**Experiment No.01**

PART A

(PART A: TO BE REFFERED BY STUDENTS)

**A.1 Aim:** Finalise the problem statement and conduct the requirement gathering

**A.2 Prerequisite: -** C, C++, Software Development Life Cycle /Literature survey

**A.3 Outcome:**

**After successful completion of this experiment students will be able to**

1. Identify ambiguities, inconsistencies and incompleteness from a requirements specification
2. Identify and state functional requirements
3. Identify and state non-functional requirements
4. Frame the problem statement for the MINI project

**A.4 Theory:**

Requirements identification is the first step of any software development project. Until the requirements of a client have been clearly identified, and verified, no other task (design, coding, testing) could begin. Usually business analysts having domain knowledge on the subject matter discuss with clients and decide what features are to be implemented.

In this experiment we will learn how to identify functional and non-functional requirements from a given problem statement. Functional and non-functional requirements are the primary components of a Software Requirements Specification.

Requirements

Somerville defines "requirement" as a specification of what should be implemented. Requirements specify how the target system should behave. It specifies what to do, but not how to do. Requirements engineering refers to the process of understanding what a customer expects from the system to be developed, and to document them in a standard and easily readable and understandable format. This documentation will serve as reference for the subsequent design, implementation and verification of the system.

It is necessary and important that before we start planning, design and implementation of the software system for our client, we are clear about it's requirements. If we don't have a clear vision of what is to be developed and what all features are expected, there would be serious problems, and customer dissatisfaction as well.

Characteristics of Requirements

Requirements gathered for any new system to be developed should exhibit the following three properties:

Unambiguity: There should not be any ambiguity what a system to be developed should do. For example, consider you are developing a web application for your client. The client requires that enough number of people should be able to access the application simultaneously. What's the "enough number of people"? That could mean 10 to you, but, perhaps, 100 to the client. There's an ambiguity.

Consistency: To illustrate this, consider the automation of a nuclear plant. Suppose one of the clients say that if the radiation level inside the plant exceeds R1, all reactors should be shut down. However, another person from the client side suggests that the threshold radiation level should be R2. Thus, there is an inconsistency between the two end users regarding what they consider as threshold level of radiation.

Completeness: A particular requirement for a system should specify what the system should do and also what it should not. For example, consider a software to be developed for ATM. If a customer enters an amount greater than the maximum permissible withdrawal amount, the ATM should display an error message, and it should not dispense any cash.

Categorization of Requirements

Based on the target audience or subject matter, requirements can be classified into different types, as stated below:

User requirements: They are written in natural language so that both customers can verify their requirements have been correctly identified

System requirements: They are written involving technical terms and/or specifications, and are meant for the development or testing teams

Requirements can be classified into two groups based on what they describe:

Functional requirements (FRs): These describe the functionality of a system -- how a system should react to a particular set of inputs and what should be the corresponding output.

Non-functional requirements (NFRs): They are not directly related what functionalities are expected from the system. However, NFRs could typically define how the system should behave under certain situations. For example, a NFR could say that the system should work with 128MB RAM. Under such condition, a NFR could be more critical than a FR.

Non-functional requirements could be further classified into different types like:

Product requirements: For example, a specification that the web application should use only plain HTML, and no frames

Performance requirements: For example, the system should remain available 24x7

Organizational requirements: The development process should comply to SEI CMM level 4

Functional Requirements: Identifying Functional Requirements

Given a problem statement, the functional requirements could be identified by focusing on the following points:

Identify the high level functional requirements simply from the conceptual understanding of the problem. For example, a Library Management System, apart from anything else, should be able to issue and return books.

Identify the cases where an end user gets some meaningful work done by using the system. For example, in a digital library a user might use the "Search Book" functionality to obtain information about the books of his interest.

If we consider the system as a black box, there would be some inputs to it, and some output in return. This black box defines the functionalities of the system. For example, to search for a book, user gives title of the book as input and get the book details and location as the output.

Any high level requirement identified could have different sub-requirements. For example, "Issue Book" module could behave differently for different class of users, or for a particular user who has issued the book thrice consecutively.

**A.5 Task to be completed in PART B**

**A.5.1. Task 1:**

**Every student needs to follow following steps and record the findings in appropriate section of PART B**

1. Conduct the literature survey
2. Shortlist at least 5 domain of project and possible nature of problem.
3. Identify the scope of the problems selected
4. Identify the end user of the solution
5. Identify the functional requirements of the project
6. Identify the non-functional requirements of the project
7. Identify the feasibility of the project
8. Finalize one problem out of five selected
9. Frame the final problem statement.
10. List down the list of user requirements, system requirements.
11. Identify the ambiguities, inconsistencies, incompleteness from the requirements gathered.
12. List down the development plan for the selected problem statements

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**PART B**

(PART B: TO BE COMPLETED BY STUDENTS)

**(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)**

|  |  |
| --- | --- |
| Roll No. | Name: |
| Program : | Division: |
| Batch: | Date of Experiment: |
| Date of Submission: | Grade : |

**B.1 Tasks given in PART A to be completed here**

*(****Students must write the answers of the task(s) given in the PART A )***

**B.2 Observations and Learning:**

*(****Students must write the observations and learning based on their understanding built about the subject matter and inferences drawn)***

**B.3 Conclusion:**

*(****Students must write the conclusive statements as per the attainment of individual outcomes listed above and learning/observation noted in section B.2)***

**B.4 Question of curiosity:**

1. Justify the statement, *“ User requirements must be understood well before beginning the software development”*
2. What is SRS? Download sample SRS format of any organization and prepare the relevant part of the same for this experiment
3. What is feasibility study? When is the feasibility study done?
4. Differentiate between functional requirements and nonfunctional requirements

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*