LECTURE 2

Software requirements Engineering

Requirement development and management

Instructor:
Dr. Natalia Chaudhry,
Assistant Professor, PUCIT, University of the Punjab, Lahore.

Recommended readings

- **Chapter 1**, Software Requirements, Wiegers K. &Beatty J., 3rd Ed. Microsoft Press, 2013
- Requirements Engineering, Elizabeth Hull, Ken Jackson and Jeremy Dick. 3rd Ed, Springer-Verlag London Limited, 2011.

Product vs. project requirements

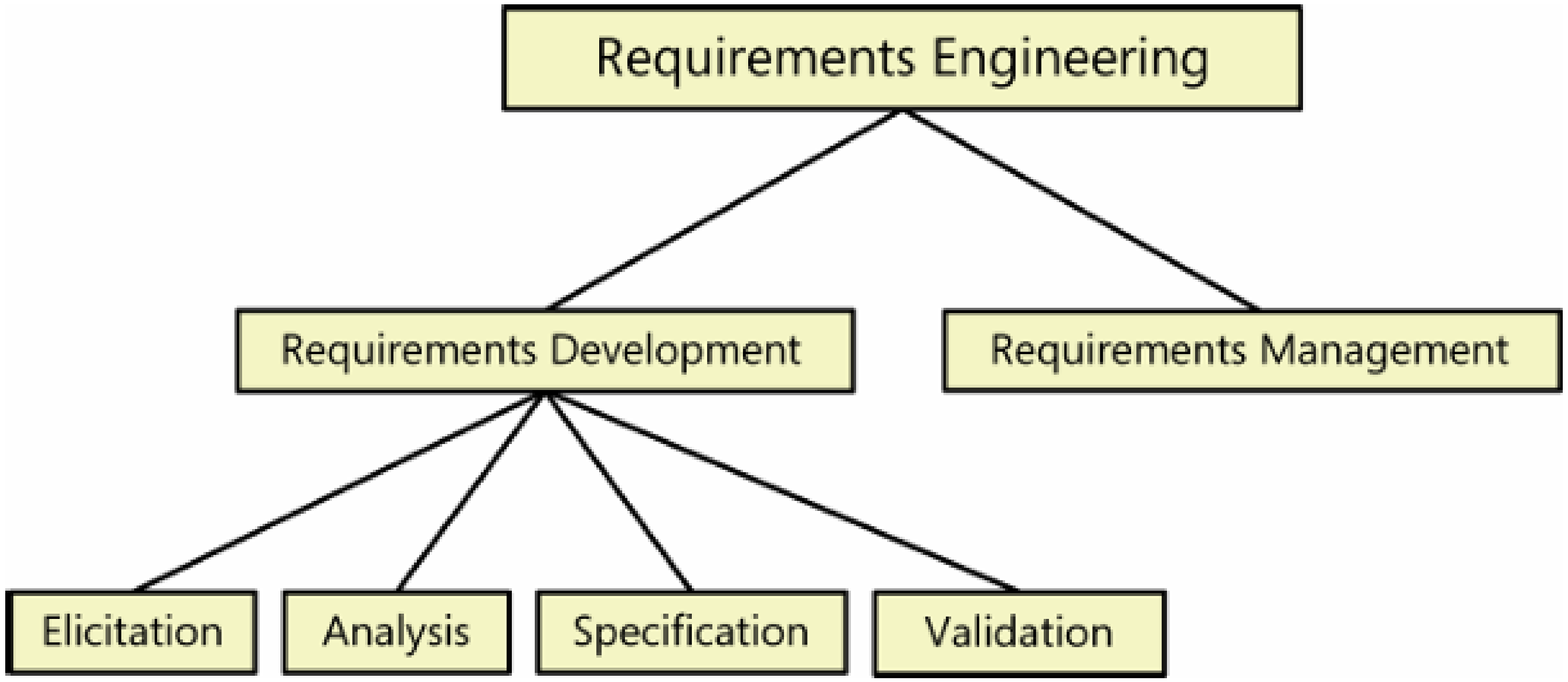
- Projects certainly do have other expectations and deliverables that are not a part of the software the team implements, but that are necessary to the successful completion of the project as a whole.
- These are project requirements but not product requirements.
- An SRS houses the product requirements, but it should not include design or implementation details (other than known constraints), project plans, test plans, or similar information.
- Separate out such items so that requirements development activities can focus on understanding what the team intends to build.

Project requirements include:

- Physical resources the development team needs, such as workstations, special hardware devices, testing labs, testing tools and equipment, team rooms, and videoconferencing equipment.
- Staff training needs.
- User documentation, including training materials, tutorials, reference manuals, and release notes.
- Support documentation, such as help desk resources and field maintenance and service information for hardware devices.
- Infrastructure changes needed in the operating environment.
- Requirements and procedures for releasing the product, installing it in the operating environment, configuring it, and testing the installation.

- Requirements and procedures for transitioning from an old system to a new one, such as data migration and conversion requirements, security setup, production cutover, and training to close skills gaps; these are sometimes called transition requirements
- Product certification and compliance requirements.
- Revised policies, processes, organizational structures, and similar documents.
- Sourcing, acquisition, and licensing of third-party software and hardware components.
- Beta testing, manufacturing, packaging, marketing, and distribution requirements.
- Customer service-level agreements.
- Requirements for obtaining legal protection (patents, trademarks, or copyrights) for intellectual property related to the software.

Subdisciplines of software requirements engineering



Requirements development-> Elicitation

Elicitation encompasses all of the activities involved with discovering requirements, such as interviews, workshops, document analysis, prototyping, and others. The key actions are:

- Identifying the product’s expected user classes and other stakeholders.
- Understanding user tasks and goals and the business objectives with which those tasks align.
- Learning about the environment in which the new product will be used.
- Working with individuals who represent each user class to understand their functionality needs and their quality expectations.

Requirements development-> Analysis

Analyzing requirements involves reaching a richer and more precise understanding of each requirement and representing sets of requirements in multiple ways. Following are the principal activities:

- Analyzing the information received from users to distinguish their task goals from functional requirements, quality expectations, business rules, suggested solutions, and other information
- Decomposing high-level requirements into an appropriate level of detail
- Deriving functional requirements from other requirements information
- Understanding the relative importance of quality attributes
- Allocating requirements to software components defined in the system architecture
- Negotiating implementation priorities
- Identifying gaps in requirements or unnecessary requirements as they relate to the defined scop

Requirements development-> Specification

Requirements specification involves representing and storing the collected requirements knowledge in a persistent and well-organized fashion. The principal activity is:

- Translating the collected user needs into written requirements and diagrams suitable for comprehension, review, and use by their intended audiences

Requirements development-> Validation

Requirements validation confirms that you have the correct set of requirements information that will enable developers to build a solution that satisfies the business objectives. The central activities are:

- Reviewing the documented requirements to correct any problems before the development group accepts them.
- Developing acceptance tests and criteria to confirm that a product based on the requirements would meet customer needs and achieve the business objectives.

Who Should Be Involved in Requirements Elicitation?

Requirements elicitation requires the active involvement of various stakeholders, including:

•**Clients or project sponsors**: They provide the project’s overall goals and vision.

•**End-users:** They represent the individuals who will be using the software and can provide insights into their needs and preferences.

•**Business analysts:** They facilitate the elicitation process, guiding the conversation and documenting the requirements.

•**Subject matter experts:** They possess in-depth knowledge of the domain and can provide valuable input regarding the specific functionalities and constraints.

What Are Common Requirements Elicitation Techniques?

To facilitate effective requirements elicitation, several techniques can be employed, including:

Interviews: Conduct one-on-one sessions with stakeholders to gather detailed information about their needs and expectations.
Workshops: Organize collaborative sessions where stakeholders can engage in discussions and brainstorming activities.
Document analysis: Review existing documentation, such as business plans, user manuals, or process flows, to extract relevant requirements.

Observation: Observe users performing their tasks in their natural environment to understand their workflow and pain points.
Prototyping: Create interactive prototypes to gather feedback and validate requirements with stakeholders.

JIRA

Jira is a very popular tool for agile project management and bug tracking

Capture Requirements

You can identify project goals as high-level features and describe them in Jira issues of type **Epic** stored in the product Backlog. Break down epics into smaller chunks of functionality. Describe them in Jira issues of type **Story** using simple statements with the following structure:

As a <type of user>, I want <some goal>, so that <some reason>.

- Specify also clear acceptance criteria making each story testable.
- Prioritize stories using issue field Priority.
- Note that Jira assigns each user story a unique ID automatically.

What are stories, epics, and initiatives?

- Stories, also called “user stories,” are short requirements or requests written from the perspective of an end user.
- Epics are large bodies of work that can be broken down into a number of smaller tasks (called stories).
- Initiatives are collections of epics that drive toward a common goal.

Initiative: Digital Banking Transformation

Epic: Enhanced Security Features

1. **Two-Factor Authentication (Requirement):** Implement two-factor authentication (2FA) for user logins to enhance account security.
    1. Description: Users will be required to enter a one-time code sent to their registered mobile number or email after entering their password during login.
2. **Biometric Authentication (Requirement):** Integrate biometric authentication methods such as fingerprint or face recognition for user authentication.
    1. Description: Users can opt to use their device's biometric features for a more convenient and secure login experience.
3. **Device Authorization (Requirement):** Implement device authorization checks to ensure that only registered devices can access the mobile banking app.
    1. Description: Users will need to authorize each new device they use for accessing their account, adding an extra layer of security.

A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer.

User story template and examples

User stories are often expressed in a simple sentence, structured as follows:

“As a [persona], I [want to], [so that].”

Breaking this down:

•"As a [persona]": Who are we building this for? We’re not just after a job title, we’re after the persona of the person. Max. Our team should have a shared understanding of who Max is. We’ve hopefully interviewed plenty of Max’s. We understand how that person works, how they think and what they feel. We have empathy for Max.

•“Wants to”: Here we’re describing their intent — not the features they use. What is it they’re actually trying to achieve? This statement should be implementation free — if you’re describing any part of the UI and not what the user goal is you're missing the point.

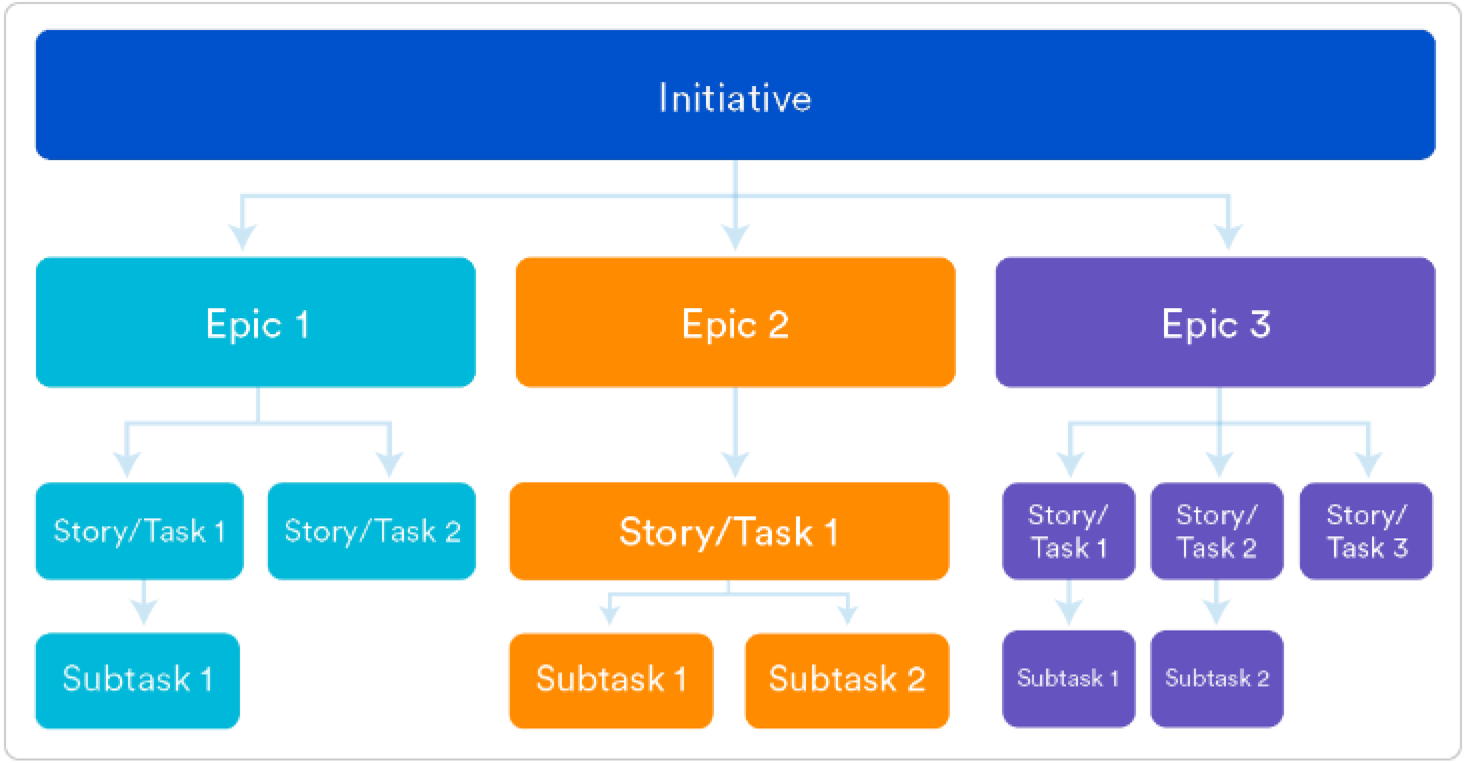
•“So that”: how does their immediate desire to do something this fit into their bigger picture? What’s the overall benefit they’re trying to achieve? What is the big problem that needs solving?

For example, user stories might look like:

- As Max, I want to invite my friends, so we can enjoy this service together.
- As Sascha, I want to organize my work, so I can feel more in control.
- As a manager, I want to be able to understand my colleagues progress, so I can better report our success and failures.

Epics are large bodies of work that can be broken down into a number of smaller tasks (called stories).

An epic is a large body of work that can be broken down into a number of smaller stories, or sometimes called "Issues" in Jira. Epics often encompass multiple teams, on multiple projects, and can even be tracked on multiple boards.



Let’s say it’s 2050 and we work for a recreational space-travel organization. We do about a dozen launches a year, so each launch isn’t the single biggest thing we do in a year, but it’s still far from routine and will take many person-hours to complete. That sizing is just right for an epic.

An example epic, “March 2050 Space Tourism Launch” includes stories for routine work items as well as stories aimed to improve key aspects of the shuttle launch, from customers buying space travel tickets to the launch of the rocket itself. As such, multiple teams will contribute to this epic by working on a wide range of stories.

The software team supporting the purchasing of tickets for the March 2050 launch might structure their epic as so:

Epic: March 2050 Launch
Story: Update date range to include March 2050 Launch dates.
Story: Reduce load time for requested flight listings to < 0.45 seconds
Story: Promote Saturn Summer Sale on confirm page for First Class bookings.

Concurrently, the propulsion teams might contribute to the same epic with these stories:

Epic: March 2050 Launch
Story: Keep fuel tanks PSI > 250 PPM on launch
Story: Reduce overall fuel consumption by 1%.

Example 1: E-commerce Website

User Story: As a shopper, I want to be able to add items to my cart so that I can purchase them later.

Acceptance Criteria:

1. I should be able to see an "Add to Cart" button next to each product.
2. Clicking the "Add to Cart" button should add the selected item to my cart.
3. I should see a confirmation message confirming that the item has been added to my cart.
4. The total number of items in my cart should be updated accordingly.

Example 2: Ride-Sharing App

User Story: As a commuter, I want to be able to see nearby available rides so that I can quickly book a ride to my destination.

Acceptance Criteria:

1. There should be a map interface showing nearby drivers and their availability.
2. I should be able to enter my current location and desired destination.
3. Upon entering the destination, I should see a list of available drivers nearby.
4. Each driver should display their rating, estimated time of arrival, and fare.
5. I should be able to select a driver and book a ride with a single click.

Example:

Unstructured Note: "Users are complaining about the checkout process being too lengthy. They find it frustrating to enter their payment details every time they make a purchase."

User Story: "As a frequent shopper, I want a one-click checkout option so that I can complete my purchases more quickly and easily."

Unstructured Note: "Users have reported issues with the search functionality. Some say the results are irrelevant, while others find it difficult to filter and sort through the large number of matches. Additionally, users want to be able to save their search criteria for future reference."

Step 1: Review the Notes

Users are experiencing problems with the search functionality, including irrelevant results and difficulty filtering and sorting. They also desire the ability to save search criteria.

Step 2: Identify Actors

Primary actors include general users who utilize the search functionality.

Step 3: Define Goals

Users want to conduct effective searches, receive relevant results, easily filter and sort through matches, and save their search criteria for future use.

Step 4: Break Down into Features

1. Improved relevance of search results.
2. Enhanced filtering and sorting options.
3. Ability to save and recall search criteria.

Step 5: Write User Stories

1. As a user, I want search results to be more relevant so that I can find what I'm looking for more quickly.
2. As a user, I want to be able to filter and sort search results by various criteria (e.g., relevance, date, category) to narrow down my options.
3. As a user, I want the option to save my search criteria so that I can easily repeat a search without re-entering the same parameters.

Step 6: Add Acceptance Criteria

1.Improved Relevance:

1. Search results should prioritize exact matches over partial matches.
2. Results should be based on relevance to the search query and user preferences if available.
3. Users should not encounter irrelevant matches within the first page of results.

2.Enhanced Filtering and Sorting:

1. Users should be able to filter search results by category, date, relevance, and other relevant criteria.
2. Sorting options should include relevance, date, and alphabetical order.
3. Users should be able to apply multiple filters simultaneously.

3.Save Search Criteria:

1. Users should see a "Save Search" button next to the search bar.
2. Clicking the "Save Search" button should open a dialog where users can name and save their search criteria.
3. Saved searches should be accessible from a dedicated section of the user's profile.

Step 7: Iterate and Refine

Gather feedback from stakeholders to refine the user stories further, ensuring they accurately reflect user needs and expectations.

After user stories..

After collecting these user stories, the next step is to analyze them to extract the essential information and ensure they are clear, feasible, and aligned with the project goals.

User Story:

As a customer, I want to be able to track my orders so that I know when they will arrive.

Analysis:

1. **Identify Stakeholders:**
    1. Customers: Primary users who want to track their orders.
    2. Customer Service Representatives: May need access to order tracking information to assist customers.
2. **Clarify Acceptance Criteria:**
    1. What details do customers need to see when tracking orders? (e.g., order status, expected delivery date, tracking number)
    2. Should customers receive notifications when the order status changes?
    3. Is there any sensitive information that shouldn't be displayed during tracking?
3. **Assess Impact and Feasibility:**
    1. Is the necessary data available within the system to provide order tracking?
    2. Are there any technical limitations or dependencies that could affect implementation?
    3. How will this feature integrate with existing systems or processes?

4. Iterate and Refine:

1. Review the analysis with stakeholders to validate assumptions and address any concerns or questions.
2. Incorporate feedback into the user stories and acceptance criteria as necessary.

5. Document Decisions:

1. Record key decisions made during the analysis process, including rationale and any trade-offs considered.
2. Ensure that the documentation is accessible to all team members for future reference.

By thoroughly analyzing user stories after they've been elicited, you can ensure that the requirements are well-understood and can serve as a solid foundation for the development process.

Requirements development-> Specification

Requirements specification involves representing and storing the collected requirements knowledge in a persistent and well-organized fashion. The principal activity is:

- Translating the collected user needs into written requirements and diagrams suitable for comprehension, review, and use by their intended audiences

Demo Example: Specification after Analysis

Project Name: Online Shopping Website

1. Functional Requirements:

- **User Registration and Authentication:**
  - Users can register with the site, providing necessary details.
  - Registered users can log in securely using credentials.
  - Passwords must be securely stored using hashing algorithms.
- **Product Browsing and Search:**
  - Users can browse products by category or search using keywords.
  - Search results should be relevant and sorted by relevance.
  - Filters such as price range, brand, and ratings should be available.
- **Adding to Cart and Checkout:**
  - Users can add items to their shopping cart.
  - Cart should display total price and allow quantity updates.
  - Users can proceed to checkout, providing shipping and payment details.
  - Secure payment gateway integration for processing transactions.

2. Non-Functional Requirements:

- **Performance:**
  - Web pages should load within 3 seconds on average.
  - System should handle concurrent user loads during peak hours without significant degradation in performance.
- **Security:**
  - All communications should be encrypted using HTTPS.
  - User data should be stored securely with appropriate access controls.
  - Regular security audits and updates to prevent vulnerabilities.
- **Usability:**
  - Intuitive user interface design with clear navigation.
  - Mobile-responsive design for seamless browsing on various devices.
  - Accessibility features for users with disabilities, complying with WCAG guidelines.

3. System Architecture:

•**Frontend:** ReactJS for dynamic UI rendering, with responsive design principles.

•**Backend:** Node.js for server-side logic, Express.js for RESTful APIs.

•**Database:** MongoDB for flexible data storage, with Mongoose ORM for schema management.

•**Hosting:** AWS EC2 for scalable hosting, with CDN integration for content delivery.

4. User Stories:

•As a user, I want to be able to save items to my wishlist for future purchase.

•As a user, I want to receive email notifications for order confirmation and shipment updates.

•As an admin, I want to view and manage inventory levels for each product.

This specification document would be the result of thorough analysis and collaboration with stakeholders, ensuring that all requirements are clearly defined and aligned with the project goals.

Requirements development-> Validation

Requirements validation confirms that you have the correct set of requirements information that will enable developers to build a solution that satisfies the business objectives. The central activities are:

- Reviewing the documented requirements to correct any problems before the development group accepts them.
- Developing acceptance tests and criteria to confirm that a product based on the requirements would meet customer needs and achieve the business objectives.

Requirement Validation Process:

1. **Review with Stakeholders**: Present the specification document to stakeholders for review and validation. This ensures that their expectations and requirements are accurately captured.
2. **Prototyping and Mockups**: Develop prototypes or mockups based on the specification to provide stakeholders with a tangible representation of the proposed solution. Gather feedback on these prototypes to validate if they align with stakeholder expectations.
3. **User Acceptance Testing (UAT)**: Conduct UAT sessions where representative users interact with the system based on the specification. Their feedback and observations help validate if the system meets their needs and expectations.
4. **Requirement Traceability Matrix (RTM)**: Create an RTM to map each requirement in the specification document to corresponding test cases. This ensures that all requirements are tested and validated during the testing phase.
5. **Cross-Functional Reviews**: Engage stakeholders from different departments or teams to review the specification document. This helps ensure that all aspects of the system, such as functionality, usability, security, and performance, are adequately addressed.
6. **Prototyping and Mockups**: Develop prototypes or mockups based on the specification to provide stakeholders with a tangible representation of the proposed solution. Gather feedback on these prototypes to validate if they align with stakeholder expectations.

Requirement Validation Process:

1. **Review with Stakeholders**: Present the specification document to stakeholders for review and validation. This ensures that their expectations and requirements are accurately captured.
2. **Prototyping and Mockups**: Develop prototypes or mockups based on the specification to provide stakeholders with a tangible representation of the proposed solution. Gather feedback on these prototypes to validate if they align with stakeholder expectations.
3. **User Acceptance Testing (UAT)**: Conduct UAT sessions where representative users interact with the system based on the specification. Their feedback and observations help validate if the system meets their needs and expectations.

Requirement Traceability Matrix (RTM): Create an RTM to map each requirement in the specification document to corresponding test cases. This ensures that all requirements are tested and validated during the testing phase.

Cross-Functional Reviews: Engage stakeholders from different departments or teams to review the specification document. This helps ensure that all aspects of the system, such as functionality, usability, security, and performance, are adequately addressed.

Prototyping and Mockups: Develop prototypes or mockups based on the specification to provide stakeholders with a tangible representation of the proposed solution. Gather feedback on these prototypes to validate if they align with stakeholder expectations.

Demo Example: Validation After Specification

Project Name: Online Shopping Website

Validation Steps Taken:

1. **Stakeholder Review**: The specification document was circulated among stakeholders, including end-users, developers, and managers. Feedback was collected through meetings and email communication, and revisions were made accordingly to address any discrepancies or misunderstandings.
2. **Prototype Demonstration**: A clickable prototype was developed based on the specification, showcasing key functionalities such as user registration, product browsing, and checkout process. Stakeholders interacted with the prototype and provided feedback on the user interface, workflow, and feature implementation.
3. **User Acceptance Testing (UAT)**: Representative users were invited to participate in UAT sessions where they tested the system based on predefined test scenarios derived from the specification. Feedback was collected on aspects such as ease of use, performance, and adherence to requirements.

Requirement Traceability Matrix (RTM): An RTM was created to ensure that each requirement listed in the specification was covered by corresponding test cases during the testing phase. This helped track the validation status of each requirement and ensure comprehensive test coverage.

Cross-Functional Reviews: Cross-functional review sessions were conducted involving stakeholders from different departments, including marketing, IT, and customer support. This ensured that the specification addressed the needs and concerns of all relevant parties and that the proposed solution was aligned with the overall business objectives.

By following these validation steps, the specification was thoroughly vetted and validated to ensure that it accurately captured the requirements and expectations of stakeholders, laying a solid foundation for the development phase.

Requirements management

Requirements management activities include the following:

- Defining the requirements baseline, a snapshot in time that represents an agreed-upon, reviewed, and approved set of functional and nonfunctional requirements, often for a specific product release or development iteration
- Evaluating the impact of proposed requirements changes and incorporating approved changes into the project in a controlled way
- Keeping project plans current with the requirements as they evolve
- Negotiating new commitments based on the estimated impact of requirements changes
- Defining the relationships and dependencies that exist between requirements
- Tracing individual requirements to their corresponding designs, source code, and tests
- Tracking requirements status and change activity throughout the project

Key Differences

Focus:

Requirement development is concerned with identifying and defining what the system needs to achieve and specifying the features and functionalities.

Requirement management is concerned with organizing, documenting, and controlling changes to requirements throughout the project lifecycle.

Activities:

Requirement development involves activities such as gathering, analyzing, and documenting requirements.

Requirement management involves activities such as establishing baselines, tracking changes, and ensuring traceability.

Timing:

Requirement development occurs early in the project lifecycle, often during the planning and initiation phases.

Requirement management is an ongoing process that continues throughout the project lifecycle to handle changes and updates.

Purpose:

The purpose of requirement development is to understand and define what needs to be built.

The purpose of requirement management is to ensure that the defined requirements are well-managed, consistent, and align with the project's goals.

most common requirements risks

Insufficient user involvement

Customers often don’t understand why it is so essential to work hard on eliciting requirements and assuring their quality. Developers might not emphasize user involvement, perhaps because they think they already understand what the users need

Insufficient user involvement leads to late-breaking requirements that generate rework and delay completion.

Inaccurate planning

Vague, poorly understood requirements lead to overly optimistic estimates, which come back to haunt you when the inevitable overruns occur.

The top contributors to poor software cost estimation are frequent requirements changes, missing requirements, insufficient communication with users, poor specification of requirements, and insufficient requirements analysis

Estimating project effort and duration based on requirements means that you need to know something about the size of your requirements and the development team’s productivity.

Creeping user requirements

As requirements evolve during development, projects often exceed their planned schedules and budgets

As new requirements come along, they are placed into the backlog of pending work and allocated to future iterations based on priority.

Change might be critical to success, but change always has a price.

Ambiguous requirements

One symptom of ambiguity in requirements is that a reader can interpret a requirement statement in several ways

Another sign is that multiple readers of a requirement arrive at different understandings of what it means.

Ambiguous requirements cause wasted time when developers implement a solution for the wrong problem. Testers who expect the product to behave differently from what the developers built waste time resolving the differences.

Ambiguous requirements

Ambiguous requirements cause wasted time when developers implement a solution for the wrong problem. Testers who expect the product to behave differently from what the developers built waste time resolving the differences.

Requirements engineering tasks have potential payoff including:

- Fewer defects in requirements and in the delivered product.
- Reduced development rework.
- Faster development and delivery.
- Fewer unnecessary and unused features.
- Lower enhancement costs.
- Fewer miscommunications.
- Reduced scope creep.
- Reduced project chaos.
- Higher customer and team member satisfaction.
- Products that do what they’re supposed to do.

Assignment 1

- Explain how Jira can be utilized to establish and maintain effective end-to-end requirements traceability.
- Provide a step-by-step guide or outline of how you would implement and manage traceability within a Jira project.

- Discuss the different types of links available in Jira and their respective purposes.
- Demonstrate how to create and manage links between requirements using Jira, including practical steps.

Deadline: 6 feb, 2024, till 11:59 pm

That’s it