Software Requirements Engineering SEC2071

Session 01

Requirements Engineering Fundamentals

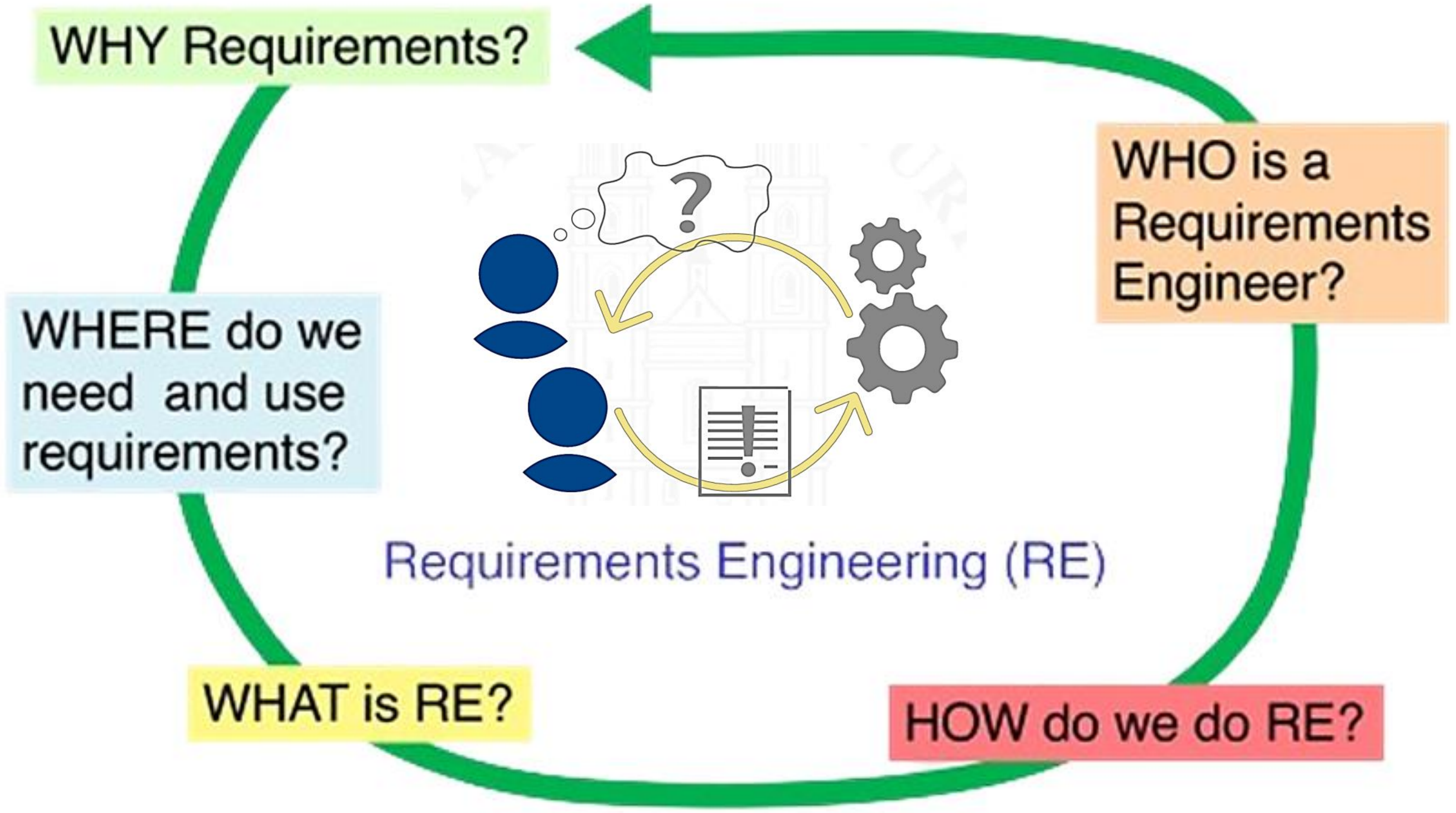
Muhammad Shahid
Department of Computer Science
National Textile University

shahid.Abdullah@ntu.edu.pk

This Session: Objectives

- Requirements Engineering
  - Where Do We Need and Use Requirements?
  - How do we do Requirements Engineering?
- Levels and Types of Requirements
  - Business Requirements
  - User Requirements
  - Functional Requirements
- Working with the Three Levels
- Requirements Development & Management
- Common Requirement Risks

Requirements Engineering (RE)


WHY Requirements?
WHO is a Requirements Engineer?
HOW do we do RE?
WHAT is RE?
WHERE do we need and use requirements?
?
Requirements Engineering (RE)

A Case: Sales and Customer Support Platform

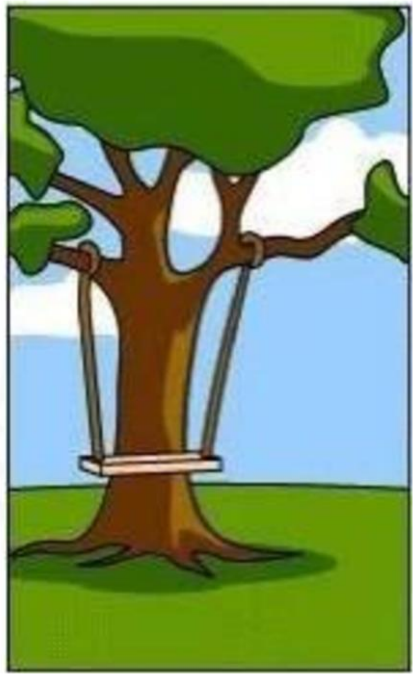
- Context: Corporate customer business in a a big international insurance company
- Current state:
    - Insurance agents have a laptop with
    - Powerpoint presentations of products
    - PDFs with terms and conditions
    - Editor for filling-in applications
    - Front-end app for querying the host system
    - Access to CRM system
    - No web interface for corporate customers
- Problems: Complicated, missing features, inflexible
- Decision: Build a new platform for sales and customer support
- Challenge: What, precisely, shall be built?

A Communication and Understanding Problem

WHY

Need, Problem

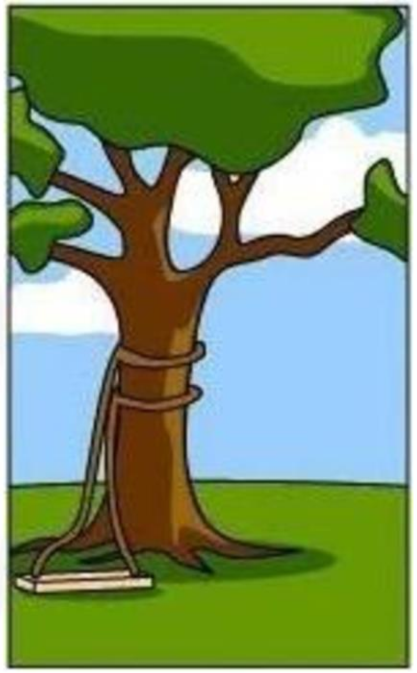
What the customer


Analysis

What the analyst

Design

What the architect


Deployed System

What the programmers

We need to know the requirements

What is a Requirement?

1. A need perceived by a stakeholder
2. A capability or property that a system shall have
3. A documented representation of a need, capability or property

[Glinz 2020]

Note: A Stakeholder is a person or organization who influences a system’s requirements or who is impacted by that system

Where Do We Need and Use Requirements?

WHERE

- In principle: For any kind of system
- In particular: for software-intensive systems

- More than just software:
  - Cyber-physical systems contain both software and physical components
  - Socio-technical systems span software, hardware, people and organizational aspects

What is a System?

- System – 1. A principle for ordering and structuring 2. A coherent, delimitable set of elements that – by coordinated action – achieve some purpose
- A system may comprise other systems
- The purpose achieved by a system may be delivered by
  - deploying it at the place(s) where it is used
  - selling/providing it as a product to its users
  - having providers who offer the system's capabilities as services to users
- Requirements Engineering is primarily concerned with systems in which software plays a major role

Forms of occurrence

- Requirements occur in various forms:
  - System requirements – How a system shall work and behave
  - Stakeholder requirements – Stakeholders’ desires and needs from a stakeholder perspective
  - User requirements – A subset of the stakeholder requirements
  - Domain requirements – Required domain properties of a socio-technical or cyber-physical system
  - Business requirements – Focus on business goals, objectives and needs of an organization

A Sample Problem

WHAT

- A ski resort operates several chairlifts. They need to renovate the access control to the chairlifts.
- **The idea**: Skiers buy RFID-equipped day access cards. Access to the lifts is controlled by RFID-enabled turnstiles. Whenever a turnstile senses a valid access card, it unlocks the turnstile for one turn, so that the skier can pass.
- **The task**: Build a software-controlled system for managing the access of skiers to the chairlifts.

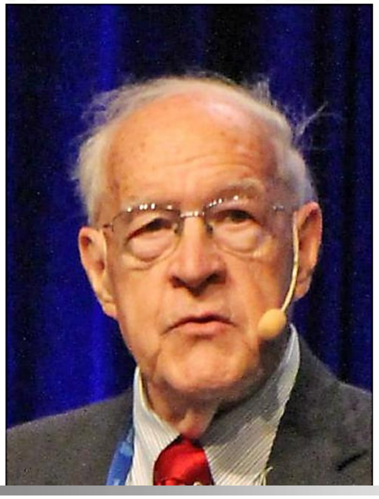
When Building Such a System...

- … we need to understand the problem and agree upon what to build

- Requirements Engineering

- "The hardest single part of building a software system is deciding precisely what to build."

  Frederick P. Brooks (1987)



Requirements Engineering: Definition

- Requirements Engineering (RE) is the systematic, iterative, and disciplined approach to develop an explicit requirements and system specification that all stakeholders agree upon

- Requirements Engineering – The systematic and disciplined approach to the specification and management of requirements with the goal of understanding the stakeholders' desires and needs and minimizing the risk of delivering a system that does not meet these desires and needs

Requirements Engineering: Definition (Traditional)

- The application of a systematic, disciplined, quantifiable approach to the specification and management of requirements; that is the application of engineering to requirements. [EEE 610.12-1990]

- Metaphor: Upfront engineering
- Goal: Complete, unambiguous requirements prior to design
- Smells: Paper, process
- Reality check: Does this always work?

Requirements Engineering: Definition (Customer-Oriented)

- Understanding and documenting the customers' desires and needs. [Glinz 1998]

- Metaphor: Customer satisfaction
- Goal: Understand the customer
- Reality check:
  1) Why not just code what the customer desires and needs?
  2) Who is "the customer"?

Risk-Oriented Requirements Engineering

- "We have no time for a complete specification."
- "This is too expensive!"
- "We're agile, so rough stories suffice."

- **Right question**:
  - "How much RE do we need such that the risk of deploying the wrong system becomes acceptable?"
- **Rule**:
  - The effort spent for Requirements Engineering shall be inversely proportional to the risk that one is willing to take.

Requirements Engineering typically concerned with "problem space"

- Requirements engineering aims at the problem space…
  - Framing the problem for a development task as comprehensively and as precisely as possible: Why is something necessary?
  - Deriving appropriate requirements towards the desired solution and agreement between stakeholders: What (capabilities, properties) is necessary?
- Design and implementation, in contrast, aims at the solution space, i.e., development and evolution of possible solutions of: How will the solution* be realized?

Differentiating the problem space from the solution space is essential!

How do we do Requirements Engineering?

HOW

- Four major tasks:
  1) Eliciting the requirements
  2) Analyzing and documenting the requirements
  3) Validating the requirements
  4) Managing and evolving the requirements

The Requirements Engineer

WHO

- Mostly a role, not a job title
- The role is part of many job functions: business analyst, system analyst, application engineer, software engineer, systems engineer, product owner, ...
- People act as requirements engineers if they
  - elicit, document, validate and manage requirements,
  - have in-depth knowledge of Requirements Engineering, enabling them to define RE processes, select appropriate RE practices and apply them properly,
  - are able to bridge the gap between the problem and potential solutions

Is Requirements Engineering worthwhile?

WHY

- Systems development is an expensive and risky business.

- Requirements Engineering
  - helps understand the problem
  - reduces the risk of failure or costly modifications in later development stages
  - provides a proper basis for estimating development effort and cost
  - is a prerequisite for testing the developed system properly

Is Requirements Engineering worthwhile?

WHY

Requirements Engineering contributes to:

- **Reducing error and rework cost**
- **Managing the development risk**
  - Meet stakeholders’ desires and needs
  - Reliable estimates for deadlines and cost

Lower cost

Higher benefit

Requirements serve as a basis for…

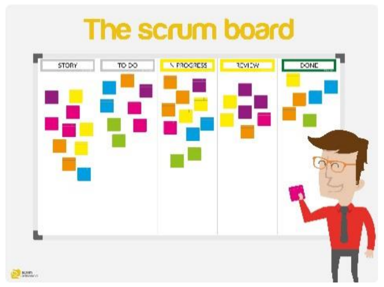
Customer relation and communication	Requirements as basis for communication between stakeholders (customer, user, engineers, regulatory bodies, …)
Software design	Requirements as basis for the structure and behaviour of a software system (from the perspective of the developer)
Project organisation and management	Requirements shape process-related aspects, i.e. the organisation of the development and project management approach
Quality assurance and acceptance	Requirements serve as basis for verification (testing) and acceptance of final product
Software maintenance and evolution	Requirements dictate the (necessary) quality of the system realization, risks, and overhead in operations

RE is a part of system development

- RE is not only a matter of software development
- It is generally applied to:
    - Software systems
    - Systems in general (incl. embedded software)
    - Software-intensive products and services
    - Processes
- We will focus on requirements engineering in the context of software development, i.e.:
    - Software
    - Software-intensive systems and products

RE has Different Forms and Interpretations

- **“Traditional” RE**
  - Result: A requirements specification (document)
  - Basis for contracting and implementation
  - Typically found in customer/supplier situations
- **“Agile” RE**
  - Result: Product backlog, user stories, epics, …
  - Basis for sprint planning
  - Typically found in “in-house” SW development projects
- **“Change-based” RE**
  - Result: Issues, Tickets, Change Requests,…
  - Basis for product maintenance and evolution
  - Typically found in in many “long-lived systems”
- **“Code-driven” RE**
  - Results: Code comments, man pages, mailing lists,…
  - Basis for understanding and changing code
  - Typically found in open-source projects



In consequence, the requirements engineer can appear in different roles

- Requirements engineer
- Business analyst
- Product owner

Responsibilities and characteristics tend to be the same

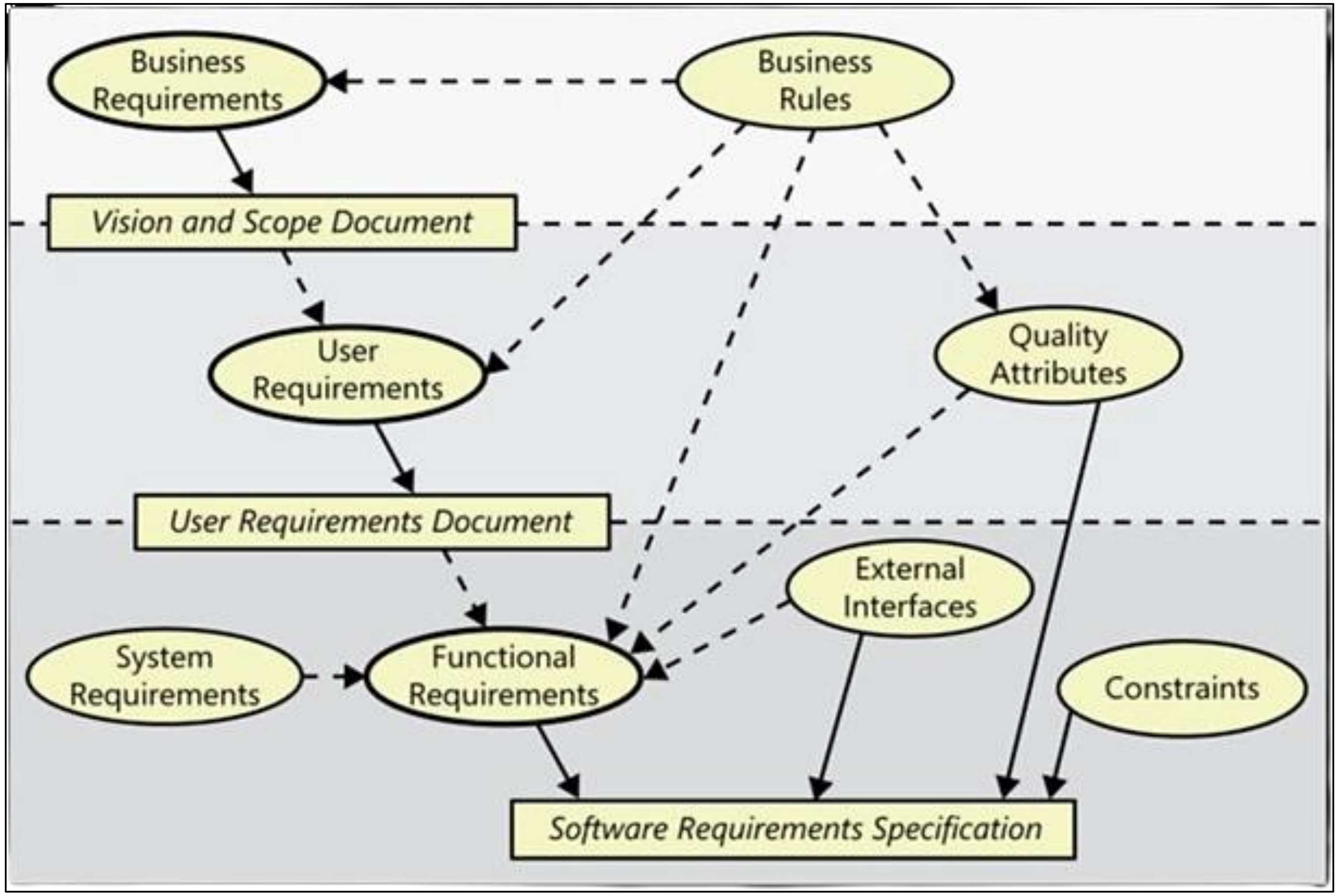
- **Responsibilities**
  - Elicit, analyse, document, validate, and manage requirements
  - Plan, estimate, organise, control the RE process

- **Characteristics**
  - Good communication skills
  - Critical thinking
  - Empathy
  - Conflict resolution skills
  - Not necessarily a domain expert
  - Not necessarily a technical expert

Levels and Types of Requirements

- Software requirements include three distinct levels:
  - Business Requirements
  - User Requirements
  - Functional Requirements

Levels and Types of Requirements



Business Requirements

- Business requirements describe why the organization is implementing the system—the business benefits the organization hopes to achieve
- The focus is on the business objectives of the organization or the customer who requests the system
- We like to record the business requirements in a vision and scope document
- Example:
  - *Suppose an airline wants to reduce airport counter staff costs by 25%. This goal might lead to the idea of building a kiosk that passengers can use to check in for their flights at the airport.*

User Requirements

- Describe goals or tasks the users must be able to perform with the product that will provide value to someone
- The domain of user requirements also includes descriptions of product attributes or characteristics that are important to user satisfaction
- Ways to represent user requirements include use cases, user stories, and event-response tables
- Example:
  - Use case: *"Check in for a flight"* using an airline's website or a kiosk at the airport
  - User story: *"As a passenger, I want to check in for a flight so I can board my airplane."*

Functional Requirements

- Functional requirements specify the behaviors the product will exhibit under specific conditions
- They describe what the developers must implement to enable users to accomplish their tasks (user requirements), thereby satisfying the business requirements
- *"The Passenger shall be able to print boarding passes for all flight segments for which he has checked in" or "If the Passenger's profile does not indicate a seating preference, the reservation system shall assign a seat."*

Business Analyst

- The business analyst (BA) documents functional requirements in a software requirements specification (SRS), which describes as fully as necessary the expected behavior of the software system

- The SRS is used in development, testing, quality assurance, project management, and related project functions

System Requirements

- System requirements describe the requirements for a product that is composed of multiple components or subsystems

- A “system” in this sense is not just any information system. A system can be all software, or it can include both software and hardware subsystems

- People and processes are part of a system, too, so certain system functions might be allocated to human beings

Business Rules

- Business rules include corporate policies, government regulations, industry standards, and computational algorithms

Quality Attributes

- Quality attributes are aka quality factors, quality of service requirements, and constraints

- They describe the product’s characteristics in various dimensions that are important either to users or to developers and maintainers, such as performance, safety, availability, and portability

- Other classes of nonfunctional requirements describe external interfaces between the system and the outside world

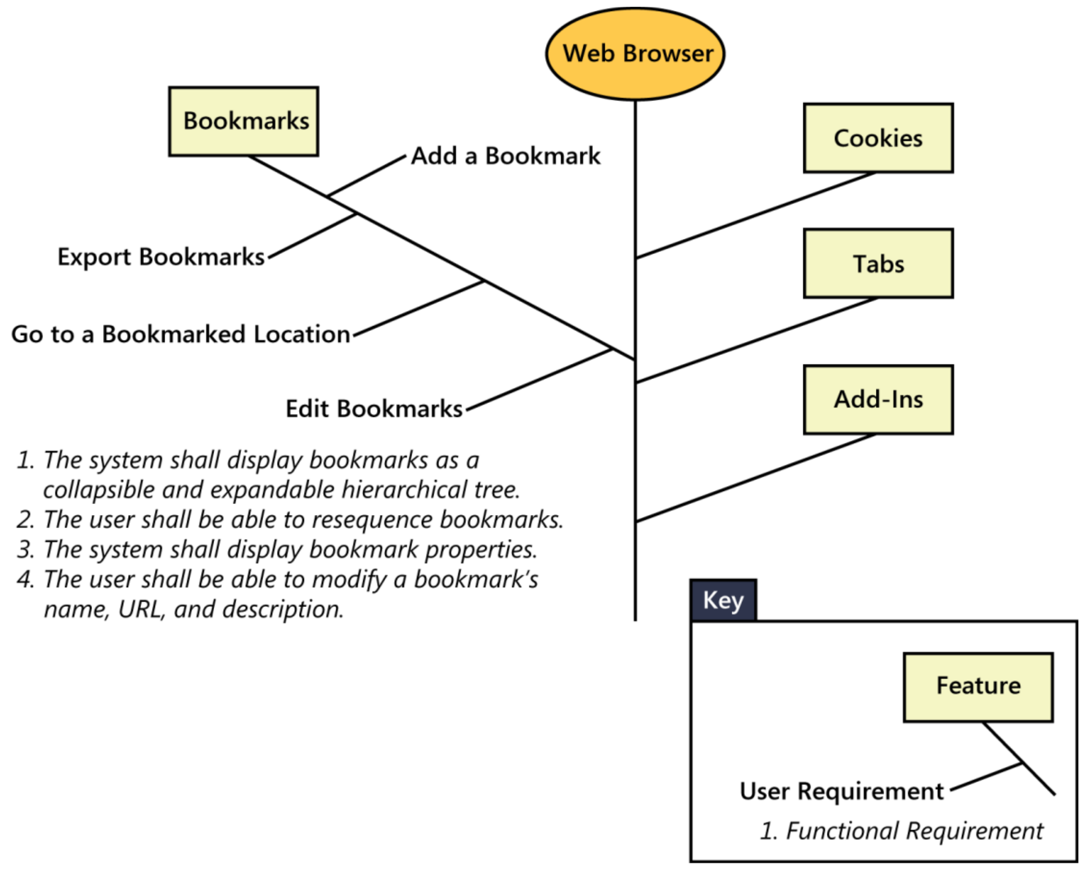
Feature

- A feature consists of one or more logically related system capabilities that provide value to a user and are described by a set of functional requirements

- A feature can encompass multiple user requirements, each of which implies that certain functional requirements must be implemented to allow the user to perform the task described by each user requirement

Feature

- Figure 1-2 illustrates a feature tree, an analysis model that shows how a feature can be hierarchically decomposed into a set of smaller features, which relate to specific user requirements and lead to specifying sets of functional requirements

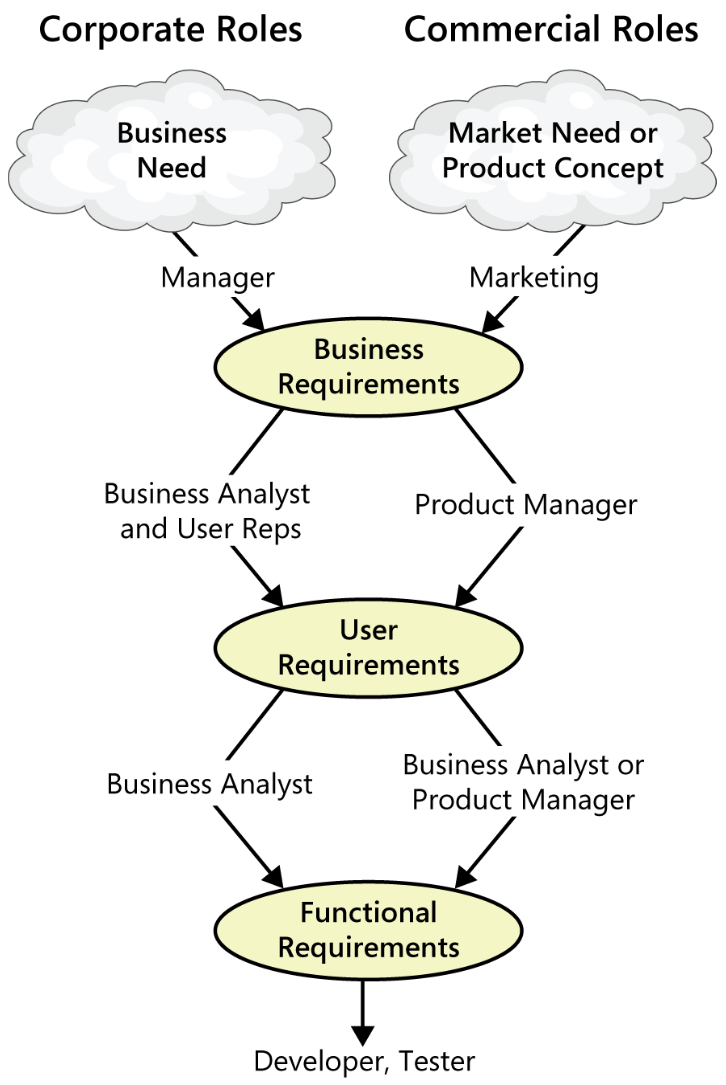


Web Browser
Bookmarks
Add a Bookmark
Cookies
Export Bookmarks
Tabs
Go to a Bookmarked Location
Edit Bookmarks
Add-Ins
1. The system shall display bookmarks as a collapsible and expandable hierarchical tree.
2. The user shall be able to resequence bookmarks.
3. The system shall display bookmark properties.
4. The user shall be able to modify a bookmark's name, URL, and description.
Key
Feature
User Requirement
1. Functional Requirement

FIGURE 1-2 Relationships among features, user requirements, and functional requirements.

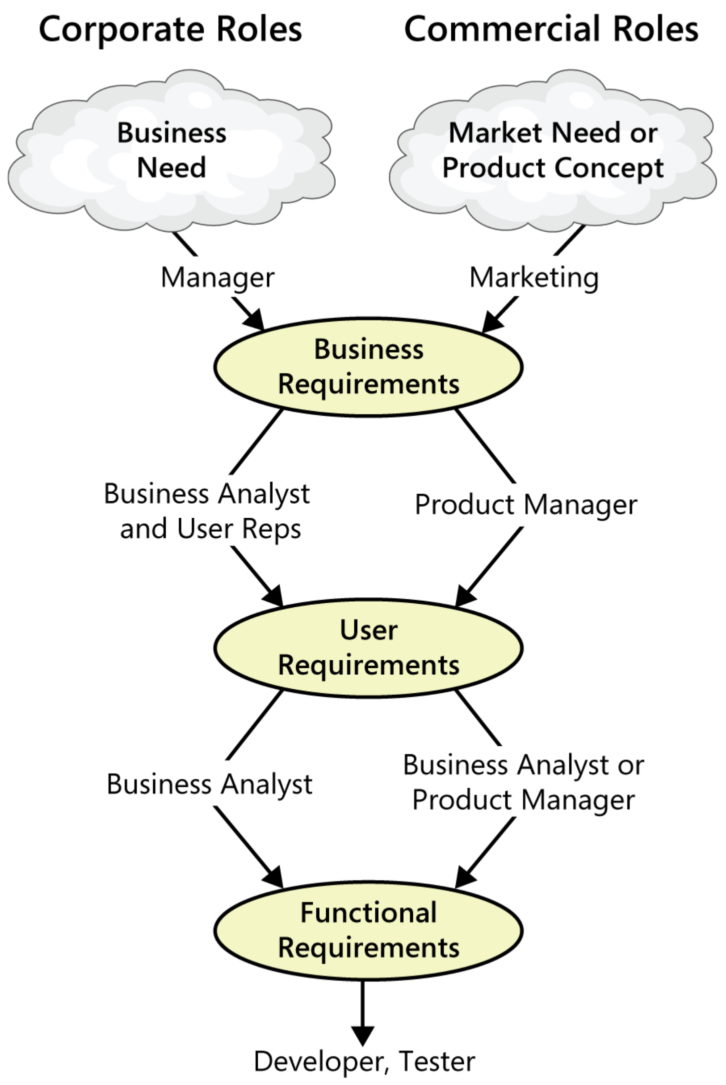
Working with the Three Levels

- Based on an identified business need, a market need, or an exciting new product concept, managers or marketing define the business requirements for software that will help their company operate more efficiently or compete successfully in the marketplace
- A business analyst works with user representatives to identify user requirements
- Companies developing commercial products often identify a product manager to determine what features to include in the new product

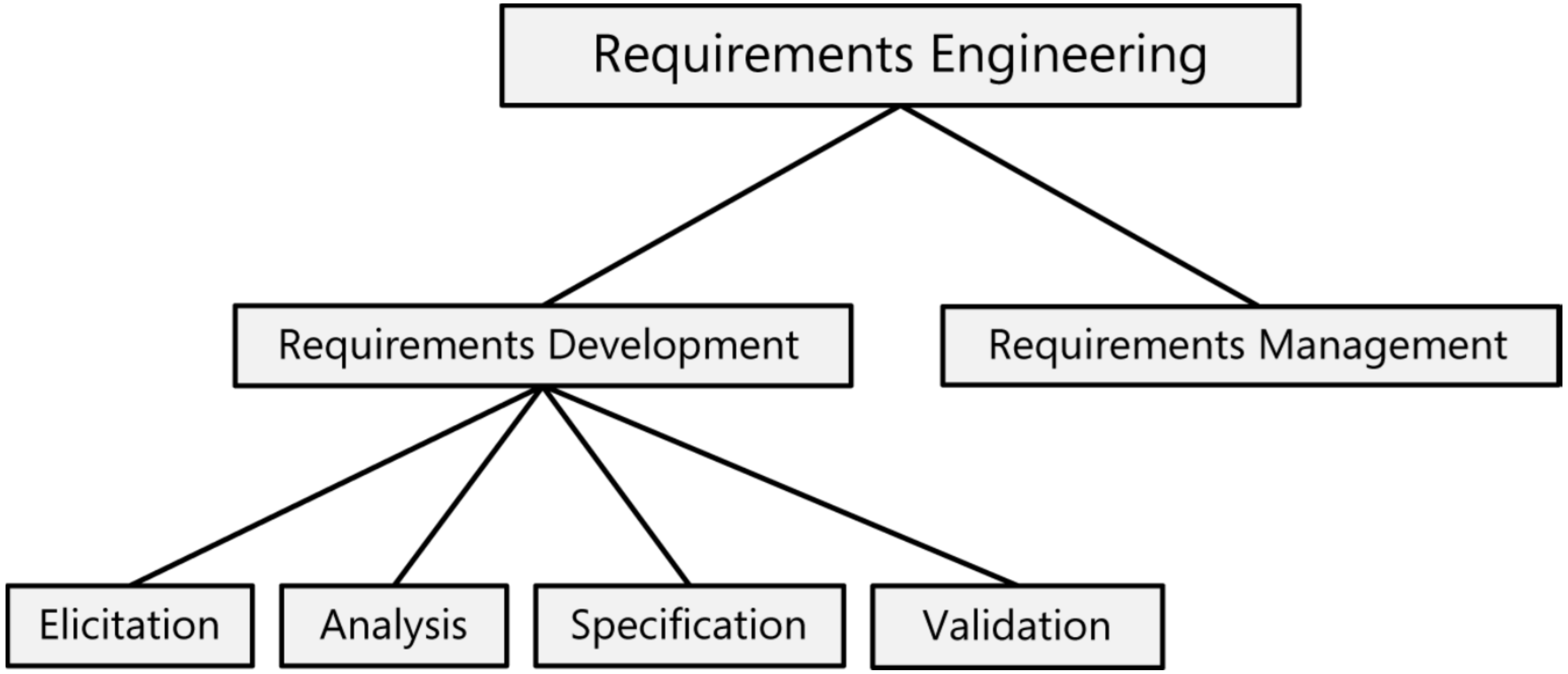


Working with the Three Levels

- Each user requirement and feature must align with accomplishing the business requirements
- From the user requirements, the BA or product manager derives the functionality that will let users achieve their goals. Developers use the functional and nonfunctional requirements to design solutions that implement the necessary functionality, within the limits that the constraints impose
- Testers determine how to verify whether the requirements were correctly implemented



Requirements Development & Management



Requirements Development: Elicitation

- Elicitation encompasses all the activities involved with discovering requirements, such as brainstorming, interviews, workshops, document analysis, prototyping, and others
- The key actions are:
  - Identifying the product’s expected user classes and other stakeholders
  - Understanding user tasks and goals and the business objectives with which those tasks align
  - Learning about the environment in which the new product will be used
  - Working with individuals who represent each user class to understand their functionality needs and their quality expectations

Requirements Development: Analysis

- Analyzing requirements involves reaching a richer and more precise understanding of each requirement and representing sets of requirements in multiple ways

- Following are the principal activities:
  - Analyzing the information received from users to distinguish their task goals from functional requirements, quality expectations, business rules, suggested solutions, and other information
  - Decomposing high-level requirements into an appropriate level of detail
  - Deriving functional requirements from other requirements information
  - Understanding the relative importance of quality attributes
  - Allocating requirements to software components defined in the system architecture
  - Negotiating implementation priorities
  - Identifying gaps in requirements or unnecessary requirements as they relate to the defined scope

Requirements Development: Specification

- Requirements specification involves representing and storing the collected requirements knowledge in a persistent and well-organized fashion
- The principal activity is translating the collected user needs into written requirements and diagrams suitable for comprehension, review, and use by their intended audiences
- “As a ‘who,’ I want to ‘what,’ so that ‘why’”

Requirements Development: Validation

- Requirements validation confirms that you have the correct set of requirements information that will enable developers to build a solution that satisfies the business objectives

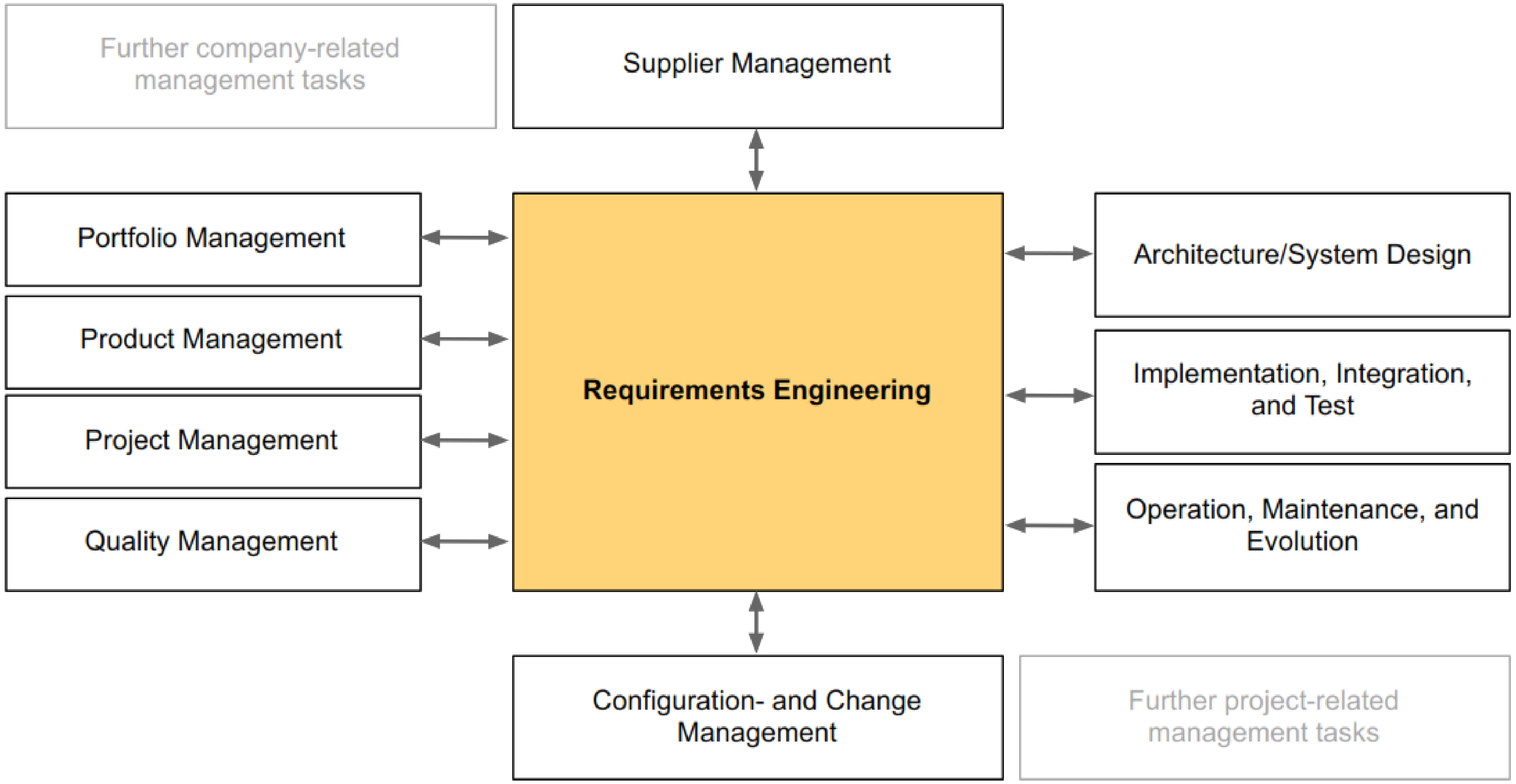
- The central activities are:
  - Reviewing the documented requirements to correct any problems before the development group accepts them
  - Developing acceptance tests and criteria to confirm that a product based on the requirements would meet customer needs and achieve the business objectives

Requirements Management: Definition

- Requirements Management (RM) aims at efficiently and effectively managing and using requirements along the whole system lifecycle

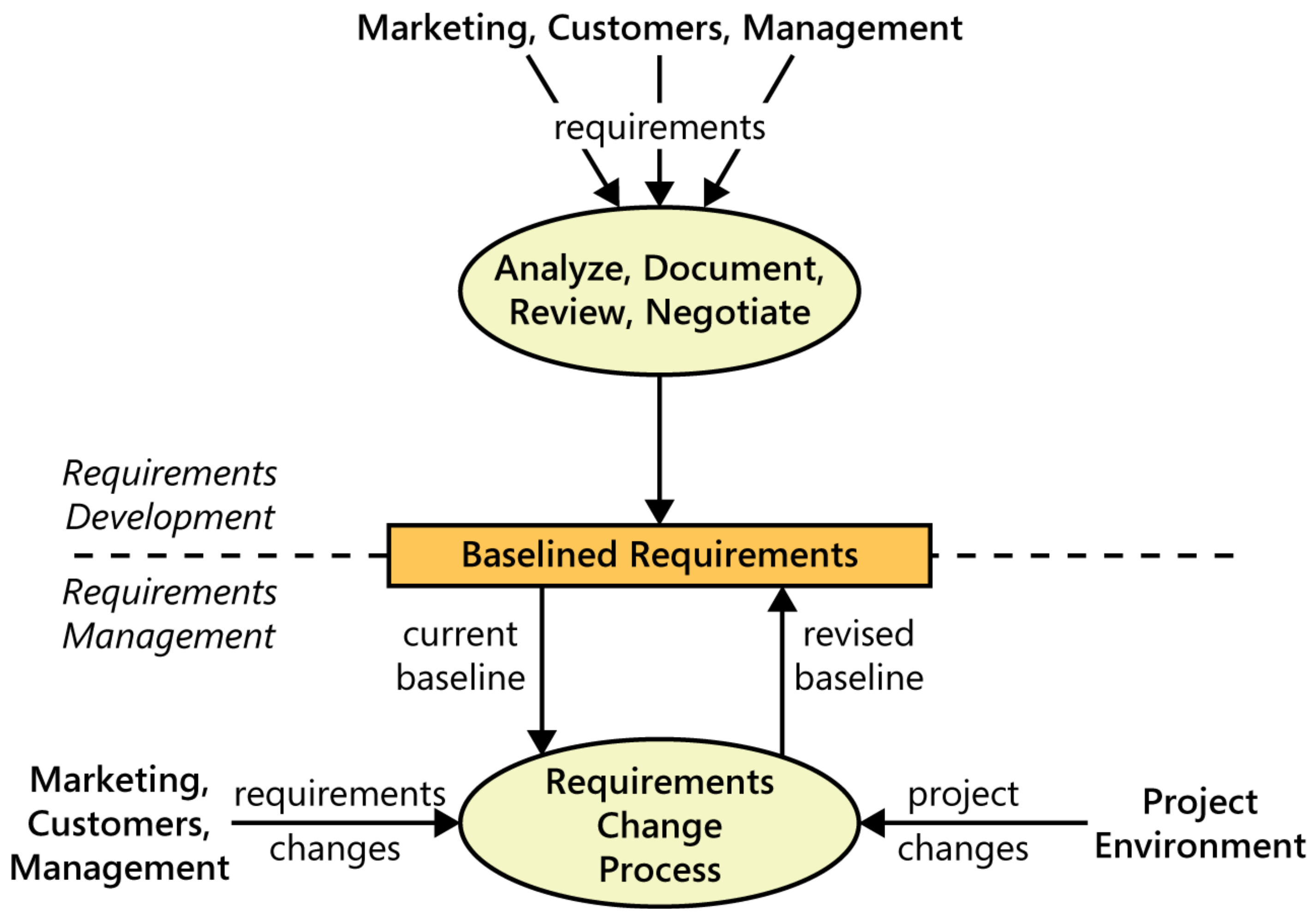
- Elementary tasks include, but are not limited to:
  - Archiving and baselining requirements
  - Modifying requirements due to new knowledge/insights
  - Tracing and verifying requirements (impact analysis, support of change processes)

Requirements Engineering and Requirements Management …


Further company-related management tasks
Supplier Management
Portfolio Management
Product Management
Project Management
Quality Management
Requirements Engineering
Architecture/System Design
Implementation, Integration, and Test
Operation, Maintenance, and Evolution
Configuration- and Change Management
Further project-related management tasks

RE and RM build a key interface to several activities in the development life cycle

Requirements Management



Common Requirement Risks

- It can cost far more to correct a defect that’s found late in the project than to fix it shortly after its creation. Suppose it costs $1 to find and fix a requirement defect while you’re still working on the requirements. If you discover that error during design instead, you must pay the $1 to fix the requirement error, plus another $2 or $3 to redo the design that was based on the incorrect requirement. Suppose, though, that no one finds the error until a user calls with a problem. Depending on the type of system, the cost to correct a requirement defect found in operation can be $100 or more on this relative scale.

Common Requirement Risks

- Insufficient user involvement:
  - Customers often don’t understand why it is so essential to work hard on eliciting requirements and assuring their quality
  - Developers might not emphasize user involvement, perhaps because they think they already understand what the users need.

- When reviewing and validating the requirements, is that the business analyst might not understand and properly record the true business or customer needs

Common Requirement Risks

- Inaccurate planning: Vague, poorly understood requirements lead to overly optimistic estimates, which come back to haunt you when the inevitable overruns occur
- The top contributors to poor software cost estimation are
  - Frequent requirements changes
  - Missing requirements
  - Insufficient communication with users
  - Poor specification of requirements
  - Insufficient requirements analysis

Common Requirement Risks

- Creeping user requirements:
  - As requirements evolve during development, projects often exceed their planned schedules and budgets
  - To manage scope creep, begin with a clear statement of the project's business objectives, strategic vision, scope, limitations, and success criteria

- Ambiguous requirements:
  - One symptom of ambiguity in requirements is that a reader can interpret a requirement statement in several ways
  - Another sign is that multiple readers of a requirement arrive at different understandings of what it means

Common Requirement Risks

- Gold plating:
  - Gold plating takes place when a developer adds functionality that wasn’t in the requirements specification but which the developer believes “the users are just going to love.” If users don’t care about this functionality, the time spent implementing it is wasted
  - Rather than simply inserting new features, developers and BAs should present stakeholders with creative ideas for their consideration
  - Customers sometimes request certain features or elaborate user interfaces that look attractive but add little value to the product. Everything you build costs time and money, so you need to maximize the delivered value

Common Requirement Risks

- Overlooked stakeholders:
    - Most products have several groups of users who might use different subsets of features, have different frequencies of use, or have varying levels of experience
    - If you don’t identify the important user classes for your product early on, some user needs won’t be met
    - After identifying all user classes, make sure that each has a voice

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

Functional Requirements: Examples

- The user shall be able to search either all the initial set of databases or select a subset from it
- The system shall provide appropriate views 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

Non-Functional Requirements

- Non-functional requirements (NFR) 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

Non-Functional Requirements

- Three main categories:
  1) **Performance requirements** reflecting: usability, efficiency, reliability, maintainability and reusability (note: also security requirements)
     - Response time, throughput
     - Resource usage
     - Reliability, availability
     - Recovery from failure
     - Allowances for maintainability and enhancement
     - Allowances for reusability
  2) **Design constraints:** Categories constraining the environment and technology of the system
     - Platform (minimal requirements, OS, devices…)
     - Technology to be used (language, DB, …)

Non-Functional Requirements

3) **Commercial constraints:** Categories constraining the project plan and development methods
   - Development process (methodology) to be used
   - Cost and delivery date

Non-Functional: Examples

- **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
- **Process requirement**
  - The system development process and deliverable documents shall conform to the process and deliverables defined in ABC100
- **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

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

Goals

- A Goal
  - Conveys the intention or the objective of one or many stakeholders
  - Can guide the discovery of verifiable non-functional requirements that can be tested objectively

Goal and NFR: Example

- 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

Application-Domain Requirements: Examples

- **Library system**
  - The system interface to the database must comply with standard Z9.5
  - 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.

Application-Domain Requirements: Problem

- **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

- 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

Session: Recap

- Requirements Engineering
    - Where Do We Need and Use Requirements?
    - How do we do Requirements Engineering?
- Levels and Types of Requirements
    - Business Requirements
    - User Requirements
    - Functional Requirements
- Working with the Three Levels
- Requirements Development & Management
- Common Requirement Risks

Questions