

Software Requirements Engineering

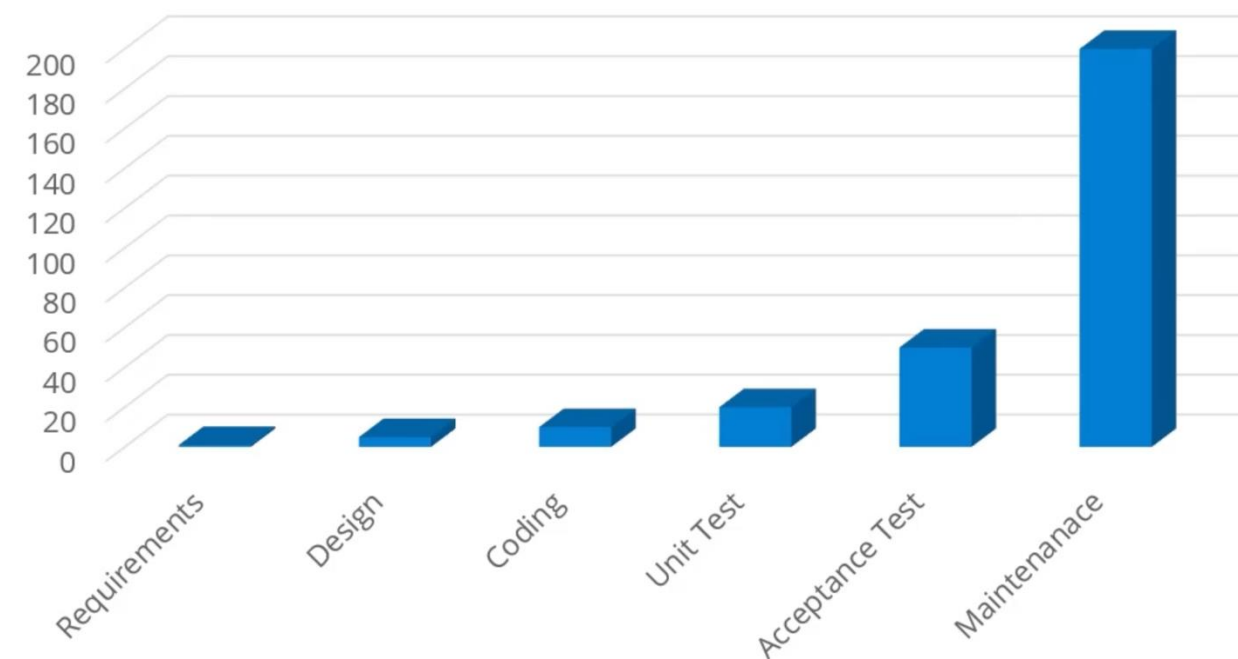
Asst. Prof. Bilge Kağan DEDETÜRK

Week-2

Software Requirements Engineering

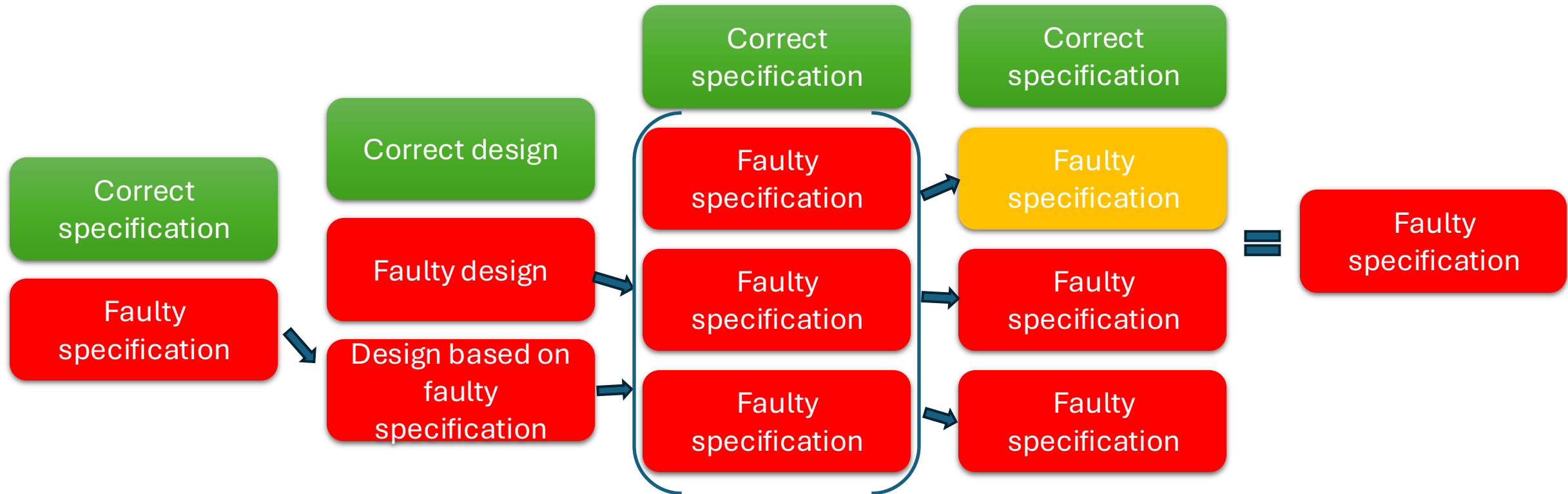
- Cost of Repair
- Propagation of errors
- Why Do We Need Requirements Engineering?
- Benefits of Having Solid Requirements
- Definition of a Software Requirement
- Requirements vs Design
- What Do Requirements Define?
- Process Requirements
- Product Requirements
 - Functional and Non-Functional Requirements

Cost of Repair



Life Cycle Phase	Relative Cost of Repair
Requirements	1
Design	5
Coding	10
Unit Test	20
Acceptance Test	50
Maintenance	200 - 1000

Propagation of errors



Why Do We Need Requirements Engineering?

- We need to know what we should build before we build it.
- We have trouble understanding the requirements that we acquire from the customer
- Failures to specific at this stage are more costly than at any other development phase.

Benefits of Having Solid Requirements

- Providing a documented basis of what the software product is to do
- Providing a baseline for software capabilities
- Providing a basis for the estimation of cost and schedule.
- Reducing the effort needed to produce software
- Facilitating future evolution, adaptation, and migration of software items.
- Technical writer can use the requirements to work on the user manuals

Definition of a Software Requirement

Software Requirement

- **Q:** True or false: the term requirement is consistently defined across the software industry.
- **A:** False. The term requirement is subjective from one company to another.
- The term requirements is not standard among companies.

Definition of a Software Requirement

IEEE Definition of a Software Requirement

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component.
3. A documented representation of a condition or capability as in (1) or (2).

Definition of a Software Requirement

SWEBOK Definition of a Software Requirement

Requirements: a property that must be exhibited by the software developed or adapted to solve a particular problem

Requirements vs Design

- Requirements refer to what the system will do, and the design refers to how we'll do it.
- Requirements should specify all the externally visible expected behavior.
- Analysts get lost in realizing the functionality and discussing the desing approach for a requirement instead of documenting what the software shall do.

What Do Requirements Define?

Functional and
capability
characteristics

Types of users and
roles

External interfaces
to the software

Safety and security
specifications

Human factors for
the user interface

Data definitions

Product and Process Requirements

Product Requirements

- Describe the software being created

Process Requirements

- Constraints and limits on how this software will be created

Process Requirements

- Requirements could include the programming language, the database, the verification techniques to use
- Requirements could include the overall process that should be followed.
- Process requirements may be imposed directly by the development organization, the customer, or a third party such as a safety regulator.

Product Requirements

Functional Requirements

- what the software must do and what users must be able to do with the software

Non-functional Requirements

- the software's expected performance behavior, such as minimum memory requirements and acceptable minimum speed.

Product and Process Requirements

A product requirement specify the implementation required to support a process requirement

- A requirement that the product is maintainable (a product requirement) often is addressed by imposing requirements to follow particular development styles, style guides, or a review/inspection process (process requirements).


A process requirement could specify the activities required to support a product requirement.


- A maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement).


Functional and Non-Functional Requirements

- The functional requirements are "what" the software can do.
 - Also known as capabilities or features
- Non-functional requirements are qualification criteria, characteristics of the software, quality attributes or ability attributes.
- Non-functional requirements describes how the software will execute its tasks.
- It is the "how" the software will perform its functional requirements.

Functional and Non-Functional Requirements

- The system will **print** the record **quickly**


The diagram for this requirement shows the word **print** with a blue arrow pointing to the label **Functional** below it. Similarly, the word **quickly** has a blue arrow pointing to the label **Non-Functional** below it.
- The system shall **assign** a unique tracking number to each shipment.


The diagram for this requirement shows the word **assign** with a blue arrow pointing to the label **Functional** below it.
- The system shall **display** the due date of payment as MM/DD/YY.


The diagram for this requirement shows the word **display** with a blue arrow pointing to the label **Functional** below it.

Functional and Non-Functional Requirements

- With 100 concurrent users, a database record shall be **fetch**ed over the network in **less than three milliseconds**.

Non-Functional Requirements

- Non-functional requirements, or NFR, or quality attributes are sometimes called the "ilities" requirements.

Non-Functional Requirements

Efficiency

- Efficiency is how the software will use the resources (e.g., disk space, ram, and bandwidth).
- Effective means the job is being done correctly.
- Efficient means it does the job correctly while using the right resources in the right way as much as possible.
- Efficiency is picking the right tools to get the job done with the lowest costs, effort in the shortest time.

Non-Functional Requirements

Flexibility

- Flexibility is how easy or hard it is to add a new feature to the software.
- Generally, this means the software was not well-designed.

Non-Functional Requirements

Integrity

- Integrity (Data integrity) defines data quality which guarantees the data is complete and has a whole structure.
- Data integrity deals with the validity of data and making sure the data is correct and not corrupt.
- Backing up, designing a suitable user interface and error detection/correction in data are some of the means to preserve integrity.

Non-Functional Requirements

Interoperability

- Interoperability is exchanging data or services with other software systems.
- Interoperability is making sure our system can operate with more than one other system.

Non-Functional Requirements

Usability

- Usability is when the system is easy to use, user-friendly, the interface is appealing, and documentation is understandable.

Non-Functional Requirements

Maintainability

- Maintainability is how easy and fast it is to fix a particular bug or add a new feature.
- Maintainability is also defined as the probability that a successful repair operation will occur within a given time.

Non-Functional Requirements

Portability

- Portability is the effort required to move or migrate software from one platform to another.
- A requirement like this must be known from day 0. It's a different style of design and coding.

Non-Functional Requirements

Reusability

- Reusability is an attribute in which software or its module is reused with very little or no modification.
- The higher the reusability, the shorter the development life cycle would be.
- It is challenging to design or code or put requirements while bearing in mind all the possible opportunities that can be used in the future.

Non-Functional Requirements

Recoverability

- Recoverability is the ability to restore your deployment to the point at which a failure occurred.