Requirements Engineering

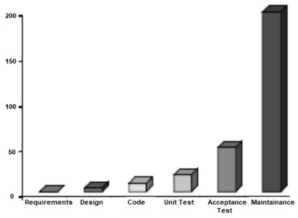
21 September 2023 13:59

What is Requirement?
Requirement is the property which must be exhibited by software developed/adapted to solve a particular problem.

- What not how
- First step in any software development lifecycle model

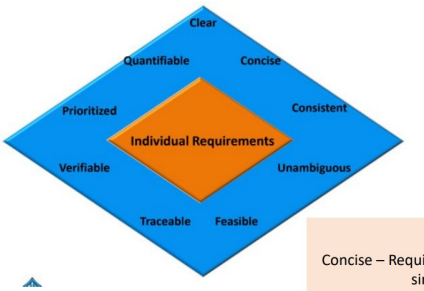
- Difficult, error prone and costly
- Critical for successful development of all down stream activities
- Requirement errors are expensive to fix

Cost of repair as a function of time



Life Cycle Stage	Relative cost of Repair
Requirements	0.1 to 0.2
Design	0.5
Coding	1
Unit test	2
Acceptance test	5
Maintenance	20

Properties of requirements



Concise – Requirements should describe a single property

See more details here:

Properties of a set of requirements

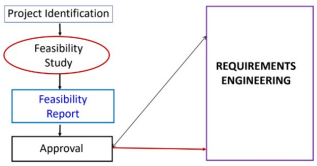
RR CC M

- When there is limited time/ budget we need to choose which requirements to satisfy
- We must consider
  - Changes in customer needs
  - Improved developer understanding of products
  - Changes in organizational policy

Feasibility Study

Short, low-cost study to asses the practicality of the project and whether it should be done.

When is Feasibility Study conducted?
Mostly done before beginning a project



ACTIVITIES IN FEASIBILITY STUDY

- Figure out the client or the sponsor or the user who would have a stake in the project
- Find the current solution to the problem
- Find the targeted customers and the future market place
- Potential benefits
- Scope
- High level block level understanding of the solution
- Considerations to technology
- Marketing strategy
- Financial projection
- Schedule and high level planning and budget requirements
- Issues, assumptions, risks and constraints
- Alternatives and their consideration
- Potential project organization

Ends with GO or NO-GO

Requirements Engineering process

A "four + one" set of activities to produce specifications or requirements

It is an iterative process

EASV

1. **Requirements Elicitation:**

Establishing clear scope and boundary for project by working proactively with all stakeholders, gathering their needs, articulating their problem, identifying and negotiating potential conflicts

- Understanding the problem
- Understanding the domain
- Identifying clear objectives
- Understanding the needs
- Understanding constraints of the system stake holders
- Writing business objectives for the project

Elicitation techniques

Approach is based on:
- ➢Nature of the system being developed
- ➢Background and experience of stakeholders

Elicitation techniques – **Active** and **Passive**

Active elicitation techniques
Ongoing interaction between the stake holders and users.
- ➢Interviews
- ➢Facilitated meetings
- ➢Role-playing
- ➢Prototypes
- ➢Ethnography
- ➢Scenarios

Passive elicitation tecniques
Infrequent interaction between the stake holders and users.
- ➢Use cases
- ➢Business process analysis & modeling
- ➢Workflows
- ➢Questionnaires
- ➢Checklists
- ➢Documentation

2. **Requirements Analysis**

Process of Requirements Analysis

1. Understand requirements in depth
2. Classify requirements into coherent clusters
3. Model the requirements
4. Analyze requirements using fishbone diagram
5. Recognize and resolve conflicts
6. Negotiate requirements
7. Prioritize requirements – MoSCoW **(Must have, Should have, Could have, Won't have)**
8. Identify risks
9. Decide on build or buy – COTS solution.

COTS: Commercial off the shelf

Classify requirements into coherent clusters:

1. Functional requirements :How system will behave for each situation
Ex) System shall assign a unique tracking number to each shipment
2. Non-functional requirements: Constraints on services offered by system (standards, development constraints, time constraints)
Ex) With 100 concurrent users a database record shall be fetched over the network in less than 3ms
3. User requirements : Statements in natural language plus informal context diagrams, system/sub-system and their interconnections and operational constraints. Written for/by customers
4. System requirements : A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented; so may be part of a contract between client and contractor.

5. Domain requirements : Constraints on the system from the domain of operations.
Ex) The software must be developed in accordance with IEC 60601 regarding the basic safety and performance for medical electrical equipment

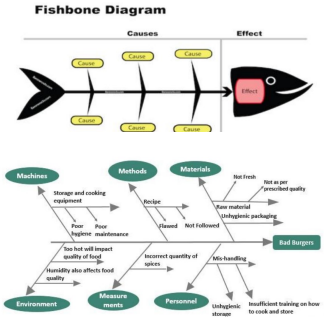
Model requirements

- Provide an understanding of the system
- Communicating and interpreting requirements

1. Structural models:
   a. Captures static entities of system
   b. What entities exist and how they're related
2. Behavioural models:
   a. Captures dynamic aspects of system
   b. How do entities react in response to a stimulus

Analyse using fish bone diagram

List out all the reasons/causes on why the requirement (effect) has come in and become relevant.

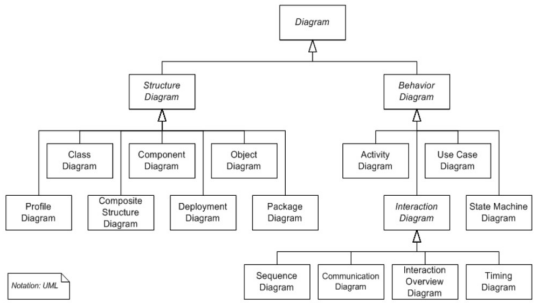


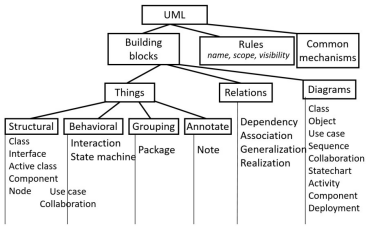
UML (Unified Modelling Language)

What is UML?
The Unified Modeling Language has become the de-facto standard for building Object-Oriented software.

UML plays an important role in defining different perspectives of a system :Requirements, Design, Implementation, Deployment

UML Use-case models are predominantly used with Modelling Systems, to discuss the dynamic behaviour of the system when it is running/operating. It is often used to used to gather the requirements of a system including internal and external influences



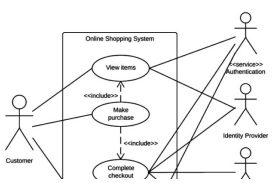


Using Use Case diagram

Use case from a user's point of view outlines how the proposed system will perform a task expected to be performed, while responding to a request or task of a role/actor/user.

Use case diagrams are used to visualize, specify, construct, and document the (intended) behavior of the system, during requirements capture and analysis. Used by developers, domain experts and end-users.

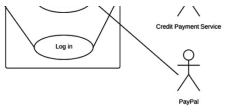
Actor is someone (can be a human or other external system) interacting with the use case (system function), named by noun but is not part of the system.



Use case diagram symbols and notation

Use cases: Horizontally shaped ovals that represent the different uses that a user might have.

Actors: Stick figures that represent the people actually employing the use cases.



Associations: A line between actors and use cases.

3. Requirements Specification

- Documentation of a set of requirements that is reviewed and approved by the customer and provides direction for the software construction activities in the next stage of the life cycle.
- Software requirements specification (SRS): Document is the basis for customers and contractors/suppliers agreeing on what the product will and will not do. It describes both the functional and non-functional requirements

Software Requirement Specification (SRS)

Functionality: What is the software supposed to do?

External interfaces: How does the software interact with people, the system's hardware, other hardware, and other software?

Non Functionality: This includes all of the Quality criteria which drive the functionality. Example: Performance, Availability, Portability etc.

Design constraints imposed on an implementation:
- Required standards in effect
- Implementation language
- Policies for database integrity
- Resource limits
- Security
- Operating environment(s) etc.

4. Requirements Validation

Validation determines whether the software requirements if implemented, will solve the right problem and satisfy the intended user needs
Verification determines whether the requirements have been specified correctly

- Repairing requirement errors as we go downstream becomes more expensive.

Requirements validation

Requirement reviews

Prototyping
Prototype facilitates user involvement during requirements engineering phase and ensures engineers and users have the same interpretation of the requirements.
Prototyping is most beneficial in systems – With many user interactions
Example: Design of online billing systems

Systems with little or no user interaction may not benefit as much from prototyping.
Example: Batch processing

Model Validation

- Ensuring that the models represent all essential functional requirements
- Demonstrating that each model is consistent in itself
- Usage of the Fish Bone Analysis technique for validation

Acceptance Criteria

To check if there are requirements matching with that the Acceptance criteria.

Requirements Management

Requirements specification is the baseline on which the future lifecycle phases will need to build upon.

Can you think of reasons why requirements might change?
- Better understanding of the problem
- Customer internalizing the problem and solution
- Evolving environment and technology landscape

Facets of Requirements Management
1. Ensuring that the requirements are all addressed in each phase of the lifecycle
2. Ensuring that the changes in the requirements are handled appropriately

Requirements Traceability Matrix (RTM)

Req Id	Architectural Section	Design Section	File/ Implementation	Unit Test Id	Functional Test ID	System Test ID	Acceptance Test Id

Requirements are traced across the SDLC using the requirement traceability matrix (RTM)
- Forward Tracing
- Backward Tracing

Every phase of the SDLC progressively fills the RTM

Requirements Change Management

REQUIREMENT CHANGE PROCESS

Log the request for change and assign a change request Identifier → Log
- Who is requesting the change
- Why is the request coming in
- What is being requested to

→ Perform impact analysis & estimate impact.
Review impact with stakeholders

Requirements Change Management