International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 11 – No. 4, September 2016 – www.ijais.org

Test Case Generation from UML Sequence Diagram for Aadhaar Card Number based ATM System

Wasiur Rhmann
Deptt. of Computer Science
B. B. Ambedkar University
(A Central University)
Lucknow, U. P., India

Prof. Vipin Saxena
Deptt. of Computer Science
B. B. Ambedkar University
(A Central University)
Lucknow, U. P., India

ABSTRACT
Software testing plays an important role to uncover the errors during the programming phase of the software development. It is also used for validation of the software. For representing the dynamic behavior of the software system, a sequence diagram from Unified Modeling Language is used. In the present work, test cases are generated from sequence diagram by converting it into the Sequence Flow Graph. Test scenarios are generated from Sequence Flow Graph by defining pre and post conditions using Object Constraint Language. Test case outputs are determined from the final message in the test scenario. A real case study for cash withdraw from Aadhaar card based ATM is considered. In India, Aadhaar card number is mandatory for every citizen and it increases security during transaction of cash from ATM machine. Generated test cases from the present study satisfy message path coverage criteria. Finally, cyclomatic complexity is also computed for optimizing or validating the generated test cases.

Keywords
UML, Sequence Diagram, Test Case, OCL, Sequence Flow Graph.

1. INTRODUCTION
Software testing is the primary and core activity for the development of a high quality software. In software testing, a program is executed with the intent to find errors and correct them [1]. Reliability of the system is fully dependent on the software testing. Software quality assurance activities are performed to review and verify the software but these activities are not sufficient to produce good quality software. Effective testing strategy reduces the development cost of the software. In software testing, a set of test cases are designed to execute the program and verify the output with expected output. Generation of test cases is very crucial activity. Generation of test cases mainly involve test case design and execution of designed test cases. With increasing in functionality of software caused more complexity in the development of the software. Object-oriented analysis and design strategy are used to reduce the software development cost and increase the reusability of the developed software. Object-oriented software needs different type of testing strategy as these types of software use the concept of class, object, inheritance, polymorphism, etc. UML is used for analysis and design of object-oriented software system. For testing the object-oriented software system, test cases may be designed from UML models. UML models are developed at the early stage of software development process. So, errors may be available during early stages of the development process. Since test case generation is a time consuming activity, so, there is need of automatic test case generation strategy. In automatic test case generation, different UML diagrams are used by researcher for testing purpose. Design models in software testing are used to finds defects at early stage of the software development. Quality of generated test cases depends on which extent they cover the functionality of the system under test. UML is used to model the requirements of the system. UML sequence diagrams are important for visualizing the dynamic aspects of the system. Sequence diagram represents the flow of control among objects during interaction between objects. Sequence diagram is an interaction diagram and can be viewed as a table in which objects are placed at X axis and messages are in increasing order of time at Y axis [2]. Sequence diagram contains different flow of control elements like alt, par, loop, etc. Various researchers have proposed many test case generation techniques from different UML diagrams. Initially, OCL is an extension of UML and a formal language. OCL may now be used with OMG meta model including UML [3]. OCL describes rule that are applied to UML. OCL is an extension to UML as a formal specification language. OCl is a précised text based language which provides constraint and object query expression for any UML models. OCL provides expressions without ambiguity which usually occurs in using natural language. Liet al. [4] presented a new approach of test case generation from UML sequence diagram and OCL expression. Authors constructed a tree from the sequence diagram. Then they traversed the tree to find out conditional predicates. OCL is used to define the pre and post conditions. Function minimization techniques are applied to generate test data on conditional predicates. This method covers message coverage and constraint attribute coverage of all objects which relate to the message. Ali et al. [5] used state diagram for generating the test cases. Here, authors first transform state diagram into finite state machine then information mined from OCL expression is used to build test cases. Generated test cases achieve transition coverage, state coverage and transition pair coverage. Swain et al. [6] have presented a novel approach to test the object-oriented software based on UML interaction diagram. Authors used message guards of interaction diagram and created conditional slice with respect to each conditional predicate. Message Flow Graph (MFG) is created from UML sequence diagram then applied conditioned slicing on predicate node. Generated test cases satisfy message path coverage and slice coverage criteria. Tonella and Potrich [7] have generated test cases from reverse engineering of the sequence diagram.

In the present work, we presented a novel approach for the generation of test cases from UML Sequence diagram. Sequence diagram is used to model the dynamic behavior of

the system and it is transformed into Sequence Flow Graph (SFG) which is traversed to generate different paths. OCL is used for pre and post conditions on a study of cash withdrawal from Aadhaar Card Number enabled ATM. Aadhaar card number is issued by Government of India which is used as unique identity number for Indian Citizen. Generated test cases satisfy message path coverage criteria and validated through the cyclomatic complexity of the flow graph.

2. PROPOSED METHODOLOGY

Different activities are performed for the generation of test cases; represented through the UML activity diagram which is shown in figure 1 and steps are given below:

Step 1. Design a UML Sequence Diagram;

Step 2. Design a Sequence Flow Graph (SFG) from UML Sequence Diagram;

Step 3. The parallel activities are to be performed like

(a) Identification of Pre and Post conditions and design of UML Use case diagram; (b)Traverse different paths of SFG.

Step 4. Further two parallel activities are to be performed like

(a) Use Pre and Post conditions for Test cases;

(b) Write output for last message from each path.

Step 5. The resultant of Steps 3 and 4 is used for writing test cases.

The above five steps are represented in Fig. 1. In SFG, each message and conditions are represented by nodes and transition between nodes is represented by the edge. Let us explain proposed methodology on alt, loop and nested if. Let us construct a sequence diagram for alt which is used to represent parallel activity. It is represented in Fig. 2 which is converted into SFG represented in the Fig. 3.

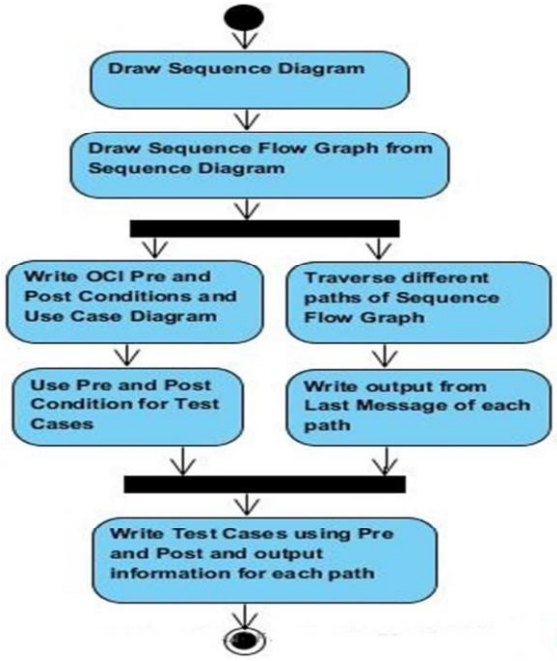

Fig.1. UML Activity Model for Generation of Test Cases

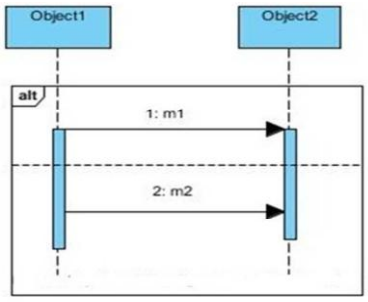

Fig. 2. UML Sequence diagram for alt

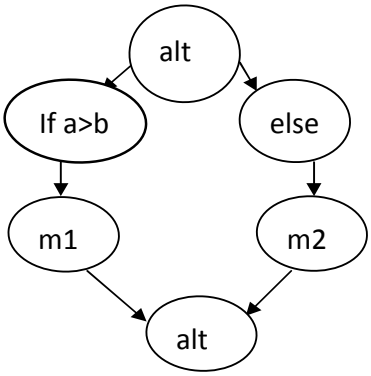

Fig. 3 SFG from UML sequence diagram for alt

In a Fig. 4, a sequence diagram for loop is constructed which is used to represent iteratively sending of messages. SFG of the sequence diagram for loop is shown below in figure 5.

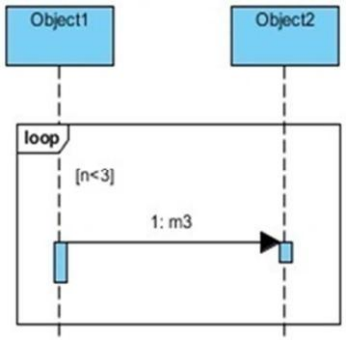

Fig. 4 UML Sequence for loop

In sequence flow graph each message and conditions are represented by a nodes and transition between nodes is

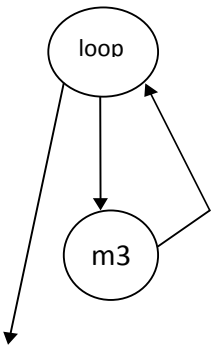

Fig. 5. SFG from UML sequence diagram for loop

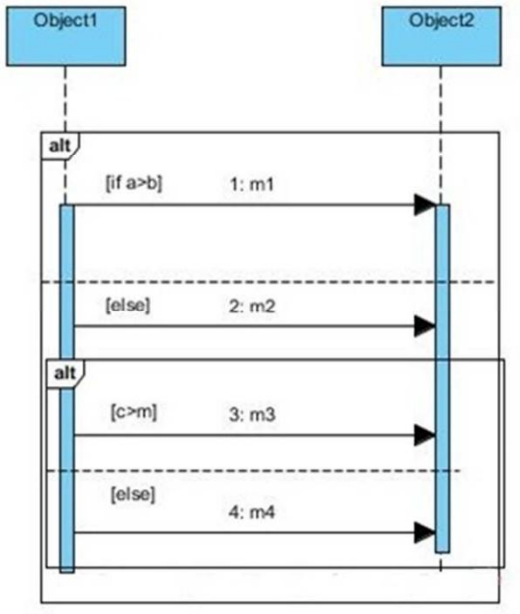

Fig. 6. UML Sequence diagram of nested if else

Fig. 6 represents the nested if else message sending statement in the form of sequence diagram. In nested if else message sending statement, there may be more than one alt statement within single alt statement. Fig. 7 shows the SFG from Fig. 6 for the nested if else statement. These three constructs are used in the real case study demonstrated in the next section.

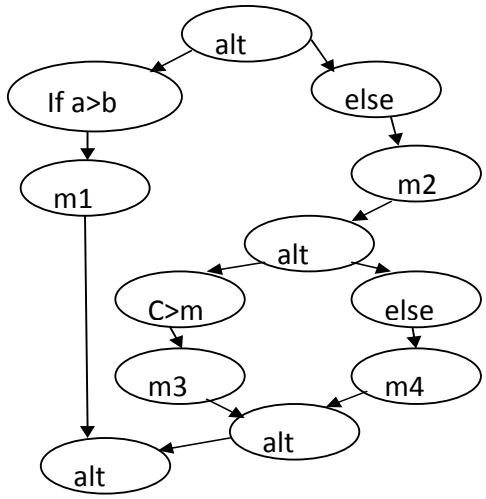

Fig. 7. SFG from UML Sequence diagram for nested if else

The above sequence diagrams cover interaction faults at the cluster level testing of the software system. In the present work, sequence diagram is converted into flow graph which is traversed to generate different combinations of message paths to ensure maximum coverage. The steps for generation of test cases are proposed below in brief:

Input: Sequence Flow Graph (SFG)
Output: Test _Cases, where Test _Case t= {Pre_condition, Input, Output, Post_ condition}

1. Start
2. generate all paths from start to end represented as P= {P_1, P_2,…….P_n}, where n=numbers of paths;
3. for each path, use OCL and Use Case diagram for pre and post condition to generate Pre and post condition of test case;
4. for each path i.e. P_i (i=1(1)n)
5. Let s=starting node, f=final node, cn=current node while(cn!=f) print the input as cn end while if (cn==f) print the output of cn end if
6. For this path print Pre and Post conditions from OCL expressions
7. end for
8. Print the test cases as {pre condition, input, output, post condition}

3. CASE STUDY:Aadhaar Card Number (ACN) Enabled ATM

Let us consider a real case study of the ACN based ATM system. ACN is issued by Unique Identification Authority of India (UIAI) which is a central agency of India. It consists of 12 digit number provided to each Aadhaar card holder and based on biometric and demographic data [8]. In ACN enabled ATM, Personnel Identification Number (PIN) number and fingerprint with Aadhaar card number are used for authentication. PIN number entered by customer is verified from Bank while Fingerprint and Aadhaar card number are verified from Central Identities Data Repository. Fig. 8 shows UML Use Case diagram for cash withdrawal from the ATM.

Here customer can perform three different activities. We will consider pre and post conditions for Use Case withdrawal for which we have drawn a sequence diagram. Pre conditions of a Use Case are set of states in which system remains. Post conditions are states that may exist after Use Case finished. A sequence diagram for cash withdrawal from ACN enabled ATM is presented in the Fig. 9. In this diagram, there are five objects Customer, ATM, Bank, ASA, and CIDR**.** Personal identity data of all account holders is contained by Central Identities Data Repository (CIDR). Authentication User agency (AUA) connects CIDR through Authentication Service Agency (ASA). Here AUA is Bank Authentication Service Agency which has secured connectivity with CIDR [9].

Customer first swipe ATM card inside the ATM machine and enters PIN number. It is to be verified by the Bank controlled by the Bank object. If PIN number is correct then customer enters ACN and finger print. ACN and fingerprint of customer are further verified by the Bank. Customer's Bank then passes these to CIDR for verification through ASA. If ACN and fingerprint are correct then CIDR sends clearance message to Bank through ASA. After clearance of dual verifications, Customer enters amount of cash withdrawal. If entered amount is less than the available balance then ATM displays *"not sufficient amount"* message else requested amount is dispensed.

These communications are represented in Fig. 9 through message passing techniques among the said five objects. The designed sequence diagram is converted into the SFG where messages are transformed into nodes and edges represent transfer of information flow between them. It is represented in Fig. 10 and it consists of four independent paths which will generate four test scenarios. From these one can write test

cases from these test paths. Pre and Post conditions are derived from OCL for each testing path.

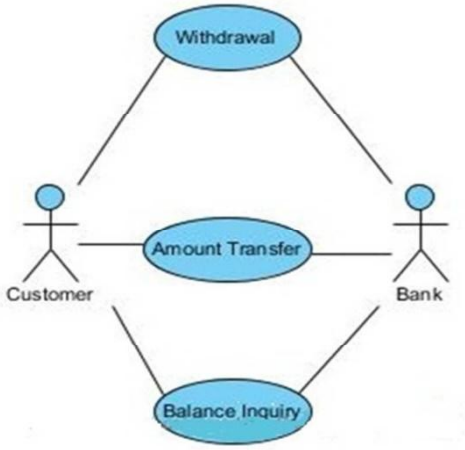

Fig. 8. Use Case Diagram for ATM

Pre conditionATM is working well and displaying Welcome message

Post condition Display Welcome message

Pre condition

Context SATM::withdraw (amount)

* pre: Bank.atm.valid=true;

Post condition

If PIN number=wrong

* It should give message ***"Wrong PIN number"***

If (ACN and Fingerprint=wrong)

* Display Message ***"Wrong ACN or Fingerprint"***

If (Amount>Customer.Account.Balance)

* Display Message ***"Not Sufficient Amount"***

If (Amount<=Customer.Account.Balance)

* Then Customer.Account.Balance@Pre-Amount

and result=Transaction successful

4. CYCLOMATIC COMPLEXITY

Cyclomatic complexity is software metric. Minimum number of test cases for sequence diagram can be calculated from cyclomatic complexity. Sequence flow graph is used to evaluate the cyclomatic complexity.

Cyclomatic complexity can be calculated by following ways.

V(G)=number of bounded region+1

Where bounded regions are area in flow graph which is surrouned by node and edges

V(G)=3+1=4

V(G)=n-e+2=23-21+2=4

Cyclomatic complexity is equal to predicate node plus one. A node which contains more than one branch out coming from that node in flow graph is called predicate node. Here there are three predicate nodes.

V (G) =Predicate node+1=3+1=4

From all three ways, cyclomatic complexity for SFG as shown in Fig. 9 is computed as 4 which shows that there are 4 independent paths which are as follows:

Path P_1: Insert Atm card, Request for Pin, Enter Pin,Verify Pin, alt, Pin is not OK, Pin number is wrong, Pin number is wrong, alt, End

Path P_2: Insert Atm card, Request for Pin,Enter Pin, Verify Pin, alt, Pin is OK, Enter Aadhaar card number and Fingerprint, Aadhaar card number and Fingerprint entered, Aadhaar card number and Fingerprint entered, Aadhaar card number and Fingerprint entered, Verify Aadhaar card number and Fingerprint, alt, Aadhaar card number and Fingerprint not OK,Aadhaar card number and Fingerprint not OK Adhaar card number and Fingerprint not OK, alt, alt, End

Path P_3: Insert Atm card, Request for Pin, Enter Pin Verify Pin, alt, Pin is OK, Enter, Aadhaar card number and Fingerprint, Aadhaar card number and Fingerprint entered, Aadhaar card number and Fingerprint entered, Aadhaar card number and Fingerprint entered, Aadhaar card number and Fingerprint entered, Verify Aadhaar card number and Fingerprint, alt, Aadhaar card number and Fingerprint OK, Enter Amount, Amount Entered, Verify Amount, alt, Amount>Balance, Insufficient amount, alt, alt, alt End

Path P_4: Insert Atm card, Request for Pin, Enter Pin Verify Pin, alt, Pin is OK,EnterAadhaar card number and Fingerprint, Aadhaar card number and Fingerprint entered,Aadhaar card number and Fingerprint entered, Aadhaar card number and Fingerprint entered, Aadhaar card number and Fingerprint entered, Verify Aadhaar card number and Fingerprint, alt, Aadhaar card number and Fingerprint OK, Enter Amount, Amount Entered, Verify Amount, alt, Amount<Balance, Receive Cash, alt, alt, alt, End

Generated test cases using java programming are presented in Fig. 11.

5. CONCLUSIONS

From the above work it is observed that Aadhaar number enhance the security of the cash withdrawal from ATM. Sequence diagram is used to model the Aadhaar number enabled Atm. Then we generated test cases from the sequence diagram. OCL used for pre and post condition. Here we used UML Sequence diagram to model the dynamic behavior of Aadhaar number enabled ATM for cash withdrawal activity. This sequence diagram is then converted into flow graph which then is used to generate test cases. Generated test cases satisfy message coverage and can be used for cluster level testing of objects. Generated test cases are validated with cyclomatic complexity metric. It is also observed that cyclomatic complexity can be helpful in estimating the maximum number of test cases required to achieve complete message coverage for UML Sequence diagram.

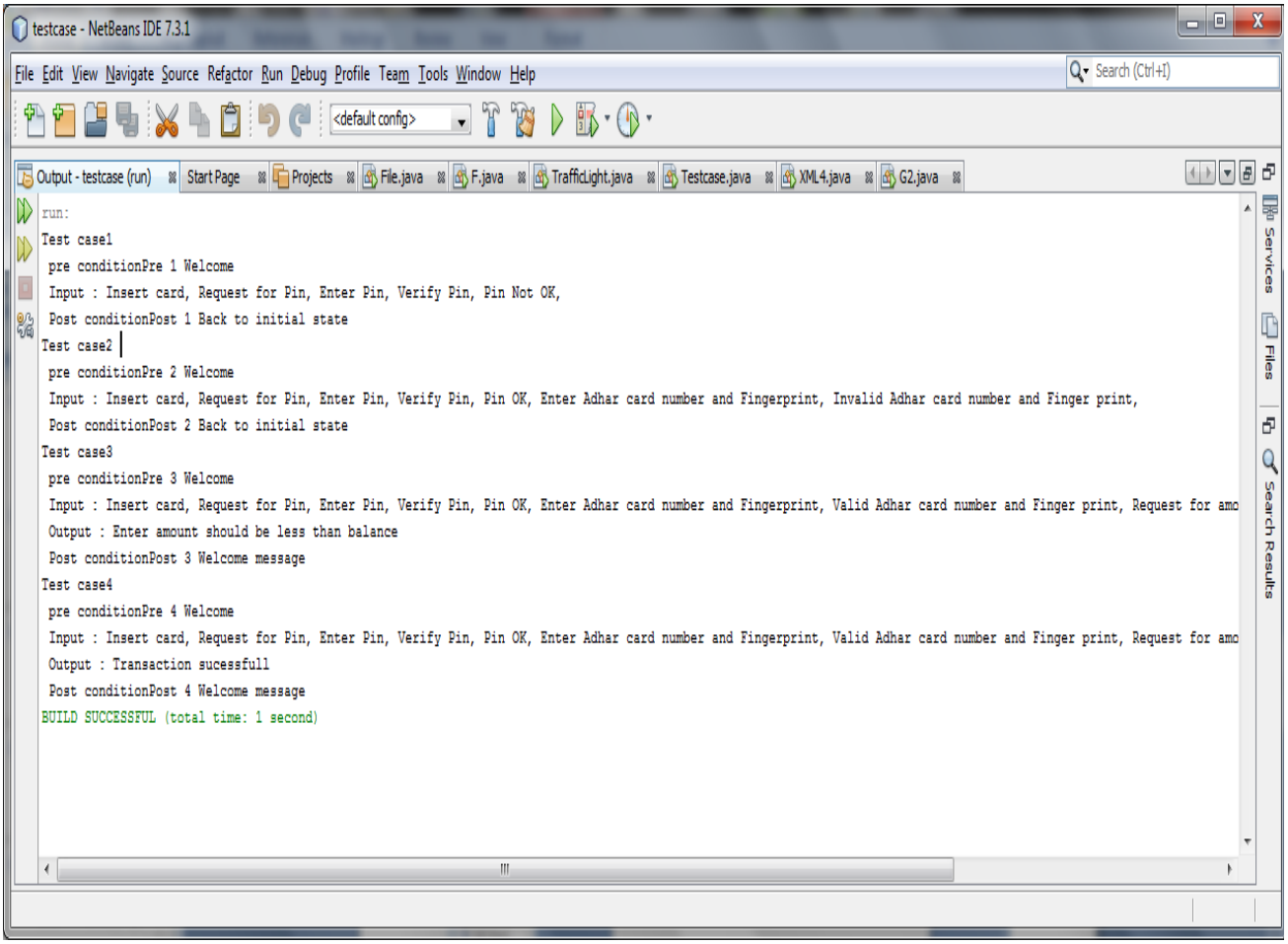

Fig. 11 Generated test cases using java programming

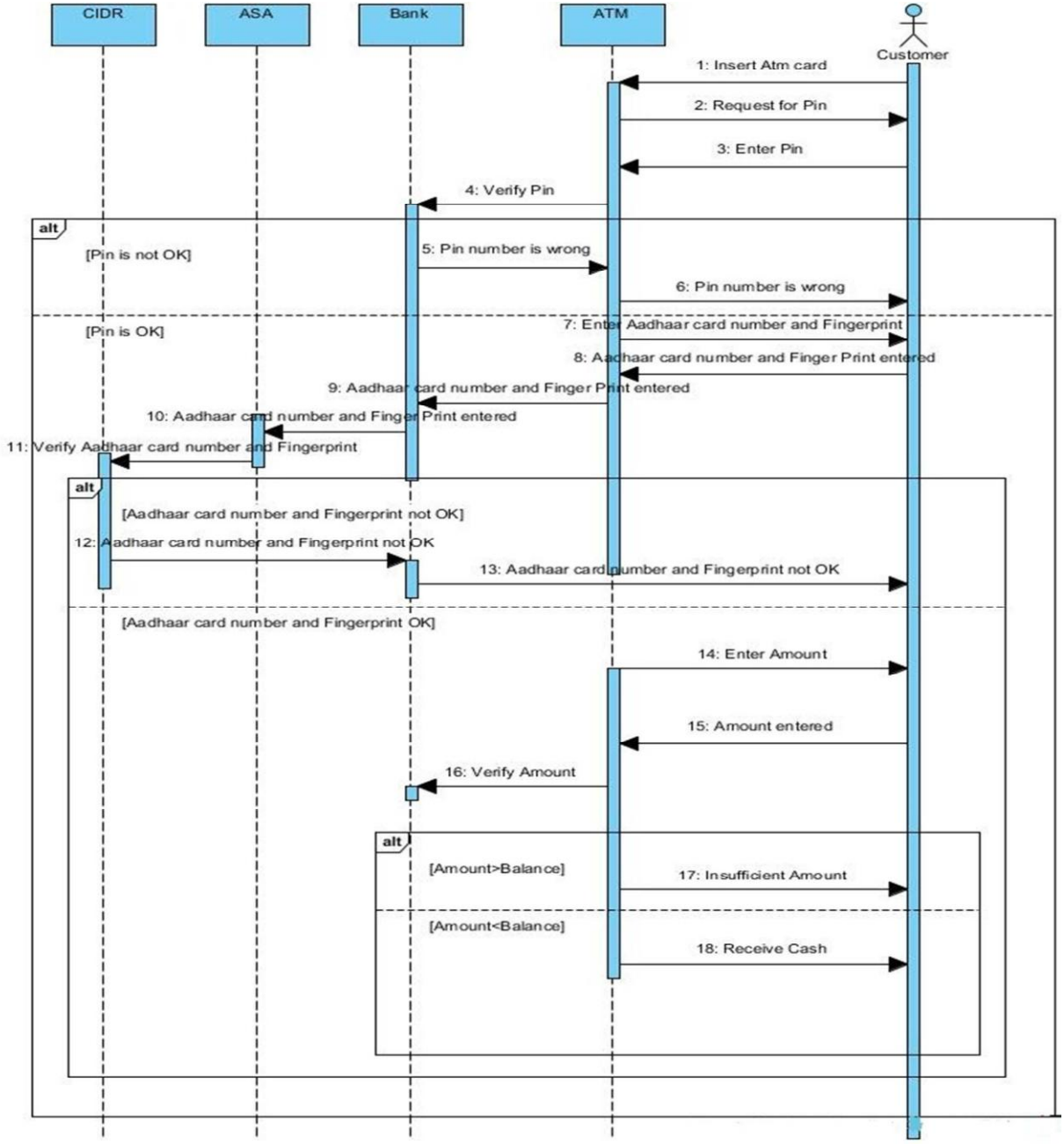

Fig 9: UML Sequence diagram of cash withdrawal from ACN enabled ATM

Fig 10: Sequence Flow Graph from Sequence diagram

6. REFERENCES

[1] Pressman, R. S. 2005. Software Engineering: A Practitioner's Approach. Mc-Graw Hill.

[2] Object Management Group (OMG). 2006. Object Constraint language (OCL) OMG, http://www.omg.org

[3] Booch G., Jacobson I. and Rambaugh G. 2009. The Unified Modeling Language Guide. Pearson.

[4] Li, L.B., Shu, L. Z. and Chen, Y. H. 2007. Test Case Automate Generation from UML Sequence Diagram and OCL Expression'. International Conference on Computational Intelligence and Security, Harbina, China, pp. 1048-1052

[5] Ali, M.A., Shaik, S. and Kumar, S. 2014. Test Case generation using UML State diagram and OCL Expression', International Journal of Computer Applications, Vol. **95**, No. 12, pp. 7-11

[6] Swain, R. K., Panthi V. and Behera, P. K. 2012. Test Case design using Slicing of UML Sequence diagram', Second International Conference on Communication, Computing and Security, pp. 136-144

[7] Tonella, P., Potrich. A. 2003. A Reverse Engineering of the Interaction Diagram from C++ code', Proc. of IEEE third International Conference on Software Maintenance, pp. 159-168.

[8] https://en.wikipedia.org/wiki/Unique_Identification_Authority_of_India.

[9] https://uidai.gov.in/authentication-2/operation-model.html