Software Requirements Analysis & Specification

- The goal of the requirements activity is to produce the software requirement specification (SRS) that describes what the proposed software should be do without describing how the s/w will do it.

* Value of a good SRS :-

- There are three major parties interested in a new system : the client, the developer & the users.
- The requirements for the system that will satisfy the needs of the clients and concerns of the users have to be communicated to the developer.
- The problem is that the client usually does not understand software or software development process.
- The developer often does not understand the client's problem & application area.
- This causes communication gap between the parties involved in the development project.
- The basic purpose of SRS is to bridge that communication gap
- Advantages of good SRS:-

1) An SRS establishes the basis for agreement between client & the supplier on what the software product will do.

- This basis for agreement is frequently formalized into a legal contract between the client & the developer.
- Through SRS client clearly describes what it expects from the supplier & the developer clearly understands what capabilities to built in the software.

2) An SRS provides a reference for validation of the final product.

- The SRS helps the client determine if the software meets the requirments.

- Providing the basis of agreement & validation should be strong enough reasons for both the client & the developer. to do thorough & rigorous job of requirments understanding & specification.

3) A High quality SRS is a prerequisite to high-quality software.

4) High quality SRS requeces the development cost.

* Requirement Process :-

- The requirement process is the sequence of activities that need to be performed in the requirements phase of that produces high-quality document containing the SRS.

- The requirement process consist of three basic task:
  1) Problem or requirement analysis
  2) Requirement specification.
  3) Requirements validation.

1) * Problem analysis :-

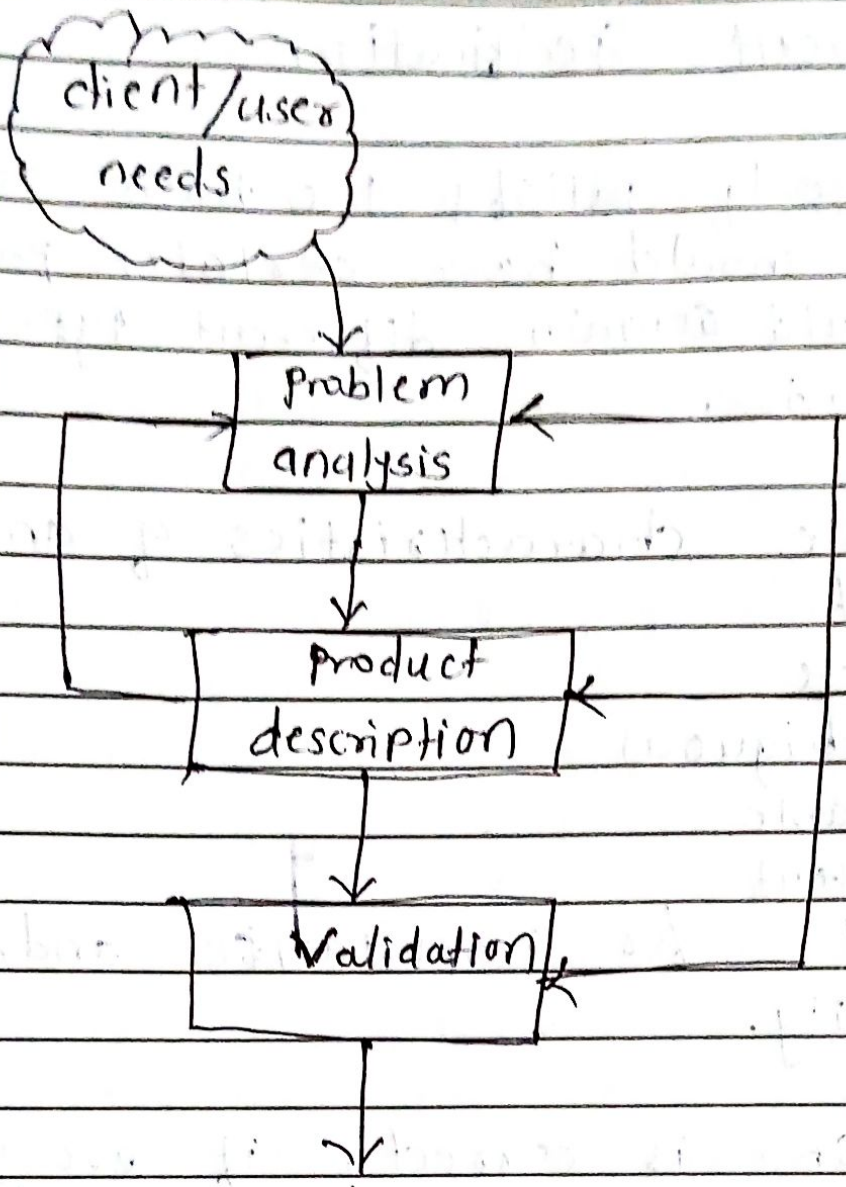
- Problem analysis often starts with a high-level problem statement.
- The basic purpose of this activity is to obtain a thorough understanding of what the software needs to provide
- During analysis, the analyst will have series of meetings with the clients & end users.
- In these early meetings, the ~~client~~ & analyst is basically the listener, absorbing the information provided.
- Once the analyst understands the system to some extent, he uses the next few meetings to seek clarification of the parts he does not understand.
- In the final few meetings, the analyst explains to the client what he understands the system should do & uses the meetings

as a means of verifying if what he proposes the system should do is indeed consistent with the objectives of the clients.

2)* Requirement specification :-
- Requirement specification focuses on clearly specifying the requirements in a document.
- In this activity, the issues like representation, specification languages, and tools are addressed.
- The important goal of this activity is properly organizing and describing requirements.

3)* Requirements validation :-
- Requirement validation ~~focuses on~~ ensuring that what have been specified in the SRS are indeed all the requirements of software & making sure that the SRS is of good quality
- The requirement process terminates with the production of the validated SRS.

- The requirement process is not a linear sequence of these three activities
- There is considerable overlap & feedback between these activities.
- The overall requirement process is shown in figure.



validation SRS.

fig. The requirement process

* Requirement Specification:-

- To properly satisfy the basic goals, an SRS should have certain properties
- It should contain different types of requirements.

Desirable characteristics of an SRS:-

1. Correct
2. complete
3. Unambiguous
4. Verifiable
5. Consistent
6. Ranked for importance and/or stability.

- An SRS is correct, if every requirement included in the SRS represents something required in the final system.
- It is complete, if everything the software is supposed to do and the responces of the software to all classes of input data are specified in the SRS.
- It is unambiguous if & only if every requirement stated has one & only one interpretation.
- Requirements are often written in natural language, which is inherently ambiguous.

- An SRS is verifiable if and only if every stated requirements is verifiable.
- A requirement is verifiable if there exists some cost-effective process that can check whether the final software meets that requirments.

- It is consistent if there is no requirement that conflicts with another.

- All the requirements for software are not of equal importance.
- some are important but not critical, and there are some which are desirable but not very important.
- An SRS is ranked for importance and/or stability if for each requirements the importance and the stability the future.
- In all these characteristics, completeness is perhaps the most important & also the most difficult property to establish.
- One of the most common defects in requirments specification is incompleteness.

Components of an SRS:-

1. Functionality :-
2. Performance
3. Design constraints imposed on an implementation.
4. External interfaces.

* Functional requirements :-

- Functional requirements specify the expected behavior of the system — which outputs should be produced from the given inputs.
- They describes the relationship between the input & output of the system
- For each functional requirements, a detailed description of all the data inputs and their source, the units of measure, and the range of valid inputs must be specified.
- All the operations to be performed on the input data to obtain the output should be specified.
- An important part of the specification is the system behavior in abnormal situations, like invalid inputs or error during computation.
- In short, the system behavior for all foreseen inputs & all foreseen ~~outputs~~ system states should be specified.

* Performance :-

- The performance requirements part of an SRS specifies the performance constraints on the software system.
- All the requirements relating to the performance characteristics of the system must be clearly specified.
- There are two types of performance requirements.
  1. static
  2. Dynamic.

- The static requirements are those that do not impose constraint on the execution characteristics of the system.
- These include requirements like the number of terminals to be supported, the number of simultaneous users to be supported, & no. of files that the system has to process and their sizes.
- These are also called capacity requirements of the system.

- Dynamic requirements specify constraints on the execution behaviour of the system.
- These typically include response time & throughput constraints on the system.

* Design constraints :-

- There are a number of factors in the client's environments that may restrict the choice of a designer leading to design constraints.
- An SRS should identify & specify all such constraints for eg.

① Standards compliance :-

- This specifies the requirements for the standards the system must follow.
- The standards may include the report may include the report format & accounting procedures.

② Hardware Limitations :-

- The software may have to operate on some existing or predetermined hardware thus imposing restrictions on the design.
- Hardware limitations can include the type of hardware or machines to be used, operating system available on the system, language supported, and limits on Primary & secondary storage.

③ Reliability & Fault Tolerance :-

- Fault Tolerance requirements can place a major constraint on how the system is to be designed,

(4) Security:

- Security requirements are becoming increasingly important.
- These requirements place restrictions on the use of certain commands, control access to data, provide different kinds of access requirments for different people.
- Require the use of passwords & cryptography techniques, & maintain a log of activities in the system.

* External interface:-

- All interactions of the software with people, Hardware & other software should be clearly specified in the external interface specification part.
- for hardware interface requirments, the SRS should specify the logical characteristics of each interface between the software product & the hardware components.
- The interface requirement should specify the interface with other s/w the system will use or that will use the system.

* Structure of a Requirements Document :-

All the requirements for a system, stated using a formal notation or natural language, have to be included in a document that is clear & concise.

Example :-

The organization based on the IEEE guide to software requirements specification

- The IEEE standards recognize the fact that different projects may require their requirements to be organized differently.
- The introduction section contains the purpose, scope, overview, etc. of the requirement document. as shown in figure.

Fig. General Structure of an SRS.

1. Introduction
   - 1.1 Purpose
   - 1.2 Scope
   - 1.3 Definition, Acronyms & abbrivation
   - 1.4 References
   - 1.5 Overview
2. Overall Description
   - 2.1 Product perspective
   - 2.2 Product functions
   - 2.3 User charcteristics
   - 2.4 General constraints
   - 2.5 Assumptions & Dependencies
3. Specific Requirements.

- The next section gives an overall perspective of the system.
- A general abstract description of the fuctions to be performed by the product is given.
- schematic diagram showing a general view of different fuctions & their relationships with each other can often be useful.

fig. one organization for specific requirements.

3. Detailed Requirements.
   - 3.1 External Interface Requirements
     - 3.1.1 User Interfaces
     - 3.1.2 Hardware Interfaces
     - 3.1.3 Software Interfaces
     - 3.1.4 communication Interfaces
   - 3.2 Functional Requirements
     - 3.2.1 mode 1
       - 3.2.1.1 functiondl Requirement 1.1
       - ⋮
       - 3.2.1.n Functional Requirements 1.n
     - ⋮
     - 3.2.m Mode m
       - 3.2.m.1 Functional Requirements m.1
       - ⋮
       - 3.2.m.n Functiondl Requirements m.n
   - 3.3 Performance Requirements
   - 3.4 Design constraints.
   - 3.5 Attributes
   - 3.6 Other Requirements.

- The detailed requirements section describes the details of the requirements that a developer needs to know for designing & developing the system.
- One method to organize the specific requirments is to first specify the external interfaces, followed by functional requirements, performance requirements, design constraints & system attributes.
- This struction is shown in figure.
- The external interface requirment section specifies all the interfaces of the software to people, other software, hardware & othe systems.
- In the fuctional requirment section, the functional capabilities of the system are described.
- The performance section specify both static & dynamic performance requirments.
- The attribut section specifies some of the overall attributes that the system should have.

Other approaches for Analysis:-

- The basic aim of problem analysis is to obtain a clear understanding of the needs of the clients & the users.
- The basic principle used in analysis is the same as in any complex tasks: divide & conquer.
- That is, partition the problem into subproblems & then try to understand each subproblem & its relationship to other subproblems in an effort to understand the total problem.
- In partitioning process, the state & projection concepts can be used.
- A state of a system represents some conditions about the system.
- In projection, a system is defined from multiple points of view
- Two other methods for problem analysis:-

* Data flow diagrams:-

- Data flow diagrams (also called data flow graphs) are commonly used during problem analysis.
- DFD's are very useful in understanding a system & can be effectively used during analysis.
- DFD's shows the flow of data through a system.
- It view a system as a function that transforms the inputs into desired outputs.
- The agent that performs the transformation

of data from one state to another is called process.

- DFD shows the movement of data through the different transformations or processes in the system.
- The processes are shown by named circles.
- And data flows are represented by named arrows entering or leaving the processes.
- A rectangle represents a source or sink and is a net originator or consumer of data.

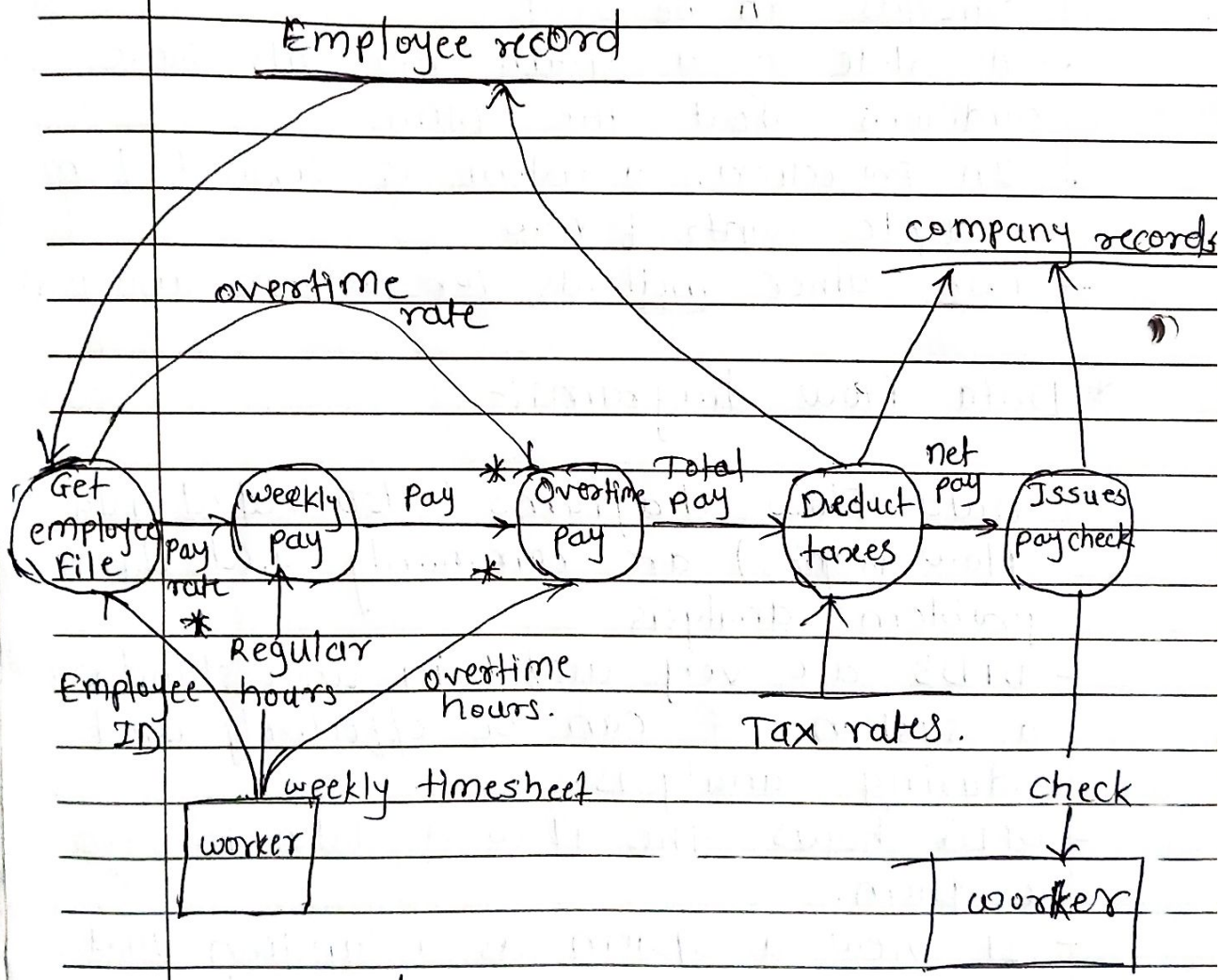


fig. DFD of system that pays workers.

- In DFD, there is one basic input data flow, the weekly timesheet, which originated from the source worker.
- The basic output is the paycheck,
- The sink for DFD is the worker
- In this system, first the employees record is retrieved, using the employee ID, which is contained in the timesheet.
- From employee record, the rate of payment & overtime are obtained.
- These rates & the regular hours & overtime hours are used to compute the pay.
- After total pay is determined, taxes are deducted
- To compute the tax deduction, information from the tax-rate file is used.
- Amout of tax deducted is recorded in the employee & company records.
- Finally the pay check is issued for the net pay.
- The amount paid is also recorded in company records.
- The need for multiple data flows by a process is represented by a "*" between the data flows.
- '*' This symbol represents 'AND' relationship
- It should be pointed out that DFD is not a flowchart.
- A DFD represents the flow of data, while flowchart shows flow of control

* Entity Relationship Diagrams:- (ERDs)

- An ERD can be used to model the data in the system & how the data items relates to each other,
- But does not cover how the data is actually manipulated & changed in the system.
- It is used by database designer ~~tool~~ to represents the structure of the database & is useful tool for analyzing software systems. which employ databases
- ERD's have two main concepts & notations to representing them.
- Entities & Relationships.
- Entities are main information holders or concepts in system
- Entities are represented as boxes in ERD.
- An entity is essentially equivalent to a table in a database with each row representing an instance of this entity.
- Attributes can be viewed as the columns of the database table & are represented as ellipses attached to entities.
- Relationships between two entities are represented by a line connecting the boxes representing the entities.
- In some notations, the name of the relationship is mentioned inside a diamond.

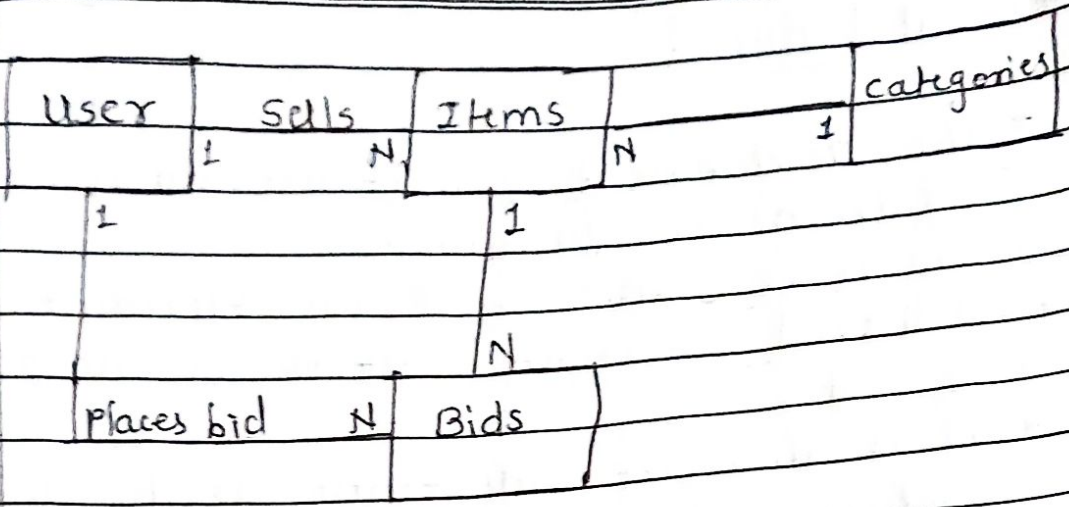


Fig. ERD for an auction system.

← The above diagram shows the university auction system. In that entities are - users, categories, items & bids.

- The relationship beteen them are also defined in diagram.
- A user can sell many items, But each item has only one seller. so there is one-to-many relationship. "sell" between the user & items.
- similarly there is a one to many relationship between users & bids, & between categories & items.
- Each entity represents a table & relationship determine what fields a table must have to support the relationship

Validation:-

- The development of software starts with a requirements document.
- which is also used to determine, the delivered software system is acceptable or not.
- It is therfore important that the requirments specification contains no errors & specifies the clients requirments correctly.
- The basic objective of the requirments validation activity is to ensure that the SRS reflects the actual requirments accurately & clearly.

- Types of errors:-
① omission ② inconsistency ③ incorrect fact ④ ambiguity.

- Omission is a common error in requirements.
- In this type of error, some user requirement is simply not included in the SRS.
- The omitted requirments may be related to the behavior of the system, its performance, constraints or any other factor.
- omission directly affects the external completeness of SRS.

- Inconsistency can be due to contradictions within the requirements themselves or due to incompatibility of the stated requirements with the actual requirements of the client or with the environment in which the system will operate.
- Errors like incorrect fact occurs when some fact recorded in SRS is not correct.
- The 4th common error type is ambiguity
- Errors of types occurs when these are some requirements that have multiple meaning that is, their interpretation is not unique
- As requirements are generally textual documents that cannot be executed, inspections & reviews are eminently suitable for requirements validation
- Inspection of SRS, frequently called requirements ~~rew~~ review, are the most common method of validation
- Requirements reviews remain the most commonly used & viable means for requirement validation.
- For validation, the most commonly used method is doing a structured group review of the requirements.