Thakur Educational Trust's (Regd.)

THAKUR RAMNARAYAN COLLEGE OF ARTS & COMMERCE

ISO 21001:2018 Certified

PROGRAMME: B.Sc (I.T)

CLASS: S.Y.B.Sc (I.T)

SUBJECT NAME: SOFTWARE ENGINEERING

SEMESTER: IV

FACULTY NAME: Ms. SMRITI DUBEY

UNIT II

Chapter 3 – Requirement Engineering Process

Concepts:

Requirement Engineering Process

Feasibility Study

Requirement Elicitation and Analysis

View Point

Scenario

Ethnography

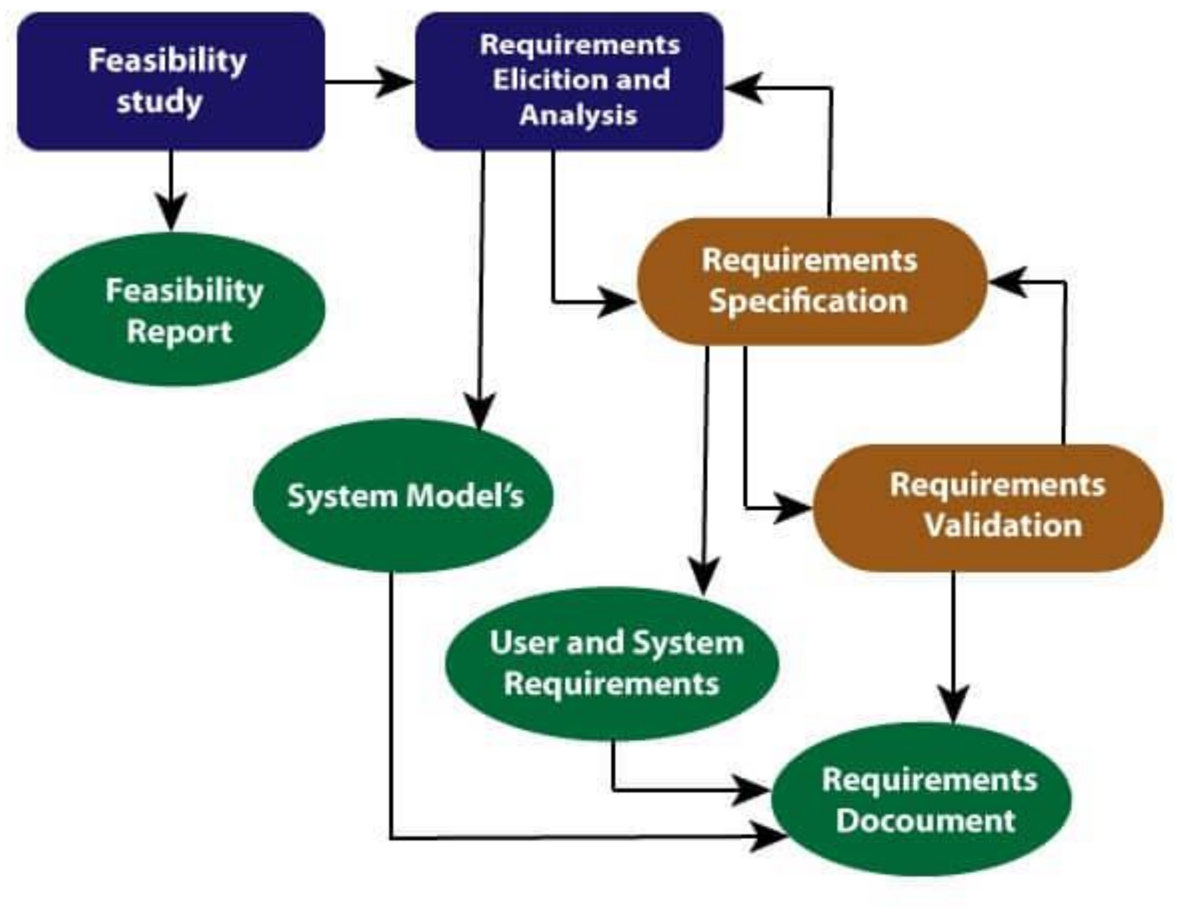
Requirement Validation Process

Requirement Management

Introduction to Requirement Engineering

The requirements for a system are the descriptions of what the system should do - the services that it provides and the constraints on its operation. These requirements reflect the needs of customers for a system that serves a certain purpose such as controlling a device, placing an order, or finding information**. The process of finding out, analyzing, documenting and checking these services and constraints is called requirements engineering (RE).**

Requirement Engineering Process



Requirement Engineering Process

The goal of the requirement engineering process is to create and maintain a system requirements document.The overall process includes four high-level requirements engineering sub-processes.These are concerned with assessing whether the system is useful to the business **(feasibility study)** ; discovering requirements **(elicitation and analysis)** ; converting these requirements into some standard form **(specification)** ; and checking that the requirements actually define the system that the customer wants **(validation)** .

These activities are concerned with the discovery,documentation and checking of requirements.The people involved develop a better understanding of what they want the software to do ; the organisation buying the system changes ; modifications are made to the system hardware,software and organisational environment.The process of managing these changing requirements is called requirement management .

Feasibility Study

For all new systems the requirement engineering process should start with a feasibility study.The input to the feasibility study is a set of preliminary business requirements an outline description of the system and how the system is intended to support business processes.The results of the feasibility study should be a report that recommends whether or not it is worth carrying on with the requirement engineering and system development process.If you are unsure whether your system solution will deliver the outcome you want,then a project feasibility study will help gain that clarity.During the feasibility study a variety of assessment methods are undertaken.The outcome of the feasibility study is a confirmed solution for implementation.

A feasibility study is a short focused study that aims to answer a number of questions:

1) Does the system contribute to the overall objectives of the organisation?

2) Can the system be implemented using current technology and within given cost and schedule constraints?

3) Can the system be integrated with other systems which are already in place?

Carrying out a feasibility study involves information assessment,information collection and report writing.

Requirement Elicitation and Analysis

After an initial feasibility study, the next stage of the requirements engineering process is requirements elicitation and analysis. In this activity, software engineers work with customers and system end-users to find out about the application domain,what services the system should provide, the required performance of the system,hardware constraints, and so on.

Requirement Engineering Process

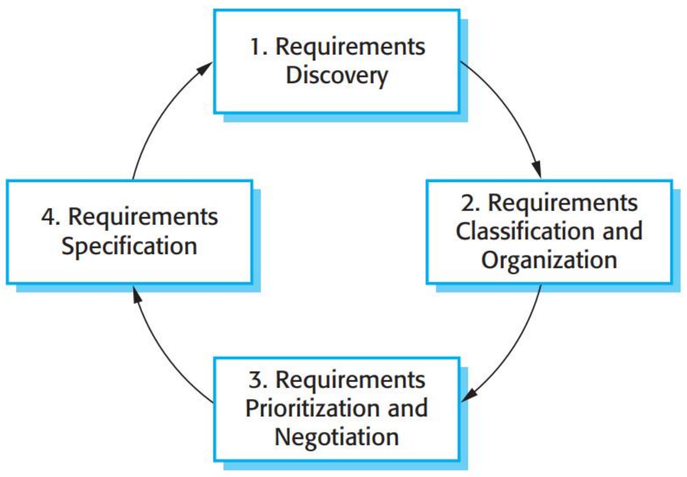

1. Requirements Discovery
2. Requirements Classification and Organization
3. Requirements Prioritization and Negotiation
4. Requirements Specification

Figure 4.13 The requirements elicitation and analysis process

Requirements elicitation and analysis may involve a variety of different kinds of people in an organization. A system stakeholder is anyone who should have some direct or indirect influence on the system requirements. Stakeholders include end users who will interact with the system and anyone else in an organization who willbe affected by it.

Each organization will have its own version or instantiation of this general model depending on local factors such as the expertise of the staff, the type of system being developed, the standards used, etc.

The process activities are:

1**. Requirements discovery** - This is the process of interacting with stakeholders of the system to discover their requirements. Domain requirements from stakeholders and documentation are also discovered during this activity.

2. **Requirements classification and organization** - This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters.

3**. Requirements prioritization and negotiation** - Inevitably, when multiple stakeholders are involved, requirements will conflict. This activity is concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation. Usually, stakeholders have to meet to resolve differences and agree on compromise requirements.

4. **Requirements specification** - The requirements are documented and input into the next round of the spiral. Formal or informal requirements documents may be produced

Figure 4.13 shows that requirements elicitation and analysis is an iterative process with continual feedback from each activity to other activities. The process cycle starts with requirements discovery and ends with the requirements documentation. The analyst's understanding of the requirements improves with each round of the cycle. The cycle ends when the requirements document is complete.

Viewpoint

A viewpoint is way of collecting and organizing a set of requirements from a group of stakeholders who have something in common. Each viewpoint therefore includes a set of system requirements. Viewpoints might come from end-users, managers, etc. They help identify the people who can provide information about their requirements and structure the requirements for analysis.

Viewpoints can be used as a way of classifying stakeholders and other sources of requirements.There are three generic types of viewpoint:

1) **Interactor viewpoints** represent people or other systems that interact directly with the system.In the bank ATM system,examples of interactor viewpoints are bank's customers and the bank's account database.

2**) Indirect viewpoints** represent stakeholders who do not use the system themselves but who influence the requirements in some way.In the bank ATM system examples of indirect viewpoints are the management of the bank and the bank security staff.

3) **Domain viewpoints** represent domain characteristics and constraints that influence the system requirements. In the bank ATM system examples of domain viewpoint would be the standards that have been developed for interbank communications.

There are various ways to discover requirements

1. Interviews

Formal or informal interviews with system stakeholders are part of most requirements engineering processes. In these interviews, the requirements engineering team puts questions to stakeholders about the system that they currently use and the system to be developed. Requirements are derived from the answers to these questions.

Interviews may be of two types:

1. **Closed interviews**, where the stakeholder answers a pre-defined set of questions.

2**. Open interviews**, in which there is no pre-defined agenda. The requirements engineering team explores a range of issues with system stakeholders and hence develop a better understanding of their needs.

2. Scenarios

People usually find it easier to relate to real-life examples rather than abstract descriptions. They can understand and criticize a scenario of how they might interact with a software system. Requirements engineers can use the information gained from this discussion to formulate the actual system requirements.Scenarios can be particularly useful for adding detail to an outline requirements description. They are descriptions of example interaction sessions. Each scenario usually covers one or a small number of possible interactions. Different forms of scenarios are developed and they provide different types of information at different levels of detail about the system.

A scenario starts with an outline of the interaction. During the elicitation process, details are added to this to create a complete description of that interaction. At its most general, a scenario may include:

1. A description of what the system and users expects when the scenario starts.
2. A description of the normal flow of events in the scenario.
3. A description of what can go wrong and how this is handled.
4. Information about other activities that might be going on at the same time.
5. A description of the system state when the scenario finishes.

Scenario-based elicitation involves working with stakeholders to identify scenarios and to capture details to be included in these scenarios. Scenarios may be written as text, supplemented by diagrams, screen shots, etc. Alternatively, a more structured approach such as event scenarios or use cases may be used.

3. Use cases

Use cases are a requirements discovery technique that were first introduced in the Objectory method. In their simplest form, a use case identifies the actors involved in an interaction and names the type of interaction. This is then supplemented by additional information describing the interaction with the system.The additional information may be a textual description or one or more graphical models such as UML sequence or state charts. Use cases are documented using a high-level use case diagram. The set of use cases represents all of the possible interactions that will be described in the system requirements. Actors in the process, who may be human or other systems, are represented as stick figures. Each class of interaction is represented as a named ellipse.Lines link the actors with the interaction. Optionally, arrowheads may be added to lines to show how the interaction is initiated.

4. Surveys

Organization may conduct surveys among various stakeholders by querying about their expectation and requirements from the upcoming system.

5. Questionnaires

A document with pre-defined set of objective questions and respective options is handed over to all stakeholders to answer, which are collected and compiled.A shortcoming of this technique is, if an option for some issue is not mentioned in the questionnaire, the issue might be left unattended.

6. Observation

This technique involves observing the stakeholders in their work environment to gather information about their needs and expectations.

7. Prototyping

This technique involves creating a working model of the software system, which can be used to gather feedback from stakeholders and to validate requirements.

Ethnography

Ethnography is an observational technique that can be used to understand operational processes and help derive support requirements for these processes. An analyst immerses himself or herself in the working environment where the system will be used. The day-to-day work is observed and notes made of the actual tasks in which participants are involved. The value of ethnography is that it helps discover implicit system requirements that reflect the actual ways that people work, rather than the formal processes defined by the organization.

People often find it very difficult to articulate details of their work because it is second nature to them. They understand their own work but may not understand its relationship to other work in the organization. Social and organizational factors that affect the work, but which are not obvious to individuals, may only become clear when noticed by an unbiased observer. For example, a work group may self-organize so that members know of each other’s work and can cover for each other if someone is absent. This may not be mentioned during an interview as the group might not see it as an integral part of their work.

Ethnography is particularly effective for discovering two types of requirements:

1. Requirements that are derived from the way in which people actually work, rather than the way in which process definitions say they ought to work. For example, air traffic controllers may switch off a conflict alert system that detects aircraft with intersecting flight paths, even though normal control procedures specify that it should be used. They deliberately put the aircraft on conflicting paths for a short time to help manage the airspace. Their control strategy is designed to ensure that these aircrafts are moved apart before problems occur and they find that the conflict alert alarm distracts them from their work.

2. Requirements that are derived from cooperation and awareness of other people’s activities. For example, air traffic controllers may use an awareness of other controllers’ work to predict the number of aircrafts that will be entering their control sector. They then modify their control strategies depending on that predicted workload. Therefore, an automated ATC system should allow controllers in a sector to have some visibility of the work in adjacent sectors.

Requirements validation

Requirements validation is the process of checking that requirements actually define the system that the customer really wants. Requirements validation is important because errors in a requirements document can lead to extensive rework costs when these problems are discovered during development or after the system is in service.

During the requirements validation process, different types of checks should be carried out on the requirements in the requirements document. These checks include:

1**. Validity checks** - A user may think that a system is needed to perform certain functions. However, further thought and analysis may identify additional or different functions that are required. Systems have diverse stakeholders with different needs and any set of requirements is inevitably a compromise across the stakeholder community.

2. **Consistency checks** - Requirements in the document should not conflict. That is, there should not be contradictory constraints or different descriptions of the same system function.

3. **Completeness checks** - The requirements document should include requirements that define all functions and the constraints intended by the system user.

4. **Realism checks** - Using knowledge of existing technology, the requirements should be checked to ensure that they can actually be implemented. These checks should also take account of the budget and schedule for the system development.

5. **Verifiability** - To reduce the potential for dispute between customer and contractor, system requirements should always be written so that they are verifiable. This means that you should be able to write a set of tests that can demonstrate that the delivered system meets each specified requirement.

There are a number of requirements validation techniques that can be used individually or in conjunction with one another:

1. **Requirements reviews** - The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.

2. **Prototyping** - In this approach to validation, an executable model of the system in question is demonstrated to end-users and customers. They can experiment with this model to see if it meets their real needs.

3. **Test-case generation** - Requirements should be testable. If the tests for the requirements are devised as part of the validation process, this often reveals requirements problems. If a test is difficult or impossible to design, this usually means that the requirements will be difficult to implement and should be reconsidered. Developing tests from the user requirements before any code is written is an integral part of extreme programming.

Requirements management

Requirements management is the process of understanding and controlling changes to system requirements. The requirements for large software systems are always changing. Because the problem cannot be fully defined, the software requirements are bound to be incomplete. During the software process, the stakeholders' understanding of the problem is constantly changing. The system requirements must then also evolve to reflect this changed problem view.

Once a system has been installed and is regularly used, new requirements inevitably emerge.

There are several reasons why change is inevitable:

1. The business and technical environment of the system always changes after installation. New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by.

2. The people who pay for a system and the users of that system are rarely the same people. System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

3. Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory. The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.