UNIT 3 : Requirement analysis

3.1 Introduction

3.2 Requirement gathering techniques & Fact Finding, Recording Outcome

3.3 Effort distribution

3.4 Importance of Requirement Specifications

3.5 SRS Characteristics

3.6 Software Requirement Specification Document

3.1 Introduction

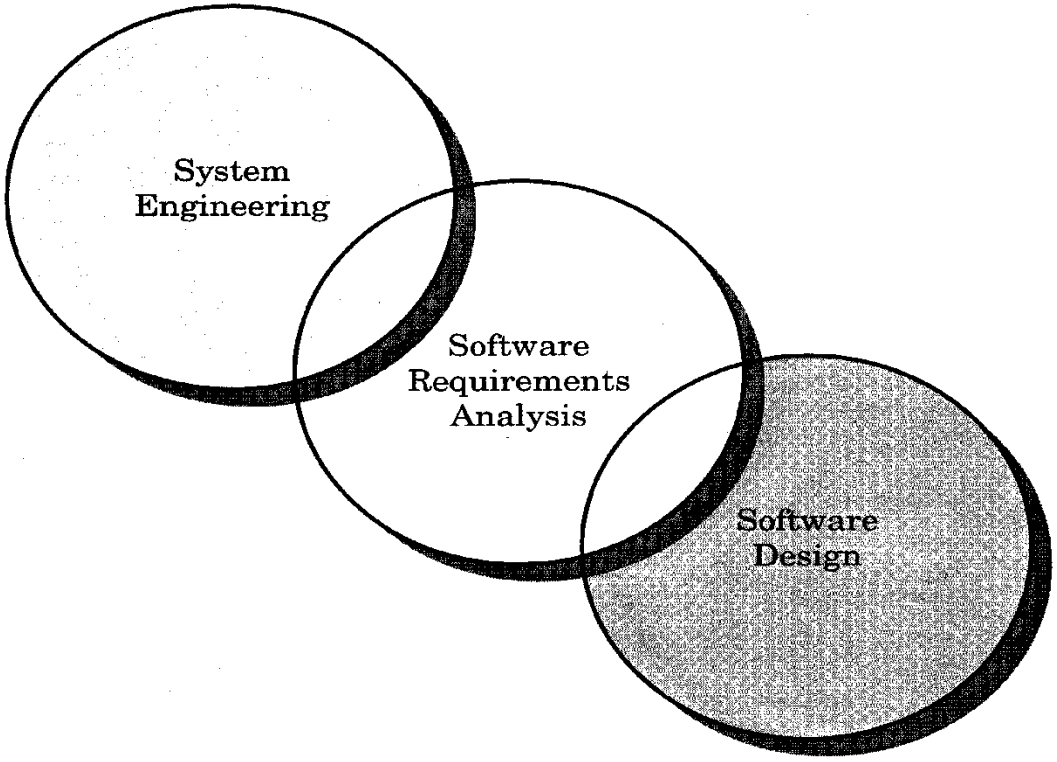

FIGURE 11.1.
Analysis and a bridge between system engineering and software design

Requirement Analysis

- Results in specifications of Software operational characteristics (function, data and behavior)
- Indicate Software interface with other system elements(language&code,technology)
- Establish constraints that Software must meet (development resources&senior's decision)

Requirement Analysis

- Five Activities
  - Problem recognition
  - Evaluation and synthesis (Separation)
  - Modeling
  - Specification (Plan)
  - Review

Problem recognition

- Discover and understand the system requirements
- Refine the requirements

Evaluation and Synthesis

- What are the alternative solution?
- Focus on what solutions should be selected or used instead of how to implement a solution

Modeling

- Required data
- Information and control flow
- Operation, behavior

Specifications

- Software functions and performance
- Interfaces between system elements
- System constraints

Review

- Uncertain stated requirements
- Conflicting requirements are not detected during requirement analysis
- Errors in the requirements elicitation and analysis

Current Application Analysis

1. Preliminary Study
2. Feasibility Study
3. Investigation Study
4. Cost / Benefit Analysis

Preliminary Study

- Preliminary system study is the **first stage** of system development life cycle.
- This is a brief investigation of the system under consideration and gives a clear picture of **what actually the physical system is?**
  - Problem Definition
  - Objectives of the Study
  - Terms of reference for Study
  - Constraints
  - Expected benefits of the new system

Feasibility Study

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility
- Social Feasibility
- Management Feasibility
- Legal Feasibility
- Time Feasibility

Investigation Study

- Identify potential problems and opportunities and consider them in light of the goals of the company.
- **Team:**
  - Stakeholders
  - Users and Managers
- **Goals for Investigation**
  - Feasibility Analysis
  - Establish Goals
  - Select methodology
  - Prepare report

Cost / Benefit Analysis

- In performing Cost benefit analysis (CBA) it is important to identify cost and benefit factors. Cost and benefits can be categorized into the following categories.
  - Hardware
  - Personnel
  - Facility
  - Operating
  - Supply costs

3.2 Requirement Gathering Techniques

1. Interviews
2. Questionnaires
3. Record review
4. Observations
5. Joint Application Development(JAD)
6. Facilitated Application Specification Technique(FAST)
7. Rapid Application Development(RAD)
8. Sampling of existing document, forms and databases

1. Interviews

- Most widely used technique.
- Requires the most skills and sensitivity.
- Structured meeting between analyst and staff.
- Discussion of one or more areas of work of the staff.
- Can be using fixed set of questions or informal questions.
- Close and Open probes(Survey).

Advantages and Disadvantages

+ Produces high Quality information.

+Provides greater depth of understanding of a person's work and expectation.

(-) Time consuming process

(-) Interviewee can provide conflicting information which becomes difficult to resolve later.

2. Questionnaires

- Effective fact finding instrument.
- Has series of questions to be answered.
- Multiple choice or Yes/No questions.
- Covers question ranging from Coding to Feedback.

Advantages and Disadvantages

+ *Economical way of gathering data.*
+ *If well defined, results are effectively analyzed.*
- *Creating a good questionnaire is difficult.*
- *No follow-up or probing can be done with answers.*

3. Record Review

- To have good understanding of the organization's business objectives.
- Kind of documents to be looked for:
  - Company Reports
  - Organization Charts
  - Policy Manuals
  - Job Descriptions
  - Reports
  - Documentation of existing system

Advantages and Disadvantages

+ *Helps understanding the organization before meeting its work force.*
+ *Helps understanding the requirements of the system in the light of business objectives.*
+ *Documentation can provide information requirements of the current system.*

(-) Difference between written policy and its application.

4. Observations

- Watching people in their normal work flow carrying out their operations.
- Analysts watch and note the type of the information the work is using and processing in the existing system.
- Can be open ended or close ended.

Advantages and Disadvantages

+ *Provides first hand experience.*

+ *Real time data collection.*

(-) Most people don't like being observed and may behave differently.

(-) Requires recursive training to have an analytical observation.

5. Joint Application Development (JAD)

- Joint application development is a system development methodology and approach that is dependent on collaboration and interaction between stakeholders through series of workshops and discussion sessions. This lesson focuses on the advantages and disadvantages of joint application development paradigm.

Advantages and Disadvantages

+ *Cost reduction*

+ *Better understanding*

+ *Improved quality*

(-) Depending on the size of the project, JAD may require a significant time commitment

(-) Different opinions within the team make it difficult to align goals and maintain focus

6. Facilitated Application Specification Technique(FAST)

- Meeting (often at neutral site)
- Establish meeting rules
- Agenda to cover important points
- A facilitator -- best if not customer or supplier
- Definition mechanism
- Understand goal -- to identify problem, specify a ***preliminary*** set of requirements

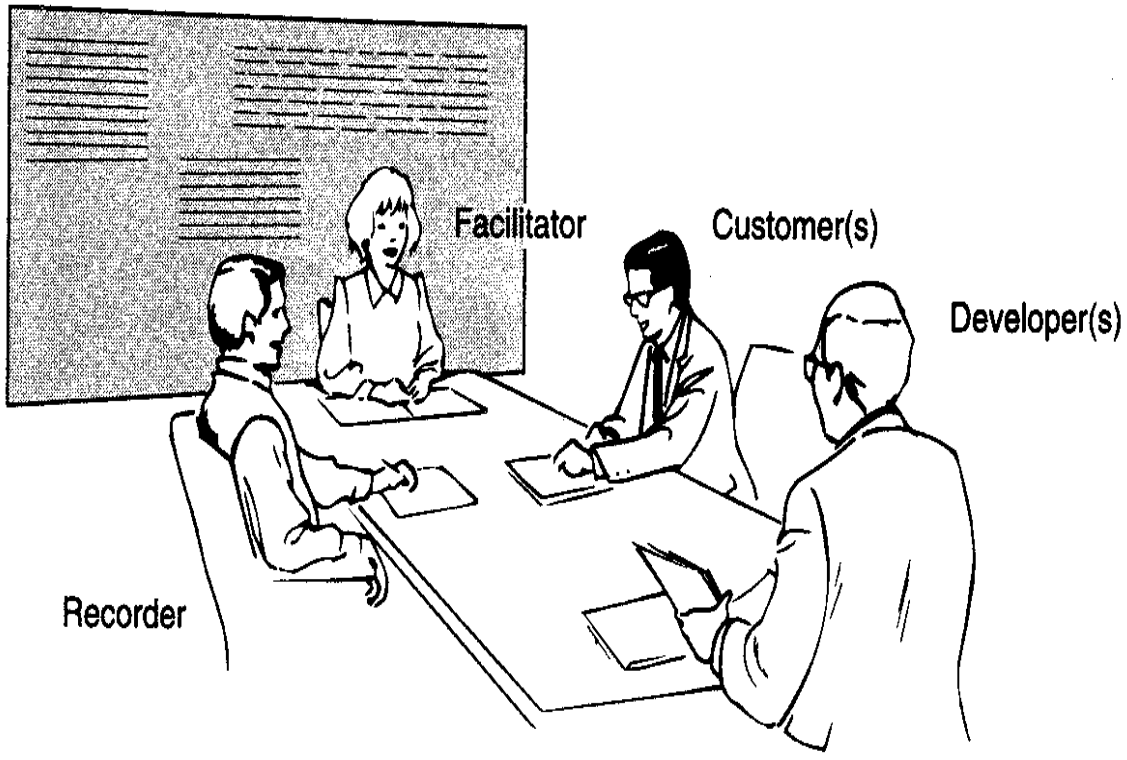

FIGURE 11.2.
The FAST meeting

7. Rapid Application Development (RAD)

- Rapid Application Development refers to the developing of programming applications and differs from programming itself in that it has a higher level of responsibility, including for requirement capturing and testing.

Advantages and Disadvantages

+ increased speed of development and increased quality

+ interoperability, extensibility, and portability

(-) reduced Scalability

8. Sampling of existing documentation, forms and databases

- Done in two (2) ways
- First, Collect copies of completed documents of the interviews and observations and carefully articulate.
- Second, Statistical analysis of the documents to find out patterns of data.

Advantages and Disadvantages

+ Used for quantitative data.
+ Used to find error rates in paper documents.
(-) Existing documents don't show what changes will be in future.

What is Fact Finding?

- **Fact finding** is process of collection of data and information based on **requirement gathering techniques .**
- Identification of what new system should be able to do.
- Specification of what the system should do as per user's requirements.
- Includes what the existing system does and what is the new one expected to do.
- Done by system or business analyst.

Why Fact Finding is important?

- Rapidly changing environment of organizations.
- Classifies data in 3 categories:
  - Functional Requirements
  - Non-Functional Requirements
  - Usability Requirements.

Functional Requirements

- Describes what a system is expected to do (***Functionality***).
- Describes the **processes** that system will carry out.
- Details of the **inputs** into the system from paper forms and documents and other systems.
- Details of the **output** expected from the system on screen display and as printouts on the paper.

Non Functional Requirements

- Describes the **quality** parameters of the processing of functional requirements.
- **Performance** criteria: Desired **Response time** for **updating** or **retrieving** data in/from the system.
- **Ability** of the system to handle with **multi** using at **multi levels**.
- **Security** parameters: **resistance** and **detection** of **attacks**.

Usability Requirements

- How s/w is Easy to use.
- Describe the usability **factors** (elements) and **facts** (reality) between the **system** and **users**.
- **Ensures** good match between the system and users **performing** tasks on the system.
- Efficient **Human-Computer** interactions.

Techniques:

1. Interviews
2. Questionnaires
3. Record review
4. Observations
5. Joint Application Development(JAD)
6. Facilitated Application Specification Technique(FAST)
7. Rapid Application Development(RAD)
8. Sampling of existing document, forms and databases

Effort Distribution

SRS (Software Requirement Specifications)

A **software requirements specification** (**SRS**) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements **engineering** phase.

SRS (Software Requirement Specifications)

WHY SRS????

- An **SRS** minimizes the time and effort required by developers to achieve desired goals and also minimizes the **development** cost. A good **SRS** defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations.

3.4 Importance of SRS

- Establish the basis for **agreement between the customers and the suppliers** on what the software product is to do.
- Reduce the development **effort**.
- Provide a basis for **estimating costs** and **schedules**.
- Provide a baseline for **validation** and **verification**.
- **Facilitate transfer**.(easier to **transfer** the software product to new users or new machines.)
- Serve as a **basis for enhancement(improvement).**

3.5 Characteristics of SRS

a) Correct

b) Unambiguous (Error Free)

c) Complete

d) Consistent (compatible)

e) Ranked for importance

f) Verifiable

g) Modifiable

h) Traceable

Correct

- This is like motherhood and apple pie. Of course you want the specification to be correct. No one writes a specification that they know is incorrect. We like to say - "**Correct and Ever Correcting**." The discipline is keeping the specification up to date when you find things that are not correct.

Unambiguous

- An SRS is unambiguous if, and only if, every requirement stated therein has only **one interpretation(meaning must be same)**. Again, easier said than done. Spending time on this area prior to releasing the SRS can be a waste of time. But as you find ambiguities - fix them.

Complete

- A simple judge of this is that it should be all that is needed by the software designers to create the software.
- Make sure about all required soft-wares which is need to for a developer to create software

Consistent

- The SRS should be consistent within itself and consistent to its reference documents. If you call an input "**Start** and **Stop**" in one place, don't call it "**Start/Stop**" in another.

Ranked for Importance

- Very often a new system has requirements that are really marketing **wish lists**. Some may not be achievable. It is useful provide this information in the SRS.

Verifiable

- Don't put in requirements like - "**It should provide the user a fast response**." Another of my favorites is - "**The system should never crash**." Instead, provide a quantitative requirement like: "**Every key stroke should provide a user response within 100 milliseconds**."

Modifiable

- SRS should be made as modifiable as possible and should be capable of easily accepting changes to the system to some extent. Modifications should be properly indexed and cross-referenced.

Traceable

- One should be able to trace a requirement to a design component and then to a code segment in the program. Similarly, one should be able to trace a requirement to the corresponding test cases.

3.6 SRS Documentation

I. Introduction
  A. System reference
  B. Overall description
  C. Software project constraints
II. Information Description
  A. Information content representation
  B. Information flow representation
    1. Data flow
    2. Control flow
III. Functional Description
  A. Functional partitioning
  B. Functional description
    1. Processing narrative
    2. Restrictions/limitations
    3. Performance requirements
    4. Design constraints
    5. Supporting diagrams
  C. Control Description
    1. Control specification
    2. Design constraints
IV. Behavioral Description
  A. System states
  B. Events and actions
V. Validation and Criteria
  A. Performance bounds
  B. Classes of tests
  C. Expected software response
  D. Special considerations
VI. Bibliography
VII. Appendix

FIGURE 11.9. Software requirements specification outline