ISSN No:-2456-2165

MSC: The Bridge between the Customer and Developer

Anup Acharya
Asst. Prof., Pokhara University
Pokhara, Nepal

Abstract:- This paper identifies an effective graphical way for capturing requirement specifications of the system during the early design phase of software engineering to reduce the software failure. Further, it proposes a concept of bridge to show; how that technique called Message Sequence Chart (ITU-T, Z.120 standardized language) can be used to gather and describe the specifications of the software system during the requirement analysis phase of Software Engineering, which is demonstrated by means of "Automated Teller Machine (ATM)".

Keywords:- *Message Sequence Chart (MSC), Requirement/Specification/Analysis, Bridge, ATM, Software System, Customer, Developer.*

I. BACKGROUND

- The air collision on the air space of New Delhi, India in which more than 200 passengers lost their lives.

- The Thai Airway Crash near Kathmandu, Nepal in which more than 100 passengers lost their lives.

- Besides these, Peter G. Neumann [2], reported that:

- A Federal district judge awarded $1.25 million to the families of three lobstermen who were lost at sea in a storm that the National Weather Service failed to predict because its parent organization (the National Oceanic and Atmospheric Administration) had not repaired a weather buoy for three months. [NY Times 13 Aug 85]
- Another Union Carbide leak (causing 135 injuries) resulted from a computer program that was not yet programmed to recognize aldicarb oxime, compounded by human error when the operator misinterpreted the results of the program to imply the presence of methyl isocyanate (as in Bhopal). A 20-minute delay in notifying county emergency made things worse. [NY Times 14 and 24 Aug 85 front pages] (There were two other serious Union Carbide incidents reported in August as well, although only this one had a computer link.)
- An untimely -- and possibly experiment-aborting -- delay of the intended 25 August launch of the space shuttle Discovery was caused when a malfunction in the backup computer was discovered just 25 minutes before the scheduled launch. The delay threatened to seriously compromise the mission. [NY Times 26 August 1985].

The Times reporter John Noble Wilford wrote, "What was puzzling to engineers was that the computer had worked perfectly in tests before today. And in tests after the failure, it worked, though showing signs of trouble." Arnold Aldrich, manager of the shuttle program at Johnson, was quoted as saying "We're about 99.5% sure it's a hardware failure." (The computers are state of the art as of 1972 and are due for upgrading in 1987.) A similar failure of just the backup computer caused a one-day delay in Discovery's maiden launch last summer.

The above incidents, which took the life of many people and made lot of damages is mainly due to the software failure. Improper use of Software Engineering Principle makes the software failure. Software Engineering has got the phases from Requirement Analysis to Maintenance. Among them, the Requirement Analysis is the most vital and most difficult phase to carry out. The most of the incident due to the software failure is concerned with the initial phase of Software Development i.e. Requirement Analysis because of various reasons, which of them main are:

- Many customers are not able to express their needs in the proper way so that developer understands it.
- Some of the potential users may be able to express their need but that may not be related to what they really need.
- Many customers may have the rough idea about their need. They may not be specific on it.
- Developer may have little knowledge or no knowledge about the domain of customer.
- Developer does not want to express his weakness about knowing the domain of others because of losing reputation.

As a result, there is high risk of misinterpretation and misinformation between the developer and customer. Due to which a big gap is created between them. With this gap developer is not capable of collecting the required information about the software system. Systems are not stagnant. Well-designed and well-coded but poorly analyzed program will not satisfy the customer's need. Customer's satisfaction is the main goal of the developer. Finally, this gap leads to the consequences of customer un-satisfaction, over budget, software delay, software failure etc.

Therefore, there is a need of making this gap very small in order to get the successful software. Some efforts have been designed to reduce this gap. High technical and complicate in nature of these efforts make the customer away rather than close. Hence, we need a kind of bridge between

the developer and the customer that should be very simple and has a capacity to communicate with each other. Customer also should be able to understand the technique and at the same time developer also should be able to extract the information from the customer. The simplicity and powerful capability of expressing characteristics features of Message Sequence Chart (MSC) makes the possibility playing the role of bridge between the customer and developer.

II. INTRODUCTION

A. Requirement Analysis

The term "Requirement analysis" refers to the collection and interpretation of the requirements so that the logical view of a software system can be made. This stage started with the gathering (collecting) of software system requirements from the customer and is known as the Requirement Gathering. The developer and the customer take an active role in this stage.

Then, the developer proceeds ahead for developing Software Requirement Specification (SRS). SRS is a document containing a complete description of requirements of the software system to be developed.

This is the initial, most precious and important stage of the software development process because the software is developed on the basis of the system requirements provided by the customer. It is very difficult to collect actual need of the customers, if one cannot collect and interpret actual needs of the customer then it is difficult to develop the system to meet the customer's need and there is a chance of software failure. To reduce such type of software failure it is necessary to capture and interpret the actual needs of the customer for which this graphical concept become a milestone because the pictorial form provide the more interesting and interactive environment than that of textual form to understand the problem. Further, this concept has minimal notations and rules, so easy to interpret the requirements.

B. Message Sequence Chart

ITU-T [1] stated Message Sequence Chart (MSC) is a standardized graphical language, which mainly concentrates on communication behavior of system components and environment by means of message exchange with the help of following features:

- *Basic elements*
- *Basic MSC*
- *Structural Concept*
- *Inline Expression*
- *High-Level MSC*

However the concept is basically focused on communication behavior of system components and environment, it is sufficient to capture the software system requirements from the customer because of the today's working scenario which are based on interactive system.

- *Basic Elements*

Fundamental concepts used to describe MSC are:

- *Instance:* Reactive entities whose communication behaviors are described by MSCs and it may be a system, block, process, service or user. An instance consists of a head (instance), axis (instance), along with the instance end.
- *Message:* Represents exchange of information between two instances or one instance and the environment. Message can be divided into two events as, message output event and input event.
- *Environment:* Describes the communication between the components of a system.
- *Action:* An event, which describes an internal activity of an instance and can have informal text associated with it.
- *Timer:* Mechanism to count time unit. Each timer is a message (or reactive entity), which belongs to some instance and can be set and reset.
- *Condition:* Conditions are multi-instance events, which may span over several instances and describes the system state.
- *MSC reference:* The MSC references can be used either to reference a single MSC or a number of MSCs using a textual MSC expression.

- *Basic MSC*

A Message Sequence Chart, normally abbreviated to MSC, describes the flow of message between instances or between environment and instance, with the help of various elements such as, action, timer, condition, message out, message in etc. It consists of a MSC name and set of instances with events within an environment (frame), as shown below:

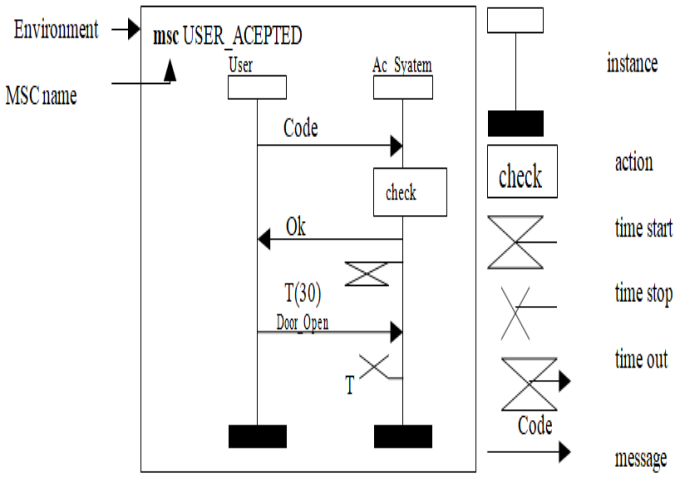

Fig 1:- msc USER_ACEPTED (example of Basic MSC)

The fig 1 is an MSC with description of basic elements that describes the scenario of user accepted mode for an

access control system. Where the user provides his particular code to the system, then granted that the user is eligible to enter the door and set a time after checking the code. The user then pushes the door open. Then the system reset the time.

➢ Structural Concept

This deals with the introduction of high-level concepts about the system with the help of inline expressions and High-level MSC (HMSC).

- *Inline Expression*

Expressions used to define composition of event structures within an MSC. From this expression alternative, parallel and optional behavior of MSC can be specified. Furthermore, exceptions and loops can also be defined. An inline expression can cover one or more instances of an MSC.

The keywords such as **alt**, **seq**, **par**, **loop**, **opt, exc** etc. can be placed in the left upper corner of the graphical notation of MCS to denote the behavior.

✓ The keyword **alt** is used to define the alternative execution of two sub-MSCs.
✓ The keyword **loop** is used to define repeated execution of a sub-MSC.
✓ The keyword **seq** is used to define the weak sequencing operation.
✓ The keyword **par** is used to define the parallel execution of MSC sections.

Inline Expression can be used as shown below:

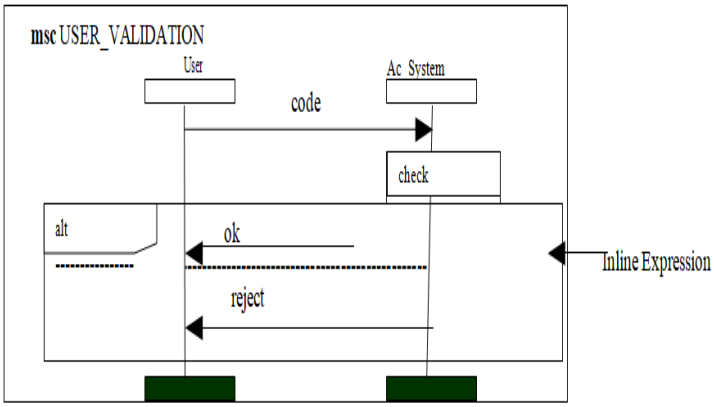

Fig 2:- Introductory example of Inline Expression

The fig 2 describe the scenario of user validation mode of an access control system, the user provide the particular code to the system, the code is checked by the system and make a decision, either the user is eligible to enter from the door or not.

- *High-level MSC (HMSC)*

High-level MSC is the graphical representation of overall concept of working scenario of a system. An HMSC contain msc name and a connected graph with in a frame; where each node may be one of the start symbols, an end symbol, an MSC reference, a connection point, a condition or a parallel frame as shown below:

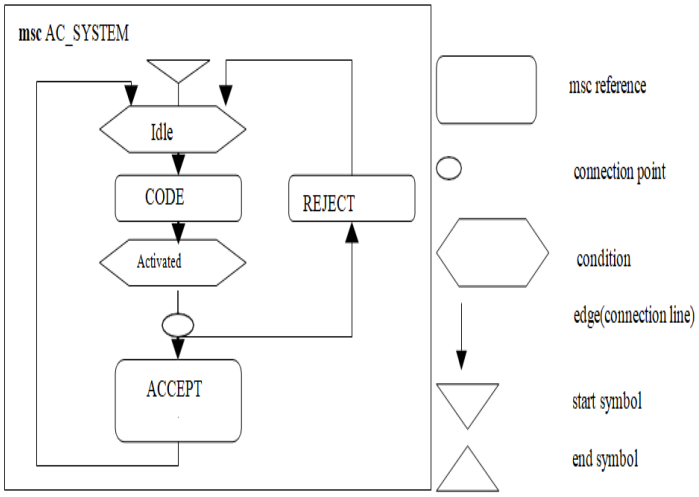

Fig 3:- introductory example of HMSC

The Fig 3 is an HMSC with description of basic elements that describes overall scenario of access control system. Where the user provide his particular code to the system, in its idle state, which then jump to the activate state. From that state the system either accepts the code or rejects the code and then from both of them the system jumps to the idle state.

III. A PROPOSED BRIDGE

When the customer and developer meet together, they come under dialogues with requirements. After listening the requirements from the customer, the developer makes a requirement specification document. Then the developer traces an MSC for the system on the basis of specification and deals it with the customer. If the customer is satisfied, then the developer proceeds ahead on the way of development process. Otherwise, modifications are made according to the customer's need then proceed ahead.

A. Method for Tracing MSC

This subsection focuses on some proposed tips for tracing MSC for a system. Which are as:

➢ Try to trace an HMSC for the system first. If the customer satisfied with that, then give the Basic MSC because if the developer goes with the Basic MSC first, it becomes more difficult, time consuming, lengthy, boring. Due to which, customer cannot understand it and is again results in software failure.
➢ While tracing an HMSC:
- Try to find out abstract reactive sub- systems from the

specification document consider them as a MSC reference.
- Try to find out the state of that system after each abstract reactive sub-system and considers them as a global condition.
- Try to give appropriate identification for the MSC reference and the condition according to the system.
- Give the system name as the MSC name then draw a connected graph within a frame.
- Connected graph always start from the initial condition with start symbol.

➢ Trace Basic MSC for each MSC reference with the same MSC head as that of reference identification. While tracing Basic MSC:
- Try to find out the set of instances for each MSC.
- Try to find out the set of events for each MSC.
- Maintain the events order for each instance.
- Don't forget to trace the global condition within MSC.
- Start the chart with the (initial) condition (if exist).
- End the chart with the (final) condition (if exist).

B. Application

The scope of this sub section is to demonstrate how the concept of "Bridge" can be applied in the real world problem to capture and describe the software requirements of the system.

➢ *Case Study*

The Case Study is about the Automated Teller Machine (ATM).

➢ *Requirement*

ATM is an electronic device that can be used to deposit cash, withdraw cash, or request the available balance from the bank account without human teller.

User uses an ATM card to communicate with an ATM. The ATM equipped with a card reader slot having magnetic stripe reader for reading an ATM card, a keyboard and screen for interaction with the user, a slot for depositing envelopes, a dispenser for cash, a slot having printer for printing user receipts. Further, ATM will communicate with the bank through computer over an appropriate network.

User inserts an ATM card when the screen displays a standard Welcome Message. After verifying the card, system asks for Personal Identification Number (PIN). Then the system displays the service menu after verifying the PIN and the selected service is provided.

➢ *Preparing Software Requirement Specification*

After collecting the system requirements from the customer, the developer will try to convert this into the specification document as:

To use an ATM, user required to insert an ATM card (provided by the user's bank) into the card reader slot when the screen displays a standard Welcome Message (idle state). The card reader attempts to read the inserted card. Unable to do so will be informed to the customer and the card is ejected, the ATM once again becomes idle. Otherwise, information to enter the Personal Identification Number (PIN) was displayed.

Once the PIN is entered (four digits), the ATM verifies the PIN number from the bank. If the PIN is invalid, the user required to re-enter the PIN number, unable to successfully enter the PIN after three tries will causes the ATM to keep the card and advice the user to contact the bank, again becomes idle. The ATM displays the menu for valid PIN, which contains a list of the transaction options that can be performed. These options are:
- Deposit Funds
- Withdraw Funds
- Transfer Funds
- Balance Inquiry

The user can select an option and perform a transaction. When the transaction has been completed, the user is asked for another transaction. If required, the system returns to the main menu. Otherwise, ejects the card and returns to the idle state.

User may cancel the interaction with ATM at any time during the entering a PIN or choosing a transaction type, which results in eject the card and return the ATM in idle state.

A user is able to make a deposit on selection of deposit option, the user is asked to specify the account to which the funds are to be deposited and the amount of deposit. Further, asked to insert a deposit envelope.

A user is able to withdraw cash on selection of withdraw option, the user is asked to specify the account type from which the funds are to be withdrawn and the amount of the withdrawal. If the account contains sufficient funds, the fund is dispensed through the cash dispenser. Otherwise, the user is informed and asked to enter a different amount or cancel the transaction.

A user is able to make a transfer of funds between any two accounts on selection of transfer option, the user is asked to specify the account from which the funds are to be withdrawn, the account to which the funds are to be deposited, and the amount of the transfer. If sufficient funds exist, the transfer is made.

A user is able to know the balance of an account on selection of balance inquiry option, the user is asked to specify the account whose balance is requested. The balance

is displayed as well as printed on the receipt.

If the user does not make a move in 30 seconds in each interaction, the ATM ejects the card and return to the idle state.

Then, the developer will try to convert this textual format into visual format with the help of Message Sequence Chart (MSC) by identifying the list of reactive sub-system from the system requirements provided by the customer as:

- Insert
- Invalid
- PIN
- Check PIN
- Reject
- Menu
- Service
- Cancel
- Option
- Stop

Further, states are found as:

- Initial Condition
- Card validation
- Waiting for Pin
- PIN validation
- Waiting for customer's need
- Provide customer's need
- Asking about the other need.

After giving the appropriate identification for the selected sub-system and the state, consider them as MSC references and the conditions respectively, an HMSC for the ATM system is traced as:

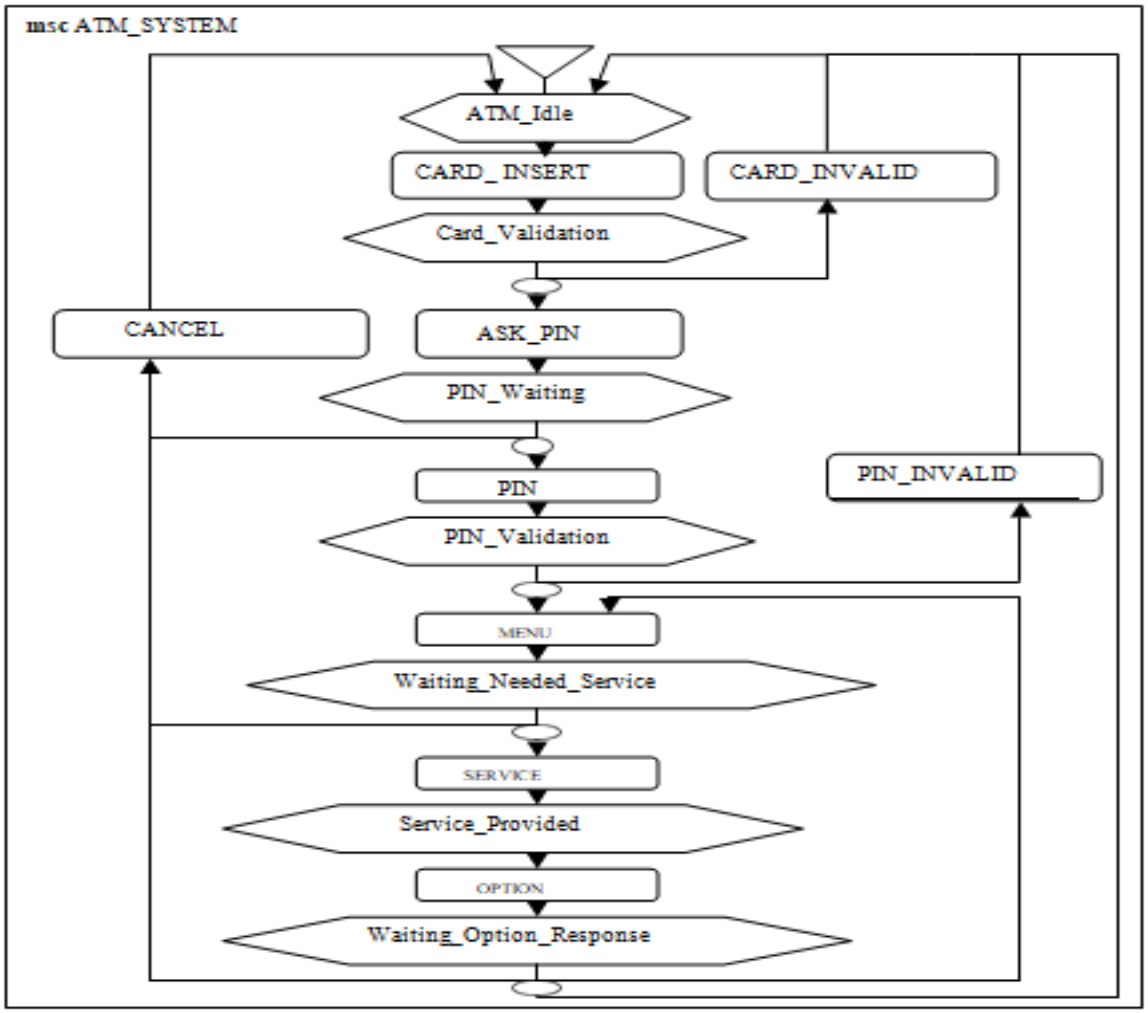

Fig 4:- HMSC of ATM_SYSYEM

After making a discussion, if the customer found to be satisfied, then the instances, events and order of events for each

MSC references are collected and basic MSC's are traced as:

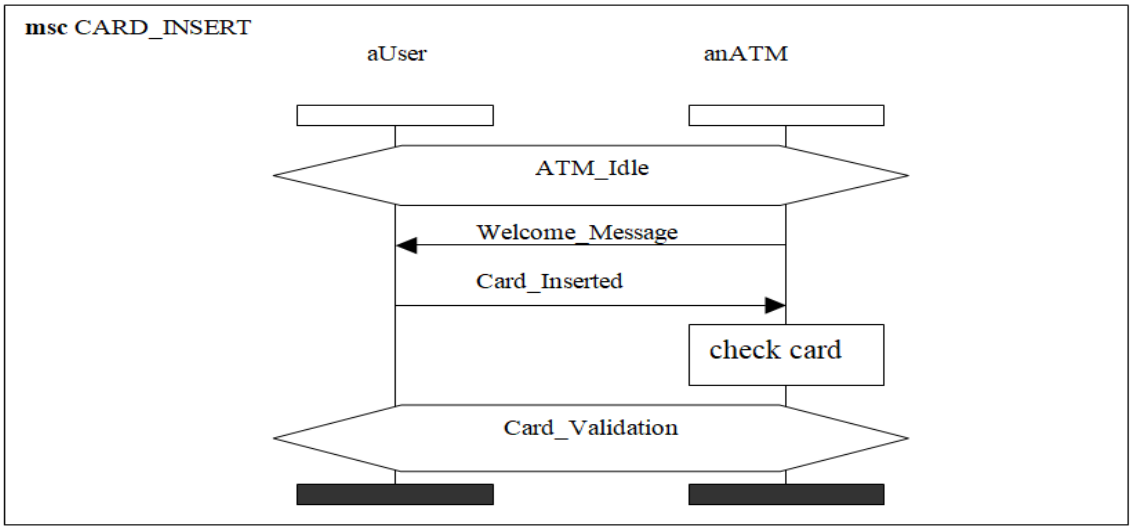

Fig 5:- msc CARD_INSERT

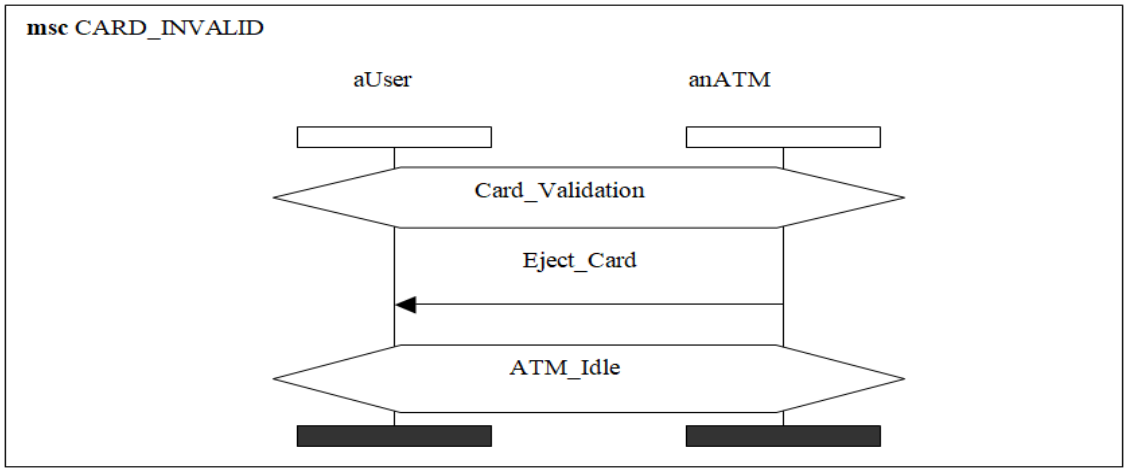

Fig 6:- msc CARD_INVALID

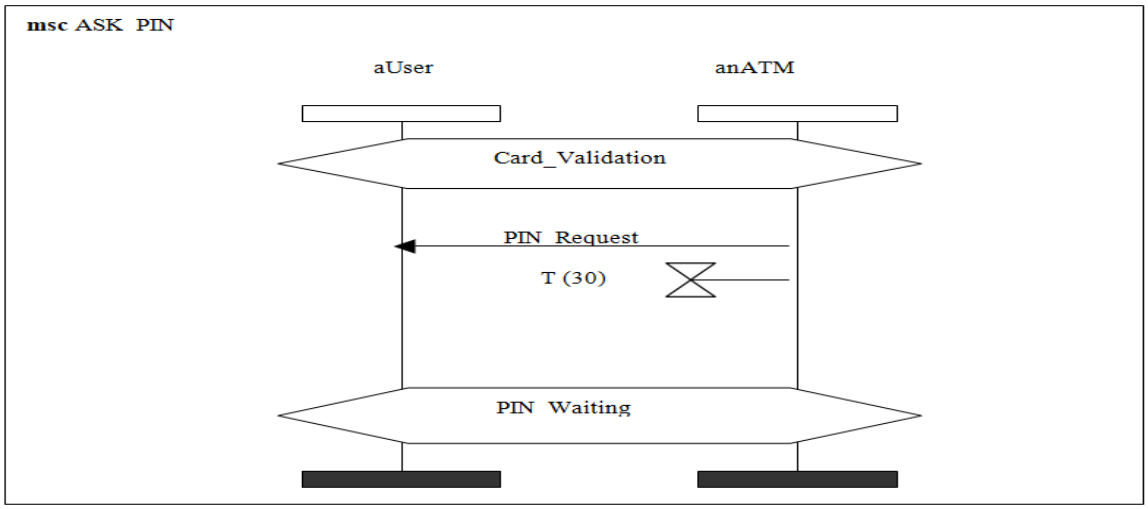

Fig 7:- msc ASK_PIN

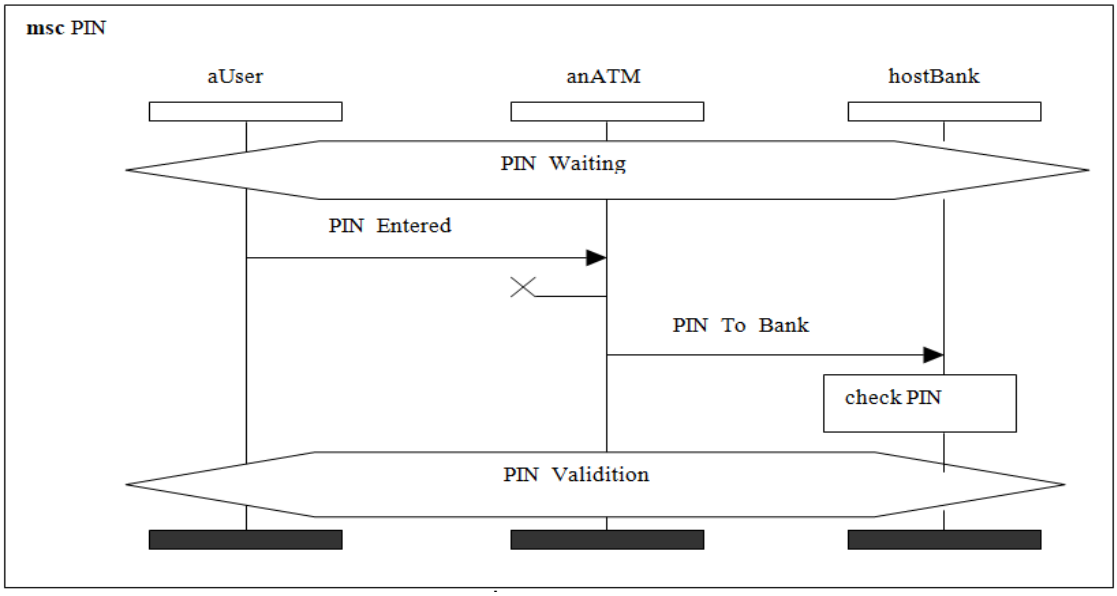

Fig 8:- msc PIN

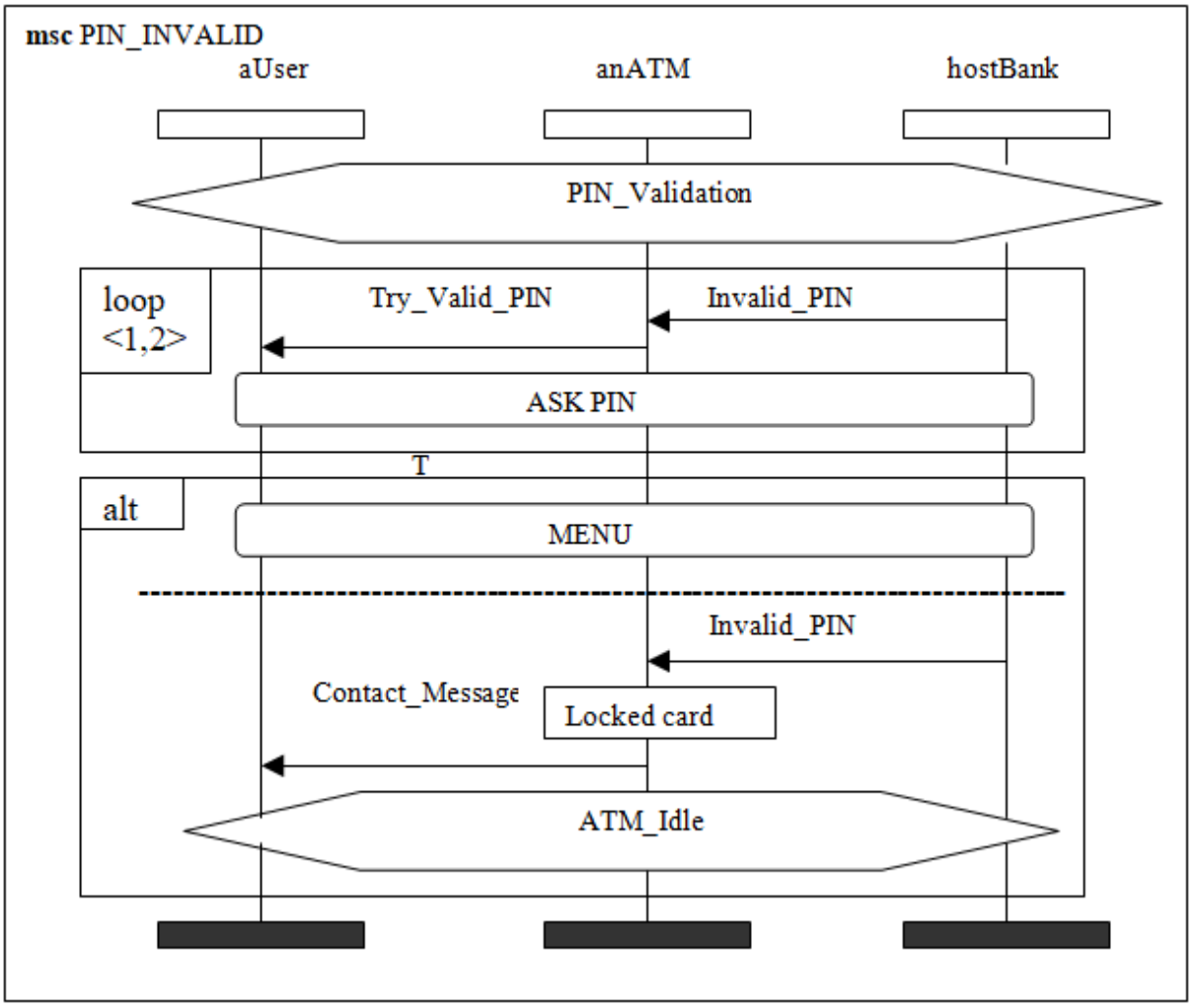

Fig 9:- msc PIN_INVALID

ISSN No:-2456-2165

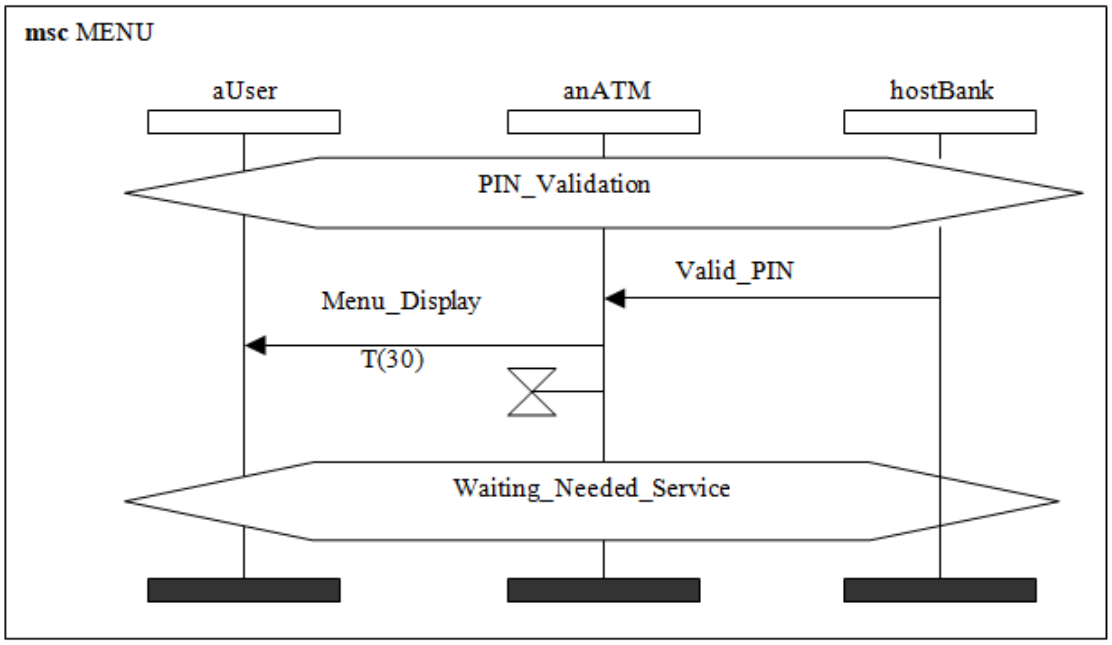

Fig 10:- msc MENU

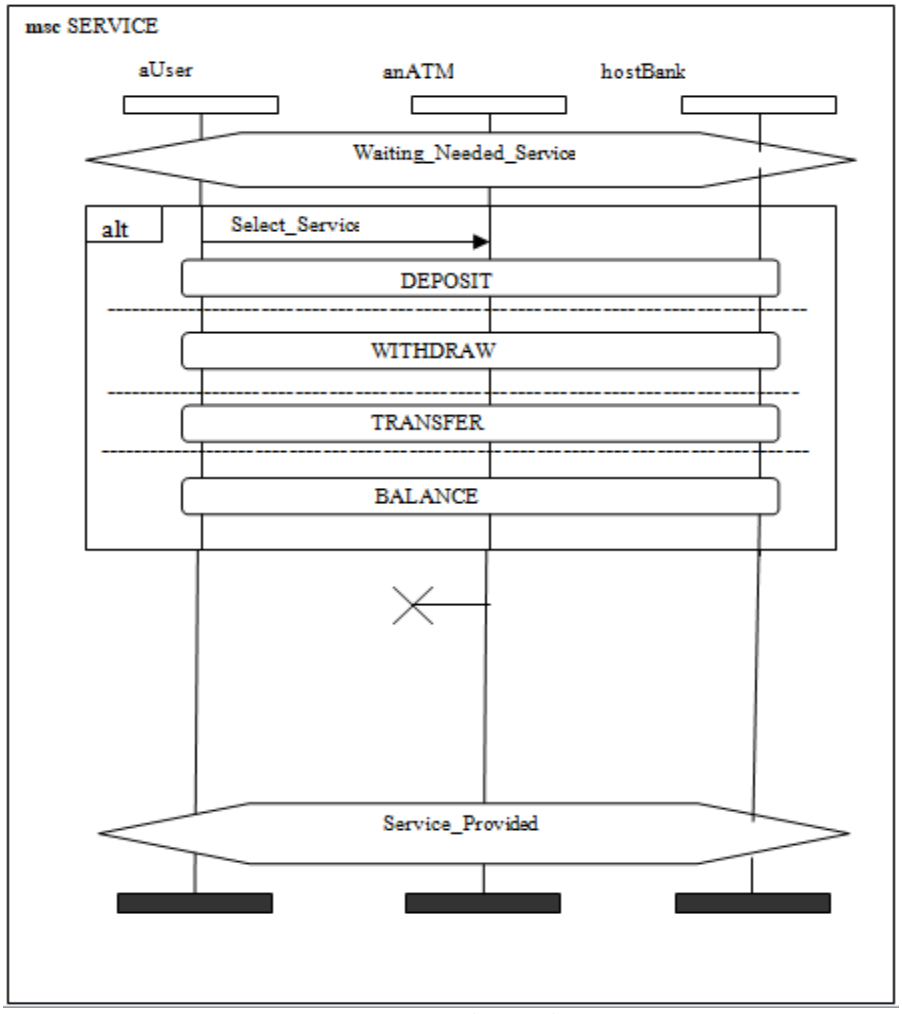

Fig 11:- msc SERVICE

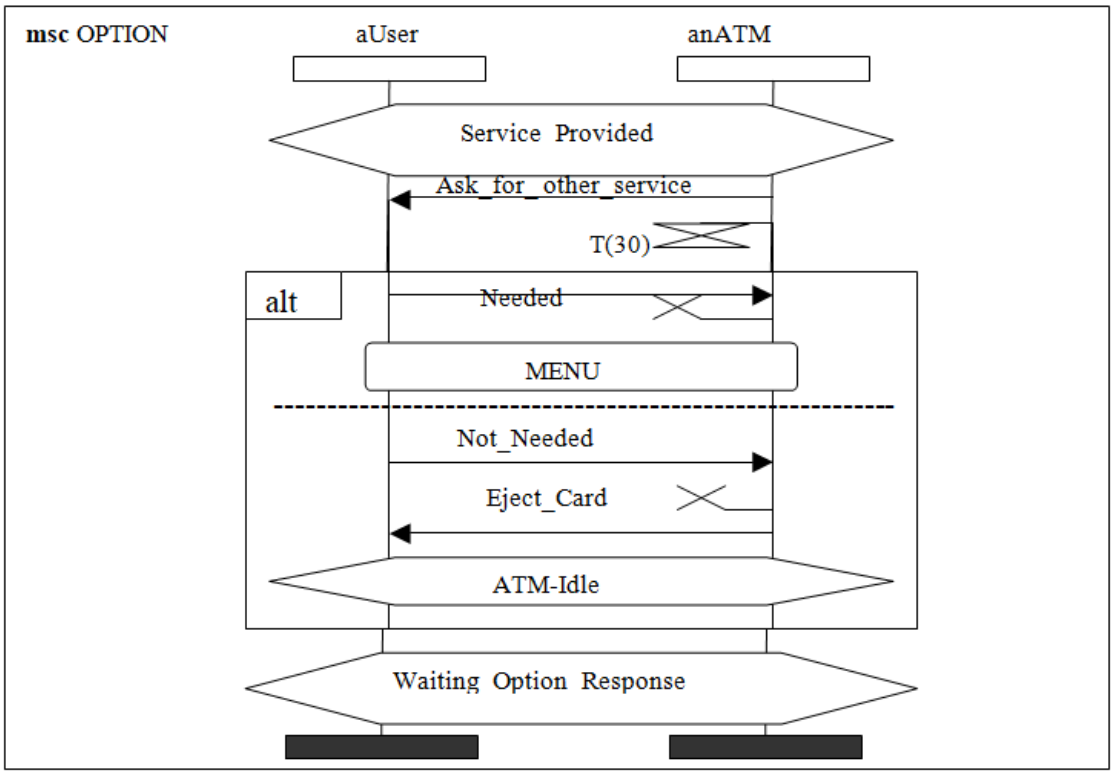

Fig 12:- msc OPTION

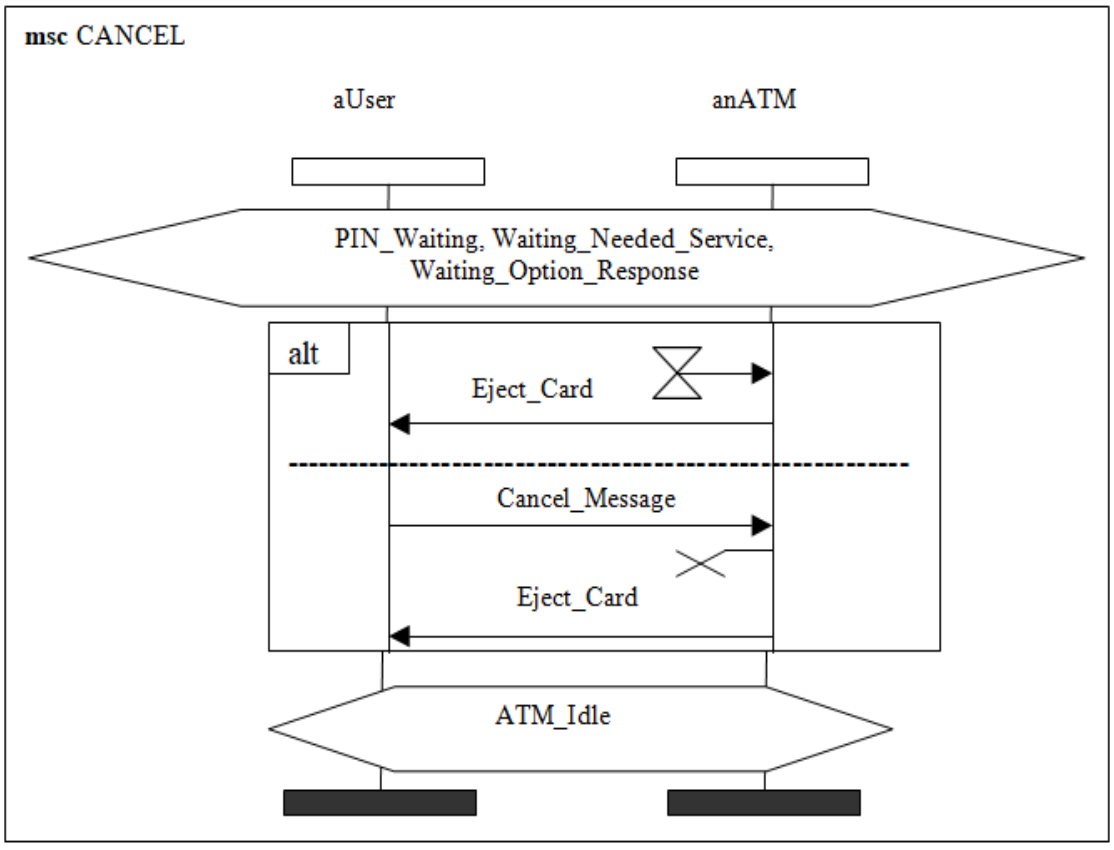

Fig 13:- msc CANCEL

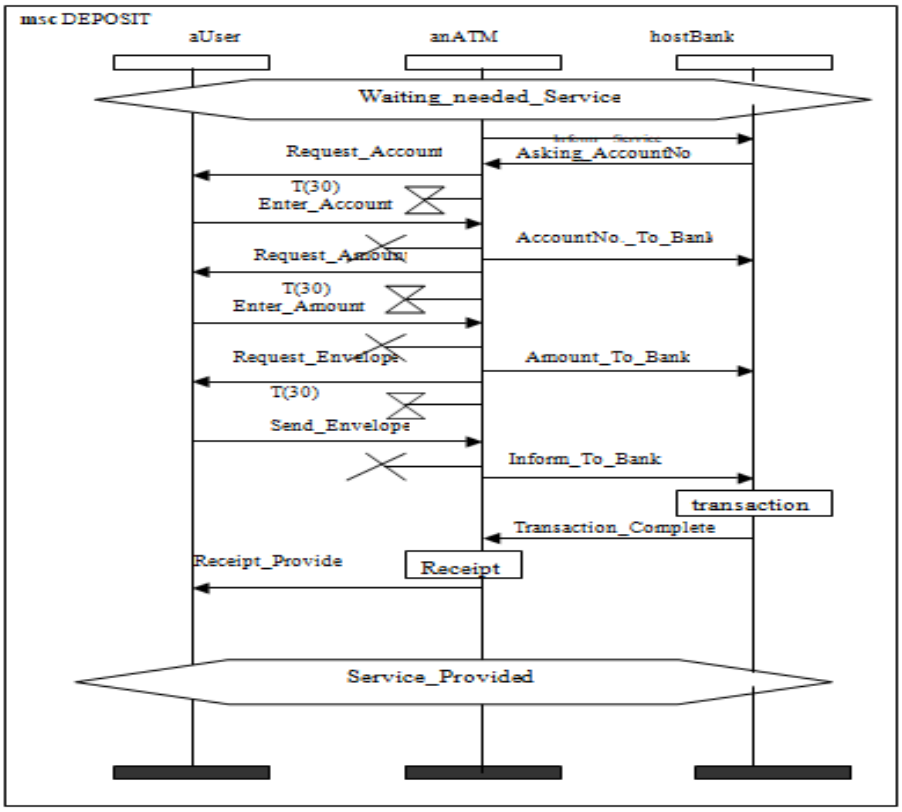

Fig 11.1:- msc DEPOSIT

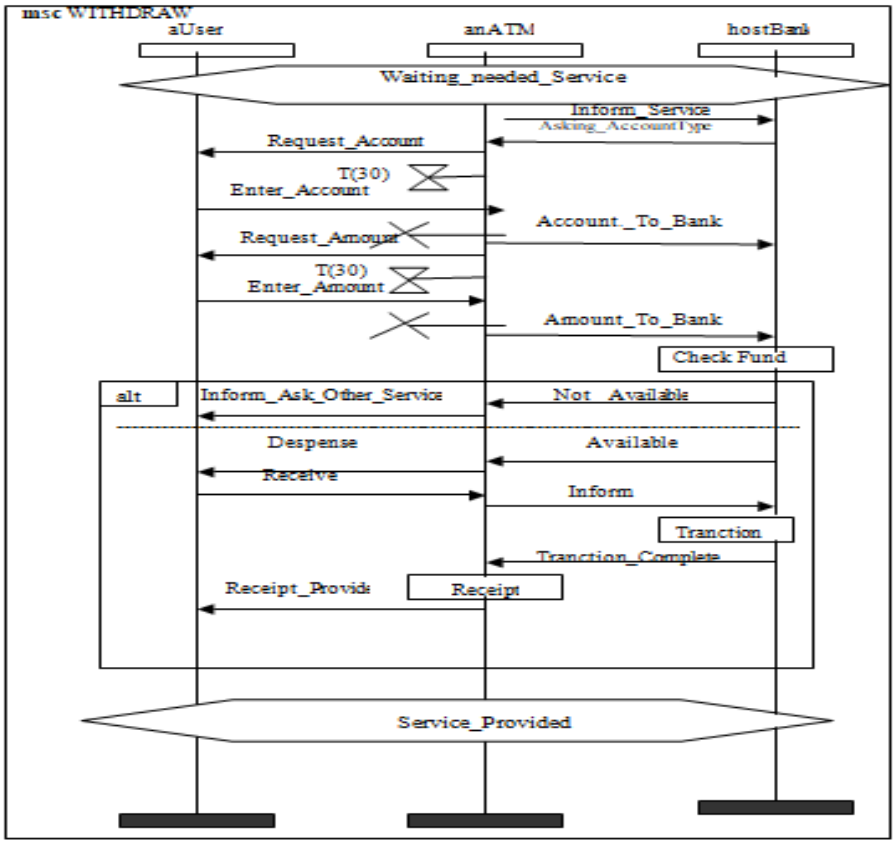

Fig 11.2:- msc WITHDRAW

ISSN No:-2456-2165

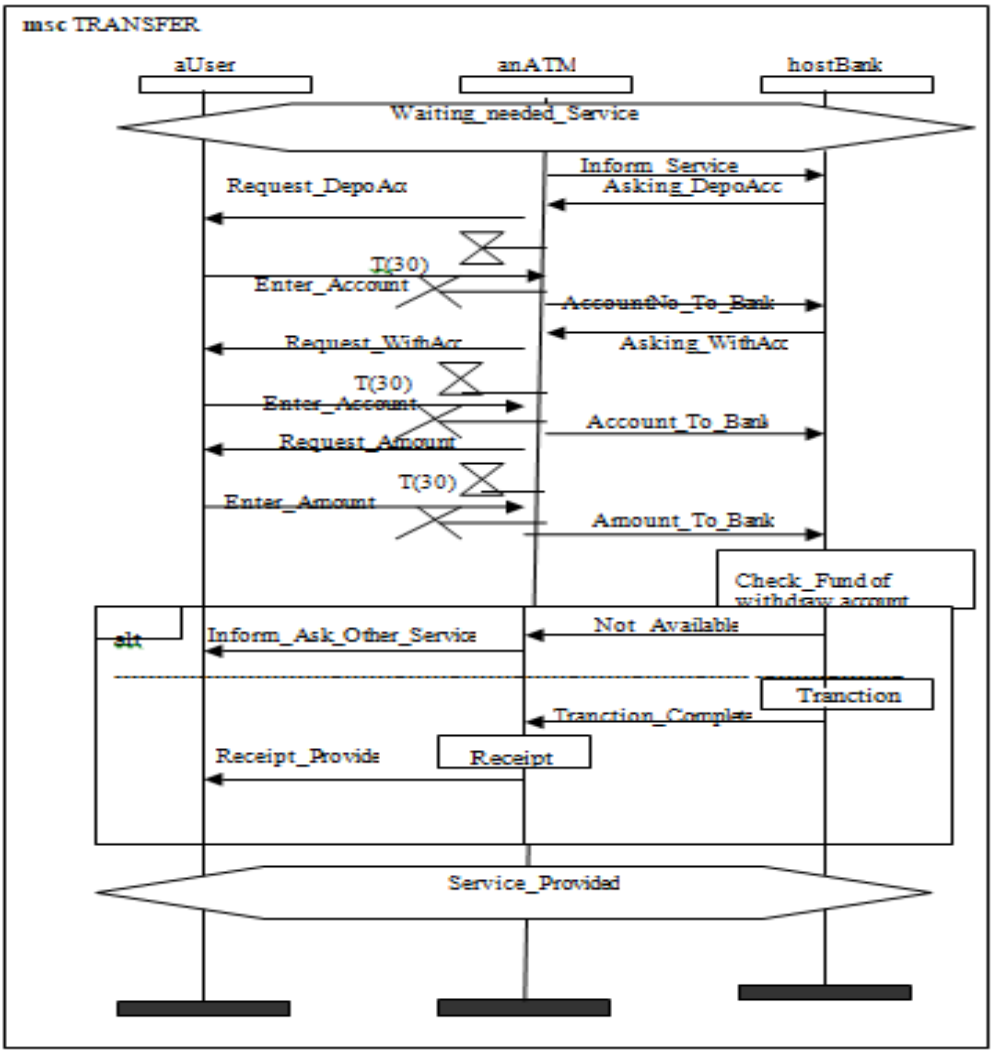

Fig 11.3:- msc TRANSFER

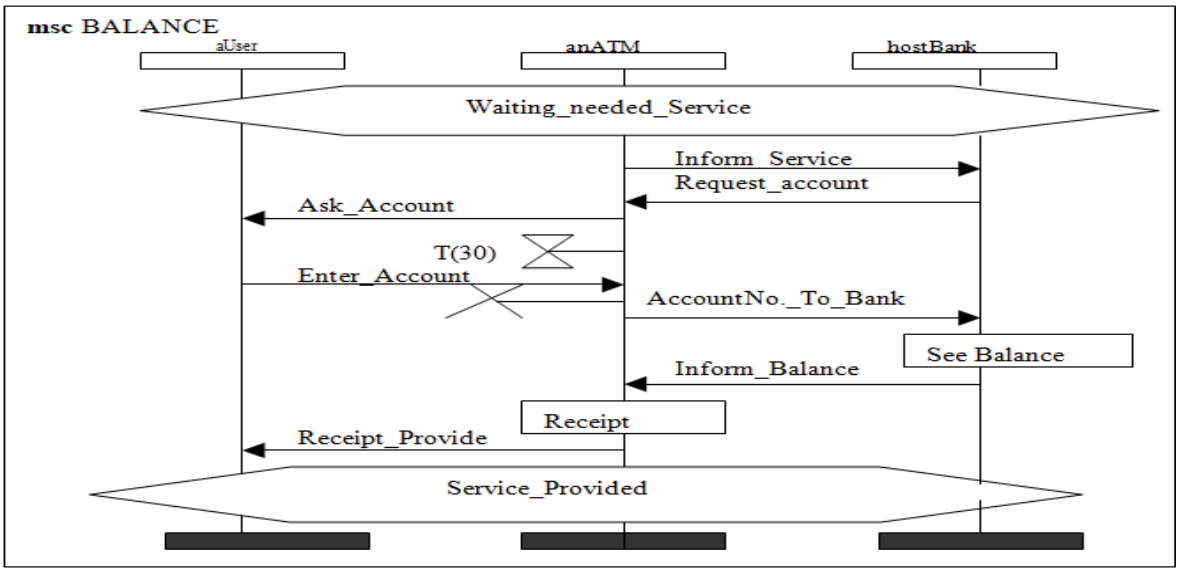

Fig 11.4:- msc BALANCE

These visual forms will help the developer to capture and understand the clear picture of the customer's need. If both are satisfied, then the developer proceed ahead on the way of development process considering these traces as an analytical view and the design matrices for the software system to be developed.

IV. CONCLUSION

MSC is very simple to understand and use. Developer can capture and interpret the software system requirements from the customer with the help of MSC easily. Since the MSC has the rich set of syntax and semantics, the developer can use these features in other phases of software engineering too.

MSC can be used to represent the functional as well as non-functional requirements of the system. It can also be used very easily to verify the requirement specification along with the customer because customers will not found difficult to understand the communication behavior of the MSC and will be able to evaluate their requirement in a very friendly environment. If the customer is not satisfied, he or she can immediately object the scenario. This makes the developer to achieve more. The simplicity feature of MSC encourages the customer to take the part in the development. Hence, the MSC reduces the gap between the developer and customers. In this way MSC is capable of building the bridge between the customer and developer.

REFERENCES

[1]. ITU-T, TELECOMMUNICATION STANDAR-DIZATION SECTOR OF ITU, *https://www.itu.int/ITUT/studygroups/com10/languages/Z.120_1199.pdf.*
[2]. The RISKS Digest, Forum on Risks to the Public in Computers and Related Systems, ACM Committee on Computers and Public Policy, Peter G. Neumann, moderator, Volume 1 Issue 2, Friday, 28 Aug 1985, *https://catless.ncl.ac.uk/Risks/1/2#subj1*
[3]. S. Leue, P. Ladkin, .What do Message Sequence Charts Mean?., *Proceedings of the Sixth International Conference on Formal Description Techniques, North-Holland, 1994*
[4]. R. S. Pressman, .Software Engineering -*A Practitioner.s Approach., McGraw-Hill 1997.*
[5]. Rajeev Alur, Gerard J. Holzmann, and Doron Peled. *An Analyzer for Message Sequence Charts. Software / Concepts and Tools, 1996.*
[6]. Grady Booch. Object-Oriented Analysis and Design. With Applications, Addison-Wesley, 2nd edition, 1994.
[7]. Ekkart Rudolph, Jens Graboski, and Peter Graubmann, "Tutorial on Message Sequence Chart(MSC'96).
[8]. Requirements document for an automated teller machine network, (aug 5, 1996), *http://www.cs.umd.edu/~mvz/cmsc435-s09/atm.pdf.*