Software Engineering Case Based Learning Exercise

Deepti Ameta (DAIICT), Ashish Sureka (Ashoka) Paramvir Singh (NITJ), Saurabh Tiwari (DAIICT)

Objective: To facilitate the concepts of understanding the problem domain, requirement elicitation and prioritization through real world case.
Topic covered in the class to undertake the study: Requirement Elicitation Techniques, Conflict Detection, Requirement Analysis, Requirement Prioritization Techniques, Concept Mapping, Use cases and user stories.

Metro TicketDistributor System

Bangalore Metro station wants to establish a TicketDistributor machine that issues tickets for passengers travelling in metro rails. Travelers have options of selecting a ticket for a single trip, round trips or for multiple trips. They can also issue a metro pass for regular passengers or a time card for a day, a week or a month according to their requirements. The discounts on tickets will be provided to frequent travelling passengers. The machine is also supposed to read the metro pass and time cards issued by the metro counters or machine. The ticket rates differ based on whether the traveller is a child or an adult. The machine is also required to recognize original as well as fake currency notes. The typical transaction consists of a user using the display interface to select the type and quantity of tickets and then choosing a payment method of either cash, credit/debit card or smartcard. The ticket or tickets are printed and dispensed to the user. Also the messaging facilities after every transaction are required on the registered number. The system can also be operated comfortably by a touch-screen. A large number of heavy components are to be used. We do not want our system to slow down, and also usability of the machine. The TicketDistributor must be able to handle several exceptions, such as aborting the transaction for incomplete transactions, insufficient amount given by the travellers to the machine, money return in case of aborted transaction, change return after successful transaction, showing insufficient balance in the card, updated information printed on the tickets e.g. departure time, date, time, price, valid from, valid till, validity duration, ticket issued from and destination station. In case of exceptions, an error message is to be displayed. We do not want user feedback after every development stage but after every two stages to save time. The machine is required to work in a heavy load environment such that at the morning and evening time in weekdays, and in weekends performance and efficiency would not get affected.

Questions:

1. Enlist all functionalities of the TicketDistributor system in the form of user stories. Can you prioritize them (using the requirement prioritization techniques, e.g., AHP, Numerical Assessment, MoSCoW method, etc.), keeping priorities of non-functional aspects into consideration? Provide details.

2. List top five parameters on which the performance of the system can be achieved. List out all the possible use cases?

3. Is it possible to measure progress of the system under development after every SDLC stage? How?

4. List out all the possible measures that can be taken if the fake currency note detecting component gets fail? How will the system withstand its failure? List all possible scenarios.

5. List all the possible features and components. But we need a high responsive system for the quick issue of the tickets so we need to neglect some. Which features and components could be neglected without affecting the systems performance?

6. Let us assume that there is a passenger who issued the ticket but he lost it somewhere. No authority would believe him as he lacks the proof. What safety features, according to you, should be incorporated as a proof against lost tickets?

7. How you can ensure that the TicketDistributor machine is usable? What are the features and constraints need to incorporate at the time of specifying the requirements?

8. Identify three different use cases where the conflicts between the requirements occur? Do you think that the conflicts can be resolved? How?

Additional Question:
How do the requirements of the similar systems (other similar applications) match with the system under study here?