Module 8: Portfolio Project

Your Own UML State Machine Diagram (SMD)

Nolan Byrnes

CSC505 – Principles of Software Development

Colorado State University – Global Campus

Professor Steven Evans

September 9th, 2022

Module 8: Portfolio Project

Pressman states that "a UML state diagram models an object’s states, the actions that are performed depending on those states, and the transitions between the states of the object" (Pressman, 2019, pg. 625). By creating a state diagram of an ATM, I was able to map out the actions that an ATM would perform based on its current state. This paper goes over the UML State Machine Diagram that I created for an ATM and the resulting Python implementation.

**UML State Machine Diagram**

There are several benefits to using State Machine diagrams such as easier communication, and it allows you to ensure that the logic for the program is complete. Wempe states that by visualizing the programs logic through a state machine diagram, it “can help to bridge the gap between business experts and developers” (Wempe, 2019, para. 12). In order to show stakeholders that the logic of a given program is correct or complete, having a diagram helps them to understand how the solution behaves much more effectively than presenting the code. By having a diagram, it is much easier to explain the logic that the developed solution uses. Creating state machine diagrams also helps to make sure that the logic you are implementing is complete, and there are clear paths of logic that the program would transition through.

In order to display the different states and transitions of an ATM, I created a UML State Machine diagram shown in Figure 1 below.

Figure 1.

UML State Machine Diagram for ATM

Diagram

Description automatically generated

Note. Diagram to represent the States and Transitions of the ATM solution .

There are two main states for an ATM, which include authentication, and the Customer’s account menu. When the ATM is in the Authentication state, the ATM needs to obtain the card number and the associated PIN. After the ATM has authenticated the user and obtained the users account details, the ATM would transition to the Customer Account Menu state, where the user would then be able to select options related to their account. When the user decides to log out, the ATM would then sit in an idle state until another user decides to use the ATM.

In order for the ATM to authenticate a user, it must first determine if the card number that they entered is a valid one, and if a valid card number is provided, it then needs to check if the associated PIN has been entered, which are two different states.

Sparx Systems mentions that a “state machine diagram may include sub-machine diagrams”, which makes the Authentication state a compound state (Sparx Systems, n.d., para. 11). Once the ATM reaches the Check Card Number state, it prompts the user to enter their card number. Once entered, it checks to make sure the card number entered exists. If it is a blocked or invalid card, it loops back to prompt the user to insert the card number. Otherwise, it transitions to the Check PIN state. If the entered PIN is invalid, it will then increase the failed login attempts of that card by one and returns to the idle state. If the invalid login attempts reached the max login attempts, it blocks the card and transitions to the idle state.

After the user has been authenticated, they have access to their account, and the ATM transitions to the Card Holder menu. This is where the user can select to either check their balance, withdraw, deposit, or log out of their account. When the user selects to withdraw, the ATM transitions to prompt the user for the amount to withdraw. It would then check to see if the account has the amount they requested in their account. If the account has sufficient funds, it will dispense the cash, and check to see if the account’s balance has reached zero. If the account does not have any fund left, it will then close the account, and log out, transitioning back to the ATM’s idle state.

**Implementation**

In my Python implementation of the diagram, the ATM begins by using ATM.run(), which gives the user the options to sign in, or shut down the ATM. When the user selects to sign in, the ATM is in an authentication state, and prompts the user for the card number. The ATM then checks with the bank to determine if the card number exists. If the entered card number is invalid, or blocked, the ATM will prompt the user to enter the card number again. After the user enters a valid card number, the ATM will then prompt the user for the PIN number. The ATM will then check with the Bank to verify if the pin of the card number matches the pin they entered.

If it is not a match, then the login attempts of the Card increases by 1. Once the Card has had three failed login attempts, the Card is then blocked. If the Pin did match, the Account information is returned, and the ATM transitions to the account menu.

In the account menu, the ATM prompts the user to view their account balance, deposit, withdraw, or logout. If the user selects to view the account balance, it is displayed, and transitions back to the account menu. If the user selects to deposit funds, the ATM requests for the amount to be deposited, and increases the users account balance by what the user has entered. If the user is withdrawing from their account, the ATM requests for the amount to be withdrawn. If the amount exceeds the account balance, the ATM informs the user that there are insufficient funds and returns to the account menu. If the amount requested is less than the current balance, the ATM will then decrease the requested amount from the user’s balance, dispense the cash to the user, and return to the account menu. If the amount requested brings the users account balance to zero, the account is closed, and the ATM transitions out of the account menu, and back to the Idle state, waiting for the next user to log in. If the user selected to log out, the ATM exits the account menu, and back to the Idle state.

Having security in mind, I used a Card() class and a Customer\_Account() class to separate the customer’s account details from the customers card. Not only is the card a separate object than the customer’s account, but it also helps to keep the customer’s account secure. By knowing an individual’s card number, you would not have access to the customer’s account information, unless they know the PIN number associated with the card. The relationship between the Customer’s account and the associated Card is by the generated id of the Customer Account and the Card’s account\_id. To create a customer account and associated card, I implemented a method in the bank class which generates a customer account and associated card. Since an applying customer does not get to decide the card number, the Bank assigns the card number that is associated with the generated card.

**Conclusion**

State machine diagrams are a useful tool for communicating to others the different states and transitions that a program has and ensuring that the logic of a program is complete. Without having a diagram to show the logic of a program, it would be difficult to explain to others how a program transitions from state to state. The State Machine diagram I created shows how my python implementation of an ATM transitions from an idle, authentication, and the customer account menu. Separating the customer’s account details from the card helped to keep their accounts secure by storing the account details in the Customer\_account() class and associating a customer’s account to a card by an auto-generated id number named account\_id in the Card() class. I found that by having a state machine diagram of the ATM that I was developing helped me to ensure that the logic to the ATM I created was complete and helped to communicate the logic that the ATM uses.

**REFERENCES**

Pressman, R. S., & Maxim, B. R. (2019). Software engineering: A practitioner's approach. McGraw-Hill Higher Education.

Sparx systems. (n.d.). State machine diagram - UML 2 tutorial | Sparx systems. UML modeling tools for Business, Software, Systems and Architecture. <https://sparxsystems.com/resources/tutorials/uml2/state-diagram.html>

Wempe, J. (2021, August 15). Benefits and Downsides of Using Statecharts. https://blog.jannikwempe.com/statecharts-benefits-and-downsides