Andrew Fernandez, Kyle Hemsley, Anthony Reveron

Dr. Valentina N. Korzhova

COT 4210

16 April 2019

The Automation Process of an ATM

An Automated Teller Machine (ATM) is a service used by banks world wide. Millions of users use this service on a daily basis. An ATM is an extremely practical use of states and transitions. That is why it is simple to abstract the way an ATM works using automata. This report demonstrates how we used Object-Oriented Programming to create a visualization of the automata of a basic-functioning ATM.

The task of the project was to demonstrate the automation process of an ATM. The approach used was to apply Object-Oriented Design principles in the C++ programming language. Classes are an efficient way to demonstrate states and transitions using objects. In order to complete the task, it was necessary to create classes that were needed to simulate an ATM. After the classes were planned with the states and transitions they would hold, the programming aspect of the project began. In order to visualize the automation of the ATM, a console application was created using C++ that gave users an interface to use the simulated ATM. This should portray the various states that an ATM goes through as users interact with it. Inspirations and ideas were derived from the two articles that can be seen in the Works Cited page of this report. For example, our class usage was inspired by the UML by Sadhana Verma and Ajay Pratap (Verma, Pratap 40).

The classes used in the console application included a class for the ATM, Customer, Account, and Transaction. The ATM class was the foundation of the application. For the visualization purposes, the ATM object that ran the simulation held all the mock Customers that were used for the demonstration using a map data structure. It also held the available funds (cash) the ATM had that was able to be withdrawn. The ATM class is the core that uses the rest of the other classes and abstracts the states of an actual ATM. The Customer class held information about each Customer object, such as their name, their Account, and their PIN. The Account class held information about the actual account, which includes the balance and its identification number. The Transaction class simply abstracted and kept track of the actions that were executed by the user during a session. When the user decides to end the session, the application uses the Transaction's information to give a brief synopsis of what was accomplished during the session.

This ATM application used various states that were used to execute various actions, such as letting a user withdraw from their Account, deposit into their Account, or just view their Account's balance. The very first state of the simulated ATM prompts the user to either create a Transaction or exit the program. If they choose to exit, then the automata will reach the final state. Otherwise, the start state would transition to a state that prompts the user to enter their Account number. This is the state that will always be reached before transitioning to the final state. As a disclaimer, in most real-world ATM's, it would prompt the user to swipe their card. For purposes of our demonstration, it just requires the Account number. From this state, the user has two subsequent states if they fail to enter a valid Account number. If it reaches the last state, and they fail again, the state would transition back to the starting state. In reality, more security

precautions would be taken for this application. If in any of these previous states the user enters a valid Account number, the state would transition to another that prompts them to enter a valid PIN associated with that Account. Again, they have three attempts to input the correct PIN. If they fail after the third time, they will be sent back to the start state of the ATM. Otherwise, they will have access to the main functions of the ATM on their Account. From here, they could either chose to view their balance, withdraw an amount, or deposit. As most ATM machines, the application limits users to withdraw amounts only in intervals of $10.00. It also does not give user access to withdraw money if they lack the funds in their Account balance. The ATM will also go to a failed state if the ATM machine itself lacks the cash funds to give the user the amount they wish to withdraw If the user enters an invalid amount to be withdrawn, the machine enters an invalid state and then prompts the user if they are finished. The View Balance and Deposit actions could also be seen as different states the machine goes through. After any action, the user will be asked if they are finished the session or if they would like to create a new Transaction. If they finish, then the information about the entire session is given to them like a receipt and then retrieve back to the start state where a new user can begin a session or the current user can begin a new session. Otherwise, it will go back and prompt the user to choose another action.

This project is a prime example that automata can be found anywhere in everyday life. It can be found whenever a machine is being used and there are various states that machine must go through to complete its purpose. A surprising discovery this project brings to attention is that an ATM must complete seemingly countless checks on user input. It is a quietly complex process

that goes through many states for the machine to be completed. These tiny processes go oblivious to the user while they complete their transactions.

Works Cited

Wang, Yingxu, et al. The Formal Design Model of an Automatic Teller Machine (ATM). International Journal of Software Science and Computational Intelligence, Jan. 2010, pdfs.semanticscholar.org/e1a4/dab983c2205b1e116ffd9cf3571f22c31a25.pdf.

Verma, Sadhana, and Ajay Pratap. Designing of Testing Framework through Finite State Automata for Object-Oriented Systems Using UML. International Journal of Computer Applications, Oct. 2015, pdfs.semanticscholar.org/d199/64294ea4d0386c3618f62aa731f4d998bf37.pdf.