**SER 415 Reflection Paper 1 Rough draft, suggestions, and ideas**

Reflection 1 - Requirements Analysis

Here, we will be constructing some requirements analysis, and then reflecting on the process and issues you encountered.

Please complete the following scenarios on your own. You can use scratch paper or do it digitally. You do not need to submit your diagrams or specifications. Remember, with these there is not a specific answer so don’t get discouraged if you feel stuck, just try your best and write about your issues in the reflection.

Once you have done all of these scenarios, please read and complete the reflection questions at the end of this document. You will then submit that reflection on Canvas as your assignment.

Problem 1 (ER Model):

“At Acme Corporation, we ask that our employees track their stats while working. These are: time on task, break time, lines of code written, time spent debugging, time spent writing tests, test pass percentage, and number of commits to the repository. Each employee also has identifying data about them such as their name, ID number, and job title. Some of these data points can be automated so that employees can spend more time working, and less time filling out forms. We would like to have two separate systems, one that auto tracks data, and a second one that employees fill out manually.”

Problem 2 (Abstract Class Diagram):

“Our app is an augmented reality game about collecting monsters from around the world, by having the player actually move in the real world to find these monsters. Each player has an ID and a list of monsters that they own. Players can trade with each other to obtain monsters they do not already have. Each of these trades involves 1 monster from each player, and each trade has its own ID in case we need to track monsters we later determine were obtained through hacking. Each monster has a name, the ID of the player who caught it, and a list of all trades it has been through.”

Problem 3 (State Machine):

“Within our Git system, individual commits go through a series of checks and updates to ensure that they are acceptable. When a commit is submitted, a series of pre-commit tests are run and if any of them fail, the employee is expected to edit and resubmit the commit. After all of those pass, the commit is then put up for review by the rest of the team. Those other team members can comment on the changes being committed, and the author needs to reply to every comment. They can also make changes to the commit based on the comments and then resubmit it, meaning it goes through the pre-commit checks again. Once all the team members have approved the commit and all comments have been replied to, the commit can be pushed to the repository. It is then deployed to the server. This involves the server being taken offline, and then the code is freshly installed on the server (this can take up to an hour!). Afterwards, the server is brought back online, and the integration tests are run. If those fail, the commit is automatically rolled back, and the author needs to edit and resubmit the code, starting with the pre-commit checks again. Once the integration tests pass, the commit has been successfully applied and the coding task is done.”

Reflection Assignment

Please answer the following questions. For each question, be sure to provide a full and well thought out explanation of your process. Your goal here is to spend some time reflecting on the activity you just completed, not just providing quick answers to the questions.

1. In Problem 1, do you feel that the ER diagram is helpful when trying to understand the system or would you prefer a class diagram?
2. In Problem 2, did you feel there was enough information to complete a meaningful diagram and if not what information would you have liked to have?
3. In Problem 3, did you feel that a state machine was an appropriate diagram for this situation or do you think a different diagram would have been more helpful?
4. Overall, is there anything not already covered that you encountered during the activity that surprised you about doing requirements analysis? (Such as difficulty creating certain types of diagrams, always feeling like you don’t have enough information, or anything similar?)

**Problem 1: ER Model**

**Scenario:**

* You need to create an ER (Entity-Relationship) diagram for a system at Acme Corporation where employees track various stats such as time on task, lines of code written, time spent debugging, etc. The company wants to have two separate systems: one that auto-tracks data and another that employees fill out manually.

**Approach:**

1. **Entities Identification:**
   * **Employee**: Name, ID number, job title.
   * **Stats**: Time on task, break time, lines of code, debugging time, testing time, test pass percentage, number of commits.
   * **Systems**: Auto-tracking system and manual input system.
2. **Relationships:**
   * Employees have stats that are either auto-tracked or manually entered.
   * Each stat is associated with one or more employees.
   * The systems (auto-tracking and manual) are linked to the stats they track.
3. **ER Diagram Creation:**
   * Draw entities (Employee, Stat, System).
   * Define relationships (e.g., "Tracks" between System and Stat, "Records" between Employee and Stat).
   * Establish cardinality (e.g., One Employee has many Stats, One System tracks many Stats).

**Reflection:** Consider how well the ER diagram helps in understanding the system compared to a class diagram.

**Problem 2: Abstract Class Diagram**

**Scenario:**

* You’re asked to create an abstract class diagram for an augmented reality game where players collect monsters. Each player has an ID and a list of monsters, and players can trade monsters with each other. Each trade has an ID and involves one monster from each player.

**Approach:**

1. **Class Identification:**
   * **Player**: ID, list of monsters.
   * **Monster**: Name, ID of the player who caught it, trade history.
   * **Trade**: Trade ID, involves one monster from each player.
2. **Relationships and Methods:**
   * A Player has a list of Monsters.
   * A Trade involves two Monsters, one from each Player.
   * Methods may include trading, adding monsters, removing monsters, etc.
3. **Class Diagram Creation:**
   * Define classes (Player, Monster, Trade).
   * Outline methods and attributes.
   * Establish relationships (e.g., Player “owns” Monster, Trade “involves” Monster).

**Reflection:** Reflect on whether enough information was provided to complete a meaningful diagram and what additional information would have been helpful.

**Problem 3: State Machine**

**Scenario:**

* You need to create a state machine diagram for a Git system where commits go through various checks before being accepted. Commits can be resubmitted, reviewed, commented on, and eventually deployed.

**Approach:**

1. **State Identification:**
   * **Initial State**: Commit is submitted.
   * **States**: Pre-commit testing, review, comment resolution, approval, deployment, rollback (if integration tests fail).
   * **Final State**: Commit is successfully applied.
2. **Transitions:**
   * Transition from one state to another based on actions like test pass/fail, comments, approvals, or integration test results.
3. **State Machine Diagram Creation:**
   * Start with the initial state (commit submitted).
   * Map out each subsequent state and the conditions that cause the transition (e.g., all tests pass, all comments resolved).
   * Define end states (commit applied or rolled back).

**Reflection:** Consider whether a state machine was the appropriate diagram for this situation or if another type of diagram would have been more helpful.

**Reflection Assignment**

**Questions to Consider:**

1. **In Problem 1, is the ER diagram helpful in understanding the system, or would you prefer a class diagram?**
   * Reflect on the strengths and weaknesses of the ER diagram in conveying the relationships and data structure compared to a class diagram.
2. **In Problem 2, was there enough information to complete a meaningful diagram?**
   * Discuss whether the provided details were sufficient for building an abstract class diagram, and if not, what additional information would have been beneficial.
3. **In Problem 3, was a state machine appropriate, or would another diagram have been more helpful?**
   * Evaluate the appropriateness of a state machine for modeling the commit process and consider alternatives that might better capture the system's behavior.
4. **Overall Reflection:**
   * Reflect on any challenges encountered during the activity, such as difficulties in creating certain diagrams, feeling like you didn’t have enough information, or anything surprising about the requirements analysis process.

**Next Steps:**

* Go through each scenario and create the respective diagrams (even if just mentally or on scratch paper).
* Reflect on the process of creating these diagrams and how well they helped you understand the system.
* Answer the reflection questions thoroughly, considering the issues and insights you encountered during the analysis.