Elliot Putnam

CS-255 – System Analysis and Design

Interpreting UML Diagrams

12/4/2023

The two diagrams provided are Activity and Sequence diagrams. They both begin with the user accessing an ATM. The Activity diagram shows the functional steps between each task required for the system to function properly. For example, from start, the first function is ‘verify PIN’, if the pin is correct, it moves on to ‘ask for amount’. The sequence diagram shows the relationship between active agents. Each agent (user, ATM, bank) for this instance is aligned with another and the steps travel from top to bottom to allow analysts to easily follow and understand the steps. An example would be the user column starts with ‘enter card’ which an arrow moves to the ATM, since the card enters the machine, then the ATM registers a card was inserted and prompts and arrow back to the user.

The use case for both diagrams is withdrawing cash from the ATM. Interactions include users entering pins, pin encryptions being sent from ATM to bank, verification goes back to the ATM, and allows the user to select their next step.

I noticed in the activity diagram there was logic that ended the process if the pin was entered incorrectly. This would cause frustration for the user due to the long wait times between menus. A better way to perform this action would be to loop the wrong pin back to the top of start. So the user has the ability to try again, eliminating the need to remove, wait, and reinsert card. Another pain point in the same diagram is the amount not available process. If the user inputs an amount that is not available, it should prompt the user to try a new value. Generating receipts, then printing them sacrifices time and resources of the machine. Receipts should be a final product upon completion.

Although it only asks for a single change to the UML diagram, I have added both as they are only small redirections. The diagram still has available alterations to make it more complete.

A diagram of a flowchart

Description automatically generated