Peter Mileff PhD

SOFTWARE ENGINEERING

Requirements Engineering

University of Miskolc
Department of Information Technology

Requirements Engineering

- The requirements for a system are the descriptions of what the system should do
  - the **services** that it provides
  - the **constraints** on its operation
- **Requirements engineering:**
  - The process of finding out, analyzing, documenting and checking these services and constraints
- The term ‘requirement’ is not used consistently in the software industry.
  - Sometimes a requirement is simply a high-level, abstract statement of a service.
  - Sometimes it is a detailed, formal definition of a system function.

Let's see it from different perspectives!

Interpretation 1.

User Requirements: natural language statements

- Describes the system services and constraints
- They are high-level abstract descriptions with
  - diagrams,
  - tables,
  - figures, etc
  - anything which serves better understanding

Objective: the document should be readable by anyone without deep technical knowledge.

- For example:
  - managers
  - and my Grandma :)

Interpretation 2.

System Requirements: natural language statements

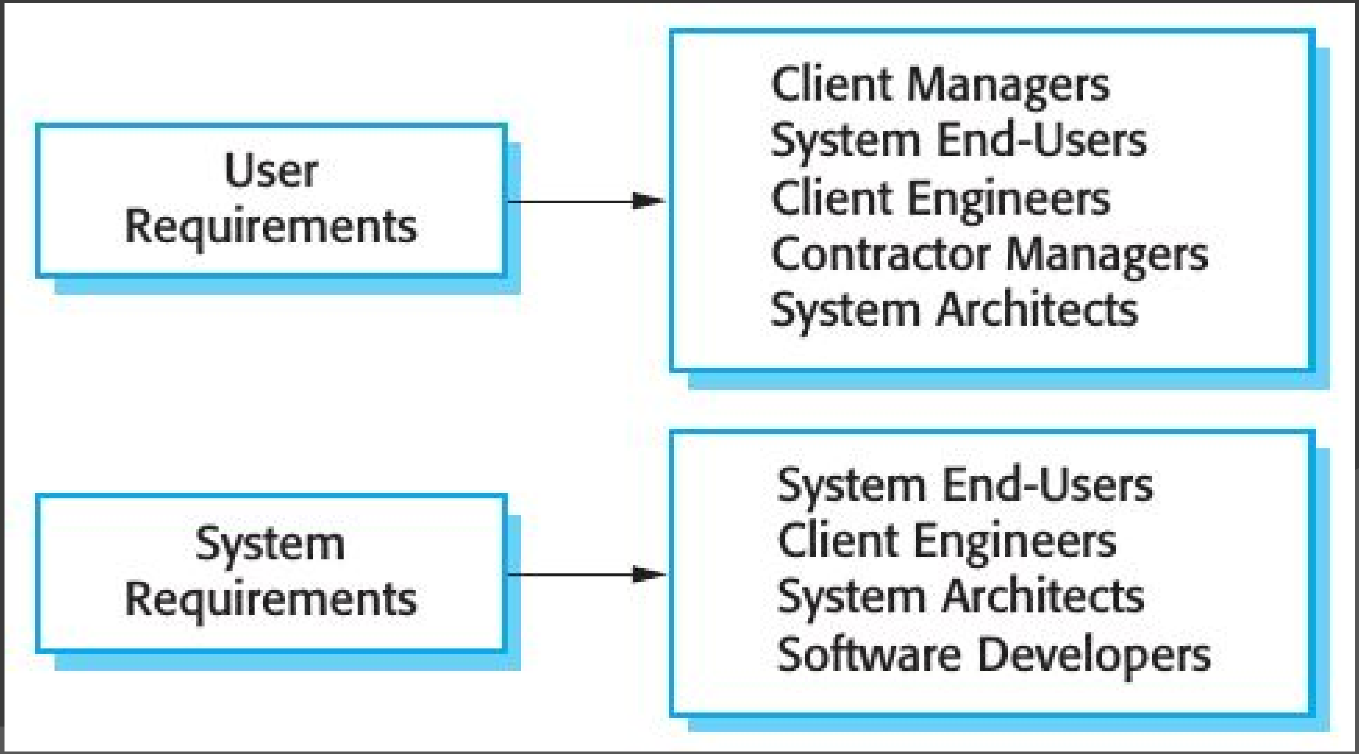
- more detailed descriptions about
  - system's functions, services, and operational constraints.
- Sometimes called a **functional specification**
- **This document should define exactly what is to be implemented!**
  - It may be part of the contract between the customer and the software developers

Objective: the document should be contain anything related to the software

- Usually it is a low level description, readable only with (deep) technical knowledge.

Who reads the requirements?

- **Different levels are very important!**
- User requirements are not usually concerned with how the system will be implemented.
- System requirements readers need to know more precisely what the system will do because they are involved in the system implementation.



Other perspectives

- There are other perspectives for grouping requirements
- **Usually 2 groups are mentioned:**
- **Functional requirements**
- **Non-functional requirement**

Functional requirements…

Functional requirements

- The functional requirements for a system describe what the system should do:
  - how the system should react to particular inputs,
  - how the system should behave in particular situations.
    - E.g.: What if I press the Login button?
    - What if my internet connection has gone?
- **Focuses only to system functions!**
  - In some cases, the functional requirements may also explicitly state what the system should not do.
- These requirements depend on
  - the type of software being developed,
  - the expected users of the software,
  - and the general approach taken by the organization when writing requirements.

Functional requirements

- The functional requirements specification of a system should be both **complete** and **consistent**.

- **Completeness:**
  - means that all services required by the user should be defined.
    - nothing is forgotten

- **Consistency:**
  - means that requirements should not have contradictory

Functional requirements

- **About consistency and completeness:**
  - For large, complex systems, it is practically impossible to achieve
  - Reason 1: it is easy to make mistakes and omissions when writing specifications for complex systems.
    - we are humans!
  - Reason 2: stakeholders have different and often inconsistent needs.
    - there are many stakeholders in a large system.
- **Who is a stakeholder?**
  - A stakeholder is a person or role that is affected by the system in some way.
    - Managers, employers, security guard, etc.

Non-Functional requirements…

Non-Functional requirements

- Non-functional requirements are not directly concerned with the specific services
  - **They are not focusing to functions!**
- So what is their role?
  - They may relate mainly to emergent system properties
  - Non-functional requirements specify or constrain characteristics of the system as a whole.

Example:

  - reliability, response time, and store occupancy.
- They may define constraints on the system implementation
  - e.g: the capabilities of I/O devices
  - or the data representations used in interfaces
  - CPU time and Memory limitation, etc

Non-Functional requirements

- **They are often more critical than individual functional requirements!**
  - System users can usually find ways to work around a system function that doesn't really meet their needs.
  - However, failing to meet a non-functional requirement can mean that the whole system is unusable.

- Example:
  - if an aircraft system does not meet its reliability requirements, it will not be certified as safe for operation
  - if an embedded control system fails to meet its performance requirements, the control functions will not operate correctly.

<u>Non-Functional requirements</u>

- **These requirements can be dangerous!**
  - The implementation of these requirements may be diffused throughout the system.

<u>Reason 1:</u>

- Non-functional requirements may affect the overall architecture of a system
  - rather than the individual components.

<u>Example:</u>

- <u>To meet the performance requirements:</u> we may have to organize the system to minimize communications between components.

Non-Functional requirements

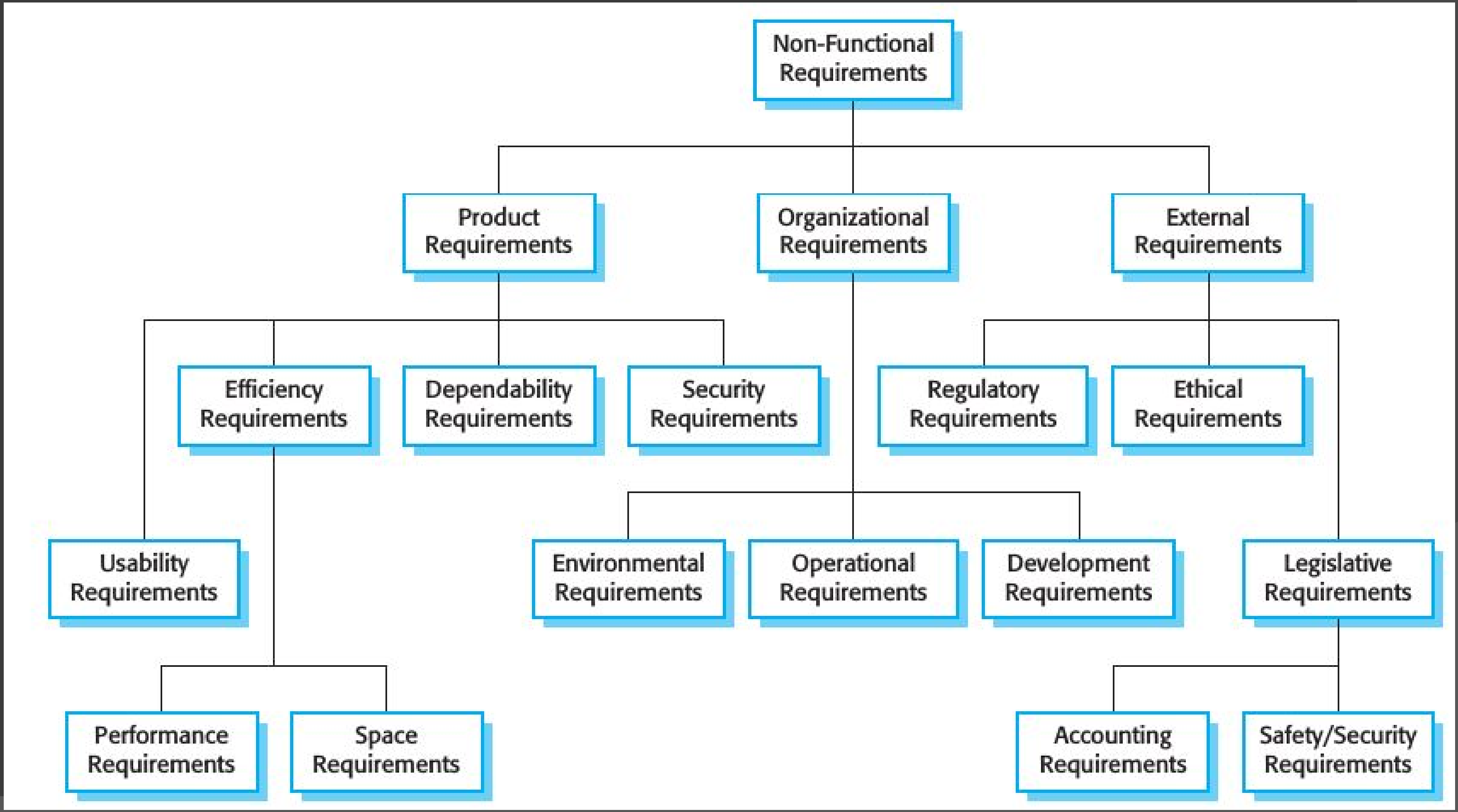
Reason 2:

- A single non-functional requirement can generate numerous new functional requirements
  - such as a security requirement

- In addition, it may also generate requirements that restrict existing requirements.

Non-Functional requirements

- **Classification**



Non-Functional requirements

- **1. Product requirements:**
  - These requirements specify or constrain the behavior of the software.
- **Examples:**
  - performance requirements
    - how fast the system must execute
    - how much memory should be used
    - loading time
  - reliability requirements
    - e.g. setting the acceptable failure rate,
  - security requirements
    - e.g. password encryption method
  - usability requirements

Non-Functional requirements

- **2. Organization requirements:**
  - broad system requirements derived from policies and procedures in the customer's and developer's organization.
- **Examples**
  - operational process requirements
    - define how the system will be used
      - e.g software should operate only between 1am and 2am
  - development process requirements
    - specify the programming language, the development environment or process standards
  - environmental requirements
    - specify the operating environment of the system.

Non-Functional requirements

- **3. External requirements:**
  - all requirements that are derived from factors external to the system and its development process.
- **Examples:**
  - regulatory requirements
    - set out what must be done for the system to be approved for use by a regulator
  - legislative requirements
    - must be followed to ensure that the system operates within the law;
  - ethical requirements
    - must be followed to ensure that the system will be acceptable to its users and the general public.

Problems with Non-Functional requirements?

Non-functional requirements (Common problems)

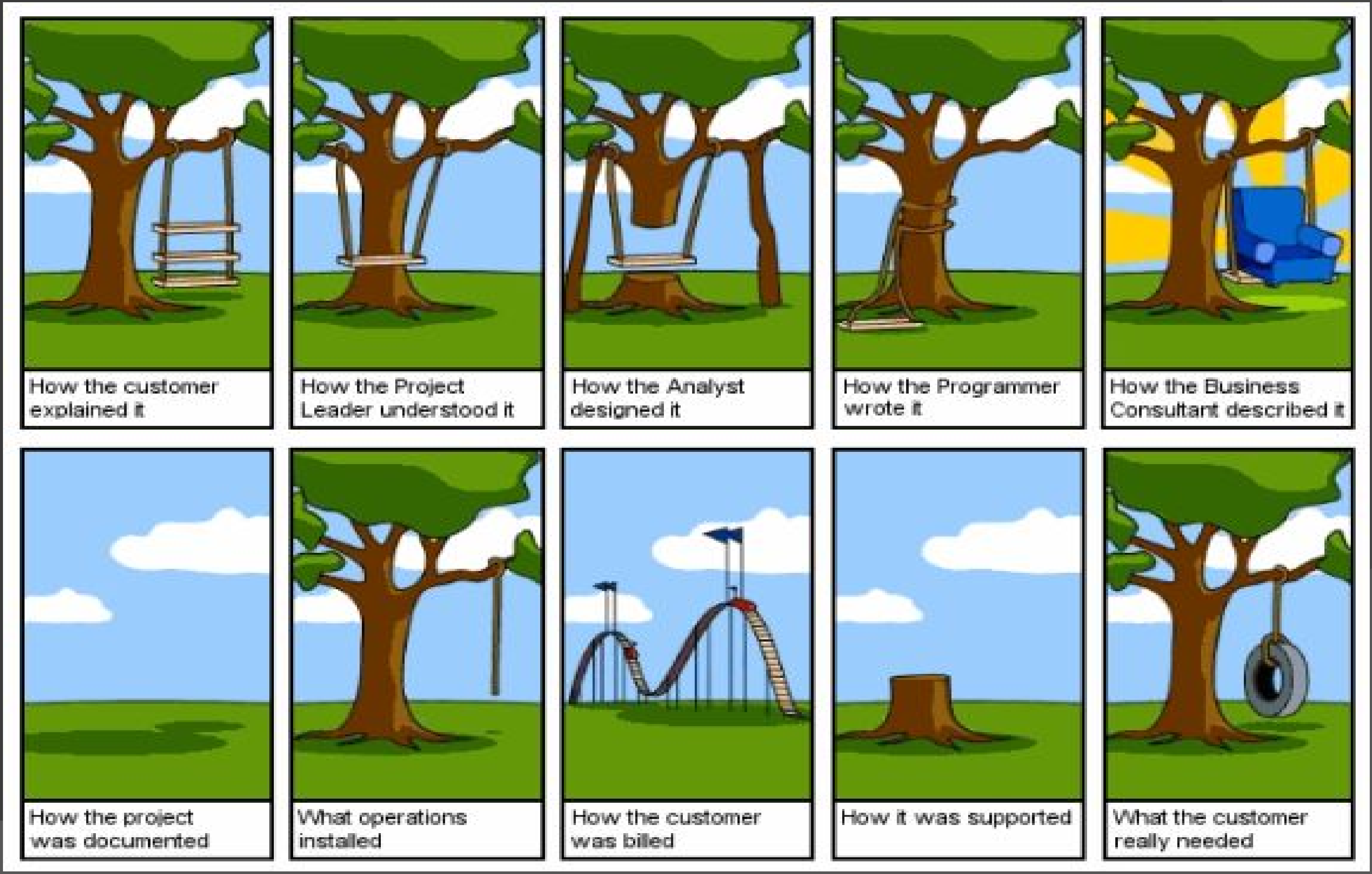
- **Users or customers often propose these requirements as general goals.**
- **Example:**
  "The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized."
- Goals are good, so what is the problem?
- **These goals cannot be measured objectively!**
  - these cause problems for system developers

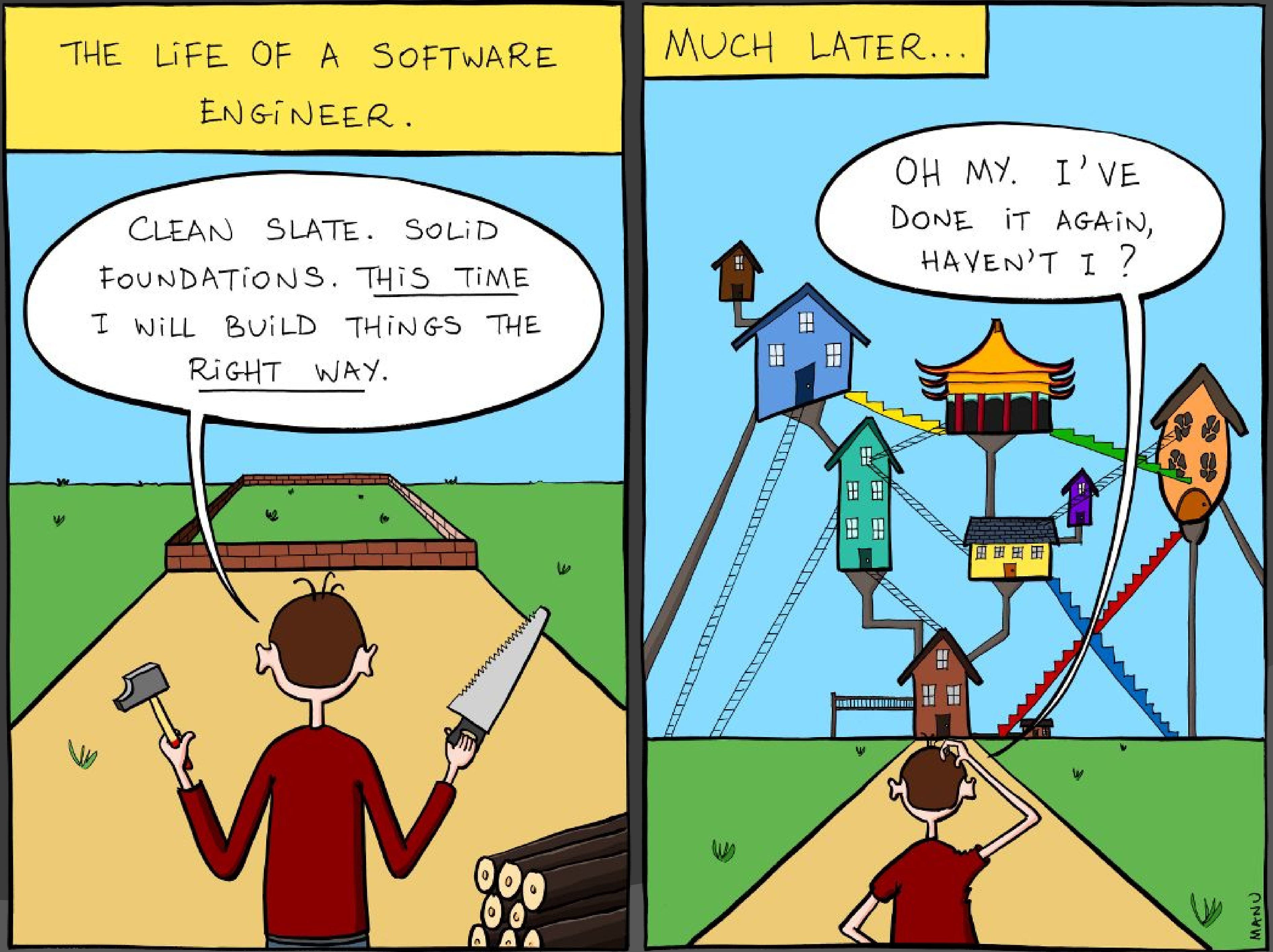
Non-functional requirements (Common problems)

- **If we cannot exactly define a goal:**
  - developers cannot implement it properly
  - can cause problems when the system is delivered to the customer
- **Example:**
  - **Customer goal:** My website must be easy to use and beautiful
- **Problem:**
  - terms like “easy to use” and “beautiful” cannot defined exactly
    - therefore they cannot be measured or validated

And the real life...

The Software design


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed


THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?
MANU

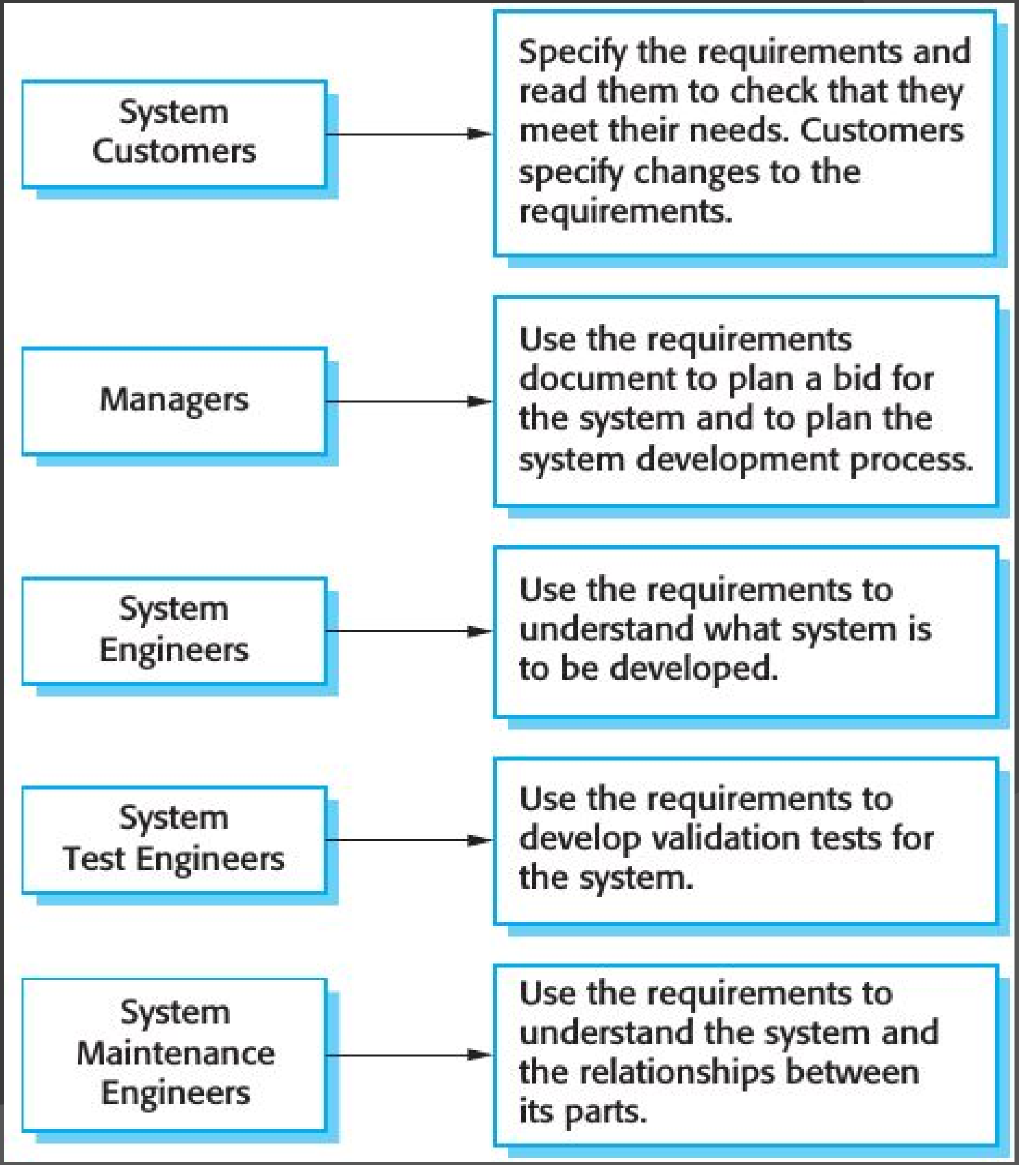
Software requirements document...

Software requirements document

- **It is an official statement of what the system developers should implement.**
  - every part of the software is described here
- **It should include:**
  - the user requirements for a system
  - and a detailed specification of the system requirements.
- Sometimes, the user and system requirements are integrated into a single description.
- In case of large number of requirements
  - the detailed system requirements may be presented in a separate document.

Who reads the document?

- In short: everybody!


System Customers
Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers
Use the requirements document to plan a bid for the system and to plan the system development process.
System Engineers
Use the requirements to understand what system is to be developed.
System Test Engineers
Use the requirements to develop validation tests for the system.
System Maintenance Engineers
Use the requirements to understand the system and the relationships between its parts.

Software requirements document

- **The level of detail**
  - depends on the type of system that is being developed and the development process used.
  - Critical systems need to have detailed requirements
    - because safety and security have to be analyzed in detail
      - e.g. nuclear power plant, aircraft control systems
  - When the system is to be developed by a separate company
    - the system specifications need to be detailed and precise.
  - If an inhouse, iterative development process is used
    - the document can be much less detailed
    - and any ambiguities can be resolved during development of the system.

Software requirements document

- **The structure of the document**
  - can be organized in any way
  - only requirement is to describe the system properly

There is an IEEE standard (1998):

1. Preface: define the expected readership, the version history, who made it and when, etc

2. Introduction: describe the need for the system. Brief description of system's functions and how it works.

- How system fits into the overall business objectives of the organization.

3. Glossary: define the technical terms

4. User requirements definition: describes the services for the user and customer

5. System architecture: a high level overview of the system's logical structure

Software requirements document

IEEE standard (1998):

6. System requirements specification: describes functional and non-functional requirements in detail.

7. System models: this might include graphical system models about the system. How the components are connected.

- e.g. object models, data-flow models, semantic data models

8. System evolution: describes the needs of the long-term operation

9. Appendices: specific information about the part of the system. E.g. hardware or database descriptions

Thank you for your attention!