Lecture 03

Requirements Engineering

Requirements Engineering

1. Introduction
2. Understanding Requirements
3. Process Modeling (Data Flow Diagram)
4. Data Modeling (Entity Relationship Diagram-ERD)
5. Other Techniques
6. Computer Aided Software Engineering (CASE)

Objectives

- To establish the importance / relevance of Requirement Specification
- To enlist the problems involved in specifying requirements
- To use modeling techniques to minimize problems in specifying requirements

What are Requirements ?

- **A condition or capability needed by a user to solve a problem or achieve an objective**
- **A condition or capability that must be met or possessed by a system to satisfy a contract, standard, specification, or other formally imposed document**
- **A software requirements specification (SRS) is a document containing a complete description of what the software will do without describing how it will do it.**

Requirements…

- May range from a **high-level** abstract statement of a service or of a system constraint to a **detailed** mathematical functional specification
- This is inevitable as requirements may serve a **dual function**
  - May be the basis for a bid for a contract - therefore must be open to interpretation
  - May be the basis for the contract itself - therefore must be defined in detail
  - Both these statements may be called requirements

Requirements...

- **Software requirements specify what the software must do to meet the business needs.**
- **For example, a stores manager might state his requirements in terms of efficiency in stores management.**
- **A bank manager might state his requirements in terms of time to service his customers.**
- **It is the analyst's job to understand these requirements and provide an appropriate solution.**
- **To be able to do this, the analyst must understand the client's business domain: who are all the stake holders, how they affect the system, what are the constraints, what are the alterables, etc**

Requirements...

- **The analyst should not blindly assume that only a software solution will solve a client's problem.**
- **He should have a broader vision.**
- **Sometimes, re-engineering of the business processes may be required to improve efficiency**
- **A detailed statement of what the software must do to meet the client's needs should be prepared. This document is called Software Requirements Specification (SRS) document.**

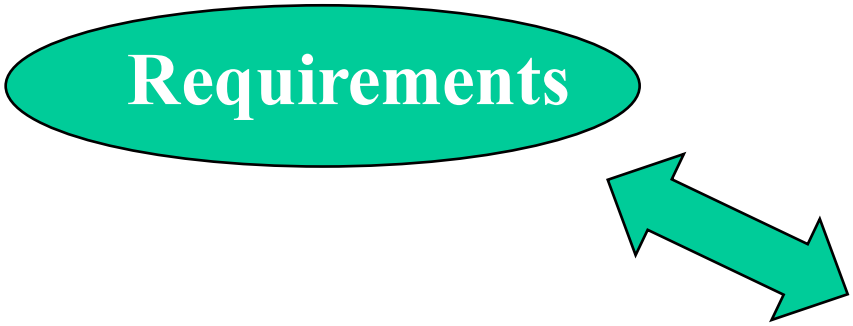
Types of requirement

- **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 services. Written as a contract between client and contractor
- **Software specification**
  - A detailed software description which can serve as a basis for a design or implementation. Written for developers

Requirements engineering

Requirements engineering is the process of establishing

- the services that the customer requires from a system
- the constraints under which it operates and is developed



The descriptions of the system services and constraints

that are generated during the requirements engineering process

Why the Term Requirements Engineering ?

Engineering, not just Analysis

- **Standards/Guidelines, Tools**
- **Breaking down of requirements, building up of solutions**
- **(IEEE)**
  - Analysis
  - Specification
  - Management

Library Problem

- **Check out a copy of a book, return a copy of a book**
- **Add a copy of a book to the Library**
- **Remove a copy of a book from the library**
- **Get the list of books by a particular author or in a particular subject area**

Library Problem ...Contd...

- **Find out the list of books currently checked out by particular borrower**
- **Find out which borrower last checked out a particular copy of a book**
- **There are two types of users : Staff Users and Ordinary Borrowers. Transactions 1 and 2 are restricted to staff users**

Constraints

1. **Copies in the library must be available for checkout or be checked out**

2. **Copy of the book may be both available and checked out at the same time**

3. **A borrower may not have more than a predefined number of books checked out at one time**

Ambiguities

- **What is a Library ?**
- **Who is a user ?**
- **Confusion regarding " *Book* " and " *Copy* "**
- **What does " *Available* " mean ?**
- **What do " *Last Checked Out* " and " *Currently* " mean ?**

Incompleteness

- **Initialization**
- **Addition of a new book**
- **Remove all copies of a book**
- **Create a library**
- **Add a staff user / ordinary user**
- **Error Handling**
- **Missing Constraints**
  - Period of Lending
  - Not more than one copy of the same book can be borrowed

Further Examples for Bad Specification

- **The counter value is picked up from the last record.**
- **Inversion of a square, matrix 'M' of size 'n' such as LM ML = I_n, where 'L' is the inverse matrix and I_n is the identity matrix of size 'n'.**
- **The software should be highly user friendly**
- **The output of the program shall usually be given within 10 seconds.**
- **The software should be developed on DOS system, but should be portable to other systems.**

Further Examples for Bad Specification

- **In first statement, the word 'last' is ambiguous. It could mean the last accessed record, which could be anywhere in a random access file, or, it could be physically the last record in the file**

- **Second statement though appears to be complete, is missing on the type of the matrix elements. Are they integers, real numbers, or complex numbers. Depending on the answer to this question, the algorithm will be different.**

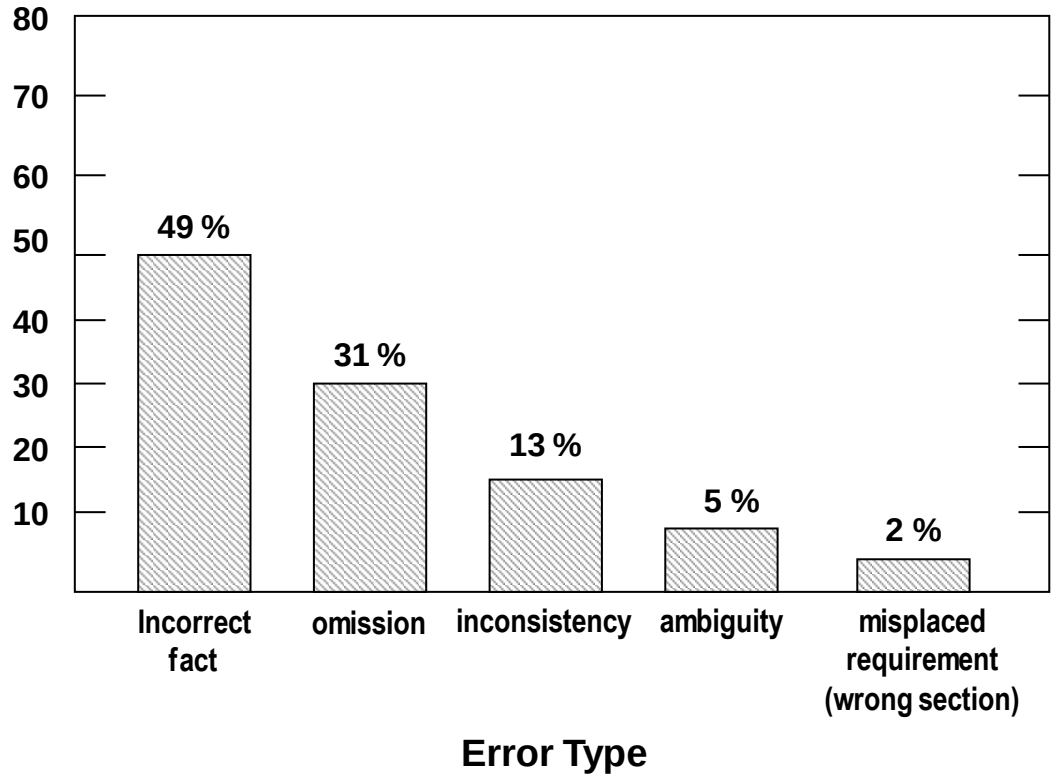
- **How does one determine, whether this requirement is satisfied or not.**

- **What are the exceptions to the 'usual 10 seconds' requirement?**

Characteristics of a Good SRS

- **Correct (No errors)**
- **Unambiguous**
- **Complete**
- **No use of TBDs *(To Be Determined)***
- **Verifiable (words like "highly", "usually" not to be used)**
- **Consistent (non-conflict)**
- **Modifiable**
- **Traceable**

Distribution of Errors in Requirements Specification



[NAVY A - 7E AIRCRAFTS OPERATIONAL FLIGHT PROGRAM; 5th INT. CONF. ON SOFTWARE ENGINEERING, 1981]

Requirements Definition & Analysis

The Requirements Definition and Analysis Phase is Critical to keep the Development and Maintenance Costs to a Minimum

Types Of Requirements

- **Functional Requirements**

  - what system should do

- **Non-Functional Requirements**

  - specify the overall quality attributes the system must satisfy.

Functional Requirements

The requirements that specify the Inputs (Stimuli) to the system, the Outputs (Responses) from the system, and the behavioral relationship between them

Functional Requirements (Examples)

- **To calculate the compound interest @ 8% per annum on a Fixed Deposit for a period of three years**
- **To calculate the Tax @ 30% on an annual income equal to and above Rs.2,00,000 but less than Rs.3,00,000**
- **To invert a Square Matrix (Maximum size 100 X 100) of Real Numbers**

Non-Functional Requirements

- **The requirements that describe the overall attributes of the system :**
  - Portability
  - Reliability
  - Performance
  - Testability
  - Modifiability
  - Security
  - Presentation
  - Reusability
  - Understandability
  - Acceptance Criteria

Non-Functional Requirements (Examples)

Performance

- **Number of significant digits to which accuracy should be maintained in a numerical solution**
- **Maximum response time in a transaction processing system**
- **Delays and task completion time limits in a real-time system**

Non-Functional Requirements (Examples) …Contd...

Environment

- **To run the software under a given operating system or use a specific programming language**

Security

- **The addition and deletion of the book in the system can be carried out by the Library-Chief only**

Non-Functional Requirements (Examples)

Testability

- All the off-diagonal elements, equal to or more than 10^{-3}, should be printed out by the matrix diagonalisation program

Understandability / Usability

- Experienced Officers should be able to use all of the system functions after a total training of two hours. After this training, the average number of errors made by experienced officers should not exceed two per day

Categories of Requirements

- **Satisfiability**
- **Service**
- **Criticality**
- **Stability**
- **User Categories**

Types of Satisfiability

- **Normal Requirements**
- **Expected Requirements**
- **Exciting Requirements**

Normal Requirements

- **User responses to specific questions**
- **Satisfy / dissatisfy in proportion to their presence / absence in system**
- **The satisfaction is linear and Bi-Directional**

Expected Requirements

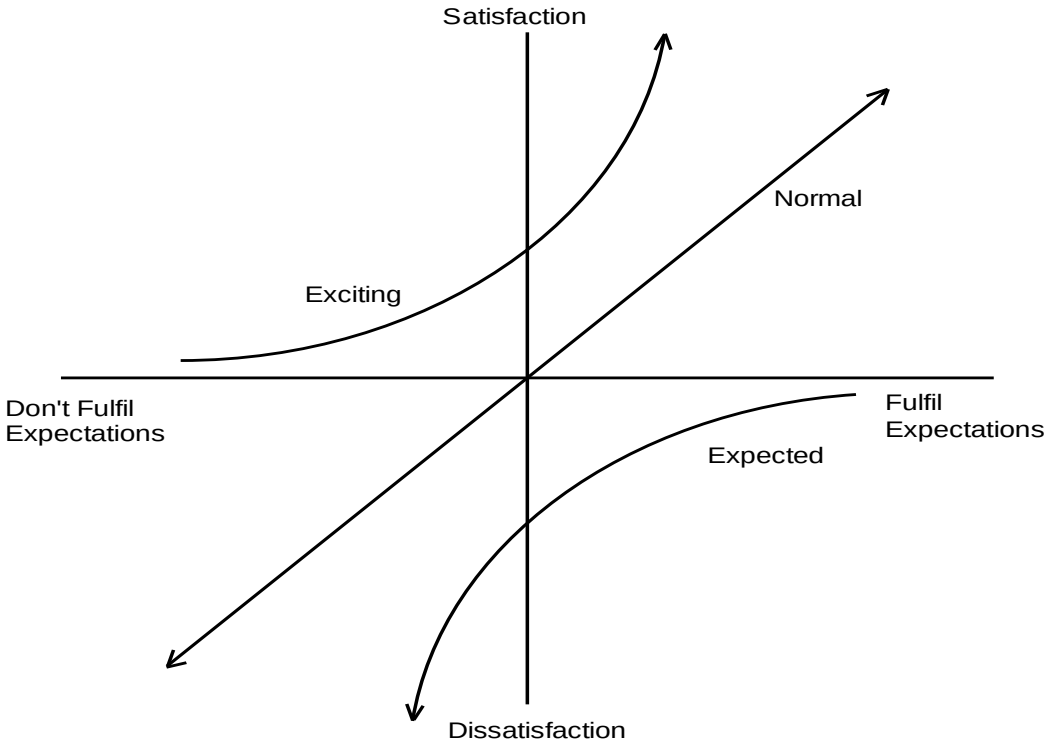
***Expected* requirements may not be stated by the users, but the developer is expected to meet them. So basic that users may neglect to mention them**

- **Their presence in solution may meet expectation but not enhance satisfaction**
- **Absence is very dissatisfying**
- **"Most Important" for analyst**

Exciting Requirements

- **Beyond the users' expectations; cannot be elicited from users**
- **Highly satisfying when met**
- **Their absence doesn't dissatisfy because they are not expected**

Satisfiability



Normal, Expected and Exciting Requirements

The Trend over the Years

- **Exciting requirements often become normal requirements**

- **Some normal requirements become expected requirements**

- **For example, on-line help feature was first introduced in the UNIX system in the form of *man* pages. At that time, it was an exciting feature. Later, other users started demanding it as part of their systems. Now a days, users do not ask for it, but the developer is expected to provide it.**

Type of Services

Function-Related Requirements

- *Users' Information Needs*
- *Information Processing Aspects*

Presentation-Related Requirements

- *Manner or form of presenting information*

Type of Services ...Contd...

Performance-Related Requirements

- ***Time-criticality of Information***
- ***Optimum Use of Resources***

Administration-Related Requirements

- ***Controls on the information processing***
- ***Organizational Climate***

Types of Stability

- **Stable**
- **Unstable**
  - Indicate the changes and their expected dates

Criticality

- **Classification**
  - *Mandatory*
  - *Desirable*
  - *Non-Essential*
- **Decision**
  - *In consultation with the users by the application of appropriate prioritization rules*
- **These define the problem domain**
- **Can help in phased development, based on criticality**

User Categories

- **Compile requirements against each defined user category to assure completeness of requirements**

- **Broadly they are of two kinds. Those who dictate the policies of the system and those who utilise the services of the system. All of them use the system. It is important that all stakeholders are identified and their requirements are captured.**

Modeling Requirements

- Every software system has the following essential characteristics:
- It has a boundary. The boundary separates what is with in system scope and what is outside
- It takes inputs from external agents and generates outputs
- It has processes which collaborate with each other to generate the outputs
- These processes operate on data by creating, modifying, destroying, and querying it
- The system may also use data stores to store data which has a life beyond the system