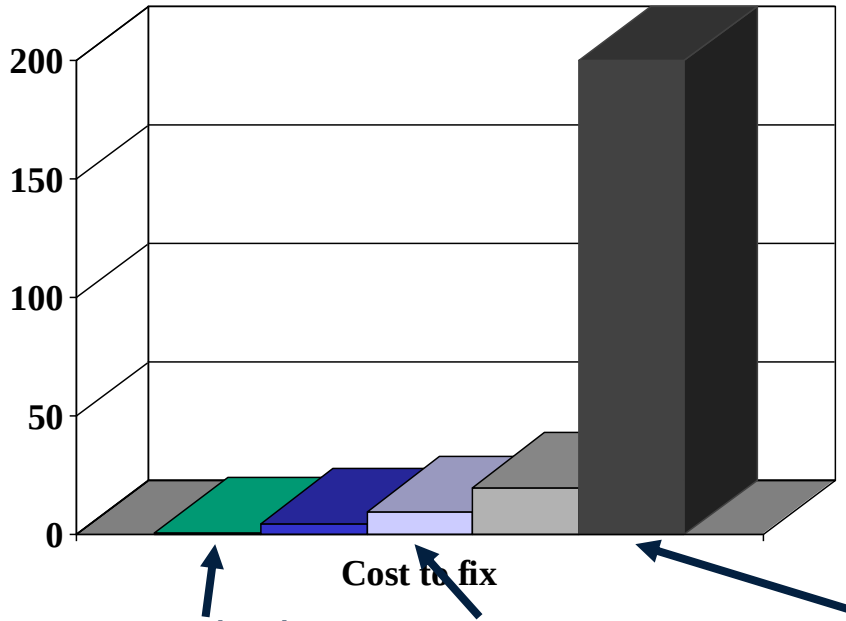
Projekt



On time, within budget and **meet requirements**!

LULEÅ UNIVERSITY OF TECHNOLOGY

The Cost of Delay in Fixing Requirements Errors



Data: Boehm & Papaccio (1988)
IEEE Trans. Software Eng.

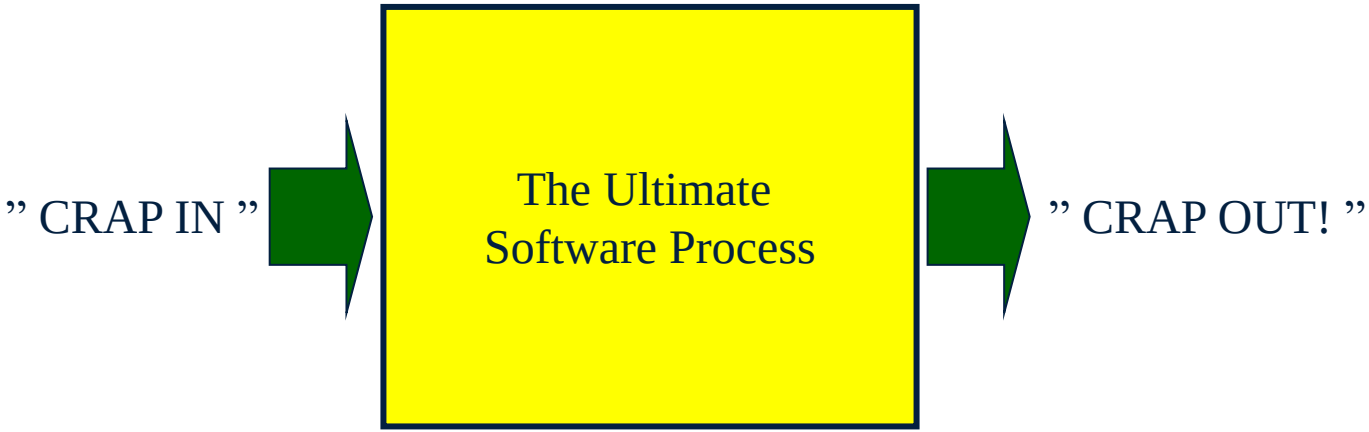
Nominal unit cost

20-fold increase during development

200-fold increase after delivery

LULEÅ UNIVERSITY OF TECHNOLOGY

The Ultimate Software Process...

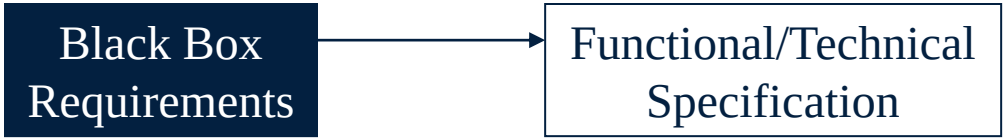


Get the requirements done right!

Do not underestimate design!

LULEÅ UNIVERSITY OF TECHNOLOGY

Requirements:
Black Box (Customer) vs. Functional (Detailed)



Customers!
- Actors
- Use Cases
- Use Case Diagrams
- Activity Diagrams
- State Charts
- Data Flow Diagrams
- Storyboards and UIs
- etc

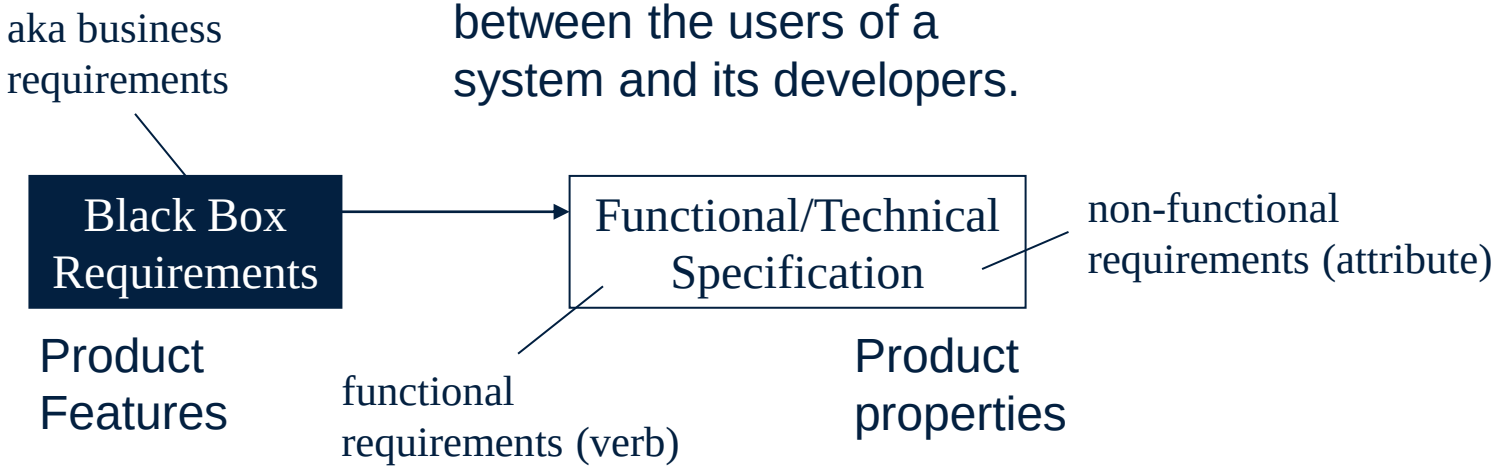
Detailed! (Developers)
- Mostly text-based
- Sequence Diagrams
- Classes/Modules
- Test Plans
- Non-functional
  - Performance
  - Reliability
  - Availability
  - Error Handling

LULEÅ UNIVERSITY OF TECHNOLOGY

Requirements

Ask **what** the system should do and **not how**!

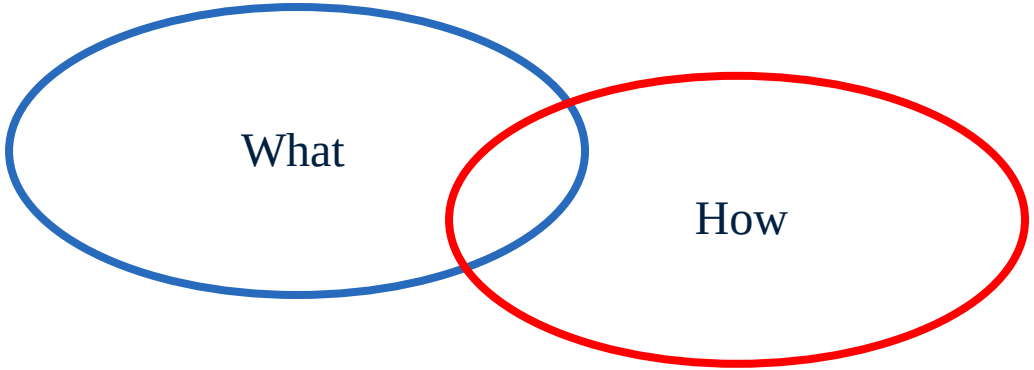
There is a difference between the users of a system and its developers.



LULEÅ UNIVERSITY OF TECHNOLOGY

Problem Space vs Solution Space

The system shall allow the user to access his account balance



Customers' account balances will be stored in a table called 'balance' in an Access database

LULEÅ UNIVERSITY OF TECHNOLOGY

Functional requirements

- A **Functional Requirement** (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behavior, and outputs.
  - E.g., a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.
- Functional Requirements in Software Engineering are also called **Functional Specification**.

LULEÅ UNIVERSITY OF TECHNOLOGY

Non-functional requirements

- **Non-Functional Requirement** (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system.
  - E.g., *"how fast does the website load?"*
- Non-functional Requirements typically found in a separate section of a **Functional Specification**.

LULEÅ UNIVERSITY OF TECHNOLOGY

Prioritize Requirements!

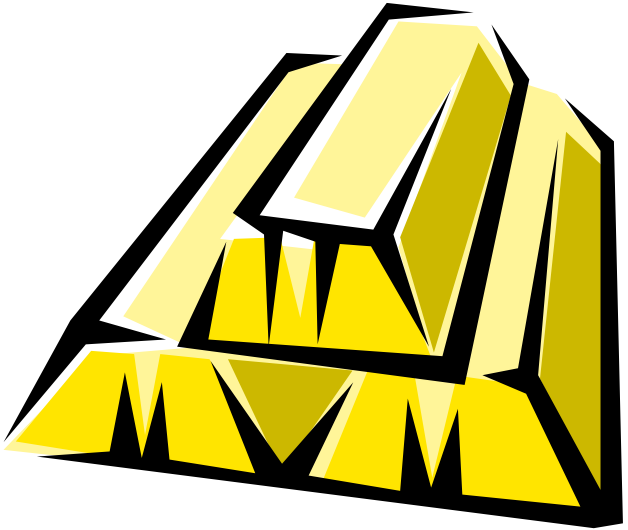
[Essential] Every game character has the same set of qualities.

[Desireable] Every area has a set of preferred qualities.

[Optional] The players character shall age.

LULEÅ UNIVERSITY OF TECHNOLOGY

Gold-Plating/Unfeasibility



- It is so easy to decorate the requirements with half promises...
- ...making way too many requirements unfeasible!
- Keep requirements as simple as possible!

LULEÅ UNIVERSITY OF TECHNOLOGY

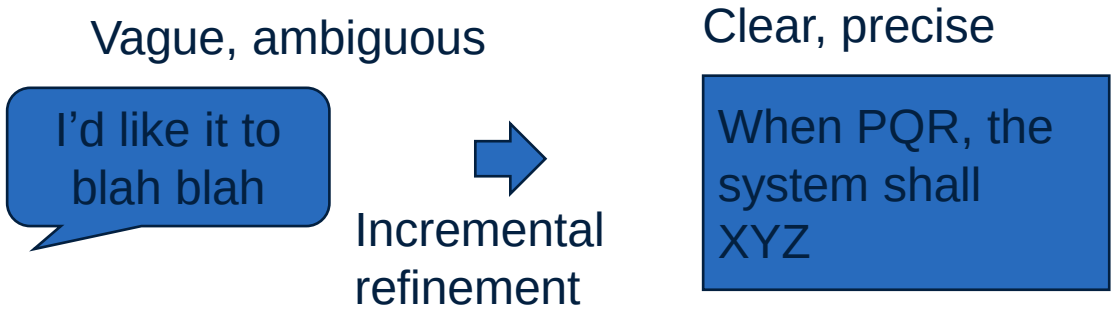
Testable Requirements

- Performance
  - If there are feasible meeting times, the system should in 95% cases schedule a meeting within 5 seconds of meeting constraints having been entered

- Usability
  - A person should be able to fill in the scheduling constraints for an N-person meeting in fewer than 5*N keystrokes/mouse-clicks

LULEÅ UNIVERSITY OF TECHNOLOGY

Requirements refinement

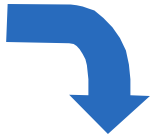


- What kinds of refinement are there?
  - Making a vague statement more precise
  - Saying what the system should do vs. its users
  - Dealing with not-yet considered situations

LULEÅ UNIVERSITY OF TECHNOLOGY

Refining a fuzzy requirement

The system shall improve the responsiveness to customer complaints



The system shall improve the responsiveness to customer complaints

….

When the customer-service clerk enters the customer code, the system shall recommend the next customer-service action.

LULEÅ UNIVERSITY OF TECHNOLOGY

Good Requirements are…

CORRECT	**A requirement that the SW shall meet**	
UNAMBIGOUS	**Only one interpretation**	
COMPLETE	**All significant reqs**	**No TBDs**
CONSISTENT	**No conflicts**	
IMPORTANCE/STABILITY	**Each req is prioritized (essential, desireable, option) (skall, bör, option)**	
VERIFIABLE	**There exists a process that can check that the software meets the requirement**	
MODIFIABLE	**Exist only once, not mixed**	
TRACEABLE	**Enumerable - backward/forward**	

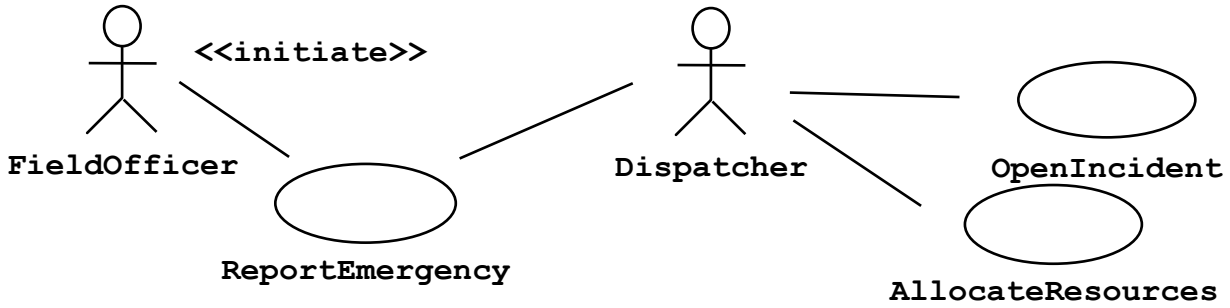
LULEÅ UNIVERSITY OF TECHNOLOGY

An example use case with requirements

Use case name	`ReportEmergency`
Participating actor	Initiated by `FieldOfficer` Communicates with `Dispatcher`
Entry condition	1. The `FieldOfficer` activates the “Report Emergency” function of her terminal.
Flow of events	2. FRIEND responds by presenting a form to the officer. 3. The `FieldOfficer` fills the form, by selecting the emergency level, type, location, and brief description of the situation. The `FieldOfficer` also describes possible responses to the emergency situation. Once the form is completed, the `FieldOfficer` submits the form, at which point the `Dispatcher` is notified. 4. The `Dispatcher` reviews the submitted information and creates an `Incident` in the database by invoking the `OpenIncident` use case. The `Dispatcher` selects a response and acknowledges the emergency report.
Exit condition	5. The `FieldOfficer` receives the acknowledgment and the selected response.
Special requirements	The `FieldOfficer's` report is acknowledged within 30 seconds. The selected response arrives no later than 30 seconds after it is sent by the `Dispatcher`.

LULEÅ UNIVERSITY OF TECHNOLOGY

UML Use Case Diagram



LULEÅ UNIVERSITY OF TECHNOLOGY

Questions to ask the user (system expert)

- Do not ask “What should the system do?”
  - Too general
- Questions prepared in a “question route”
  - Not a strict questionnaire
  - What -> How
- Ask open-ended questions
  - Avoid yes/no questions
  - ”Fishing”
- Do not ask negative “why” questions
  - Trying to find out about habits, not rationale!

LULEÅ UNIVERSITY OF TECHNOLOGY

Software Requirements Specification (SRS)

"*A software requirements specification (SRS) is a description of a* ***software system to be developed****. It is modeled after business requirements specification (CONOPS). The software requirements specification lays out* ***functional and non-functional requirements****, and it may include a set of* ***use cases*** *that describe user interactions that the software must provide to the user for perfect interaction*."*

IEEE/ISO/IEC 29148-2018

ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering

* https://en.wikipedia.org/wiki/Software_requirements_specification

LULEÅ UNIVERSITY OF TECHNOLOGY


LULEÅ
UNIVERSITY
OF TECHNOLOGY