ATM MANAGEMENT SYSTEM

Cluster Innovation Centre

University of Delhi

ABHISHEK YADAV (11802)
SATYAM SINHA (11834)
YASH KUMAR (11844)

APRIL 2019

Month Long Project submitted for the paper

OBJECT-ORIENTED PROGRAMMING

Certificate of Originality

The work embodied in this report entitled **"ATM Management System"** has been carried out by **Abhishek Yadav, Satyam Sinha and Yash Kumar** for the paper "**Object Oriented Programming"**. I declare that the work and language included in this project report is free from any kind of plagiarism.

Abhishek Yadav
Satyam Sinha
Yash Kumar

Acknowledgment

I have taken efforts in this Project (ATM Management System). However, it would not have been possible without the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

Firstly I express my gratitude to project coordinator the Prof. **Anjani Kumar Verma** for their support. I would like to thank my project guide Prof. **Anjani Kumar Verma** for giving me this and guiding me throughout my work with patients. He has taken the pain to go through my project and make necessary corrections as and when needed. Without her guidance and persistent efforts, this project would not have been possible.

I extend my gratefulness to my institute's library and its members for their support and heartfelt thanks to all the faculty members of the college.

Abstract

ATM Management System

by

Abhishek Yadav, Satyam Sinha, and Yash Kumar,

Cluster Innovation Centre, 2018

An Automated Teller Machine (ATM) is a safety-critical and real-time system that is highly complicated in design and implementation. This paper presents the formal design, specification, and modeling of the ATM system using denotational mathematics known as Real-Time Process Algebra (RTPA). The conceptual model of the ATM system is introduced as the initial requirements for the system. The architectural model of the ATM system is created using RTPA architectural modeling methodologies and refined by a set of Unified Data Models (UDMs), which share a generic mathematical model of tuples. The static behaviors of the ATM system are specified and refined by a set of Unified Process Models (UPMs) for the ATM transition processing and system supporting processes. The dynamic behaviors of the ATM system are specified and refined by process priority allocation, process deployment, and process dispatch models. Based on the formal design models of the ATM system, the code can be automatically generated using the RTPA Code Generator (RTPA-CG), or be seamlessly transformed into programs by programmers. The formal models of ATM may not only serve as a formal design paradigm of real-time software systems, but also a test bench for the expressive power and modeling capability of existing formal methods in software engineering.

I. INTRODUCTION

Automated Teller Machine enables the clients of a bank to have access to their account without going to the bank. This is achieved only by developing the application using online concepts. When the product is implemented, the user who uses this product will be able to see all the information and services provided by the ATM, when he enters the necessary option
and arguments. The product also provides services like a request for cheques, deposit cash and other advanced requirements of the user. The data is stored in the database and is retrieved whenever necessary. The implementation needs ATM machine hardware to operate or similar simulated conditions can also be used to successfully use the developed product.
To develop this ATM system the entire operation has been divided into the following
step:

1. Verification process
2. Banking services
3. Transactions
4. Special services

The program is designed in such a way that the user has to card and pin number. Once verified, he is provided a menu and he/she had to enter the option provided in the menu. For example, when the user wants to view the list of payment history than he/she had to enter the

option for payment history provided in the main menu. When the option is entered along with the respective argument, then the payment history is displayed on the screen. The user also must be given the option to browse through the pages like the previous page, next page, etc. The user may experience a delay in retrieving or viewing the data when there are many users logged on to the same bank branch system.

II. FORMULATION OF THE PROBLEM

II.1Methodoloy

ATM (asynchronous transfer mode) has been considered by the telecommunication industry to be the ultimate solution to the networking requirements of broadband communication. We discuss the implementation of an ATM cell processing system that provides guaranteed quality of service (QoS) for a broad range of services by per-connection maintenance. The complexity in this application domain is based on the implementation and the verification of complex hierarchical control structures. To overcome this problem, we propose a design methodology, which is three-fold. First we use abstract executable specifications in C++ for functional verification, second, we introduce a design for verification (DfV) strategy to reduce the number of test cases and third to increase modeling efficiency, we have chosen behavioral VHDL modeling and high-level behavioral synthesis for the applicable parts of the design. In this paper, we present the area overhead introduced due to our methodology compared to the advantage in terms of reducing the verification effort.

II.2 Results

An **automated teller machine** (**ATM**) is an electronic telecommunications device that enables customers of financial institutions to perform financial transactions, such as cash withdrawals, deposits, transfer funds, or obtaining account information, at any time and without the need for direct interaction with bank staff.

ATMs are known by a variety of names, including **automatic teller machine** in the United States. Although ABM is used in Canada, ATM is still very commonly used in Canada and many Canadian organizations use ATM over ABM. In British English, the terms **cash point, cash machine, mini bank** (the official name of the Yorkshire bank ATMs), and **"hole in the wall"** are most widely used**.** Other terms include **any time money**, **cash line**, **nibank**, **tyme machine**, **cash dispenser**, **bankomat** or **bancomat**. Many ATMs have a sign above them, indicating the name of the bank or organization that owns the ATM, and possibly including the networks to which it can connect. In Canada, ABMs that are not operated by a financial institution are known as "white-label ABMs".

CONCLUSION

All good things must come to an end, so does our ATM program. We are done with the basic transaction processes for our ATM. Now we have in hand, two unique features of our ATM which help in reducing the complex nature of banking processes such as PIN CHANGE and FUND TRANSFER. I am not sure about the fund transfer, but now we can change the pin using an ATM without visiting the bank. This amendment is one of the most advantageous features of ATM today.

I have read that some of the banks also have the feature of fund transfer nowadays, so today we will write the code for both cases. During the actual process of PIN CHANGE, we have to enter the OTP which would be sent to our registered mobile number. But in our case we are not dealing with mobile phone, we will just enter the old and new pin.

References

https://www.oracle.com/technetwork/java/javase/documentation/index.html

Books: Head First Java, 2nd Edition

- **Thinking in Java (4th Edition)**
- **Java the complete reference**

APPENDIX

Account.java | ATM.java | Bank.java | Transaction.java | User.java

```java
import java.security.NoSuchAlgorithmException;

public class User {

    private String firstname;

    @SuppressWarnings("unused")
    private String lastname;

    private String uuid;

    private byte pinHash[];

    private ArrayList<Account> accounts;
    public User(String firstname, String lastname, String pin, Bank theBank){

        this.firstname = firstname;
        this.lastname = lastname;

        try {
            MessageDigest nd = MessageDigest.getInstance("MD5");
            this.pinHash = nd.digest(pin.getBytes());
        }catch (NoSuchAlgorithmException e){
            System.err.println("error, caught NoSuchAlgorithmException");
            e.printStackTrace();
            System.exit(1);
```