Cavalier Institute - https://cavalierinstitutions.com

Date	Sep-19-2024	Session No	4

Topic : Requirements Engineering / Questions / Scenarios

Exercise on Requirements Engineering

4.1 Identify and briefly describe four types of requirements that may be defined for a computer-based system.

4.2 Discover ambiguities or omissions in the following statement of requirements for part of a ticket-issuing system:

An automated ticket-issuing system sells rail tickets. Users select their destination and input a credit card and a personal identification number. The rail ticket is issued and their credit card account charged. When the user presses the start button, a menu display of potential destinations is activated, along with a message to the user to select a destination. Once a destination has been selected, users are requested to input their credit card. Its validity is checked and the user is then requested to input a personal identifier. When the credit transaction has been validated, the ticket is issued.

4.3 Rewrite the above description using the structured approach described in this chapter. Resolve the identified ambiguities in an appropriate way.

4.4 Write a set of non-functional requirements for the ticket-issuing system, setting out its expected reliability and response time.

4.5 Using the technique suggested here, where natural language descriptions are presented in a standard format, write plausible user requirements for the following functions:

- An unattended petrol (gas) pump system that includes a credit card reader. The customer swipes the card through the reader, then specifies the amount of fuel required. The fuel is delivered, and the customer's account debited.
- The cash-dispensing function in a bank ATM.
- The spelling-check and correcting function in a word processor.

4.6 Suggest how an engineer responsible for drawing up a system requirements specification might keep track of the relationships between functional and non-functional requirements.

4.7 Using your knowledge of how an ATM is used, develop a set of use cases that could serve as a basis for understanding the requirements for an ATM system.

4.8 Who should be involved in a requirements review? Draw a process model showing how a requirements review might be organized.

4.9 When emergency changes have to be made to systems, the system software may have to be modified before changes to the requirements have been approved. Suggest a model of a process for making these modifications that will ensure that the requirements document and the system implementation do not become inconsistent.

4.10 You have taken a job with a software user who has contracted your previous employer to develop a system for them. You discover that your company's interpretation of the requirements is different from the interpretation taken by your previous employer. Discuss what you should do in such a situation. You know that the costs to your current employer will increase if the ambiguities are not resolved. However, you also have a responsibility of confidentiality to your previous employer.

4.1. Identify and briefly describe four types of requirements that may be defined for a computer-based system.

Scenario:

When defining requirements for a computer-based system, various types of requirements may need to be identified and described to ensure a clear understanding of what the system must accomplish. These include functional requirements, non-functional requirements, domain requirements, and user requirements.

Solutions:

1. **Functional Requirements**: These specify the specific behaviors or functions the system must support. They describe what the system should do, such as generating reports, calculating totals, or processing transactions.
    - *Example*: The system should allow users to select a rail destination and issue tickets.

2. **Non-Functional Requirements**: These specify criteria that judge the system's operation, rather than specific behaviors. Non-functional requirements focus on qualities such as performance, security, usability, and reliability.
    - *Example*: The system must process a transaction in under 2 seconds.
3. **Domain Requirements**: These originate from the system's operating environment and may be specific to the domain in which the system will operate. They might impose specific constraints or rules.
    - *Example*: In a rail ticketing system, the system should follow the rail network's pricing rules and regulations.
4. **User Requirements**: These are statements in natural language that express what the system must provide to fulfill the user's goals. They are usually written from the user's perspective.
    - *Example*: The user must be able to select a destination from a displayed list of options.

4.2. Discover ambiguities or omissions in the following statement of requirements for part of a ticket-issuing system:

Scenario:

The system description mentions an automated ticket-issuing system that sells rail tickets, but it contains several ambiguities and omissions.

- "An automated ticket-issuing system sells rail tickets."
- "Users select their destination and input a credit card and a personal identification number."
- "The rail ticket is issued, and their credit card account charged."
- "When the user presses the start button, a menu display of potential destinations is activated, along with a message to the user to select a destination."

Solutions:

1. **Ambiguity**: It is not specified what happens if the card is invalid or the transaction is declined.
    - **Omission**: There is no mention of error handling for incorrect input (e.g., wrong PIN or card).
2. **Ambiguity**: It is unclear if the system supports multiple languages or if the user can cancel the operation at any stage.
    - **Omission**: There is no clarity on how the system handles input errors or invalid destinations.
3. **Ambiguity**: The description does not explain what happens after the transaction fails—does the user retry or get a refund if the card is charged incorrectly?

    - **Omission**: No mention of the timing between steps (e.g., how long the system waits for input before canceling the operation).
4. **Ambiguity**: The statement doesn't specify if users are presented with a confirmation screen after selecting the destination or after entering credit card information.

4.3. Rewrite the above description using the structured approach described in this chapter.

Scenario:

The initial description is ambiguous and lacks structure. A better-structured description should clarify these ambiguities and omissions.

Solutions:

1. **Start Button**: When the user presses the start button, the system activates a menu that displays potential rail destinations. A message prompts the user to select a destination from the displayed list.
2. **Destination Selection**: Once the user selects a destination, the system requests the user to input a credit card.
3. **Card Validation**: The system checks the validity of the credit card. If the card is invalid, the user is informed, and the transaction is canceled. If valid, the system prompts the user to input their personal identification number (PIN).
4. **PIN Validation**: Upon entering the PIN, the system validates the personal identifier. If the identifier is invalid, the user is prompted to retry. If valid, the transaction is processed.
5. **Ticket Issuance**: Once the transaction is successfully processed, the ticket is printed, and the user's account is debited. If the transaction fails at any step, an error message is displayed, and no ticket is issued.
6. **Cancellation**: At any point, the user should be able to cancel the transaction.

4.4. Write a set of non-functional requirements for the ticket-issuing system, setting out its expected reliability and response time.

Scenario:

Non-functional requirements set performance, reliability, and quality benchmarks for the system to ensure it functions as expected under real-world conditions.

Solutions:

1. **Reliability**: The system must be available 99.9% of the time and have less than 1% transaction failure rate.
2. **Response Time**: The system should respond to any user input (e.g., button presses or card entry) within 2 seconds.
3. **Security**: The system must comply with PCI DSS (Payment Card Industry Data Security Standard) for handling credit card transactions and personal identification numbers.
4. **Capacity**: The system should handle up to 500 transactions per hour without a degradation in performance.
5. **Maintainability**: The system should be easy to update, with system downtime for updates limited to non-peak hours (between 12 AM and 4 AM).

4.5. Write plausible user requirements using a standard format.

Scenario:

Here are plausible user requirements for three systems based on their functionalities.

Solutions:

1. **Unattended Petrol Pump System**:
    - **Description**: The system allows customers to purchase fuel without assistance by using a credit card.
    - **User Action**: The customer swipes the card and inputs the desired amount of fuel.
    - **System Action**: The system verifies the card and dispenses the fuel. Once the delivery is complete, the customer's account is debited.
2. **Bank ATM Cash-Dispensing Function**:
    - **Description**: The system allows users to withdraw cash from their bank accounts.
    - **User Action**: The user inserts their card, inputs their PIN, and selects the amount of cash to withdraw.
    - **System Action**: The system verifies the card, checks the account balance, and dispenses the requested cash.
3. **Word Processor Spelling-Check Function**:
    - **Description**: The spelling check automatically identifies and highlights spelling errors in a document.
    - **User Action**: The user writes a document, and the system checks spelling in real-time or when prompted.
    - **System Action**: The system underlines misspelled words, suggests corrections, and allows the user to apply the correction.

4.6. Suggest how an engineer might track the relationships between functional and non-functional requirements.

Scenario:

Tracking relationships between functional and non-functional requirements is crucial to ensure consistency and coverage in system development.

Solutions:

1. **Traceability Matrix**: Engineers can use a traceability matrix that maps each functional requirement to its corresponding non-functional requirements, ensuring that all aspects of performance, reliability, and security are addressed for each feature.
2. **Requirement Management Tools**: Tools like Jira, IBM DOORS, or Rational RequisitePro can be used to link functional requirements with their non-functional counterparts.

4.7. Develop a set of use cases for an ATM system.

Scenario:

Understanding ATM use cases can help in defining the system requirements.

Solutions:

- **Use Case 1**: Withdraw Cash
  - User inserts card, enters PIN, selects withdrawal option, chooses amount, and collects cash.
- **Use Case 2**: Check Balance
  - User inserts card, enters PIN, and selects the balance inquiry option.
- **Use Case 3**: Deposit Cash
  - User inserts card, enters PIN, selects the deposit option, inserts cash, and confirms.

4.8. Who should be involved in a requirements review?

Scenario:

A thorough review of system requirements requires input from various stakeholders to ensure completeness and accuracy.

Solutions:

1. **Review Participants**: Project managers, developers, testers, domain experts, customers, and end-users should all be involved in the review process.
2. **Process Model**:
    - Collect requirements.
    - Review by stakeholders.
    - Conduct feedback sessions.
    - Make revisions.
    - Final approval.

4.9. Suggest a model of a process for emergency software changes.

Scenario:

In some cases, emergency changes to system software must be made without updating the requirements first.

Solutions:

1. **Model**:
    - Log the emergency change request.
    - Perform an impact analysis.
    - Implement and test the change.
    - Update the requirements document post-implementation.
    - Conduct a review to ensure consistency.

4.10. What should you do when there is a disagreement in requirement interpretation?

Scenario:

When two companies interpret requirements differently, there can be cost and confidentiality implications.

Solutions:

- **Step 1**: Report the ambiguity to both employers, emphasizing the need for clarity.
- **Step 2**: Facilitate a discussion to resolve the ambiguity while maintaining confidentiality.
- **Step 3**: Document the final agreed-upon requirements to avoid future misunderstandings.

END