Jose Grijalva

8/26/2024

SER 415 Fall A 2024 Online

Reflection #1

1) When constructing an Entity-Relationship (ER) Diagram, you only focus on the entities, relationships, values, and attributes of any given problem. In this problem, Acme Corporation is concerned about their employees tracking all their stats of time spent on various work-related things, the test past percentage, the number of commits to a repository, and the number of lines of code written. ER Diagrams are helpful in this case as this type of tracking system would require solid documentation to provide descriptions of crucial information like organized very specific sets of data. This is a rather database-centric problem so I would prefer an ER diagram over a Class Diagram. Class Diagrams tend to emphasize more on unnecessary details like hierarchies and polymorphisms, which isn't Acme Corporation's focus on their employees. ER diagrams would make more sense than Class Diagrams for the provided reasons.

2) After sketching the abstract class diagram, I see a lot of clear details of this augmented reality game based on all of the provided information for this problem. With all of the operations inside the classes of the abstract class diagram, I could see all of the behavioral information for the players and monsters. I could also tell how important the visibility is of certain attributes like IDs as players should be able to see each other's IDs

but they shouldn’t have access to specific monsters if they did not capture them. The players also should not see each other’s trade history. The Abstract Class diagram is great for the security of the attributes for classes. If there is any information I would like to add, it would be a validation rule for the trading mechanism, where the players would have to agree to trade before the trading is executed or else any player can make unauthorized trades.

3) After thoroughly reading through the git system problem and then sketching the state machine diagram for it, it makes perfect sense that this would be the most appropriate diagram to utilize for this situation. The way a git repository functions includes branches, statuses, and directories consolidated into a file system of its own. To access different branches and directories, you need to enter the correct command lines and names to reach them, and if you don’t enter the correct command lines and correct names, you get stuck in the same state or position where you are. You also have a command line where you can see your git commit history. The git commit history is similar to how you see the history of transitions on a state machine diagram. It is important to be able to see the git commit history to find any possible mistakes or positions to return to if there is a bug causing issues for the team. I see no other diagrams more appropriate than the state machine diagram.

4) There isn't any lack of specific content already covered that surprised me. However, I was intrigued to discover unexpected details in most of the problems I have been looking

through. For problem #2, I've noticed that there were still a few more details that can be added to the information, such as whether this augmented game plays similarly to Pokémon, considering that you capture monsters and can trade them between players. However, problem #2 never mentions if each monster is a different type from each other, or if there are any rules or restrictions for monster trading between players. I was surprised by being able to point out missing details after sketching all 3 of the diagrams.