**State Machine Diagrams**

Lucy Mills

Colorado State University, Global

CSC470 - Software Engineering

Dr. Luis Gonzalez

December 4, 2022

**State Machine Diagrams**

To illustrate an object’s transition in state of condition due to various factors, a state machine diagram can be created for the system. While the overall format of a state machine diagrams is loose with many different implementations, there are seven key elements used in these diagrams for illustrating the flow of status change (Jin, 2022). For state machine diagrams, the components and corresponding shapes are the first state (black circle) for the beginning of the proecess, the terminator (black circle with a band around it) to end the process, an exit point (white circle with a cross in it) for creating an exit point inside the status change, choice pseudostates (black diamond) for scenarios where multiple outcomes are possible, events (arrows with text) represent shifts in state, transitions (arrows) shows the flows between states, and finally states (white rectangles with rounded corners) (Opinaldo, 2021). With these components present, the diagram can take on whatever format that most appropriately illustrates the main object’s change in status through various processes.

**ATM System**

The first state machine diagram included below, figure 1, is for showing the potential states of an ATM machine after it has been turned off. Promptly after turning off, the machine initiates a startup action, known as self-test state, to ensure that the machine is capable of performing more transactions. A pseudostate is added to this intersection as the test has the potential to both fail or succeed. If the test fails, then the machine enters an out-of-service state; an exit point is added to this state as the machine is unable to provide any service to a customer until it has been inspected by a bank employee. Alternatively, if the test is successful, then the machine enters the idle state where it waits until a customer interacts with the ATM by touching the screen or pressing a button on the pin pad. Once the machine is serving a bank customer, then the process is finished.

**Online Ordering System**

Next, the second state machine diagram, figure 2, illustrates the potential states of a customer’s account for an online shopping outlet. The user begins by creating an account on the website by entering personal information like name, birthdate, email address, mailing address, and billing information. Then, a temporary account, lasting 48 hours, is created with an activation box showing that it only exists for the timeframe or until the user click on the verification link sent to their email address and activate the user’s account. The other option in the pseudostate is for users who fail to click on the verification link in their email, rendering the user to be marked as inactive. While an active user has clearance to place online orders on the website, active users can have their account locked, limiting access to the account, if website rules are broken i.e., taking too many tries to login to the account or using an insecure payment method. Finally, the user has the option to close their account for the online shopping outlet if they decide to quit shopping with the company.

**Figure 1**

*Diagram

Description automatically generatedATM Machine State Machine Diagram*

*Note.* In this diagram, the ATM machine’s change in status begins once the machine turns itself off and enters various states until a user comes to interact with the machine.

**Figure 2**

*Online Ordering Account State Machine Diagram*

*Diagram

Description automatically generated*

*Note.* Whereas this state machine diagram evaluates the status of a user’s online shopping account based off account verification and online behavior.

**Pseudocode - ATM Machine**

public class ATM

{

public int serveCustomer()

{

// processes for serving customer

}

private void turnOff()

{

if(ATM == off)

{

perform startup action = run self-test state;

if(self-test == fails)

{

activate out-turnOff-service state;

enter exit point as ATM cannot provide service;

}

else if(self-test == pass)

{

while(waiting for customer)

{

activate idle state;

}

serveCustomer();

}

}

}

}

**Pseudocode – Online Account**

public class ShopperAccount

{

public void createAccount()

{

input personal information;

create temporaryAccount();

}

public void temporaryAccount()

{

for(48 hours)

{

if(user verifies account through clicking link){

verify account;

mark as active user;

}

else if (user fails to click link){

do not verify account;

mark as inactive user;

}

}

}

Active users

{

if(website rules are broken){

mark as locked account;

disable access to the account;

}

else{

can place online orders;

can end membership to close account;

}

}

}

**Lessons Learned**

Throughout the software engineering course, I have learned about many vital topics that will guide me designing and evaluating software in my future computer science related career. Learning the various forms of UML diagrams has taught me the appropriate purposes and audiences for design diagrams depending on the structure being examined. Behavioral diagrams, like state machine, activity, use-case, communication, interaction, and timing diagrams, focus on the changing relationships and processes in a system. On the other hand, structural diagrams, like class, composite structure, object, component, deployment, and package diagrams, more so formats the specific steps needed to complete a process in the system. Depending on the complexity of the UML diagram, they can be used to present new system features to a nontechnical group of stakeholders or highly technical team of software developers. The flexibility of UML diagrams is beneficial in providing different structures and formats to fit the needs of every development team. Converting the UML diagrams into pseudocode has given me the opportunity to internalize the ways in which a system’s process can be applied to code.

Overall, the section that I most enjoyed learning about has to be Module 6 over sequence diagrams. The structured format and chronological order of processes in sequence diagrams made them easy for me to understand and create myself. The main external influence, or actor, begins the diagram in the top left corner with other external entities for the system being listed across the top of the diagram. Below every object, a vertical lifeline runs to hold every message the object is responsible for manipulating. The steps of the sequence will connect the object messages in chronological order for the system. With my new knowledge of different UML diagrams, I am now able to create visualizations for any process involved in the workings of a system.

**References**

Jin, J. (2022). *System Design — UML State Machine Diagram - InterviewNoodle*. Medium. <https://medium.com/interviewnoodle/system-design-uml-state-machine-diagram-428beddf395d>

Opinaldo, N. (2021). *UML State Machine Diagram Tutorial: Explain with Examples*. <https://gitmind.com/state-machine-diagram.html>

Unhelkar, B. (2017). *Software Engineering with UML* (1st ed.). Auerbach Publications.