Requirements Engineering

- Problems with requirements practices
- Requirement engineering phases

A Customer walks into your office and says;

“I know you think you understand what I said, but what you don’t understand is what I said is not what I mean”.

Invariably this happens late in the project, after deadline commitments have been made, reputations are on the line, and serious money is at stake.

Requirement Engineering

- **Requirements engineering** (RE) refers to the process of defining, documenting and maintaining requirements.

- Requirement Engineering (RE) is the science and discipline concerned with analyzing and documenting requirements.

Starter Questions

- What is a "requirement"?
- How do we determine the requirements?

Types of Requirements

- Functional
  - ex - it must email the sales manager when an inventory item is "low"
- Non-Functional
  - ex - it must require less than one hour to run
- Explicit
  - ex – required features
- Implied
  - ex – software quality
- Forgotten
  - ex – exists in current process
- Unimagined

Questions

- What makes a particular requirement good or bad?
- Why is requirements engineering difficult?

Requirements of Requirements

- Clear
- Measurable
- Feasible
- Necessary
- Prioritized
- Concise

The Problems with our Requirements Practices

- We have trouble in understanding the requirements that we do acquire from the customer
- We often record requirements in a disorganized manner
- We spend far too little time verifying what we do record
- We allow change to control us, rather than establishing mechanisms to control change
- Most importantly, we fail to establish a solid foundation for the system or software that the user wants built

The Problems with our Requirements Practices (contd..)

- Many software developers argue that
  - Building software is so compelling that we want to jump right in (before having a clear understanding of what is needed)
  - Things will become clear as we build the software
  - Project stakeholders will be able to better understand what they need only after examining early iterations of the software
  - Things change so rapidly that requirements engineering is a waste of time
  - The bottom line is producing a working program and that all else is secondary
- All of these arguments contain some truth, especially for small projects that take less than one month to complete
- However, as software grows in size and complexity, these arguments begin to break down and can lead to a failed software project

A Solution: Requirements Engineering

- Begins during the communication activity and continues into the modeling activity
- Builds a bridge from the system requirements into software design and construction
- Allows the requirements engineer to examine
  - the context of the software work to be performed
  - the specific needs that design and construction must address
  - the priorities that guide the order in which work is to be completed
  - the information, function, and behavior that will have a profound impact on the resultant design

A Bridge to Design and Construction

- Understanding the requirements of a problem is among the most difficult tasks that face a software engineer.
- Requirements engineering establishes a solid base for design and construction. Without it, the resulting software has a high probability of not meeting customer's needs.
- Requirement engineering builds a bridge to design and construction. But where does the bridge originate?
  - Begins at the feet of the project stakeholders, where business need is defined, user scenarios are described, functions and features are delineated, and project constraints are identified.

Requirements Engineering Tasks

- Requirements engineering provides the appropriate mechanism for
  - Understanding what the customer wants,
  - Analyzing need,
  - Assessing feasibility,
  - Negotiating a reasonable solution,
  - Specifying the solution unambiguously,
  - Validating the specification and
  - Managing the requirements as they are transformed into an operational system.

- Seven distinct tasks: Inception, Elicitation, Elaboration, Negotiation, Specification, Validation, Requirements Management

Inception

Elicitation

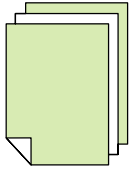
Elaboration

Negotiation

Specification

Validation

Requirements Management



Inception Task

- During inception, the requirements engineer asks a set of questions to establish…
  - A basic understanding of the problem
  - The people who want a solution
  - The nature of the solution that is desired
  - The effectiveness of preliminary communication and collaboration between the customer and the developer

- Through these questions, the requirements engineer needs to…
  - Identify the stakeholders
  - Recognize multiple viewpoints
  - Work toward collaboration
  - Break the ice and initiate the communication

The First Set of Questions

These questions focus on the customer, other stakeholders, the overall goals, and the benefits

- Who is behind the request for this work?
- Who will use the solution?
- What will be the economic benefit of a successful solution?
- Is there another source for the solution that you need?

The Next Set of Questions

These questions enable the requirements engineer to gain a better understanding of the problem and allow the customer to voice his or her perceptions about a solution

- How would you characterize "good" output that would be generated by a successful solution?
- What problem(s) will this solution address?
- Can you show me (or describe) the business environment in which the solution will be used?
- Will special performance issues or constraints affect the way the solution is approached?

The Final Set of Questions

These questions focus on the effectiveness of the communication activity itself

- Are you the right person to answer these questions? Are your answers "official"?
- Are my questions relevant to the problem that you have?
- Am I asking too many questions?
- Can anyone else provide additional information?
- Should I be asking you anything else?

Elicitation Task

- Questions asked in a manner that no further doubt remains

- Elicitation of requirements is difficult because
  - Problems of scope in identifying the boundaries of the system or specifying too much technical detail rather than overall system objectives
  - Problems of understanding what is wanted, what the problem domain is, and what the computing environment can handle (Information that is believed to be "obvious" is often omitted)
  - Problems of volatility because the requirements change over time

- Elicitation may be accomplished through two activities
  - Collaborative requirements gathering
  - Quality function deployment

Collaborative Requirement Gathering

- Team Oriented Approach
- Team of stakeholders and developers meet
  - Identify problem
  - Propose a solution
  - Negotiate different approaches
  - Specify preliminary solution
- Many approaches but all follow basic guidelines

Guidelines

Meetings are conducted and attended by both s/w engineers and customers

Rules for preparation and participation should be established

Agenda for formal and informal discussions

Facilitator- controls the meetings, can be anyone from participants

Definition mechanism- Worksheets, Flip charts, Chat rooms, Virtual Forum etc

Goal- Identify problem and solution

Scenario

- Initial meetings in inception phase – One or two pages of Product Request (PR)
- Meeting place, time & date should be selected for further meetings and facilitator need to be identified
- Members of s/w team and other stakeholders are invited to attend

  PR is distributed to all members for approval

- While reviewing PR each attendee asked to make **object list** before attending future meeting
  - Objects regarding environmental aspects
  - Objects produced by system
  - Objects used by system
- Each attendee asked to make **list of services**
  - Process and Functions
- Each attendee has to make **list of constraints** and **performance criteria**

Example: SafeHome Project (Home Security Function)

Our research indicates that the market for home management systems is growing at a rate of 40% per year. The first SafeHome function we bring to market should be the home security function. Most people are familiar with “Alarm Systems” so this would be an easy sell.

The home security function would protect against and/or recognize a variety of undesirable “situations” such as illegal entry, fire, flooding, carbon monoxide levels and others. It will use our wireless sensors to detect each situation, can be programmed by homeowner and will automatically telephone a monitoring agency when a situation is detected.

- List of Objects:
  - Control Panel
  - Smoke detector
  - Window & door sensors
  - Motion Detector
  - An Alarm
  - An Event
  - A Display
  - A PC
  - Telephone numbers
  - Telephone call
- List of Services:
  - Configuring the system
  - Setting the alarm
  - Monitoring the sensors
  - Dialing phone
  - Programming control panel
  - Reading display
- List of Constraints:
  - System should detect and recognize failure of sensors
  - User Friendly display
  - Performance of sensor recognition

Elicitation Meeting

- Need to combine all documents from all attendees and prepare agreement
- Take one topic and prepare list of points
- Start with mini specifications
  - E.g. control panel: mounted unit with some size with sensors and PC connected. User interaction through keyboard. LCD display for feedback.
- Each mini specification will be discussed: addition/deletion/further elaboration
- Uncover new objects, services and constraints
- After each meeting **Issue List** will be prepared and maintained.
- Mention **Validation Criteria** for all requirements
- Finally call outsider to write **SRS (Software Requirement Specifications)**

Quality Function Deployment (QFD)

- Technique to translate needs of the customer into technical requirements
- Concentrates on maximizing customer satisfaction from s/w engineering process
- Three types of Requirements:
  - Normal- Objectives and Goals
    - E.g. Graphical Display, Specific functions, performance
  - Expected- implicit to the product, customer does not explicitly state them
    - E.g. GUI, Reliability, Ease of S/W installation
  - Exciting- Go beyond customers expectations

Deployment Tasks

- Function Deployment: Value each function i.e. requirement by setting priority
- Information Deployment: Identifies data, objects and events
- Task Deployment: Behavior of the system
- Value Analysis: Determine relative priority of the requirements

QFD Process

- QFD uses raw data for requirement gathering:
  - Customer Requirements
  - Observations
  - Surveys
  - Examination of historical data
- The above all data will be translated into a table of requirements- Customer Voice Table – reviewed with customer
- Variety of diagrams, matrices and evaluation methods are used to extract the expected and excited requirements

Elaboration Task

- During elaboration, the software engineer takes the information obtained during inception and elicitation and begins to expand and refine it
- Elaboration focuses on developing a refined technical model of software functions, features, and constraints
- It is an analysis modeling task
  - Use cases are developed
  - Domain classes are identified along with their attributes and relationships
  - State machine diagrams are used to capture the life on an object
- The end result is an analysis model that defines the functional, informational, and behavioral domains of the problem

Developing Use Cases

- Step One – Define the set of actors that will be involved in the story
  - Actors are people, devices, or other systems that use the system or product within the context of the function and behavior that is to be described
  - Actors are anything that communicate with the system or product and that are external to the system itself
- Step Two – Develop use cases, where each one answers a set of questions

Questions Commonly Answered by a Use Case

- Who is the primary actor(s), the secondary actor(s)?
- What are the actor's goals?
- What preconditions should exist before the scenario begins?
- What main tasks or functions are performed by the actor?
- What exceptions might be considered as the scenario is described?
- What variations in the actor's interaction are possible?
- What system information will the actor acquire, produce, or change?
- Will the actor have to inform the system about changes in the external environment?
- What information does the actor desire from the system?
- Does the actor wish to be informed about unexpected changes?

Elements of the Analysis Model

- Scenario-based elements
  - Describe the system from the user's point of view using scenarios that are depicted in use cases and activity diagrams
- Class-based elements
  - Identify the domain classes for the objects manipulated by the actors, the attributes of these classes, and how they interact with one another; they utilize class diagrams to do this
- Behavioral elements
  - Use state diagrams to represent the state of the system, the events that cause the system to change state, and the actions that are taken as a result of a particular event; can also be applied to each class in the system
- Flow-oriented elements
  - Use data flow diagrams to show the input data that comes into a system, what functions are applied to that data to do transformations, and what resulting output data are produced

Negotiation Task

- During negotiation, the software engineer reconciles the conflicts between what the customer wants and what can be achieved given limited business resources
- Requirements are ranked (i.e., prioritized) by the customers, users, and other stakeholders
- Risks associated with each requirement are identified and analyzed
- Rough guesses of development effort are made and used to assess the impact of each requirement on project cost and delivery time
- Using an iterative approach, requirements are eliminated, combined and/or modified so that each party achieves some measure of satisfaction

The Art of Negotiation

- Recognize that it is not competition
- Map out a strategy
- Listen actively
- Focus on the other party’s interests
- Don’t let it get personal
- Be creative
- Be ready to commit

Specification Task

- A specification is the final work product produced by the requirements engineer
- It is normally in the form of a software requirements specification
- It serves as the foundation for subsequent software engineering activities
- It describes the function and performance of a computer-based system and the constraints that will govern its development
- It formalizes the informational, functional, and behavioral requirements of the proposed software in both a graphical and textual format

Typical Contents of a Software Requirements Specification

- Requirements
  - Required states and modes
  - Software requirements grouped by capabilities (i.e., functions, objects)
  - Software external interface requirements
  - Software internal interface requirements
  - Software internal data requirements
  - Other software requirements (safety, security, privacy, environment, hardware, software, communications, quality, personnel, training, logistics, etc.)
  - Design and implementation constraints
- Qualification provisions to ensure each requirement has been met
  - Demonstration, test, analysis, inspection, etc.
- Requirements traceability
  - Trace back to the system or subsystem where each requirement applies

Validation Task

- During validation, the work products produced as a result of requirements engineering are assessed for quality

- The specification is examined to ensure that
  - all software requirements have been stated unambiguously
  - inconsistencies, omissions, and errors have been detected and corrected
  - the work products conform to the standards established for the process, the project, and the product

- The formal technical review serves as the primary requirements validation mechanism
  - Members include software engineers, customers, users, and other stakeholders

Questions to ask when Validating Requirements

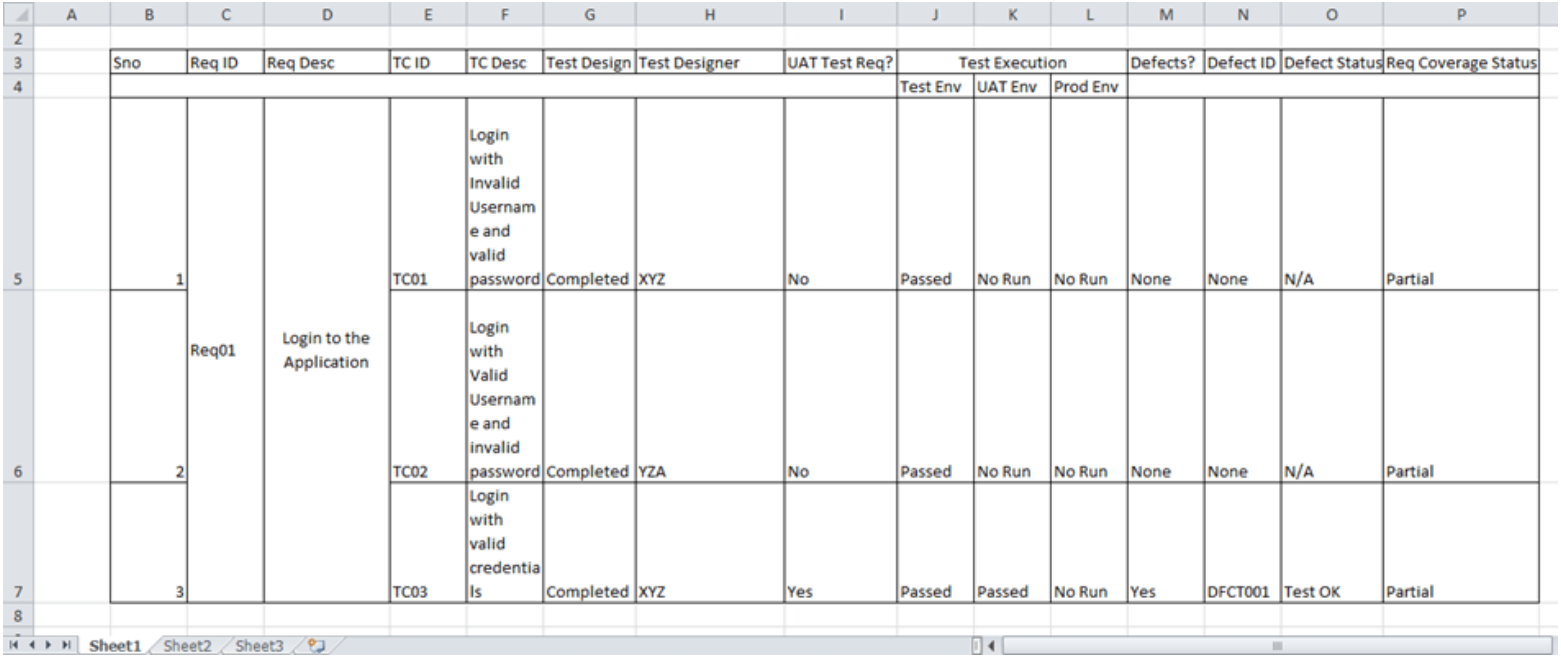
- Is each requirement consistent with the overall objective for the system/product?
- Have all requirements been specified at the proper level of abstraction? That is, do some requirements provide a level of technical detail that is inappropriate at this stage?
- Is the requirement really necessary or does it represent an add-on feature that may not be essential to the objective of the system?
- Is each requirement bounded and unambiguous?
- Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?

Questions to ask when Validating Requirements (continued)

- Do any requirements conflict with other requirements?
- Is each requirement achievable in the technical environment that will house the system or product?
- Is each requirement testable, once implemented?
  - Approaches: Demonstration, actual test, analysis, or inspection
- Does the requirements model properly reflect the information, function, and behavior of the system to be built?
- Has the requirements model been “partitioned” in a way that exposes progressively more detailed information about the system?

Requirements Management Task

- During requirements management, the project team performs a set of activities to identify, control, and track requirements and changes to the requirements at any time as the project proceeds
- Each requirement is assigned a unique identifier
- The requirements are then placed into one or more traceability tables
- These tables may be stored in a database that relate features, sources, dependencies, subsystems, and interfaces to the requirements
- A requirements traceability table is also placed at the end of the software requirements specification



Sno	Req ID	Req Desc	TC ID	TC Desc	Test Design	Test Designer	UAT Test Req?	Test Execution			Defects?	Defect ID	Defect Status	Req Coverage Status
								Test Env	UAT Env	Prod Env				
1	Req01	Login to the Application	TC01	Login with Invalid Username and valid password	Completed	XYZ	No	Passed	No Run	No Run	None	None	N/A	Partial
2			TC02	Login with Valid Username and invalid password	Completed	YZA	No	Passed	No Run	No Run	None	None	N/A	Partial
3			TC03	Login with valid credentials	Completed	XYZ	Yes	Passed	Passed	No Run	Yes	DFCT001	Test OK	Partial

Requirements Management…

https://www.guru99.com/traceability-matrix.html

Traceability tables

- **Features traceability table-** shows how requirements relate to important customer observable system/product features
- **Source traceability table-** identifies the source of each requirement.
- **Dependency traceability table-** indicates how requirements are related to one another
- **Subsystem traceability table-** categories requirements by the subsystem that they govern
- **Interface traceability table-** shows how requirements relate to both internal and external system interfaces

https://www.youtube.com/watch?time_continue=15&v=cm-cSW66Isc&feature=emb_logo