Unit-1

Requirement Analysis & Specification

Reference Book:
Software Engineering -A Practitioner’s Approach (Seventh Edition) - Roger S. Pressman.

Chapter 5:Understanding Requirements

Requirements analysis is hard

“The **hardest** single **part** of building a software system is **deciding what to build**. No part of the work so cripples the resulting system if done wrong.”

"Fred" Brooks Jr. is an American computer architect, software engineer, and computer scientist, best known for managing the development of IBM's System/360 family of computers

“The **seeds** of major software **disasters** are usually **sown** in the **first three months** of commencing the software project.”

Capers Jones is an American specialist in software engineering methodologies

What is Requirement Engineering?

Tasks and **techniques** that lead to an **understanding** of **requirements** is called **requirement engineering**.

User requirements to mean the high-level abstract requirements

- Often referred to as user needs.
- Such as what activities that users must be able to perform using the system.
- User requirements are generally documented in a User Requirements Document (URD) using narrative text.
- User requirements are generally signed off by the user and used as the primary input for creating system requirements.
- An important and difficult step of designing a software product is determining what the user actually wants it to do.
- This is because the user is often not able to communicate the entirety of their needs and wants, and the information they provide may also be incomplete, inaccurate and self-conflicting.

System requirements **to mean the detailed description of what the system should do.**

- **System requirements are the building blocks developers use to build the system.**
- **These are the traditional "shall" statements that describe what the system "shall do."**
- **System requirements are classified as either functional or supplemental requirements/non-functional requirements.**
- **A functional requirement specifies something that a user needs to perform their work. For example, a system may be required to enter and print cost estimates; this is a functional requirement.**
- **Supplemental or non-functional requirements are sometimes called quality of service requirements.**

User requirement

- UR1: A user of the work-based learning system shall be able to locate the person who is responsible for a document.
- UR2: A user of the work-based social networking system shall be able to control the amount of information about the user that the system presents to the other users like co-team member, group member and external for viewing.

System requirement

- SR1: The work-based learning system shall be able to retrieve the name, desk address, telephone number and email id fields of the record of a person responsible for the selected document.
- SR2: The work-based social networking system shall allow the user to present and hide fields from among the 48 data fields of the record of a person. The system presents these visible fields to the other users, based on three levels entitlements, namely co-team member, group member and external. The fields will be of view type only and not editable.

User requirements definition

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc.) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

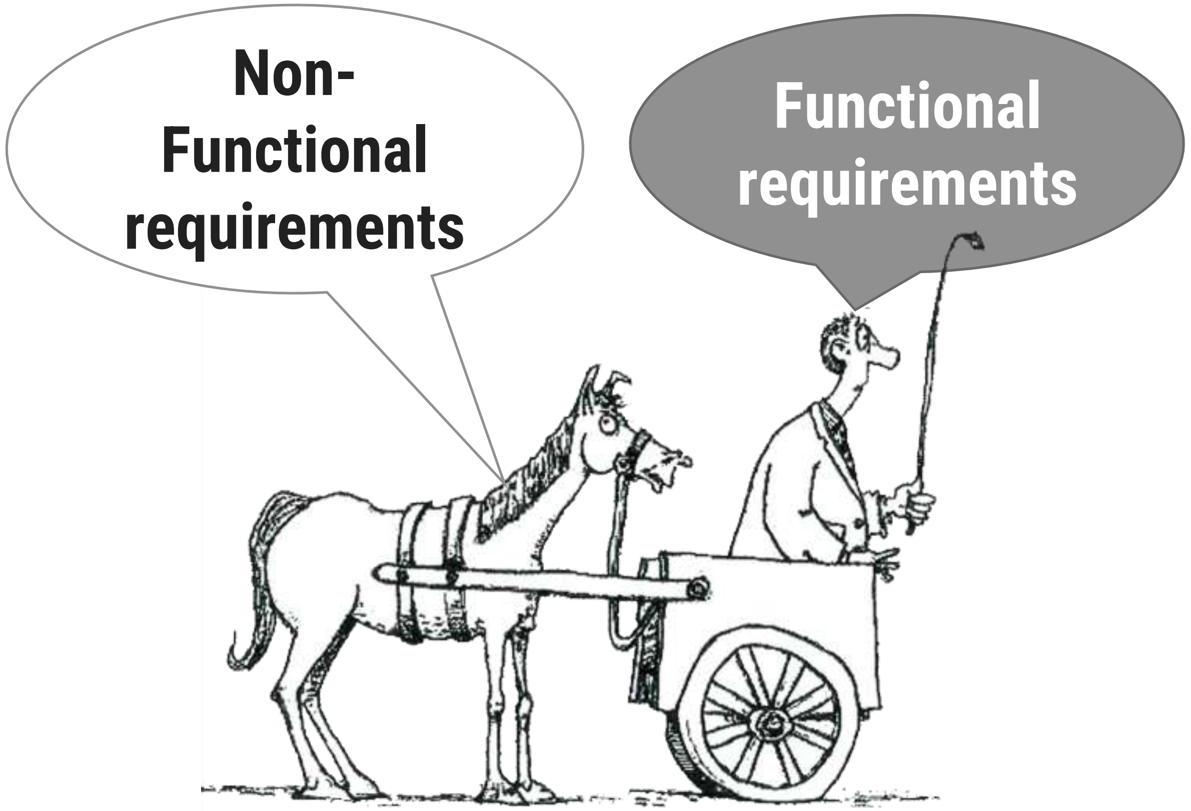
Requirement Engineering

It **provides** the appropriate **mechanism** for **understanding**

- What customer wants
- Analyzing needs
- Assessing feasibility
- Negotiating a reasonable solution
- Specifying solution unambiguously
- Validating the specification
- Managing requirements

Requirements fall into two types

Functional requirements	Non-Functional requirements



Don't put what you want to do - before how you need to do it

Functional requirements

Any requirement which specifies **what the system Should do**.

A functional requirement will describe a particular behavior of function of the system when certain conditions are met, for example: “Send email when a new customer signs up” or “Open a new account”.

Typical functional requirements

- Business Rules
- Administrative functions
- Historical Data
- Transaction corrections, adjustments and cancellations
- Legal or Regulatory Requirements
- External Interfaces
- Reporting Requirements
- Authentication
- Authorization levels
- Audit Tracking

Non-functional requirements

Any requirement which specifies **how the system performs a certain function**.

A non-functional requirement will describe how a system should behave and what limits there are on its functionality.

Typical non-functional requirements

Response time	Availability	Regulatory
Throughput	Reliability	Manageability
Utilization	Recoverability	Environmental
Static volumetric	Maintainability	Data Integrity
Scalability	Serviceability	Usability
Capacity	Security	Interoperability

Library Management System

Function Requirements

- **Add Article:** New entries must be entered in database
- **Update Article:** Any changes in articles should be updated in case of update
- **Delete Article:** Wrong entry must be removed from system
- **Inquiry Members:** Inquiry all current enrolled members to view their details
- **Inquiry Issuance:** Inquiry all database articles
- **Check out Article:** To issue any article must be checked out
- **Check In article:** After receiving any article system will reenter article by Checking
- **Inquiry waiting for approvals:** Librarian will generate all newly application which is in waiting list
- **Reserve Article:** This use case is used to reserve any book with the name of librarian, it can be pledged

Non-function Requirements

- **Safety Requirements:** The database may get crashed at any certain time due to virus or operating system failure. So, it is required to take the database backup.
- **Security Requirements:** We are going to develop a secured database for the university. There are different categories of users namely teaching staff, administrator, library staff ,students etc., Depending upon the category of user the access rights are decided.
- **Software Constraints:** The development of the system will be constrained by the availability of required software such as database and development tools. The availability of these tools will be governed by

Feasibility studies

A feasibility study is a short, focused study that should take place early in the RE process. It should answer three key questions:

(1)Does the system contribute to the overall objectives of the organization?

(1)Can the system be implemented within schedule and budget using current technology?

(1) Can the system be integrated with other systems that are used?

If the answer to any of these questions is no, you should probably not go ahead with the project.

Requirements Engineering Tasks

1 Inception

Roughly define scope

A basic understanding of a problem, people who want a solution, the nature of solution desired

2 Elicitation (Requirement Gathering)

Define requirements

The practice of collecting the requirements of a system from users, customers and other stakeholders

3 Elaboration

Further define requirements

Expand and refine requirements obtained from inception & elicitation

Creation of User scenarios, extract analysis class and business domain entities

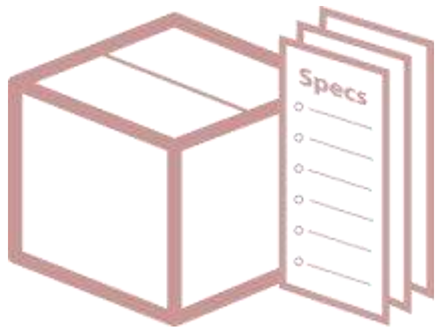
Requirements Engineering Tasks Cont.

4 Negotiation

Reconcile conflicts

Agree on a deliverable system that is realistic for developers and customers

5 Specification



Create analysis model

It may be written document, set of graphical models, formal mathematical model, collection of user scenarios, prototype or collection of these

SRS (Software Requirement Specification) is a document that is created when a detailed description of all aspects of software to build must be specified before starting of project

Requirements Engineering Tasks Cont.

6 Validation

Ensure quality of requirements

Review the requirements specification for errors, ambiguities, omissions (absence) and conflicts

7 Requirements Management

It is a set of activities to identify, control & trace requirements & changes to requirements (Umbrella Activities) at any time as the project proceeds.

Elicitation is the Hardest Part!

- **Problems of scope**
  - System boundaries are ill-defined
  - Customers will provide irrelevant information
- **Problems of understanding**
  - Customers never know exactly what they want
  - Customers don't understand capabilities and limitations
  - Customers have trouble fully communicating needs
- **Problems of volatility**
  - Requirements always change

Project Inception

During the **initial project meetings**, the following tasks should be accomplished

- **Identify the project stakeholders**
  - These are the folks we should be talking to
- **Recognize multiple viewpoints**
  - Stakeholders may have different (and conflicting) requirements
- **Work toward collaboration**
  - It's all about reconciling conflict
- **Ask the first questions**
  - Who? What are the benefits? Another source?
  - What is the problem? What defines success? Other constraints?
  - Am I doing my job right?

Collaborative Elicitation

One-on-one Q&A sessions rarely succeed in practice; collaborative strategies are more practical

Elicitation work products

Collaborative elicitation should result in several **work products**

- A bounded statement of scope
- A list of stakeholders
- A description of the technical environment
- A list of requirements and constraints
- Any prototypes developed
- A set of use cases
  - Characterize how users will interact with the system
  - Use cases tie functional requirements together

Quality Function Deployment (QFD)

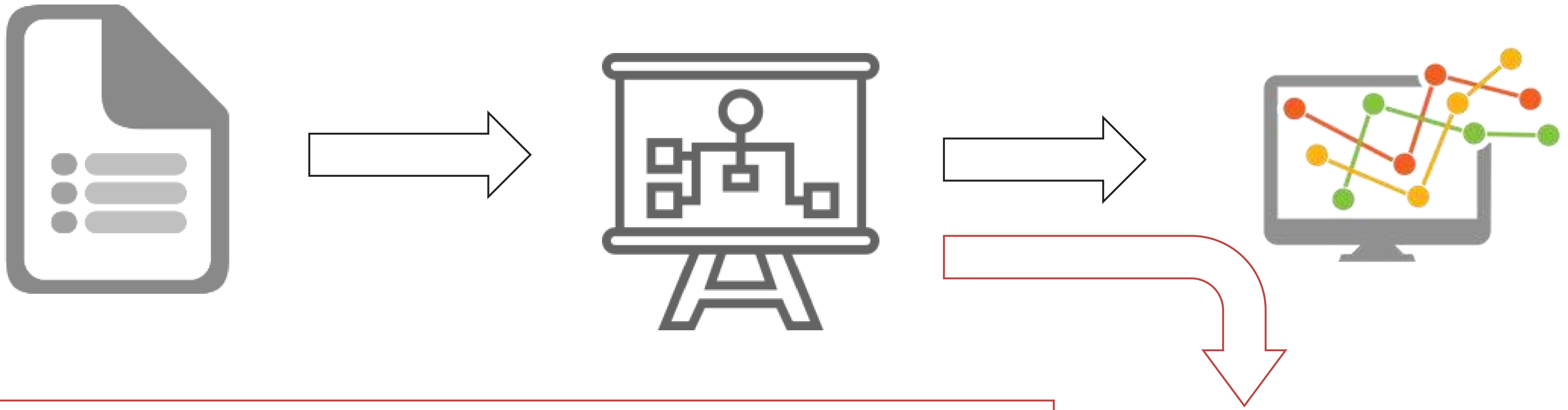
- This is a technique that **translates** the **needs** of the **customer** into **technical requirements** for software
- It **emphasizes** an understanding of **what is valuable to the customer** and then deploys these values throughout the engineering process through functions, information, and tasks
- 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

The requirement analysis model

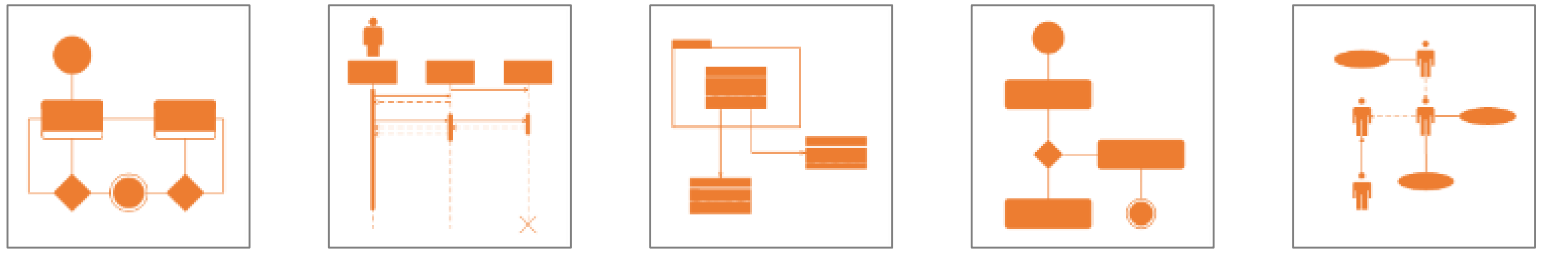


Purpose	Describe what the customer wants to built
	Establish the foundation for the software design
	Provide a set of validation requirements

System Information

System Function

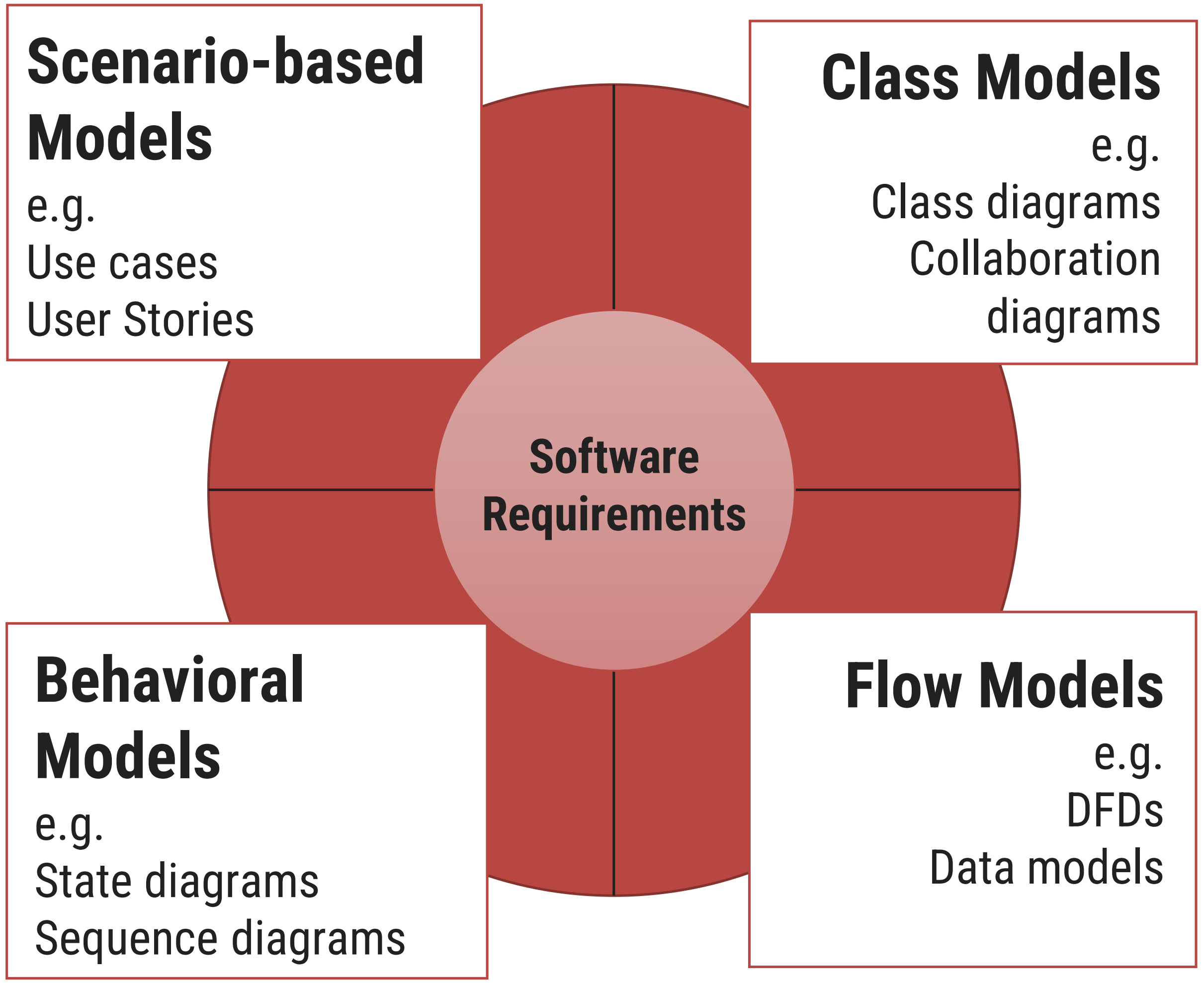
System Behaviors



Analysis rule of Thumb

- Make **sure all points** of view are **covered**
- Every **element** should **add value**
- **Keep** it **simple**
- **Maintain** a high level of **abstraction**
- **Focus** on the **problem domain**
- **Minimize** system **coupling**
- **Model** should **provide value to all stakeholders**

Elements of the Requirements Model



Elements of the Requirements Model Cont.

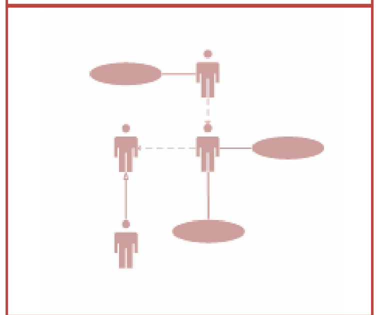
Scenario-based elements

Describe the system from the user's point of view using scenarios that are depicted (stated) in use cases and activity diagrams

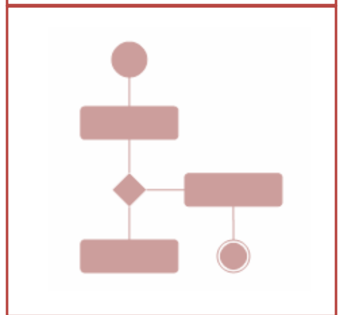
Class-based elements

Identify the domain classes for the objects manipulated by the actors, the attributes of these classes, and how they interact with one another; which utilize class diagrams to do this.

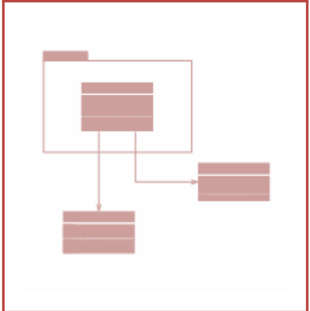
Use Case Diagram



Activity Diagram



Class Diagram



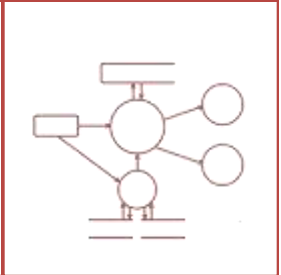
Behavioral elements

Use state diagrams to represent the state of the system, the events that cause the system to change state, and the actions that are taken as a result of a particular event. This can also be applied to each class in the system.

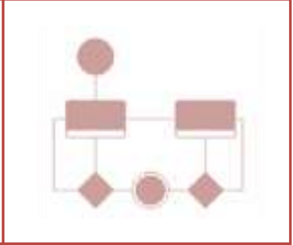
Flow-oriented elements

Use data flow diagrams to show the input data that comes into a system, what functions are applied to that data to do transformations, and what resulting output data are produced.

Data Flow Diagram



State Diagram



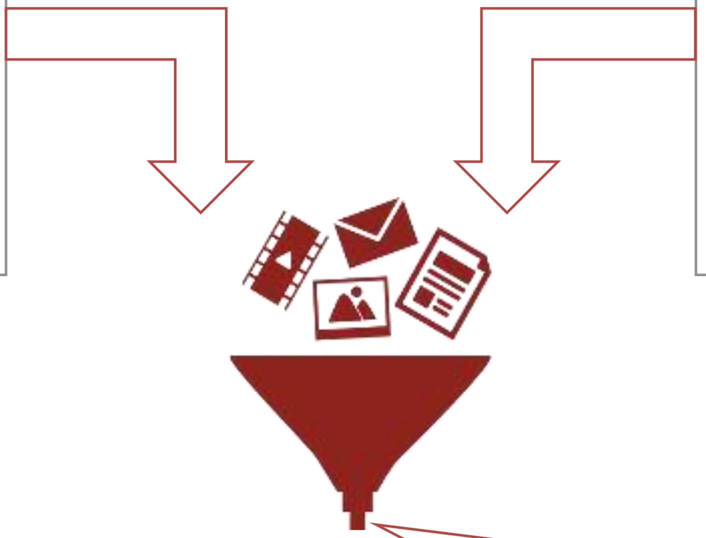
Analysis Modeling Approaches

Structured Analysis

- Models data elements
  - Attributes
  - Relationships
- Models processes that transform data

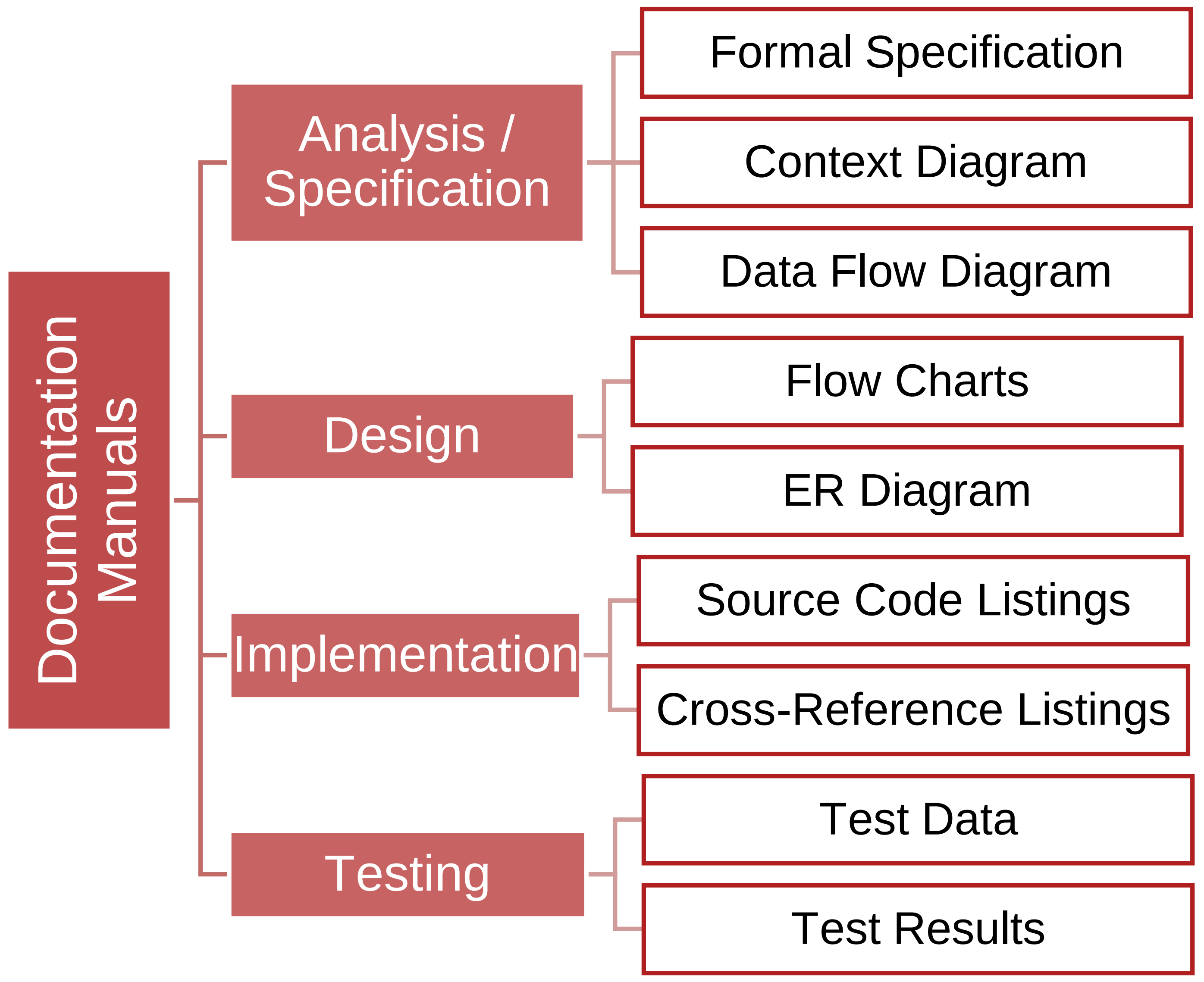
Object Oriented Analysis

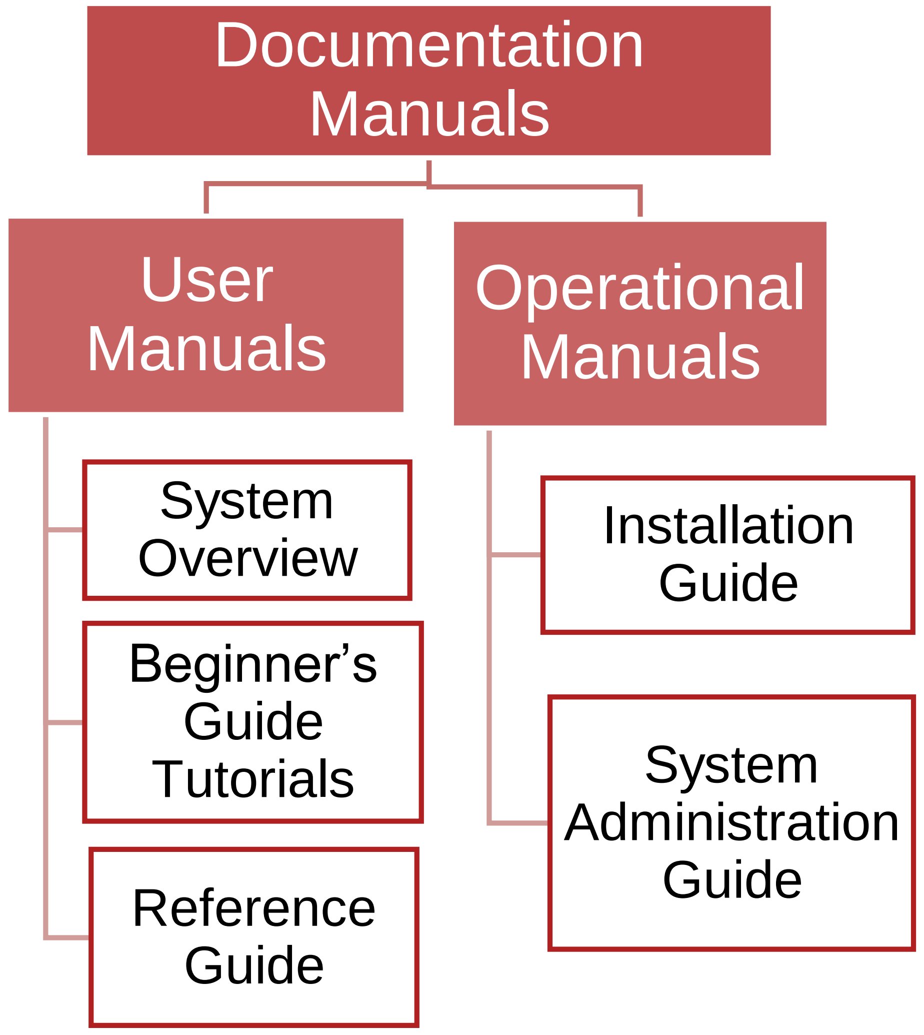
- Model analysis classes
  - Data
  - Processes
- Models class collaborations



Techniques from both approaches are typically used in practice.

List of documentation & manuals





SRS (Software Requirements Specification)

Software Requirement Specification (SRS) is a document that completely describes what the proposed software should do without describing how software will do it.

SRS is also helping the clients to understand their own needs.

SRS Contains

- A complete information description
- A detailed functional description
- A representation of system behaviour
- An indication of performance requirements and design constraints
- Appropriate validation criteria
- Other information suitable to requirements

Characteristics of a Good SRS

- SRS should be **accurate, complete, efficient, and of high quality,** so that it does not affect the entire project plan.
- An SRS is **said to be of high quality when** the developer and user **easily understand** the prepared document.
- Characteristics of a Good SRS:

Correct

SRS is correct when all user requirements are stated in the requirements document.

Note that there is no specified tool or procedure to assure the correctness of SRS.

Unambiguous

SRS is unambiguous when every stated requirement has only one interpretation.

Complete

SRS is complete when the requirements clearly define what the software is required to do.

Characteristics of a Good SRS Cont.

Ranked for Importance/Stability

All requirements are not equally important, hence each requirement is identified to make differences among other requirements.

Stability implies the probability of changes in the requirement in future.

Modifiable

The requirements of the user can change, hence requirements document should be created in such a manner that those changes can be modified easily.

Traceable

SRS is traceable when the source of each requirement is clear and facilitates the reference of each requirement in future.

Verifiable

SRS is verifiable when the specified requirements can be verified with a cost-effective process to check whether the final software meets those requirements.

Consistent

SRS is consistent when the subsets of individual requirements defined do not conflict with each other.

Problems Without SRS

- Without developing the SRS document, the system would not be properly implemented according to customer needs.
- Software developers would not know whether what they are developing is what exactly is required by the customer.
- Without SRS, it will be very difficult for the maintenance engineers to understand the functionality of the system.
- It will be very difficult for user document writers to write the users' manuals properly without understanding the SRS.

Standard Template for writing SRS

1. Introduction

1.1 Purpose

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Project Scope

1.5 References

2. Overall Description

2.1 Product Perspective

2.2 Product Features

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 User Documentation

2.7 Assumptions and Dependencies

3. System Features

3.1 System Feature 1

3.2 System Feature 2 (and so on)

4. External Interface Requirements

4.1 User Interfaces

4.2 Hardware Interfaces

4.3 Software Interfaces

4.4 Communications Interfaces

5. Other Non-functional Requirements

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

6. Other Requirements

Appendix A: Glossary | Appendix B: Analysis Models | Appendix C: Issues List

Thank You