Requirement Engineering

Requirement Engineering

- The process of establishing the services the system should provide and the constraints under which it must operate' - Roger S. Pressman S*oftware Engineering – A practitioner's Approach European Adaptation, fifth edition*

- The appropriate mechanism for understanding what the customer wants, analyzing need, assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification, and managing the requirements as they are transformed into an operational system. - Thayer, R.H. and M. Dorfman, *Software requirements engineering*.

Requirements Engineering

- The process of establishing the services that a customer requires from a system and the constraints under which it operates and is developed.
- Begins during the communication activity and continues into the modeling activity
- Builds a bridge from the system requirements into software design and construction
- Allows the requirements engineer to examine
  - the context of the software work to be performed
  - the specific needs that design and construction must address
  - the priorities that guide the order in which work is to be completed
  - the information, function, and behavior that will have a profound impact on the resultant design

Requirements Engineering Tasks

- Seven distinct tasks
  - Inception
  - Elicitation
  - Elaboration
  - Negotiation
  - Specification
  - Validation
  - Requirements Management
- Some of these tasks may occur in parallel and all are adapted to the needs of the project
- All strive to define what the customer wants
- All serve to establish a solid foundation for the design and construction of the software

Inception Task

- During inception, the requirements engineer asks a set of questions to establish…
  - A basic understanding of the problem
  - The people who want a solution
  - The effectiveness of preliminary communication and collaboration between the customer and the developer
- Through these questions, the requirements engineer needs to…
  - Identify the stakeholders
  - Recognize multiple viewpoints
  - Work toward collaboration
  - Break the ice and initiate the communication

Elicitation Task

- Elicitation may be accomplished through two activities
  - Collaborative requirements gathering
  - Quality function deployment
- Eliciting requirements is difficult because of
  - <u>Problems of scope</u> in identifying the boundaries of the system or specifying too much technical detail rather than overall system objectives
  - <u>Problems of understanding</u> what is wanted, what the problem domain is, and what the computing environment can handle (Information that is believed to be "obvious" is often omitted)
  - <u>Problems of volatility</u> because the requirements change over time

Collaborative Requirements Gathering

- Meetings are conducted and attended by both software engineers, customers, and other interested stakeholders
- Rules for preparation and participation are established
- An agenda is suggested that is formal enough to cover all important points but informal enough to encourage the free flow of ideas
- A "facilitator" (customer, developer, or outsider) controls the meeting
- A "definition mechanism" is used such as work sheets, flip charts, wall stickers, electronic bulletin board, chat room, or some other virtual forum
- The goal is to identify the problem, propose elements of the solution, negotiate different approaches, and specify a preliminary set of solution requirements

Quality Function Deployment

- This is a technique that translates the needs of the customer into technical requirements for software.
- It emphasizes an understanding of what is valuable to the customer and then deploys these values throughout the engineering process through functions, information, and tasks.
- It identifies three types of requirements
  - Normal requirements: These requirements are the objectives and goals stated for a product or system during meetings with the customer.
  - Expected requirements: These requirements are implicit to the product or system and may be so fundamental that the customer does not explicitly state them.
  - Exciting requirements: These requirements are for features that go beyond the customer's expectations and prove to be very satisfying when present.

Elicitation Work Products

- The work products will vary depending on the system, but should include one or more of the following items:

1. A statement of need and feasibility
2. A bounded statement of scope for the system or product
3. A list of customers, users, and other stakeholders who participated in requirements elicitation
4. A description of the system's technical environment
5. A list of requirements (organized by function) and the domain constraints that apply to each
6. A set of preliminary <u>usage scenarios</u> (in the form of use cases) that provide insight into the use of the system or product under different operating conditions
7. Any <u>prototypes</u> developed to better define requirements

Elaboration Task

- During elaboration, the software engineer takes the information obtained during inception and elicitation and begins to expand and refine it.
- Elaboration focuses on developing a refined technical model of software functions, features, and constraints.
- It is an analysis modeling task
  - Use cases are developed.
  - Domain classes are identified along with their attributes and relationships.
  - State machine diagrams are used to capture the life on an object.
- The end result is an analysis model that defines the functional, informational, and behavioral domains of the problem.

Negotiation Task

- During negotiation, the software engineer reconciles the conflicts between what the customer wants and what can be achieved given limited business resources.
- Requirements are ranked (i.e., prioritized) by the customers, users, and other stakeholders.
- Risks associated with each requirement are identified and analyzed.
- Rough guesses of development effort are made and used to assess the impact of each requirement on project cost and delivery time.
- Using an iterative approach, requirements are eliminated, combined and/or modified so that each party achieves some measure of satisfaction.

The Art of Negotiation

- Recognize that it is not competition
- Map out a strategy
- Listen actively
- Focus on the other party’s interests
- Don’t let it get personal
- Be creative
- Be ready to commit

Specification Task

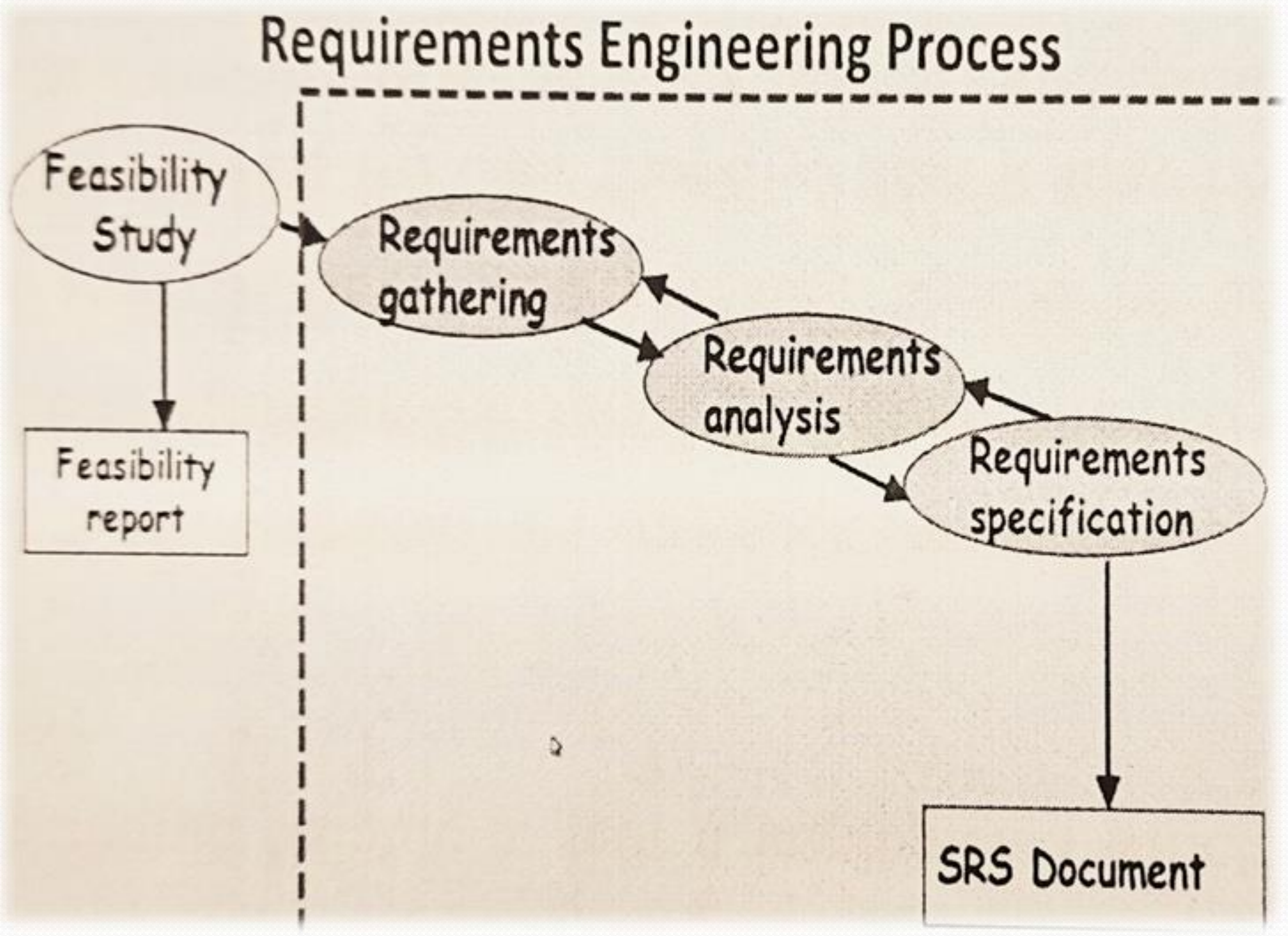
- A specification is the final work product produced by the requirements engineer.
- It is normally in the form of a software requirements specification.
- It serves as the foundation for subsequent software engineering activities.
- It describes the function and performance of a computer-based system and the constraints that will govern its development.
- It formalizes the <u>informational</u>, <u>functional</u>, and <u>behavioral</u> requirements of the proposed software in both a graphical and textual format.

Questions to ask when Validating Requirements

- Is each requirement consistent with the overall objective for the system/product?
- Have all requirements been specified at the proper level of abstraction? That is, do some requirements provide a level of technical detail that is inappropriate at this stage?
- Is the requirement really necessary or does it represent an add-on feature that may not be essential to the objective of the system?
- Is each requirement bounded and unambiguous?
- Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?
- Do any requirements conflict with other requirements?
- Is each requirement achievable in the technical environment that will house the system or product?

Requirements Management Task

- During requirements management, the project team performs a set of activities to identify, control, and track requirements and changes to the requirements at any time as the project proceeds
- Each requirement is assigned a unique identifier
- The requirements are then placed into one or more traceability tables
- These tables may be stored in a database that relate features, sources, dependencies, subsystems, and interfaces to the requirements
- A requirements traceability table is also placed at the end of the software requirements specification


Requirements Engineering Process
Feasibility Study
Feasibility report
Requirements gathering
Requirements analysis
Requirements specification
SRS Document

Software Requirement Specification (SRS)

- A software requirements specification (SRS) is a complete description of the behavior of the system to be developed.
- It may include a set of use cases that describe interactions the users will have with the software.
- A document that clearly and precisely describes, each of the essential requirements of the software and the external interfaces.
  - (functions, performance, design constraint, and quality attributes)
- Each requirement is defined in such a way that its achievement is capable of being *objectively verified* by a prescribed method; for example inspection, demonstration, analysis, or test.
- Software requirements specification establishes the basis for agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do.

An example organization of an SRS

- Introduction
  - Purpose
  - Definitions
  - System overview
  - References
- Overall description
  - Product perspective
    - System Interfaces
    - User Interfaces
    - Hardware interfaces
    - Software interfaces
    - Communication Interfaces
    - Memory Constraints
    - Operations
    - Site Adaptation Requirements
  - Product functions
  - User characteristics
  - Constraints, assumptions and dependencies
- Specific requirements
  - External interface requirements
  - Functional requirements
    - Introduction
    - Inputs
    - Processing
    - Outputs
  - Performance requirements
  - Design constraints
    - Standards Compliance
  - Logical database requirement
  - Software System attributes
    - Reliability
    - Availability
    - Security
    - Maintainability
    - Portability
  - Other requirements

Characteristics of a Good SRS (IEEE 830)

1. Unambiguous
2. Complete
3. Verifiable
4. Consistent
5. Modifiable
6. Traceable
7. Usable during the Operation and Maintenance Phase

IEEE Recommended Practice for Software Requirements Specifications

[illegible]

IEEE Software Document Definitions

- SQAP – Software Quality Assurance Plan (IEEE 730)
- SCMP – Software Configuration Management Plan (IEEE 828)
- STD – Software Test Documentation (IEEE 829)
- SRS – Software requirements specification (IEEE 830)
- SVVP – Software Validation & Verification Plan IEEE 1012
- SDD – Software Design Description IEEE 1016
- SPMP – Software Project Management Plan IEEE 1058
- SUD – Software User Documentation IEEE 1063

Requirements vs. Design

Requirements	Design
Describe what will be delivered	**Describe how it will be done**
Primary goal of analysis: UNDERSTANDING	**Primary goal of design:** OPTIMIZATION
There is more than one solution	**There is only one (final) solution**
Customer interested	**Customer not interested (Most of the time) except for external**

Types of Requirement

- User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

- System requirements
  - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Functional and Non-functional Requirements

- Functional requirements
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.
- Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
- Domain requirements
  - Constraints on the system from the domain of operation

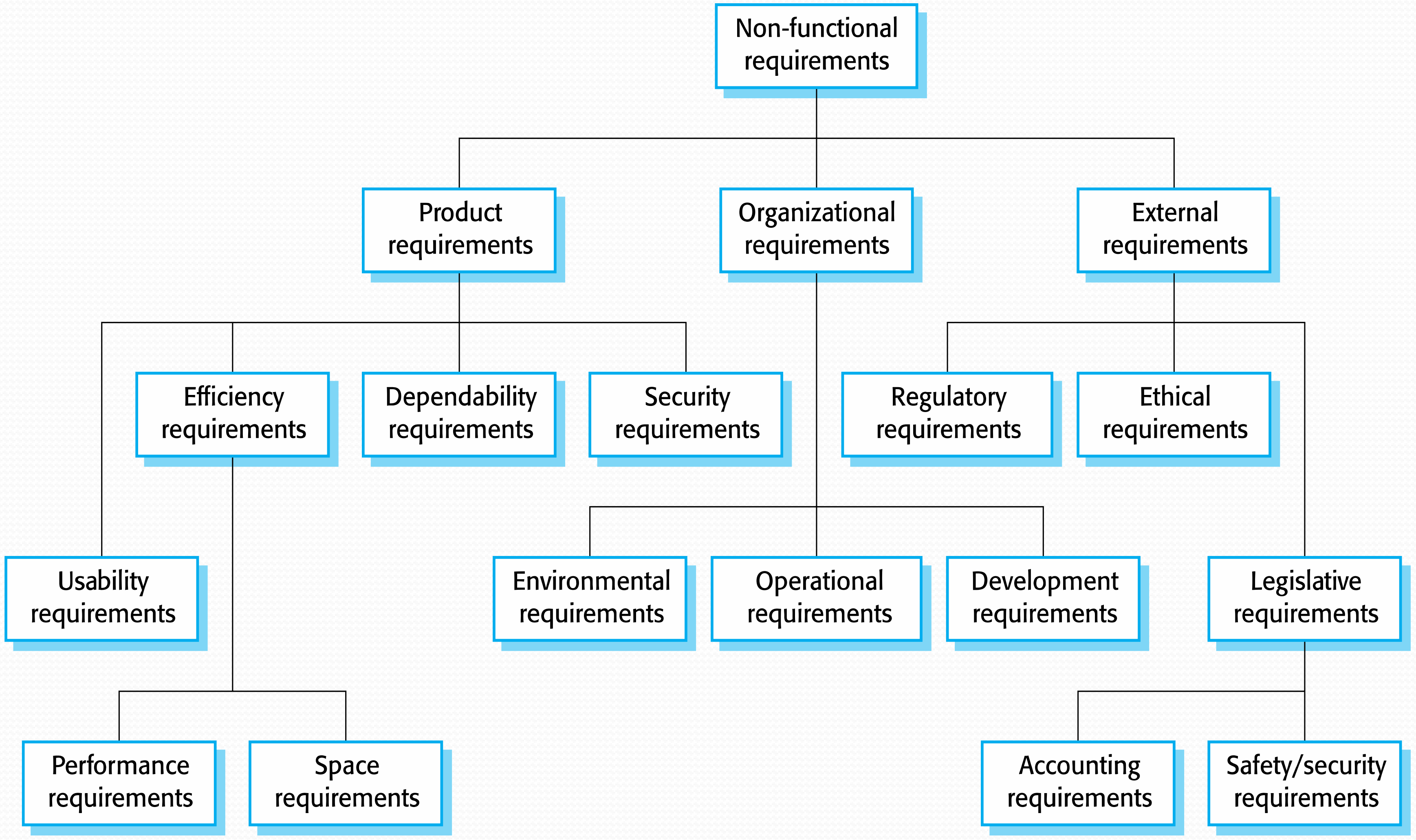
Functional Requirements

- Describe functionality or system services.
- 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.
- Functional system requirements should describe the system services in detail.

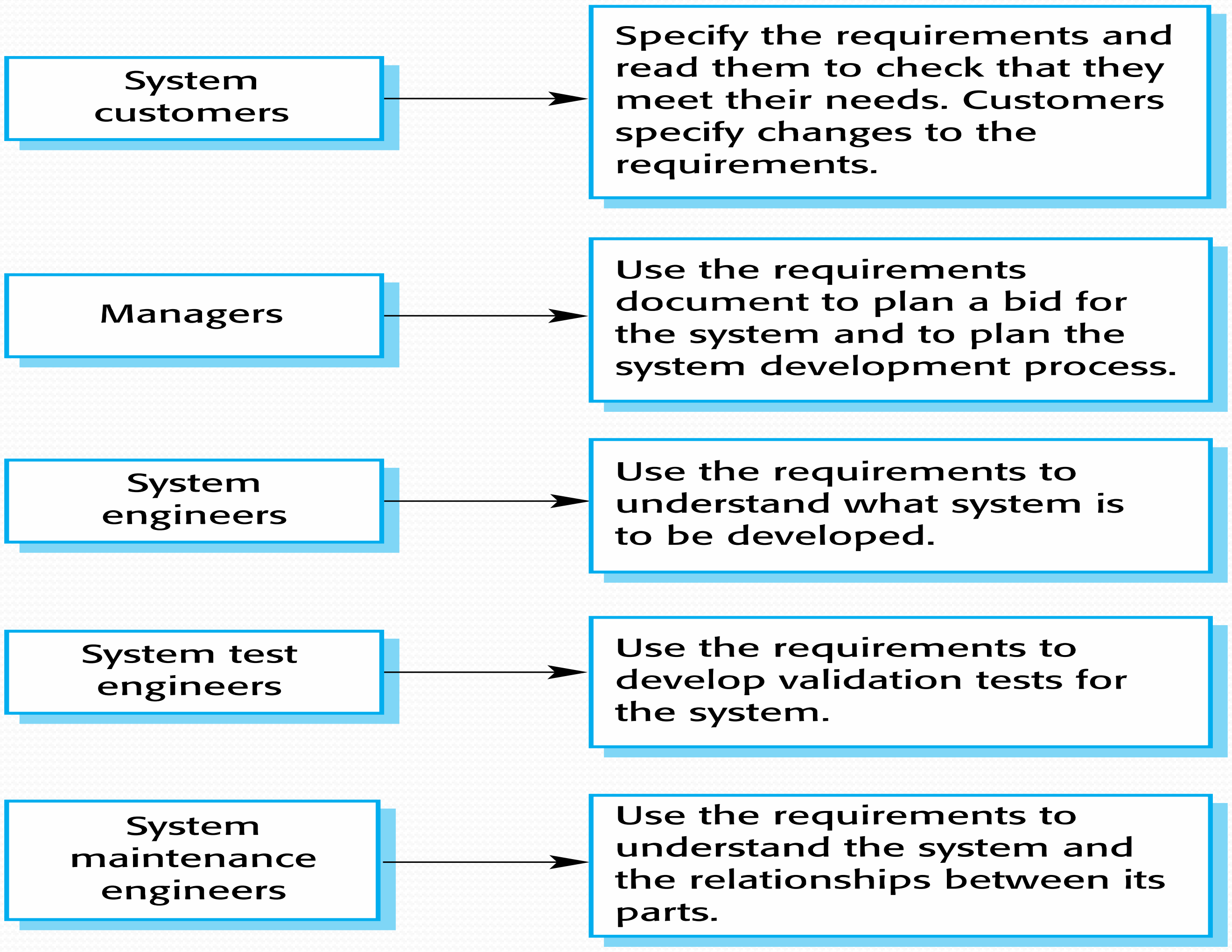
Non-functional Requirements

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Types of Non-functional Requirement



Users of a requirements document



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.

System Analyst

- A **systems analyst** is an IT professional who specializes in analyzing, designing and implementing software systems.

- System analysts assess the suitability of information systems in terms of their intended outcomes and liaise with end users, software vendors and programmers in order to achieve these outcomes.

- Requirements analysis allows the software engineer (called an *analyst* or *modeler* in this role) to:
  - elaborate on basic requirements established during earlier requirement engineering tasks
  - build models that depict user scenarios, functional activities, problem classes and their relationships, system and class behavior, and the flow of data as it is transformed.

System Analyst

A systems analyst may:

- Identify, understand and plan for organizational and human impacts of planned systems, and ensure that new technical requirements are properly integrated with existing processes and skill sets.
- Plan a system flow from the ground up.
- Interact with internal users and customers to learn and document requirements that are then used to produce business requirements documents.
- Write technical requirements from a critical phase.
- Interact with designers to understand software limitations.
- Help programmers during system development, ex: provide use cases, flowcharts or even database design.
- Perform system testing.
- Deploy the completed system.
- Document requirements or contribute to user manuals.
- Whenever a development process is conducted, the system analyst is responsible for designing components and providing that information to the developer.

Knowledge and Qualities of System Analyst

- An analyst must process various skills to effectively carry out the job. Specifically, they must be divided into two categories:

- **Interpersonal skill:** This skills deal with relationships and the interface of the analyst with people in business. They are useful in establishing trust, resolving conflict, and communicating information.

- **Technical skills:** On other hand, focus on procedures and techniques for operations analysis and systems analysis.

Knowledge and Qualities of System Analyst

The Interpersonal skills include: -

- Communication: Communication is not just reports, telephone conversations, and interviews. It is people talking, listening, feeling and reacting to one another, their experience and reactions.
- Understanding: Identifying problems and assessing their ramifications, having a grasp of company goals and objectives, and showing sensitivity to the impact of the system on people at work.
- Teaching: Educating people in use of computer system, selling the system to user, and giving support when needed.
- Selling: Selling ideas and promoting innovations in problem solving using computers.

Knowledge and Qualities of System Analyst

The technical skills include: -

- Creativity: Helping users model ideas into concrete plans and developing candidate systems to match user requirements.
- Problem solving: Reducing problems to their elemental levels for analysis, developing alternative solutions to a given problem, and delineating the pros and cons of candidate system.
- Project management: Scheduling, performing well under time constraints, coordinating team efforts, and managing costs and expenditures.
- Questioning attitude and inquiring mind: Knowing the what, when, why, where, who and how a system works.
- Knowledge of the basics of the computer and the business function.

Role of a System Analyst

- The primary role of a systems analyst is to study the problems and needs of an organization in order to determine how people, methods, and information technology can best be combined to bring about improvements in the organization. (Hoffer, George & Valachich, 1999).

- The systems analyst is a key person analyzing the business, identifying opportunities for improvement, and designing information systems to implement these ideas (Dennis, Wixom & Tegarden, 2002).

- Systems analysts are key to the systems development process. The analyst's primary focus is on *what* not *how*. *What* data does the system produce and consume, *what* functions must the system perform, *what* interfaces are defined and *what* constrains apply?" (Pressman, 1997).

Role of a System Analyst

- The systems analyst must understand both the business requirements of an organization and the workings of the various technologies - the systems analyst builds the bridges between organizational needs and technology solutions.

- “When a system developer walks away from the successful implementation of a good system, what has been achieved is the acceptance and efficient operation of a technical computer system by a human community. The system has both a technical and a social dimension - it is a socio-technical system . The project plan will have allowed for the evolution of the technical aspects of the system with the active involvement of the human community that will operate it" (Lejk and Deeks, 1998).

Feasibility Study and It's Types

- The **feasibility study** is an evaluation and analysis of the potential of a proposed project which is based on extensive investigation and research to support the process of decision making.
- It is quantifying benefits and costs
  - Payback analysis
  - Net Present Value Analysis
  - Return on Investment Analysis

Following are different components of the feasibility study:

- Operational feasibility
- Economic feasibility
- Technical feasibility
- Human factors feasibility
- Legal/Political feasibility

Feasibility Study Contents

1. Purpose & scope of *the study*
   - Objectives (of the study)
   - who commissioned it & who did it,
   - sources of information,
   - process used for the study,
   - how long did it take,...
2. Description of present situation
   - organizational setting, current system(s).
   - Related factors and constraints.
3. Problems and requirements
   - What's wrong with the present situation?
   - What changes are needed?
4. Objectives of the new system.
   - Goals and relationships between them
5. Possible alternatives
   - ...including 'do nothing'.
6. Criteria for comparison
   - definition of the criteria
7. Analysis of alternatives
   - description of each alternative
   - evaluation with respect to criteria
   - cost/benefit analysis and special implications.
8. Recommendations
   - what is recommended and implications
   - what to do next;
     - E.g. may recommend an interim solution and a permanent solution
9. Appendices
   - to include any supporting material.

User Transaction Requirement

- Each user view will involve certain transactions, stipulating how the data is to be used
- There are three broad categories:
  - Data entry: every data item needs to be created somewhere
  - Data update and deletion
  - Data queries
- Transactions should be related to the user view to ensure all functions are supported
- Do transactions needed to be atomic

User Design Requirements

- There are tools for guiding the user design process and for discussing **user design requirements** with users.
- Mostly, Use Cases are used for this.
- A **design specification** provides explicit information about the requirements for a product and how the product is to be put together.

Thank You