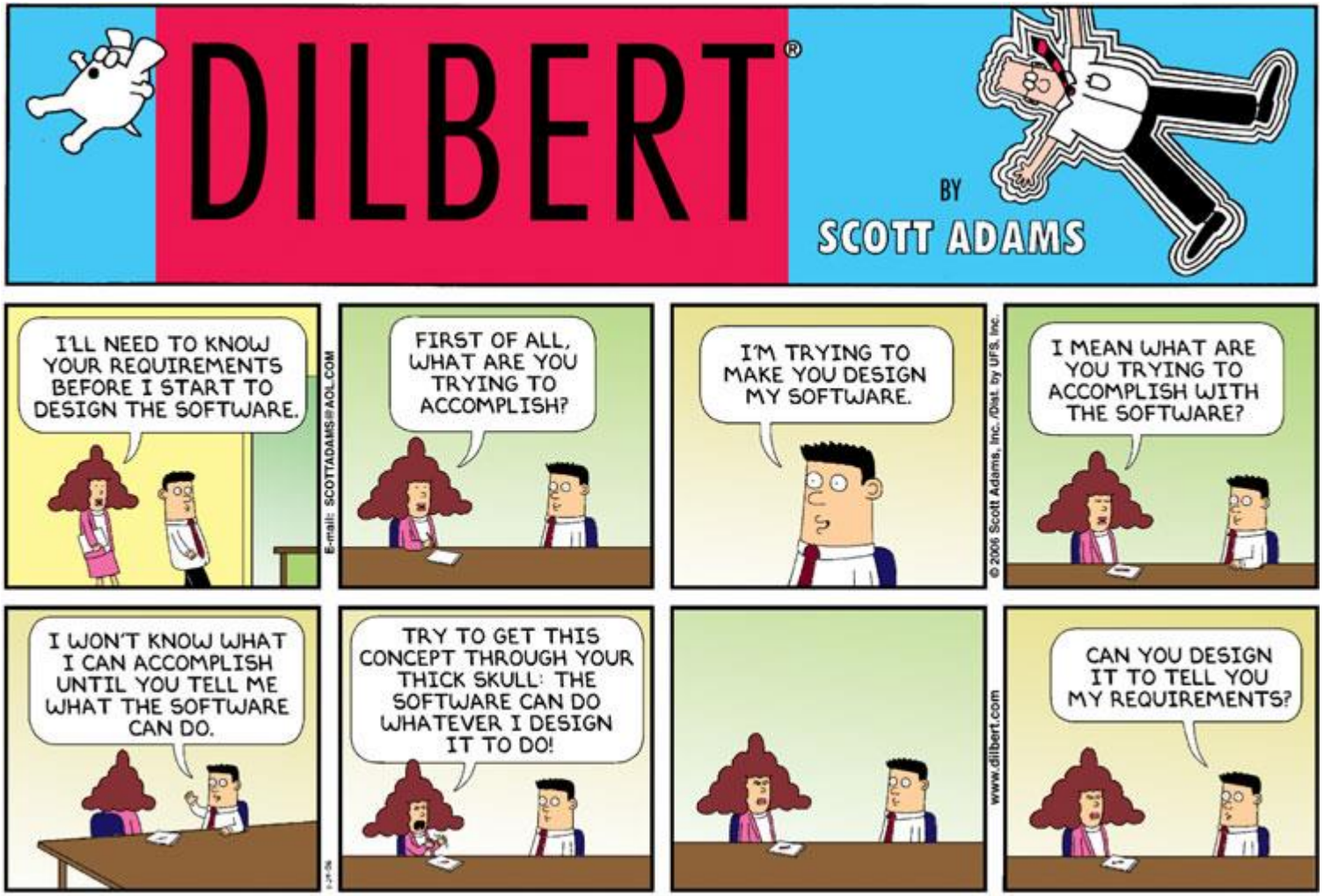
Chapter-1
Essentials of Software Requirements

Textbook Cover Page



Definition and Importance of Requirements

What are “Requirements”?

- A requirement is:
  - Capturing the purpose of a system

- An expression of the ideas to be embodied in the system or application under development

  - A statement about the proposed system that all stakeholders agree must be made true in order for the customer’s problem to be adequately solved
    - Short and concise piece of information
    - Says something about the system
    - All the stakeholders have agreed that it is valid
    - It helps solve the customer’s problem

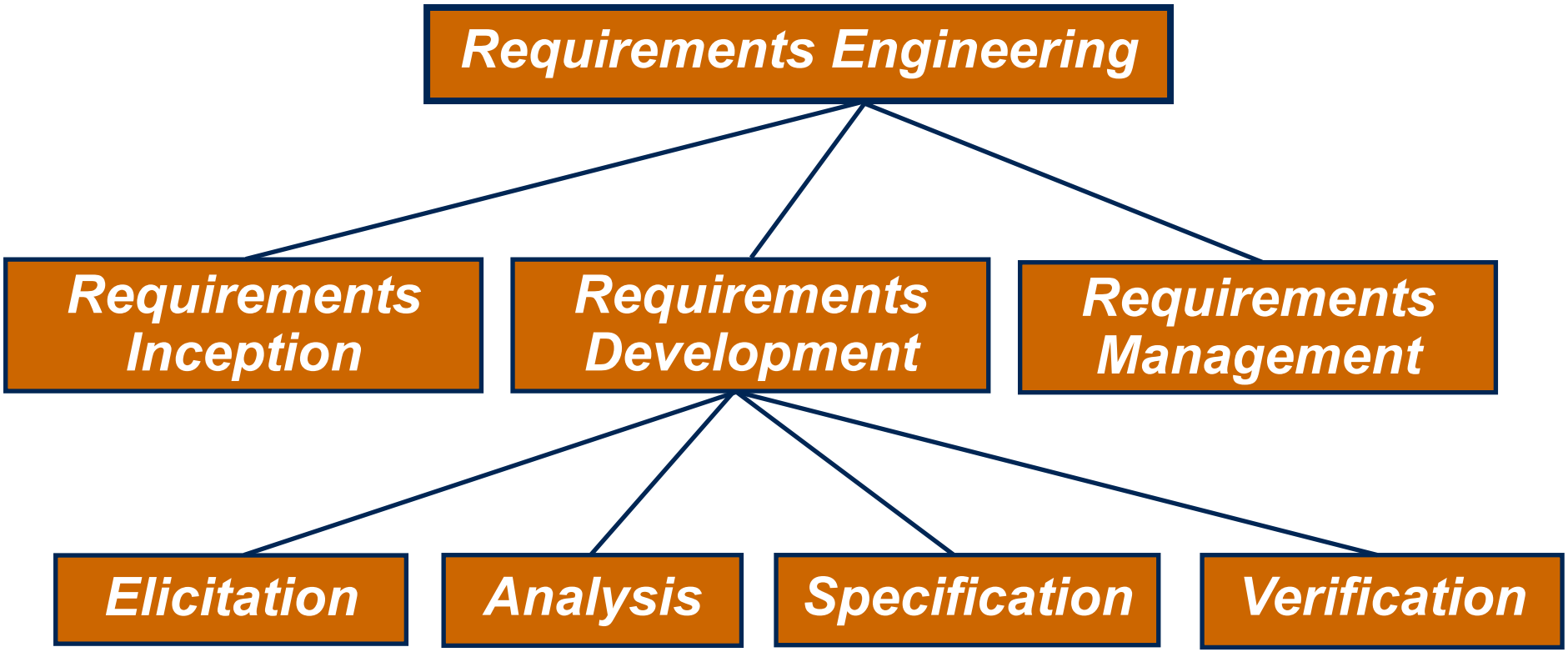
According to IEEE 830-1993

- A **requirement** is defined as:
  - A condition or capability needed by a user to solve a problem or achieve an objective
  - A condition or a capability that must be met or possessed by a system … to satisfy a contract, standard, specification, or other formally imposed document …

What is "Requirements Engineering"?

- **Requirements Engineering (RE)** is:
  - The activity of development, elicitation, specification, analysis, and management of the stakeholder requirements, which are to be met by a new or evolving system
  - RE is concerned with identifying the purpose of a software system… and the contexts in which it will be used
    - How/where the system will be used
    - Big picture is important
  - Captures real world needs of stakeholders affected by a software system and expresses them as artifacts that can be implemented by a computing system
    - Bridge to design and construction
    - How to communicate and negotiate?
    - Is anything lost in the translation between different worlds?

Requirements Engineering Activities



Source: Larry Boldt, Trends in Requirements Engineering People-Process-Technology, Technology Builders, Inc., 2001

About these Requirements Activities…

- **Inception**
  - Start the process (business need, market opportunity, great idea, ...), business case, feasibility study, system scope, risks, etc.
- **Requirements elicitation**
  - Requirements discovered through consultation with stakeholders
- **Requirements analysis and negotiation**
  - Requirements are analyzed and conflicts resolved through negotiation
- **Requirements specification**
  - A precise requirements document is produced
- **Requirements validation**
  - The requirements document is checked for consistency and completeness
- **Requirements management**
  - Needs and contexts evolve, and so do requirements!

General Problems with the Requirements Process

- Lack of the right expertise (software engineers, domain experts, etc.)
- Initial ideas are often incomplete, wildly optimistic, and firmly entrenched in the minds of the people leading the acquisition process
- Difficulty of using complex tools and diverse methods associated with requirements gathering may negate the anticipated benefits of a complete and detailed approach

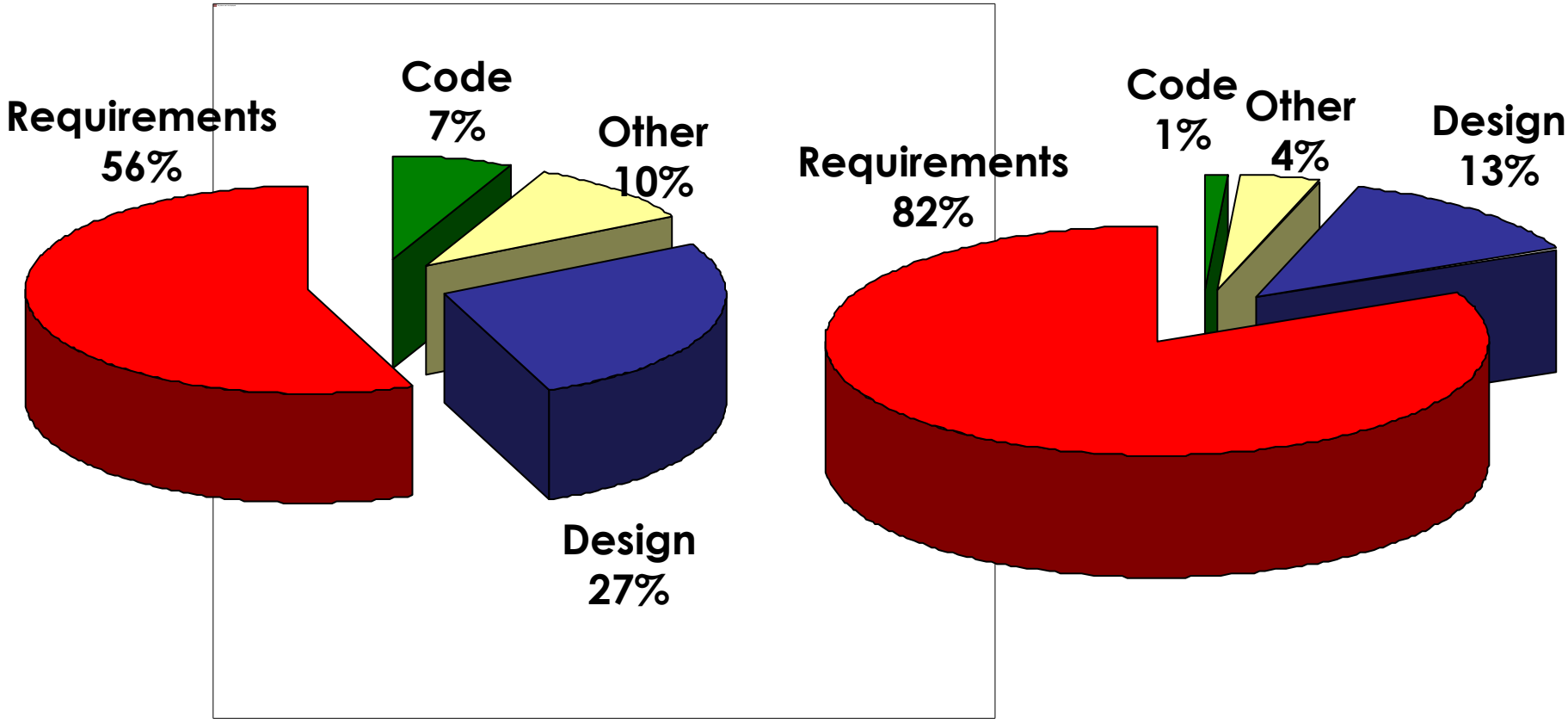
Statistics from NIST Report

- NIST (National Institute of Standards and Technology) has published a comprehensive (309 pages) and very interesting report on project statistics and experiences based on data from a large number of software projects[1]
  - 70% of the defects are introduced in the specification phase
  - 30% are introduced later in the technical solution process
  - Only 5% of the specification inadequacies are corrected in the specification phase
  - 95% are detected later in the project or after delivery where the cost for correction on average is 22 times higher compared to a correction directly during the specification effort
  - The NIST report concludes that extensive testing is essential, however testing detects the dominating specification errors late in the process

[1] http://www.nist.gov/public_affairs/releases/n02-10.htm (May 2002)

Why Focus on Requirements ?

- Distribution of Defects
- Distribution of Effort to Fix Defects



Source: Martin & Leffinwell

View of the Software Engineering Institute (SEI)

- Improve software development with the CMM/CMMI model for software development
  - Capability Maturity Model (CMM)
  - For software development, superseded by Capability Maturity Model Integration (CMMI)

- SEI's vision is:
  - The right software, delivered defect free, on time & on cost, every time
  - "Right software" implies software that satisfies requirements for functionality and qualities (e.g., performance, cost…) throughout its lifetime
  - "Defect free" software is achieved either through exhaustive testing after coding or by developing the code right the first time

Managing Evolving Requirements

"Changing requirements is as certain as death and taxes"

Requirement tools: These seem to have the biggest impact on the success of a project. This may seem strange since "Firm Basic Requirements" is number six on the top ten list. However these tools, if used as a platform for communications between all the stakeholders, such as executive sponsors and users, can provide enormous benefits. This tool needs to be at the top of the shopping list for any firm involved in developing software applications.

Source: http://standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf, 1999

Types of Requirements

1. A functional requirement is a requirement defining functions of the system under development
   - Describes what the system should do
2. A non-functional requirement is a requirement that is not functional. This includes many different kinds of requirements.
3. User requirement is a desired goal or function that a user and other stakeholders expect the system to achieve
   - Does not necessarily become a system requirement
4. Application domain requirement (sometimes called business rules) are requirements derived from business practices within a given industrial sector, or in a given company, or defined by government regulations or standards.
   - May lead to system requirements. Can be functional or non-functional

Types of Requirements

5. Problem domain requirements should be satisfied within the problem domain in order to satisfy some of the goals

6. System requirements are the requirements for the system to be built, as a whole
   - A system is a collection of interrelated components working together towards some common objective (may be software, mechanical, electrical and electronic hardware and b``e operated by people)
   - Systems Engineering is a multidisciplinary approach to systems development – software is only a part (but often the problematic part)

Types of Requirements

- Important note: Software Requirements Engineering is a special case of Requirements Engineering. Many topics discussed in this course are quite general and apply to requirements engineering, in general.
- In a system containing software, software requirements are derived from the system requirements. The system then consists of hardware and software, and the hardware (and often the operating system and other existing software modules) are part of the environment in which the software is used.

Functional Requirements

- What inputs the system should accept
- What outputs the system should produce
- What data the system should store other systems might use
- What computations the system should perform
- The timing and synchronization of the above

- Depend on the type of software, expected users, and the type of system where the software is used
- Functional user requirements may be high-level statements of what the system should do, but functional system requirements should describe the system services in detail

Examples of Functional Requirements

- The user shall be able to search either all of the initial set of databases or select a subset from it.
- The system shall provide appropriate viewers for the user to read documents in the document store.
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

Note: not all requirements on this and following slides are high quality requirements but are typical requirements found too often in documents

Non-Functional Requirements (NFR)

- Non-functional requirements are important
  - If they are not met, the system is useless
  - Non-functional requirements may be very difficult to state precisely (especially at the beginning) and imprecise requirements may be difficult to verify
- They are sometimes called quality requirements, quality of service, or extra-functional requirements.
- *Three main categories :*
  - **Performance requirements** :
    - Response time, throughput
    - Resource usage
    - Reliability, availability
    - Recovery from failure
    - Allowances for maintainability and enhancement
    - Allowances for reusability

[1] Lethbridge and Laganière, Object Oriented Software Engineering: Practical Software Development using UML and Java, 2005

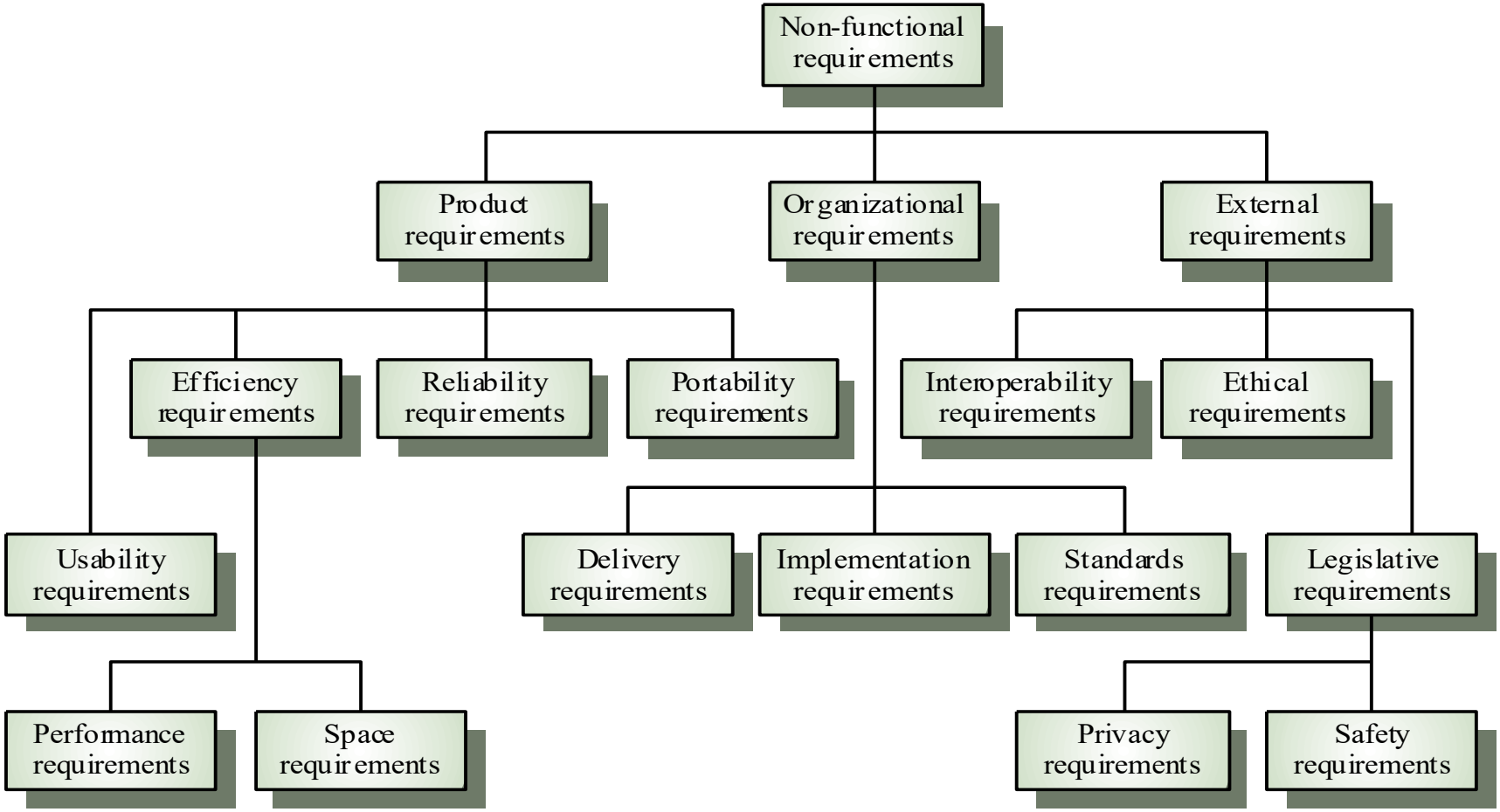
Non-Functional Requirements (NFR)

- **Design constraints:** Categories constraining the environment and technology of the system.
  - Platform (minimal requirements, OS, devices…)
  - Technology to be used (language, DB, …)

- **Commercial constraints:** Categories constraining the project plan and development methods
  - Development process (methodology) to be used and Development time frame.
  - Cost and delivery date
    - Often put in contract or project plan instead

Various NFR Types

- *Other ontologies also exist*



Source: Gerald Kotonya and Ian Sommerville, Requirements Engineering – Processes and Techniques, Wiley, 1998

Examples of Non-Functional Requirements

- **Product requirement**
  - It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard Ada character set.
- **APSE** standing for Ada Programming Support Environment was a specification for a programming environment to support software development in the Ada.
- **Process requirement**
  - The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCoSPSTAN95.
- **Security requirement**
  - The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

Measurable Non-Functional Requirements

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

Source: Gerald Kotonya and Ian Sommerville, Requirements Engineering – Processes and Techniques, Wiley, 1998

Goals

- A Goal
  - Conveys the intention or the objective of one or many stakeholders
- A goal is an objective or concern that guides the RE process. It can be used to discover and evaluate functional and non-functional requirements
  - A goal is not yet a requirement…
- Note: All requirements must be verifiable (by some test, inspection, audit etc.)

Example of Goal and NFR

- A system goal
  - The system should be easy to use by experienced controllers and should be organized in such a way that user errors are minimized.

- A verifiable usability requirement derived from this goal
  - Experienced controllers shall be able to use all the system functions after a total of three hours of training.
  - The average number of errors made by experienced controllers shall not exceed two per day.

  - Assumption: An experienced controller has at least 2 years experience with the old system (as stated by the stakeholder)

Application-Domain Requirements

- Derived from the application domain
- Describe system characteristics and features that reflect the domain
- May be new functional requirements, constraints on existing requirements, or define specific computations
- If domain requirements are not satisfied, the system may be unworkable

Examples of Application-Domain Requirements

- **Library system**
  - The system interface to the database must comply with standard Z39.50.
  - Because of copyright restrictions, some documents must be deleted immediately on arrival. Depending on the user's requirements, these documents will first be printed either locally or printed to a network printer and retrieved by the user.

- **Train protection system**
  - The deceleration of the train shall be computed as:
    $D_{train} = D_{control} + D_{gradient}$

    where $D_{gradient}$ is $9.81ms^2$ * compensated gradient / alpha and where the values of $9.81ms^2$ / alpha are known for different types of train.

Problems Concerning Application-Domain Requirements

- **Understandability**
  - Requirements are expressed in the language of the application domain
  - This is often not understood by software engineers developing the system

- **Implicitness / Tacit knowledge**
  - Domain specialists understand the area so well that they do not think of making the domain requirements explicit
  - People are often unaware of the tacit knowledge they possess and therefore cannot express it to others

Emergent Properties (when the system consists of several sub-systems)

- **Properties of the system as a whole**
  - Requirements which cannot be addressed by a single component, but which depend for their satisfaction on how all the software components interoperate
  - Only emerge once all individual subsystems have been integrated
  - Dependent on the system architecture

- **Examples of emergent properties**
  - Reliability
  - Maintainability
  - Performance
  - Usability
  - Security
  - Safety

The Requirements Engineering (RE) Process

Requirements within the software development process

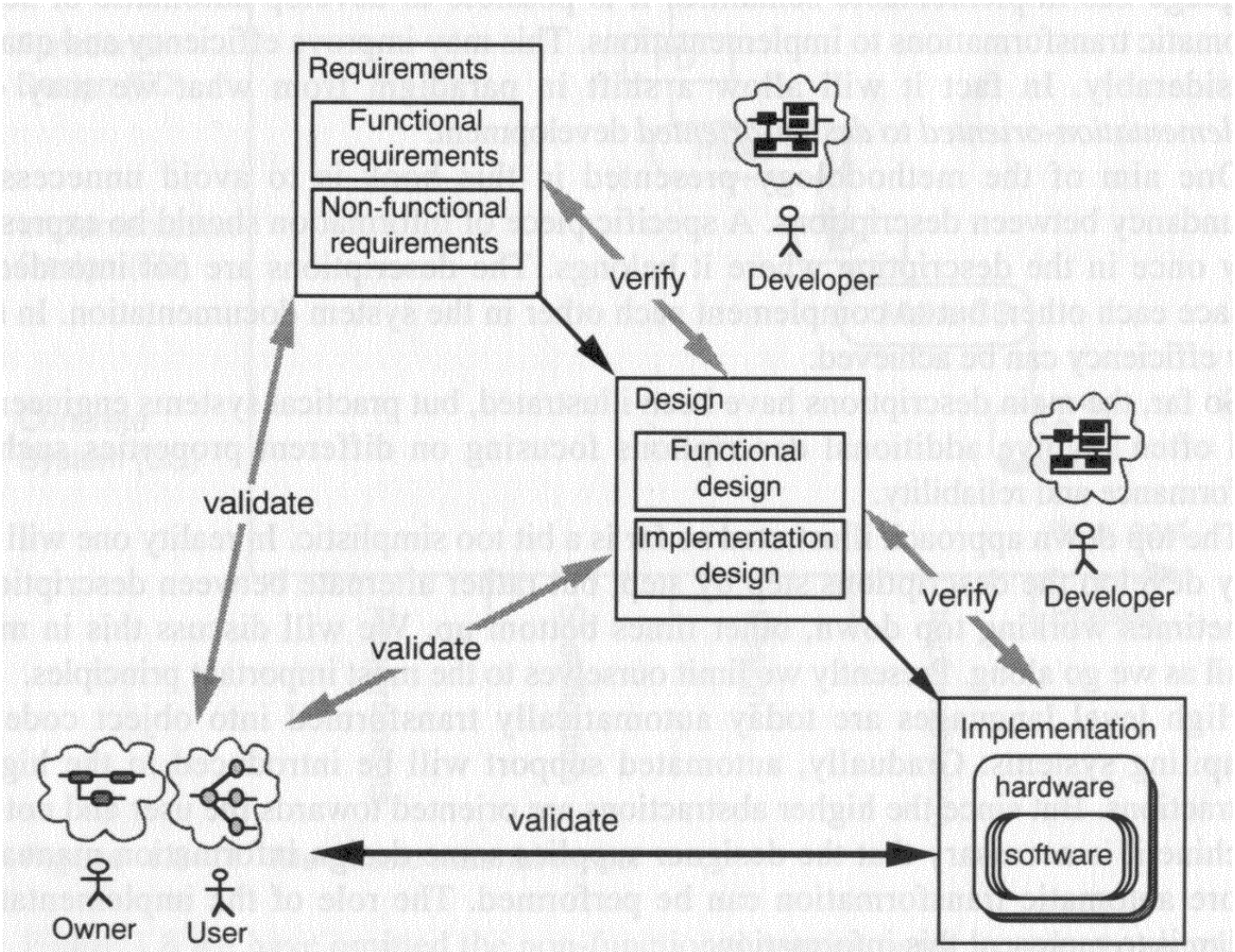
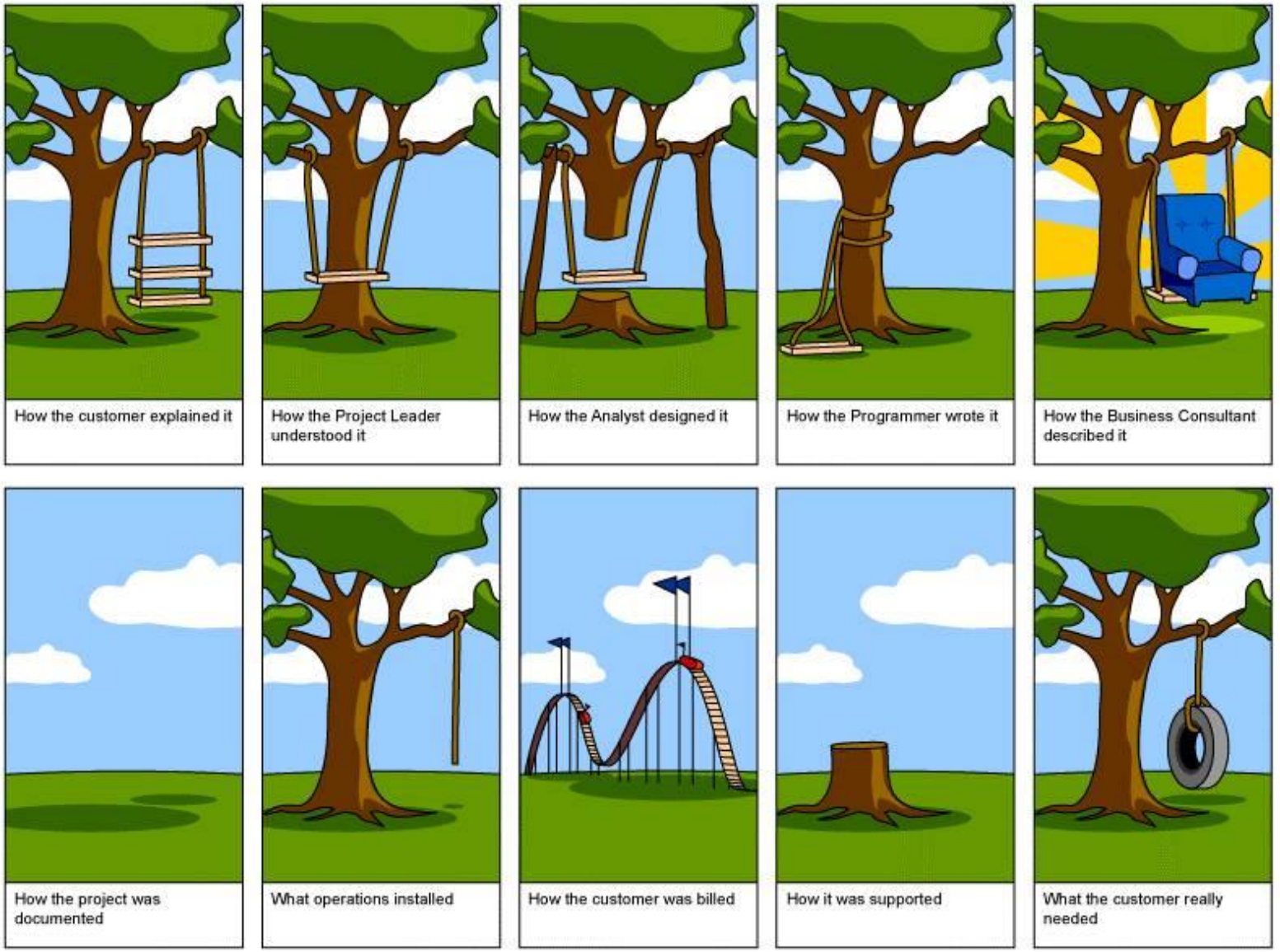


Figure 1.7 *Step-wise quality assurance*

What is the right system to build ?



RE activities and documents (Wiegers)

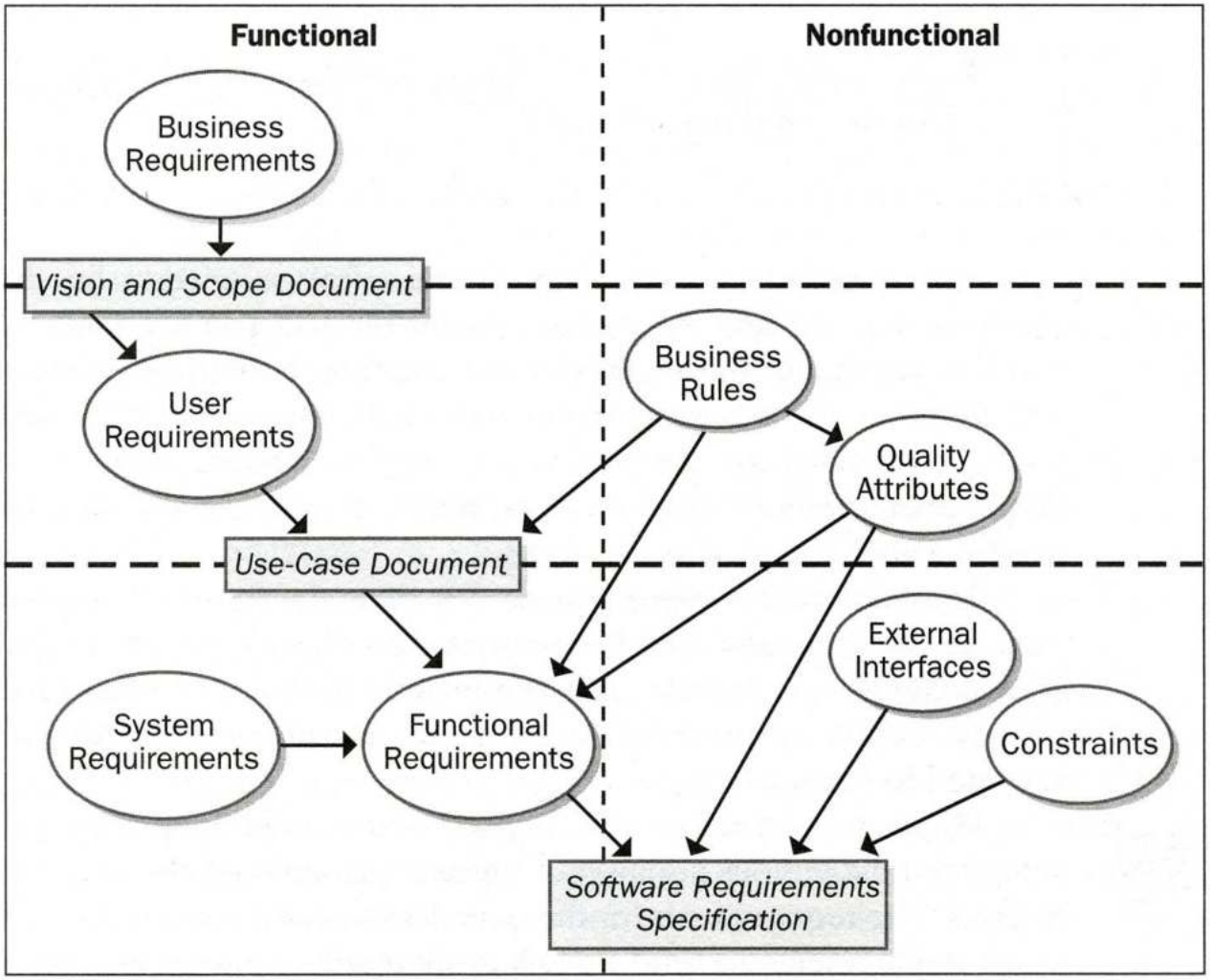


Figure 1-1 Relationship of several types of requirements information.

Notes on previous slide

- There needs to be an arrow from User requirements to System requirements. (The system has to be able to perform certain use cases. The same use cases must be supported by the software, therefore become Software requirements.)
- Business rules (including standards and regulations) are not only non-functional, they also include functional aspects (as shown by the arrows in the diagram).