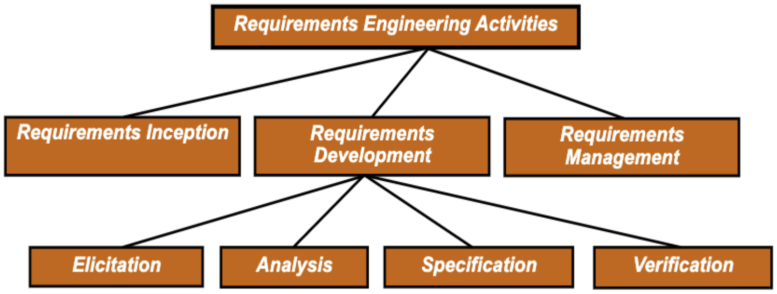
Lecture 2

➔ (Week 2 - Jan 15, 2024)

Requirements:

- A requirement is a simple statement of what the system must do or what characteristics it must have.
- It captures the purpose of a system.
- It expresses the ideas to be embodied in the system or application under development.
- It is 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.
- It is a short and concise piece of information.
- It says something about the system.
- All the stakeholders have agreed that it is valid.
- It helps solve the customer's problem.

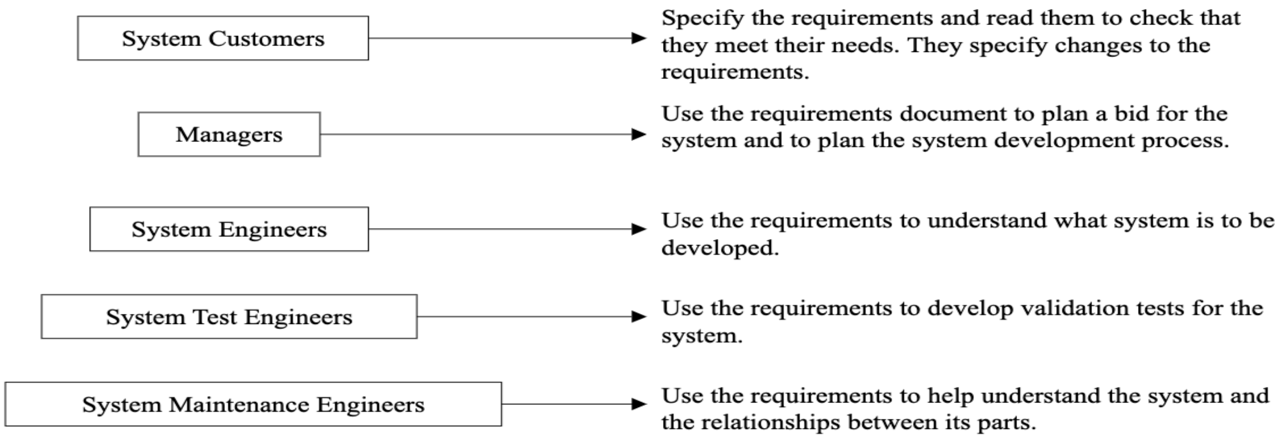


Requirements Engineering (RE):

- RE is the activity of development, elicitation, specification, analysis, and management of the stakeholder requirements 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.

Requirements in IEEE 830-1993:

- A requirement is defined as a condition or capability needed by a user to solve a problem or achieve an objective.
- It is 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.



Specify the requirements and read them to check that they meet their needs. They specify changes to the requirements.

Use the requirements document to plan a bid for the system and to plan the system development process.

Use the requirements to understand what system is to be developed.

Use the requirements to develop validation tests for the system.

Use the requirements to help understand the system and the relationships between its parts.

Requirement Engineering Activities:

- ➔ Inception: Start the process (identification of business need, new market opportunity, great ideas).
- ➔ Requirements elicitation: Requirements discovered through consultation with stakeholders.
- ➔ Requirements analysis and negotiation: Requirements are analyzed, and conflicts resolved.
- ➔ 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.

1. Requirement Inception

- Start of the process.
- Identifies the business need, market opportunity, or a new idea.
- Involves building a business case and preliminary scope definition.
- Stakeholders include business managers, marketing, and product managers.
- Techniques: Brainstorming, Joint Application Development (JAD) meetings.

2. Requirement Elicitation

- Gathering information about the problem domain, constraints, and questions.
- Difficulties due to communication challenges and changing requirements.
- Techniques: Brainstorming, interviews, task observations, use cases, prototyping, questionnaires, etc.

3. Requirement Analysis

- Process of studying stakeholder needs for a solution.
- May involve reorganizing workflows, creating new systems, or developing software.
- Objectives: Resolve conflicts, define system boundaries, elaborate system requirements.

4. Requirement Specifications

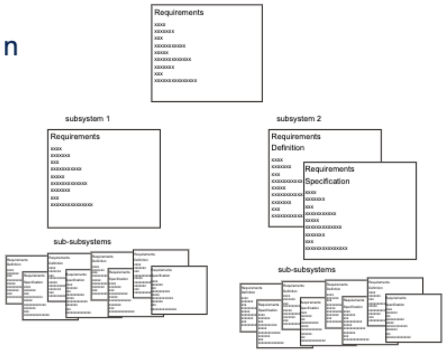
- Defines the behavior of a new system (solution domain) to address the problem domain.
- Software Requirements Specification (SRS) document describes essential requirements and interfaces.
- Establishes an agreement between customers and contractors.

5. Requirement Validation and Verification

- Validation ensures the right product is being built, verification checks if it's built correctly.
- Performed at every stage.
- Techniques include checks, formal reviews, logical analysis, prototypes, and functional tests.

6. Requirements Management

- Essential for handling changes in requirements.
- Changes can result from business process, technology, or better understanding.
- Traceability is crucial for effective requirements management.



Requirements Documents

- Types of requirements documents include Vision and Scope Document, Elicitation notes, Problem domain Requirements, System requirements document, and Software requirements document.
- Software requirements are usually part of the system requirements.

Challenges in the Requirements Process

- Challenges include a lack of expertise, incomplete initial ideas, and difficulty in using complex tools and methods.

Statistics from NIST Report

- The NIST report reveals that 70% of defects are introduced in the specification phase.
- Only 5% of specification inadequacies are corrected in the specification phase.
- Extensive testing is essential, but it detects specification errors late in the process, which is costly.

Requirements Categories

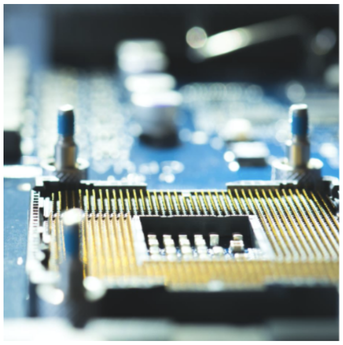
- Four main categories of requirements: Business requirements, User requirements, Functional requirements (FR), and Non-functional requirements (NFR).

Business Requirements

- Describe what must be delivered to provide value.
- Examples include increasing market share, shortening order processing time, and reducing customer service costs.

User Requirements

- Specify what the user expects the system to perform.
- Describe tasks that users perform as part of business operations, like scheduling appointments or placing orders.



Functional Requirements (FR)

- Define system functions.
- Describe what the system should do, inputs accepted, outputs produced, data stored, computations performed, and timing.

Non-Functional Requirements (NFR)

- Define important system characteristics (quality attributes) such as performance and scalability.
- Critical for the system's usefulness and can be challenging to state precisely.
- Often discovered during user conversations in the analysis phase.
- All requirements, including NFRs, must be verifiable.

Non-Functional Requirements (NFR) Types (1)

- Performance requirements characterize system properties like expected performance, usability, efficiency, reliability, maintainability, reusability, and security.
- Examples include response time, throughput, resource usage, reliability, availability, recovery from failure, maintainability, and reusability.
- Design constraints cover the system's environment and technology, including the platform, technology to be used, and commercial constraints like cost and delivery date.

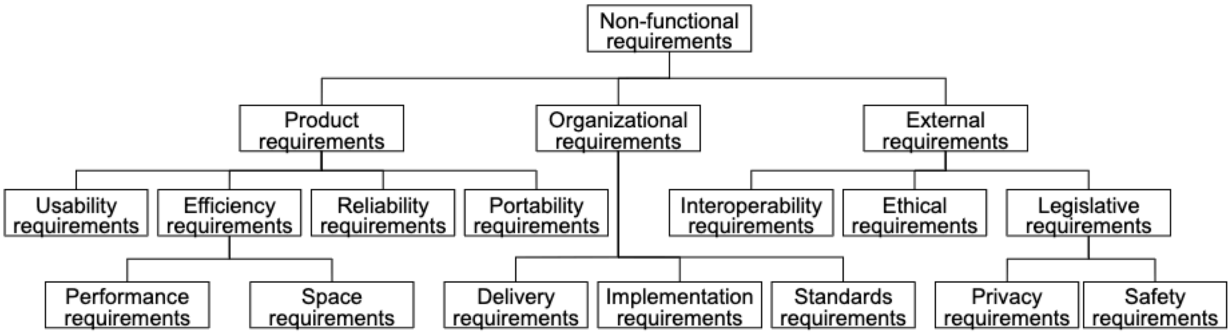
Non-Functional Requirements (NFR) Types (2)

- Important NFRs to Users: performance, security, usability, compatibility, accessibility, reliability, flexibility, and interoperability.
- Important NFRs to Developers: maintainability, portability, reusability, and testability.

Non-Functional Requirements (NFR) Types (3)

- Usability measures how easily users can operate and interact with a system.
- Dependability includes reliability, robustness, and safety.
- Reliability focuses on the system's ability to perform required functions under specified conditions for a defined period.
- Robustness evaluates the system's performance in the presence of invalid inputs or stressful conditions.
- Safety assesses the absence of catastrophic consequences to the environment.

NFR: Sommerville's Classification


Non-functional requirements

Product requirements

Organizational requirements

External requirements

Usability requirements

Efficiency requirements

Reliability requirements

Portability requirements

Interoperability requirements

Ethical requirements

Legislative requirements

Performance requirements

Space requirements

Delivery requirements

Implementation requirements

Standards requirements

Privacy requirements

Safety requirements

Non-Functional Requirements (NFR) Types (4)

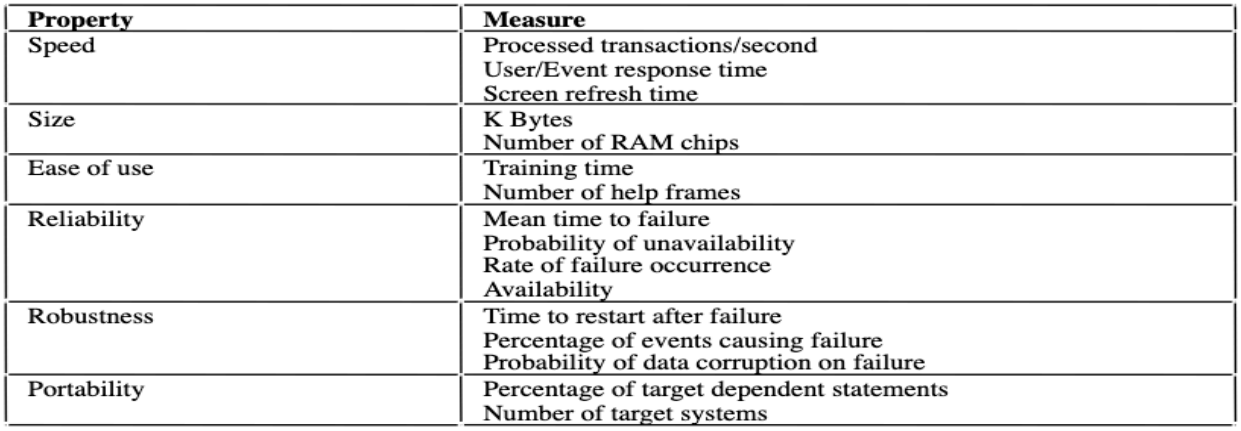
- Performance includes quantifiable attributes like response time, throughput, availability, and accuracy.
- Response time measures the system's reaction time to user inputs.
- Throughput evaluates how much work the system can accomplish in a specific timeframe.

- Availability assesses the system's operational accessibility when needed.
- Accuracy quantifies the magnitude of error.
- Supportability includes adaptability and maintainability for ease of changes and updates.

Examples of Non-Functional Requirements

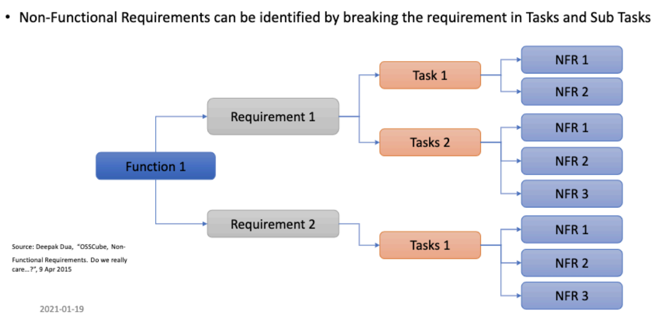
- Product Requirement: The product should be readily portable to the Linux operating system.
- Product Requirement: The product should be easy to use with one hand.
- Product Requirement: The product should function under poor lighting conditions, and users will wear gloves.
- Product Requirement: The product should identify an aircraft within 0.25 seconds.
- Process Requirement: The system development process and deliverables shall conform to XYZCoSPSTAN95.
- Security Requirement: The system shall not disclose personal information about customers except for their name and reference number to operators.

Quantitative 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

Identification of Non Functional Requirements

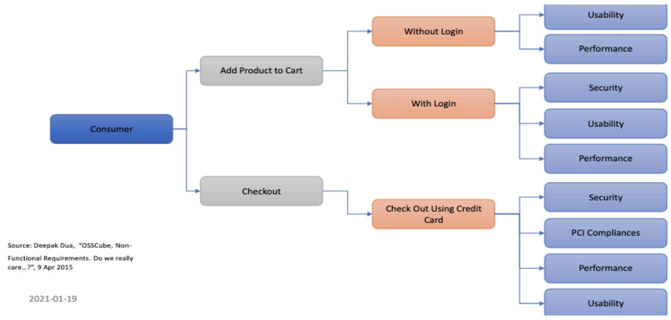


- Non-Functional Requirements can be identified by breaking the requirement in Tasks and Sub Tasks

Source: Deepak Dua, "OSSCube, Non-Functional Requirements. Do we really care…?", 9 Apr 2015

2021-01-19

Example: ECommerce Website

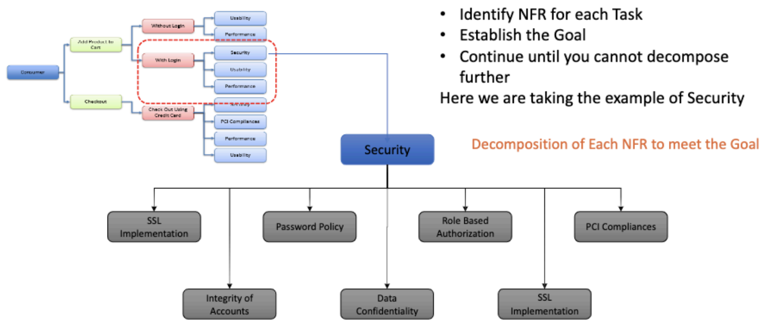


Source: Deepak Dua, "OSSCube, Non-Functional Requirements. Do we really care…?", 9 Apr 2015

2021-01-19

Identification of Non Functional Requirements

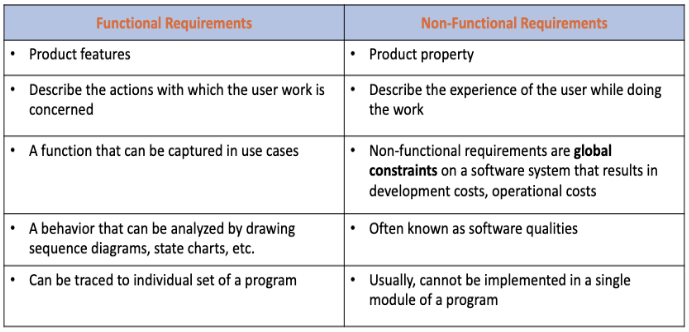
Functional vs. Non-functional



- Identify NFR for each Task
- Establish the Goal
- Continue until you cannot decompose further

Here we are taking the example of Security

Decomposition of Each NFR to meet the Goal



Functional Requirements	Non-Functional Requirements
• Product features	• Product property
• Describe the actions with which the user work is concerned	• Describe the experience of the user while doing the work
• A function that can be captured in use cases	• Non-functional requirements are **global constraints** on a software system that results in development costs, operational costs
• A behavior that can be analyzed by drawing sequence diagrams, state charts, etc.	• Often known as software qualities
• Can be traced to individual set of a program	• Usually, cannot be implemented in a single module of a program

Sources of Requirements

- Stakeholder discussions to elicit individual perspectives.
- Competitive analysis of existing systems.
- Policy and Procedure Manuals for identifying system interactions and constraints.
- Data from marketing and customer care departments, including customer surveys.
- Consideration of legacy systems, including System Manuals, Specifications, Issue Logs, and Enhancement Requests.

Requirements Characteristics

- A complete Requirement Specifications should be clear, correct, consistent, coherent, comprehensible, modifiable, verifiable, prioritized, unambiguous, traceable, and credible from a reliable source.

Quality Example (Lethbridge) - Restaurant Advisor System

- Identified quality deficiencies in the requirements include duplication, inadequately defined criteria, ambiguity, unclear handling of non-participating restaurants, unspecified options, and incompleteness regarding user identification for reservations.

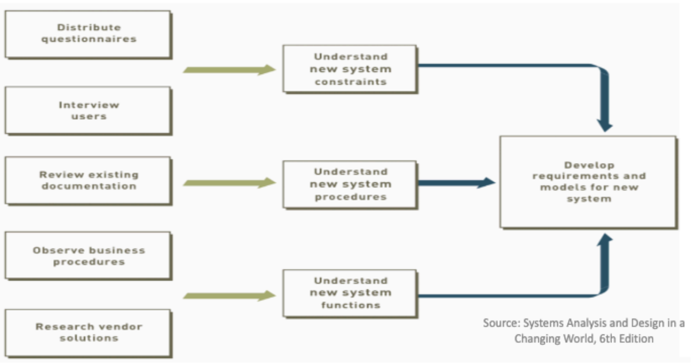
Amazon System Requirements

- Functional Requirements: Search, Browse, Shop, and Comment functionalities.
- Non-functional Requirements: Operational, Performance, Security, and Cultural aspects tailored to global users.

Modeling

Gathering and Requirements Development

Requirements and Modeling go together

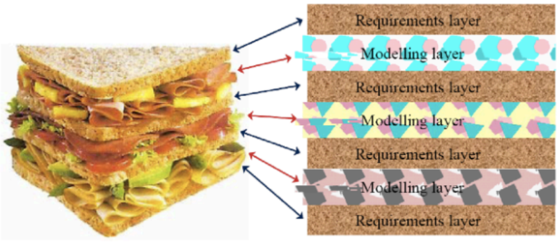

Distribute questionnaires
Interview users
Review existing documentation
Observe business procedures
Research vendor solutions
Understand new system constraints
Understand new system procedures
Understand new system functions
Develop requirements and models for new system

Source: Systems Analysis and Design in a Changing World, 6th Edition

Why Combine RE with Modeling?

- For Analysis: Models aid in understanding the problem domain more effectively.
- For Documentation: Models can describe requirements, reducing ambiguity and improving clarity.
- Reducing Ambiguity: Textual requirements can be ambiguous; models provide a visual and structured way to represent them.
- Visual Understanding: Graphical modeling provides a visual means to understand complex requirements, bridging cultural or language gaps.
- Unification of Representations: Models unify different representations of requirements, making them more comprehensible.

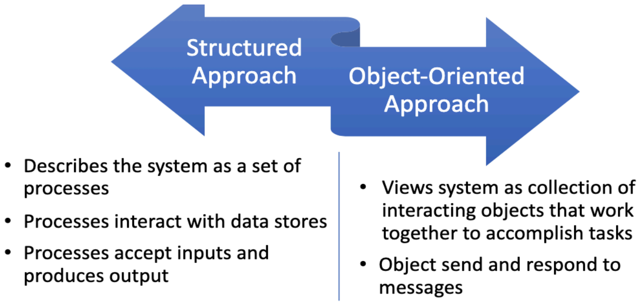
- The systems engineering sandwich!



Models and Modeling

- Model: A representation of some aspect of the system under development.
- Types of Models:
  - Textual Models: Written descriptions of requirements.
  - Graphical Models: Diagrams or schematics representing requirements visually.
  - Mathematical Models: Formulas, statistics, or algorithms used for analyzing and expressing requirements.

Two Approaches to System Development



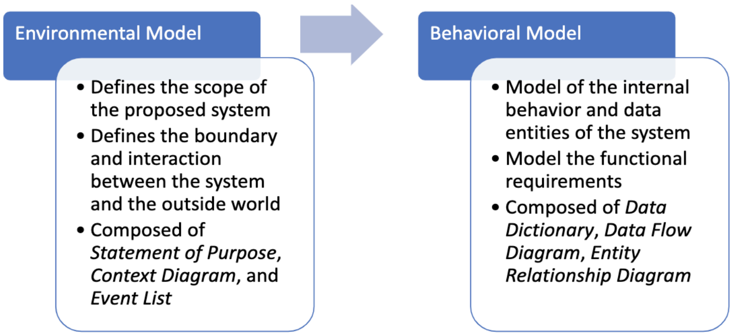
Structured Approach

- Describes the system as a set of processes
- Processes interact with data stores
- Processes accept inputs and produces output

Object-Oriented Approach

- Views system as collection of interacting objects that work together to accomplish tasks
- Object send and respond to messages

Structured System Analysis



Environmental Model

- Defines the scope of the proposed system
- Defines the boundary and interaction between the system and the outside world
- Composed of *Statement of Purpose*, *Context Diagram*, and *Event List*

Behavioral Model

- Model of the internal behavior and data entities of the system
- Model the functional requirements
- Composed of *Data Dictionary*, *Data Flow Diagram*, *Entity Relationship Diagram*

Object-Oriented System Analysis using UML

- UML (Unified Modeling Language) is used for Object-Oriented System Analysis.
- Key UML diagrams include Use-cases, Class Diagrams, Sequence Diagrams, Activity Diagrams, State Diagrams, Package Diagrams, and more.

Summary

- Definition and Importance of Requirements
  - Requirements are essential specifications for a system.
  - They serve as the foundation for system design and development.
- Requirement Engineering Activities
  - The process of gathering, analyzing, documenting, and managing requirements.

- Types of Requirements
    - Categories include Business Requirements, User Requirements, Functional Requirements, and Non-Functional Requirements.
- Requirements Quality
    - Requirements should be clear, correct, consistent, coherent, comprehensible, and verifiable.
- Business Examples
    - Real-world examples of requirements and their significance in various business scenarios.
- Modeling Techniques
    - The use of UML diagrams and modeling techniques to represent and analyze requirements in a structured and visual manner.