Analysis & Modeling of Automated Teller Machine system using Formal Methods

Developed by

Sofia Kanwal

(98-FAS/MSSE/F05)

Supervised by

Dr. Nazir Ahmad Zafar

DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF APPLIED SCIENCES
INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD
2007-2008

Global ATM System

ATM

RTCA

D-E

24-2-11

Department of Computer Science
International Islamic University Islamabad

Final Approval

Dated: ____2008

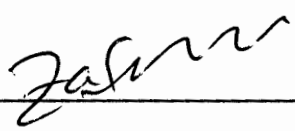
It is certified that we have read the project report submitted by Miss Sofia Kanwal (98-FAS/MSSE/F05) and it is our judgment that this project is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the MS Degree in Software Engineering.

Committee

External Examiner
Dr. Aamer Nadeem
Associate Professor,
Muhammad Ali Jinnah University,
Islamabad.

Internal Examiner
Adnan Ashraf
Lecturer,
Department of Computer Science,
International Islamic University,
Islamabad.

Supervisor
Dr. Nazir Ahmad Zafar
Principal Scientist,
Department of Computer and Information Sciences,
Pakistan Institute of Engineering & Applied Sciences,
Nilore, Islamabad.



Dedicated
To
My ever loving and Encouraging
Parents

The Dissertation is submitted to

Department of Computer Science,

International Islamic University, Islamabad

As a partial fulfillment of the requirement

For the award of the degree of

MS in Software Engineering

DECLARATION

I hereby declare that this project report, neither as a whole nor as a part thereof has been copied out from any source. It is further declared that I have developed the project and its report while working individually, and completed the report entirely on the basis of my personal efforts made under the sincere guidance of my Project Supervisor. If any part of this report is proved to be copied out or found to be reported, I shall standby the consequences. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other University or Institute of learning.

Sofia Kanwal

(98-FAS/MSSE/F05)

ACKNOWLEDGEMENT

I would like to express my gratitude to Almighty Allah who gave me the possibility to complete this thesis.

I gratefully acknowledge the support and prayers of my loving parents in each step of life. This is due to them that today I have reached up to here.

I am deeply indebted to my supervisor Prof. Dr. Nazir A. Zafar from Pakistan Institute of Engineering and Applied Sciences whose help, stimulating suggestions and encouragement helped me in all the time of research for and writing of this thesis.

I also want to thank my all teachers and friends for all their help, support, interest and valuable hints.

Especially, I would like to give my special thanks to my husband Mr. Qaddafi Khan whose patient love enabled me to complete this work.

Sofia Kanwal

(98-FAS/MSSE/F05)

PROJECT IN BRIEF

Project Title:	Analysis and Modeling of Automated Teller Machine system using Formal Methods
Undertaken By:	Sofia Kanwal (98-FAS/MSSE/F05)
Supervised By:	Dr. Nazir Ahmad Zafar Principle Scientist, Department of Computer and Information Sciences, Pakistan Institute of Engineering & Applied Sciences, Nilore, Islamabad.
Tools Used:	Z Notation Z/EVES toolset Microsoft Office 2003 Microsoft Visio 2003 Adobe Acrobat Reader & Writer
Operating System:	Windows 2000 Professional
System Used:	Pentium III Machine
Date Started:	September, 2006
Date Completed:	March, 2008

ABSTRACT

In this thesis, importance of formal methods has been demonstrated by applying them in the software development of safety critical systems and more specifically business critical systems such as automated teller machine (ATM) system. We have demonstrated the application and integration of formal methods in existing software engineering life cycle. We have used the model oriented specification language that is Z notation, to ensure the correctness, reliability and consistency of ATM system at analysis and design stage. In this way when we start implementation, we will have broader analysis and design at hand. Further more deep analysis of system at implementation stage will reduce the backward loops to previous stages. A formal model of ATM system at abstract level has been developed in Z specification language. This model covers both the static and dynamic aspects of ATM system. The requirements have been taken from the standard IFX 1.4 [21]. Main focus is on the security related issues and normal operations of ATM system have also been covered. The formal method, Z notation is based on set theory and first order predicate logic. Z notation is quiet easy to learn and use due to its simple mathematics and rich vocabulary of notations. The formal model is finally checked and analyzed using Z/EVES toolset. For analyzing, different proof techniques such as prove by reduce, simplify and rewrite have been used which helped us greatly in correcting the model.

Chapter No. *Contents* *Page No.*

References

ABBREVIATIONS

ATM	Automated Teller Machine
FM	Formal Methods
IFX	Interactive Financial Exchange
VDM	Vienna Development Method
CSP	Communicating Sequential Processes
V&V	Verification and Validation
UML	Unified Modeling Language
VDM	Vienna Development Method
LOTOS	Language of Temporal Ordering Specification

LIST OF FIGURES

Chapter 1:

Introduction

Chapter 1

Introduction

1.1 Overview of Thesis

In this thesis, importance of formal methods has been demonstrated by applying them in the software development of safety critical system and more specifically business critical system such as automated teller machine. For the development of any system either hardware or software, early stages of development are most critical and subject to errors. Hardware designers use specific languages to specify their design but in software engineering early stages like requirements engineering and requirements analysis have not yet got proper tools and methods support. There is a rare use of specification language and design language in software development. In software industry, highly qualified development of safety critical systems is a hot subject because safety of human, environment and protection of valuable assets are of vital importance. So there is a need to switch from adhoc and manual methods to rigorous engineering discipline. In order to contribute in this effort, we have demonstrated the application and integration of formal methods in existing software engineering life cycle. Integrating a formal method such as model checker into a hardware design is relatively easy because other tools are already a standard part of analysis and design process at many hardware companies [19] but this is not the case in software analysis and design. The use of proper specification language such as Z notation, which is a well known formal method, compels us to ensure the correctness, reliability and consistency at analysis and design stage, before we start the actual implementation of the software system. As already mentioned, the system under consideration in this research is Automated Teller Machine system. One of the existing software engineering approaches to analysis and design is the use of Unified Modeling Language (UML). This is an informal approach to system development as it has no mathematical foundation. In this thesis, UML use cases are used for analysis purpose and then this approach is integrated with the new one, which is the use of formal methods in terms of Z notation. Z notation [28, 29, 34] is based on set theory and first order predicate logic. A formal model at abstract level has been developed in Z specification language. This model is finally verified using a tool named Z/EVES [22] toolset.

1.2 Aim and Objectives

The aim of this thesis is to contribute in the effort for making the use of formal methods common in industry for the development of safety critical systems.

Objectives:

We have tried to achieve the following objectives in this research

(i) Applying formal methods to safety critical systems.

(ii) Developing the model of ATM system which is more secure, using Z notation.

(iii) Integration of formal and informal approaches.

A particular feature of this thesis is that we have introduced tool based validation of safety properties.

1.3 Motivation

Formal notations have now reached the level of maturity that some of them are being standardized (e.g., LOTOS, VDM and Z) [35]. Formal methods are ideal for ensuring the correctness and reliability of system. Although formal methods can be used at any stage of software system development but they are particularly suited at the initial stages of system development like requirements analysis and design. This is because capturing the errors and inconsistencies at initial stages could greatly effect the time and cost spent in later stages. If the correctness and completeness of requirements will not be guaranteed at initial stages, it is likely that inconsistencies will propagate to later stages which will become more costly and difficult to cater at that time.

It is fair to say that the practical use of formal method is limited till now, but this is not due to that formal methods have some underlying drawbacks but due to some misconceptions as discussed in [3]. Another reason for hindrance to use formal methods is that it is a new technique and software analysts, designers and programmer are reluctant to switch from existing techniques to this new one.

There is one class of computer systems, which are safety critical systems, where reliability is of paramount concern. This is the class of systems where failure cannot be tolerated. Following systems come under this category.

1) Avionics
2) Medical systems
3) Nuclear Power Plants
4) Automated Teller Machine systems
5) Secure Mobile Communication systems
6) Railway Signaling systems

Our work includes demonstrating the application of formal methods to business critical system that is Automated Teller Machine system (ATM). The reason that why we have selected ATM system is that it is a business critical system, and its incorrect or unexpected behavior may lead to large scale economic loss. So there is a need to assure the secure transactions through this machine. The formal specification model has been developed using Z. It will be seen that it will offer more clarity, correctness and completeness at the beginning of development life cycle.

1.4 Automated Teller Machine system (ATM)

An Automated Teller Machine is an electronic computerized device that allows customers to directly use a secure method to access their bank accounts, order or make cash withdrawals and check their account balances without the need for a personal cashier. Many ATMs also allow customers to deposit cash or cheques, exchange currency and transfer money between their bank accounts.

ATM system is a safety critical system because its malfunctioning may lead some one to great economic loss.

1.4.1 Components of ATM system

The typical ATM system is made up of following components:

- Magnetic card reader

  ATM card will be inserted in the magnetic card reader and it will read the card information for its expiry and account detail.

- Money Cassette

  This is a place holder to keep currency notes inside ATM machine.

- Display

  The display screen will be available to provide the display facility.

- Touch screen

  Touch sensitive screen will be there for selecting from menus.

- Real time clock

  This clock will work internally to keep track of transaction date and time information.

- Receipt Printer

  Automatic receipt will be generated after the transaction is complete.

1.5 Related Work

ATM system has been an important research area. Due to its business critical nature many researchers have been contributing in modeling of the system.

Martin Giese et al. [40] built a semi formal model of the withdraw use case of ATM system with the help of UML and OCL. They have tried to bridge a gap between formal and informal specifications. They relate the use cases with OCL. This paper does not consider all the issues of withdraw money by using ATM in considerable detail. Yingxu Wang et al. [55] presented the formal model of ATM system using Real Time Process Algebra (RTPA) in order to ensure correctness and dependability. Static and dynamic aspects relating to ATM system functionality are covered in the above mentioned paper. Here full ATM system has not been considered but the system is taken as a case study and application of RTPA have been demonstrated. Functionality of ATM as part of big banking system is presented in B method by Martin Buchi [37]. B method is a state based formal method built on set theory and predicate logic. The safety critical issues are not particularly addressed in the paper. B method is for the development of program code from a specification in the abstract machine notation. As we are at the specification level so B method is not suitable for our purpose to specify system formally [20]. Jeffrey Douglas et al. [33] specified the ATM system specification at abstract level in formal specification language Aslan. Aslan [33] is a state-based specification language built on first-order predicate calculus with equality. Jeffrey has considered the abstract model of ATM system but did not address the security related issues particularly. David W. Bustard [13] discussed the general issue of providing support for change to formal models. He formalized the very simple model of ATM in LOTOS [23] which is a formal specification language. LOTOS is suitable for considering control issues. As concurrency is not addressed in this research so it is not suitable for our purpose. Also

LOTOS does not have a widespread use in industry due to lack of rich variety of techniques and methods, and absence of powerful tools support [47]. Beum-Seuk Lee et al. [5] considered some aspects of ATM system and formalized in formal specification language Two Level Grammar (TLG) in order to bridge a gap between natural language representation of system and formal representation of system and to promote the reusability of requirements document.

Maritta Heisel et al. [39] model some aspects of ATM system in Z notation. The model is not the representation of full ATM system and also the safety critical issues have not been addressed in it. Our model reflects real world requirements as taken from standard set of specifications specified in IFX (Interactive Financial Exchange) version 1.4 [21] and mainly considers the security related issues.

1.6 Formal Methods

Formal method is that area of computer science that is concerned with the application of mathematical techniques to the analysis, design and implementation of computer software and hardware [29]. Formal methods provide frame work with in which one can specify, develop and verify system in a systematic manner [29]. Formal specification is the use of mathematics to specify the desired properties of a computer system [29]. Softwares currently used in computers that have been developed using informal techniques has itself become so complex that it is not trust-worthy and has caused human injury and death as a result. List of incidents caused by poor system development is provided in [44] and it is updated annually.

Main problem with informal specification is the inherent ambiguity of textual description. Mathematics can eliminate such ambiguities as it has the power to express the complex properties succinctly. The work of author on formal methods presented in [50, 51] also confirms the strength of this emerging technique.

1.7 Problem Statement

Our work includes:

- Formalizing the selected standard set of requirements in Z notation as specified in IFX standard version 1.4 [21].
- Focusing the safety critical and normal operation issues of ATM system.

- Model checking of resultant formal model using Z/EVES tool set [22].

1.8 Proposed Solution

As presented in related work, with respect to existing model of ATM systems in formal methods, there is very little contribution of researchers. So we have decided to make a full fledge abstract formal model of ATM system. As the ATM system is a safety critical system so it should be dependable. Dependability is defined as the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers. Dependability includes the following attributes of a computing system [1].

Availability: readiness for correct service;

Reliability: continuity of correct service;

Safety: absence of catastrophic consequences on the user(s) and the environment;

Security: the concurrent existence of (a) availability for authorized users only, (b) confidentiality, and (c) integrity.

All these attributes combine make the system dependable. As dependability is a very wide attribute to cover, it is not possible at this level of research to consider all these aspects to make a system dependable so we have selected the security attribute of the system only. The ATM system will be secure in a sense that it will be available only to authorize users and it will provide confidentiality and confidence to its users.

So the focus in proposed model will be on most of the service features and the security features of ATM system.

Another important aspect of model will be that it will not be a hypothetical representation of the system but the requirements are taken from the internationally accepted standard for ATM and POS (Point Of Sale) systems development called IFX (Interactive Financial Exchange) [21]. IFX is a mature, well-designed specification developed by the financial industry and leading technology providers. It defines the architecture that supports efficiency, extensibility and security. It is a defining standard between ATM/POS terminals and authorizing systems. IFX is an open standard that is rich in features. It already contains many features that are required today and in the future for use at ATMs, for example deposit, withdrawal, balance inquiry, transfer, bill presentment, bill payment, valuable media service, terminal management and monitoring [21].

- Keeping in mind the standard set by IFX, we will develop a formal model of ATM system, to provide better services and to make the system more secure.

Although our model is better than the existing models as it covers most of the security issues and common services but at this level it will not cover all the aspects of ATM system in full detail. The system at abstract level will be presented and the further refinement is left for future work.

1.9 Methodology

The methodology used in research is as follows. The flow of work will be first to describe the system in informal way using UML and then modeling the system in formal way using formal specification language. Finally the tool based analysis will be done. Here we used some of UML structures for initial abstract system modeling. Use case diagram is used for functional requirements understanding and representation. For more refined and detailed system model we used formal methods. Formal methods have mathematical foundation that are more precise, and in fact more rigorous and robust in stating software system properties. Z specification language is used here for formal system modeling. Accordingly, its use is especially attractive for systems in which errors are particularly costly. The tool for the analysis purpose is Z/EVES tool set which provides the syntax and to some extant semantic checking of the system.

1.10 Tools to Be Used

Following tools will be used for formal specification of ATM system.

1.10.1 Z Notation

Z is a model based specification notation used at analysis stage of system development. It is based upon set theory and first order predicate logic [34]. The characteristic features of Z include: (i)It allows type definition even at specification level. (ii) It allows refinement of the model, so that each refined model will be closer to implementation. (iii) It has a powerful structuring mechanism. There are about 90 techniques of formal methods but we chose Z notation due to following reasons.

- Z is considered suitable tool for the specification of systems which are not complex. As we are developing the model at abstract level which doesn't involve any inherent complexity, so Z is a best option at this stage of system development.
- Z is based on basic set theory and first order predicate logic which can be learnt by a person who has very little idea about mathematics. As integrating formal and informal approaches is one of the objectives set for this research, so at this stage, it is not appropriate to use any formal technique which involves complex mathematics.
- Z has got wide tool support [22].
- Z has been standardized and one of the oldest technique for formal specification.

1.10.2 Z/EVES

Z/EVES is a tool for analyzing Z specifications. It can be used for parsing, type checking, domain checking, schema expansion, precondition calculation, refinement proofs and theorem proving. The specifications developed in Z in this research will be validated using this tool.

1.11 Plateforms

Windows 98, 2000, XP.

1.12 Structure of Thesis

Chapter One, "Introduction": This chapter, of which 'Structure of Thesis' is last heading is going to finish. It gives the introduction of the research. If firstly highlights the aims and objectives and motivation of this thesis. A brief introduction of Automated Teller Machine system, use of formal methods in these systems, problem statement, and highlights of proposed solution are given in this chapter. At last methodology used and tools needed to complete this research have been mentioned.

Chapter Two, "Formal Methods": This chapter is about formal methods and motivation for their use, role of software in safety critical systems, applications of formal methods and their classification. It briefly explains the formal method used in this research i.e. Z notation.

Chapter Three, "Automated Teller Machine system": An introduction to ATM system is given in this chapter. Current advancement in technology with respect to ATM system is also presented. Brief requirements analysis has also done in this chapter on the basis of which we develop the formal model in next chapter. UML diagrams are used to represent the informal model of system.

Chapter Four, "Formal Model": Here the formal model of system in Z notation is presented. It represents the static and dynamic aspects of system in detail.

Chapter Five, "Model Checking": It includes the tool based validation of system's specification using the Z/EVES Tool set.

Chapter Six, "Conclusion and Future work": This chapter summaries the whole research. It also presents the contribution to the subject. This is concluded by some directions of future work.

Chapter 2:

Formal Methods

Chapter 2

Formal Methods

2.1 Introduction

In computer science and software engineering, formal methods are mathematically based techniques for the specification, development and verification of software and hardware system [48]. The use of formal methods for software and hardware design is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analysis can contribute to the reliability and robustness of a design [8]. These definitions show the worth of formal methods. Formal methods can be applied to the every stage of software development from specification to testing and maintenance. This is a common thinking about formal methods that their use is expensive both in terms of time and human effort. People normally hesitate to use them for software development. To minimize the misconception that formal methods are very difficult to apply and work with, they have been classified into light weight formal methods and heavy duty formal methods [9]. Light weight formal methods use very basic concepts of mathematics and need no special expertise in mathematics. Also they allow the partial specification of system properties. A person having little knowledge of mathematics can apply these methods for his requirement. The example of these methods are the Alloy object modeling notation [14], Denney's synthesis of some aspects of the Z notation with use case driven development [49], and the CSK VDMTools [52].

Heavy duty formal methods make rich use of mathematics. Only an expert mathematition can apply them. Examples of such methods are theorem provers. These methods can be very expensive so only be used were cost of failure is very high (for example development of safety critical system etc.).

With in formal methods, there is a term "formal specification" which means to specify the requirements of system in mathematical terms so as to make it consistent, precise and unambiguous. Formal specification describes what the system must do without describing how it should be done. The cost effective use of formal methods is formal specification because it is used at very basic level of system development and

secondly because it didn't use very complex mathematics. A formal specification can serve as a single, reliable reference point for those who investigate the customer's needs, those who implement program to satisfy those needs and those who write instruction manuals for the system [27].

2.2 Motivation for use of Formal Methods

A software system may fail to perform as expected because the software was not able to uncover design errors. For a system to be reliable and secure, these potential causes of failure must be handled.

Flaws in early stages of software development i.e. analysis and design are greater threats for system reliability. So the use of formal techniques at early stages of system development can help reducing these errors and ultimately increasing the confidence in system reliability. There are several reasons that we have selected to use formal methods in our research.

1. To improve the quality of whole development process
2. To improve the reliability of system
3. To reduce specification errors
4. To improve requirements definition
5. To understand design clearly
6. To be certain that the design and implementation are error free
7. To meet customer requirements

2.3 Role of FM in Safety Critical systems

Formal methods have been used and under continuous research in Europe, especially in U.K [38]. Safety critical systems make up a minority of industrial applications using formal methods [45].

Examples:

There are many examples of practical use of formal methods especially more demand to use formal methods is from defense sector [24]. Here are some of the examples quoted from many areas.

1- An early example of use of formal methods was the SIFT project, which is the most substantial US experiment in the safety critical sector. SIFT [31] is a computer controlled aircraft system which was commissioned by National Aeronautic and Space Agency (NASA). The safety requirements of SIFT were very stringent. Formal methods were used to minimize the failure rate of aircraft during its flight.

2- In 1988 GEC Alsthom and METRA Transport started working on computer control signaling system for controlling RER commute trains in Paris. Formal methods were used for formal specification and verification of code. For this purpose Abrial's B language [45] was used and proofs were done manually. Although this project consumed substantial amount of money but it was a successful application of formal methods contributing safety of system.

3- There are many types of medical equipment of which correct and precise functioning is vital for saving human life. In many cardiac care instruments formal methods have been used to enhance the quality of product. One example is [4] manufacture of bedside instrument which is used to monitor vital signs of patients in ITC (Intensive Care) units. The embedded software underwent formal specification to enhance the product quality. The development team found that the use of formal specification resulted in improved quality and minimizing ambiguity and inconsistencies which arise using informal approaches.

4- The wide safety critical implication is the control of operations concerning the storage and use of explosive articles.

5- The verifiable integrate processor for enhanced reliability is a microprocessor developed for safety critical application. The HOL (Higher Order Logic) has been used to verify parts of processor.

2.4 Applications of Formal Methods

Formal methods have wide application both to hardware and software. Software development is a lengthy and systematic process and there are many stages involved in its full development from analysis through design, coding, testing, implementation and

maintenance. FM plays its role by involving in every stage of development. There are different types of formal methods that can be used at different levels of abstractions.
The clean room approach of software development incorporates FM in its process. The normal testing phase is replaced by certification and software is checked for absence of errors, rather than correcting errors. Let us see the role of FM at very stage of software development.

2.4.1 Requirements Specification

This is the first stage in software development to work with. After gathering the requirements from real world, there is a need to specify them in a systematic and clear manner, so that they do not remain ambiguous and become precise and meaningful. For achieving the clarity and unambiguity, one good option is use of formal methods. There are different formal languages which have been proposed for this specific purpose and which are expressive enough to formalize all the real world requirements of the system. The most commonly used formal language is Z notation. It has got wide tool support and is enriched with mathematical notations. The importance of the use of FM at this stage is that, errors and inconsistencies will be uncovered early in the development and the chance to propagate these errors will be minimized. If the errors will not be caught at this stage, it will become very expensive to correct them later. One statistical calculation is that a modification in service can cost up to 1000 times more than a modification at the requirements stage, even worse two third of all errors are made at the requirements stage [2]. So it is very worthwhile to get the benefits from FM at this stage of software development.

2.4.2 Design

In design stage the refinement of requirements is done. For this purpose, we need rigorous refinement method. There are different formal methods which offer low level of abstraction and which are closer to implementation. VDM-SL is a formal language which can be used at design stage. This language also allows automatic transformation to code for most of the properties of the system. FM used at this stage is bit complex than the FM used at specification stage. So if safety is not a prime

concern, the user has the option to use or not use the FM at this stage. In hard real time systems, there is a strict requirement of achieving desired response in all circumstances, so using FM in these systems can help ensuring the correctness of the design.

2.4.3 Implementation

Implementation stage has already formalized in a sense that the computer languages have mathematical foundation and have proper syntax to follow. In [15] the author claims that programming itself is a formal specification, so when we do the formal specification two times, one at requirements level and one at implementation stage, it reduces the chances of errors.

2.4.4 Documentation

Documentation is very important part of software system. Software is not considered complete with out added documentation. Changes in documentation will reflect changes in software. If documentation is formalized, there is less chances of errors and ambiguity. In case of safety critical systems where timing issues are important, method of documenting them become significant [12]. FM provides a precise method of recording system functionality and thus used as a powerful aid to documentation. The documentation will contain the requirements and system functionality, written in a specific formal notation as well as non formal description will also be there to increase the clarity and understandability.

2.5 Classification of Formal Methods

There are five broad categories of formal methods [54] discussed as follows:

2.5.1 Model based

- **Z notation**

  It is model oriented formal method most commonly used. It is based on set theory and first order predicate logic [28]. Z is used at requirements specification level

and provides various levels of abstraction. It is also used for behavior specification of abstract data types. In Z notation, the whole specification is divided into chunks which are called schemas and which can be considered separately piece by piece. Actually the whole specification is an ordered collection of schema definitions and axiomatic descriptions. Each schema has its own name with its two portions. One for the description of variables and data types and other is the predicate part to describe the constraints over the variables. The schemas are combined to produce full system description. The Z has got power full tool support. In this thesis, the Z specification is analyzed for consistency and completeness through a tool called Z-EVES Tool set.

- **VDM**

VDM stands for Vienna Development Method [7]. It is another model based formal method used for specification and design purpose and most similar to Z notation. Like Z notation, it is us used for specifying behavior of abstract data types and sequential programs. The behavior is specified in terms of pre conditions and post conditions. There are no specific notations for specifying timing behavior, so if one needs to specify timing constraints, special features needs to be added in specification. The level of abstraction supported by VDM is lower so more specification detail near to implementation can be added. With the help of VDM, formal specification is constructed and refined stepwise until we reach the level of concrete source program.

- **B-Method**

It uses the abstract machine notation for the description of target system [36]. The B method is considered complete because it has strong tool support and it gives abstract machine specification, refinement, composition and their proofs. B method does not provide guideline for design decisions, testing and inspection.

2.5.2 Logic based

- **Hoare Logic**

Hoare Logic is extension of first order predicate calculus [16] that includes inference rules. Hoare Logic is best suited for low level specification. High level specification is a bit difficult using it. It shows the consistency between the program and its specification so it is considered a suitable mean in first stage of reverse engineering. Hoare Logic has no real time feature and it is used as a successful formal method in various applications that need program verification and mathematical proof.

- **Temporal Logic**

Temporal Logic has its own origin and it was used first time to analyze the structure of time [42]. It supports the specification of real time applications. Temporal Logic is state based and different time operations are used for system description depending upon time is linear, parallel or branching.

2.5.3 Property based

- **Larch**

It is a property oriented method for specification of sequential programs and abstract data type [30]. It was developed at MIT and Xerox PARC to support the use of formal specification. One goal of Larch was to support different programming and imperative languages. Larch specifies the state dependent behavior by giving preconditions and post conditions for operations. State independent behavior is specified by providing declarations for operations. The declaration consists of name of operation and the type of its input and out put arguments.

2.5.4 Process algebra

- **CSP**

  CSP stands for Communicating Sequential Processes [6]. CSP is for concurrent systems and it enables the description of entities that have properties that vary over time. The dynamic realities can be modeled by CSP. CSP specification is viewed as a system of processes executing independently, communicating over unidirectional channels and synchronizing over some particular events. Some times parallel processes get synchronized when one process sent message and other receive that through channel.

- **LOTOS**

  LOTOS (Language of Temporal Ordering Specification) was developed to define implementation independent formal standards of OSI protocols [23]. LOTOS has two distinct parts, one has its origin form CSS and CSP for describing control events and other has its origin from ACT ONE which is for the description of abstract data type. By combining these two, LOTOS has strong ability to describe the statics and dynamics of a given system. This formal language is not suitable for real time applications. For that purpose Timed LOTOS has been proposed which has not got much popularity.

2.5.5 Net based

As graphical notations are easier to understand and comprehend that is why many graphical notations have been developed. These notations are combined with formal semantics to bring special advantage and clarity to system development.

- **Petri net**

  Petri net is considered one of the first formalism to deal with concurrency and non determinism between events [53]. It has an advantage over other formal languages

that it provides graphical representation along with formal semantics which are easy to comprehend for user. There are different types of Petri nets like ordinary Petri nets, colored Petri nets and timed Petri nets. They can be used to model hardware, software and human behavior. In some situations, ordinary Petri nets cannot deal with fairness and data structures.

- **State Charts**

State Charts provide the mechanism for abstraction which is based on finite state machine [25]. They support top down system development method and are used to specify state transitions in reactive systems [10, 11]. Reactive systems cannot be described in terms of simple functions that map inputs to out puts. Safety critical systems are reactive systems.

Chapter 3:

Abstract Model

Chapter 3

Abstract Model

3.1 Introduction

Automated Teller Machine system

Automated Teller Machine also called ATM is a hardware device, with embedded software, used by banks for account transaction processing. The general way of processing through ATM is that user inserts a special card, called ATM card into the machine. This card contains the information like username, account number, bank information and user identification number which is transferred to bank's central computer. For the purpose of security, the user is required to enter the personal identification number (PIN) mostly of four digits. This confirms the identification of user. The computer than permits the user to perform transaction. Most ATMs withdraw cash, deposit money, transfer money from one account to another, exchange currency and provide account information on user demand. Many banks have formed nation wide networks so that customer of one bank can use an ATM of another for required transaction.

The chief conceptualist of ATM was Don WetZel, an idea he thought of while waiting in line at a Dallas Bank.

The first ATM was installed in 1969 by chemical bank at its branch in Rockville Centre, N.Y. A customer using an ATM card was dispensed a package containing money.

3.2 Requirements Analysis

The ATM system under consideration in this research provides following services. These requirements are taken from the standard set of requirements specified in IFX specification version 1.4[21]. The requirement number as mentioned in the specification document is also written here for keeping reference.

3.2.8 Account balances

4.3.5 Sign in by PIN

6.3.15 credit card account transaction record

6.3.25 Debit card

6.3.26 Debit card authorization

6.3.27 Credit card authorization

6.4.1 Balance inquiry

6.4.12 Credit card account transaction

6.10.1 Debit card reports

6.10.2 Credit card reports

3.2.1 Use cases

There is a need for an informal description that every stakeholder involved in the project can understand. This is indeed the purpose of use cases. Figure 3.1 is a concise use case diagram of the requirements of ATM system taken from IFX 1.4 specification document. In this diagram the four main use cases are identified.

1. *View account balance*
2. *Withdraw cash from account*
3. *Withdraw cash as loan*
4. *Verify card*

Use cases 1, 2 and 3 uses the use case named *Verify card.*

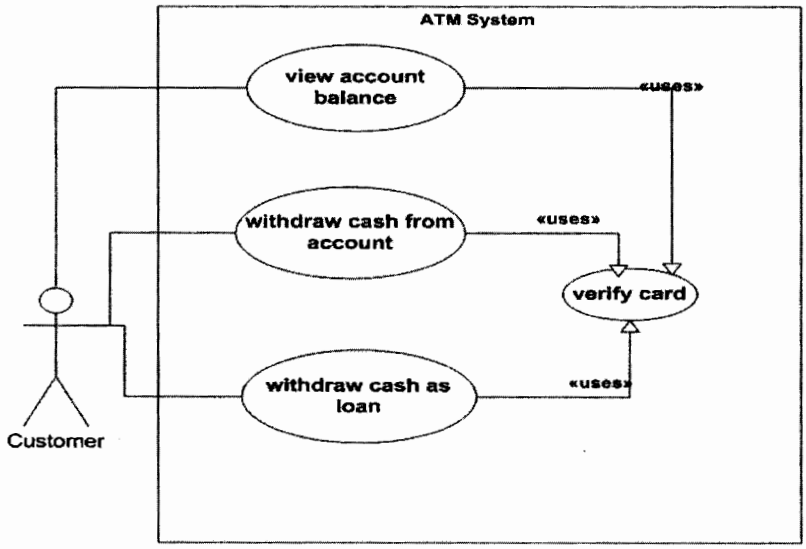


Figure 3.1. Use case diagram of ATM system

3.2.2 Description of Use cases

Use case: View account balance

Actor: Customer

Pre condition: Customer must have authentication

Post condition: Customer has viewed the account info

Flow of events:

Actor action	System response
1. Customer enters his valid account number	System matches the account number and opens the account information
2. Customer views the account balance.	System gives required account information to customer
3. Customer presses the log out button	System logs out.

Alternate flow:

1.a Customer enters invalid account number	System notifies for invalid account number to customer

Use case: Withdraw cash from account

Actor: customer

Pre Condition: Customer must have authentication

Post Condition: Customer must get money

Flow of events:

Actor Action	System Response
1. Customer enters his valid account number.	System takes customer to his account.
2. Customer enters amount of money to withdraw.	System matches entered amount with account balance and gives response to customer.
3. Customer presses ok button to withdraw money.	System gives cash to customer.
4. Customer presses log out button	System logs out from account.

Alternate flow:

1. a Customer enters invalid account number.	System does not allow the customer to enter into account.
2. a Customer enters amount of money greater than account balance.	System notifies the customer for amount of money greater than balance.

Use case: Withdraw cash as loan

Actor: Customer

Pre Condition: Customer must have authentication

Post Condition: Customer must get money

Flow of events:

Actor Action	**System Response**
1. Customer enters valid amount of money to withdraw.	System loan department looks for availability of cash and gives response.
2. Customer presses ok button to withdraw money.	System gives cash to customer.
3. Customer presses log out button.	System logs out from account.

Alternate flow:

1. a Customer enters invalid amount of money.	System notifies the customer for invalid amount of money.

3.2.3 Description of Components, Reports and State Operations of ATM system

Components

Bank

The bank holds the accounts of customers. Customers are of two types i.e. credit card holders and debit card holders. The bank has credit card department. Through ATM card either it is credit card or debit card, a customer can get money when he needs.

ATM

Every bank has ATM facility. It is a machine which is used to get money without cheque/DD/Money order. Every ATM has money cassette to place the money inside the machine. The transaction limit for a day ranges from 20,000 to 25,000 Pak Rupees. ATM is being considered active when it is used by a customer

Initial State ATM

Initially the money cassette of ATM is full and ATM is in idle state which means that it is ready to be used by a customer.

Card Database

It is the central data base for keeping the information of credit and debit cards.

Reports

There are different types of reports that will be displayed according to customer action. The report names are self explanatory.

Insufficient ATM Funds Report

This report is displayed when there is insufficient money in ATM machine.

Insufficient Loan Funds Report

This report is displayed when there is lack of money for giving loan in credit card department of bank.

Amount Too High Report

This report is displayed when the customer requests the amount of money more than the limit.

Card Authentic Not OK Report

This report is displayed when card authentication is not ok.

Transaction Limit Exceed Report

This report is displayed when customer tries to get money through ATM when he already has taken money up to maximum limit for that day.

Already Made Some Withdrawal Report

This report shows approval for customer to take some more money by telling that you have made some withdrawal in the same day but still you have option to take more money.

Sate Operations

Credit Amount Sufficient

This schema checks the sufficiency of money for credit card holder, when he or she enters the amount of money to withdraw. It says that the credit amount will be sufficient only if the input amount of money is less than the loan funds and it should be less than the money in ATM machine and also it should not exceed the per day transaction limit that is supposed to be 20,000.

Insufficient ATM Funds

This operation implies the condition that ATM funds will be insufficient if they are less than the ATM critical level which has been kept Rs. 5000.

Insufficient Loan Funds

This operation implies the condition that loan funds will be insufficient if they are less than the loan critical level which has been kept Rs. 5000.

Credit Amount Too High

This operation checks the condition for amount to withdraw. If the input amount is greater than 20,000 it will be considered too high to be withdrawn.

Card Authentic OK

In this operation, card authenticity is checked. When the customer will input the ATM card in machine, its encrypted information will be matched with the information stored in card database.

Card Authentic Not OK

This operation is negation of above operation that is Card Authentic OK.

Card Retained

This operation demonstrates the conditions in which the card will be retained by ATM machine. If the customer enters the wrong PIN code for three consecutive times, his card will be retained.

Card Ejected

This is card eject operation. The ATM card will be ejected either if the process is complete or it is cancelled by the customer or card is waiting for customer response for long time.

Transaction Log

When ever a new transaction will occur, its record comprising of account number, transaction date and with draw amount will be saved in transaction log.

Transaction Limit Exceed

The operation 'Transaction limit exceed' by taking the current date and account number checks whether the transaction limit is equal to withdraw amount. If that is the case, it means the maximum transaction have been made for the day.

Already Made Some Withdrawal

If in the same date, the user have made withdraw transaction which is less than the transaction limit of that account, then it means that user have already made some withdrawal but he also has the choice to take some more money.

Debit Amount Too High

This operation is invoked when the debit card holder, who has an account in bank requests the amount to withdraw, which is greater than the account balance or it is greater than the ATM funds kept in ATM machine.

Don't Allow Transaction

As the name implies, the operation 'Don't Allow Transaction' will work if card authentication is not ok. In that case, the corresponding report will be displayed.

Allow Transaction

The transaction will be allowed if card authentication is ok.

Refuse Withdrawal by Credit Card

The withdraw operation by credit card will be refused if amount requested is too high or there is insufficient amount of money in ATM or there is insufficient money in loan funds of credit card department of bank.

Refuse Withdrawal by Debit Card

In spite of permissions for making withdrawal, the debit card holder will not be able to make withdrawal if his requested amount is greater than his account balance or there is insufficient amount in ATM or he has already made the maximum withdrawal in the same day.

Permit Withdrawal by Debit Card

If debit card holder is allowed to proceed the transaction and requested amount is not exceeding the maximum limit and there are sufficient amount of money in ATM and the person has not taken the maximum amount of money in the same day or he has taken the money in the same day but not to the maximum limit, in all these cases, he will have permission to take money through ATM.

Permit Withdrawal by Credit Card

If the credit card holder is allowed to proceed the transaction and the requested amount is not exceeding the maximum limit and there is sufficient amount of money

in ATM machine and there is sufficient loan available for the person, then he will be given permission to make withdrawal.

Chapter 4:

Formal Model

Chapter 4

Formal Model

4.1 Overview

The proposed formal model of ATM system has been developed using Z notation. The main advantage of Z notation is that the large specification can be divided in chunks called schemas [26, 28]. By dividing the specification into schemas we can focus and understand the system piece by piece.

In Z notation, schemas are used to describe static and dynamic aspects of a system. The static aspects include the following

- The states of systems.
- The invariants which are maintained when the system changes its state.

The dynamic aspect includes

- The possible operation of system.
- The change of state.
- Relationship between inputs and outputs.

The schema language allows the description of different facets of system. For example one schema can describe the behaviour of system when it receives the correct input and other schema can describe the system when it receives the wrong input.

First of all free types have been defined globally.

4.2 Static Model

The following are free types named REPORT, TRANSACTIONSTATE and ACTION respectively.

$REPORT ::= AtmFundsInsufficient$
$\quad | \ LoanFundsInsufficient$
$\quad | \ AmountRequestedTooHigh$
$\quad | \ InvalidCard$
$\quad | \ InvalidThumbImpression$
$\quad | \ HaveMadeMaxWithdrawToday$
$\quad | \ HaveAlradyMadeSomeWithdrawalToday$

$TRANSACTIONSTATE ::= Complete \mid InProgress \mid AwaitingForLong \mid Cancelled$
$ACTION ::= RetainCard \mid EjectCard$

CardDatabase

In schema below, *CARD* is a set of cards. *cardset* is a sequence of type *CARD*.

[*CARD*]

CardDatabase

alreadypresentCard: P *CARD*
cardset: seq *CARD*

---

alreadypresentCard = ran *cardset*

Invariants:

- The elements of already present card set should be the same as elements stored in database in card set sequence.

Bank

The schema below is of Bank.

The values for account critical level (which means account critical level), Loan critical level and atm critical level have been supposed to 5000.

There are only two values *adequate* and *inadequate* for the *fundstatus* which is declared as free type.

AccountID is a set of account numbers, *CUSTOMER* is a set of customers. *AtmFunds*, *LoanFunds* and updated Loanfunds which have been shown as with prime(`) symble i.e. *LoanFunds'* are of type natural number.

DebitCardHolder and *CreditCardHolder* are power set of *CUSTOMER* which has already been defined as a set type.

Account Balance is a total function which takes the account number that is *AccountID* as input and returns the respective balance present in your account.

[*AccountID, CUSTOMER*]
AccCriticalLevel == 5000
LoanCriticalLevel == 5000
AtmCriticalLevel == 5000
FundStatus ::= *adequate* | *inadequate*

Bank

$AtmFunds, LoanFunds, LoanFunds' : \mathbb{N}$
$DebitCardHolder, CreditCardHolder : \mathbb{P}\ CUSTOMER$
$AccountBalance, AccountBalance' : AccountID \rightarrow \mathbb{N}$

$\forall c : CreditCardHolder$
$\bullet\ FundStatus = \{adequate\}$
$\Leftrightarrow AtmFunds > AtmCriticalLevel \wedge LoanFunds > LoanCriticalLevel$
$\forall a : AccountID$
$\bullet\ FundStatus = \{adequate\}$
$\Leftrightarrow AtmFunds > AtmCriticalLevel \wedge AccountBalance\ a > AccCriticalLevel$

Invariants:

- Any credit card holder who may not have account in bank is allowed to do transaction if and only if there is enough money in ATM and enough loan funds are available. In that case the fund status will be shown as adequate.
- Debit card holder is an account holder, for him to make transaction there should be enough money in ATM machine as well as in his account.

ATM

The below mentioned schema is of ATM. First of all some free types have been declared.

Money cassette is a container inside ATM machine to keep money which has two possible states either full or not.

Boolean is a variable which has two possible values true and false.

ATM active is of type boolean, if ATM is active, it means that it is processing the Current card of type *CARD* at hand. If it is not active, it means it is in idle state and ready to make new transaction.

$MoneyCassette ::= FULL \mid NOTFULL$
$TRUE == 1$
$FALSE == 0$
$boolean == \{FALSE, TRUE\}$

ATM

$\Delta Bank$
$ATMactive: boolean$
$currentcard: CARD$
$Pin\_no: CARD \rightarrow \mathbb{N}$
$Retries: CARD \rightarrow \mathbb{N}$
$Restricted: CARD \rightarrow boolean$
$TransactionLimit: AccountID \rightarrow \mathbb{N}$

$\forall a: AccountID \bullet TransactionLimit\ a = 20000$
$ATMactive = TRUE \Rightarrow Restricted\ currentcard = FALSE$
$\forall c: CARD \bullet Restricted\ c = TRUE \Rightarrow Retries\ c = 0$

Invariants:

- The Transaction limit for every account has been supposed to 20000.
- If ATM machine is active, it means that current inserted card is not restricted. and you are allowed to make the further transaction.
- If any card is restricted, it means its retries attempt have become zero.

Initial State ATM

The schema below shows the initial set state of ATM. With in its upper half, which is it declaration part, the schema ATMmain have been written with special symbol xi [Ξ] which means that the schema ATMmain can only be used with in this schema but one is not allowed to make changes in that.

InitialStateATM

ΞATM

$ATMactive = FALSE \wedge MoneyCassette = \{FULL\}$
$\forall c: CARD \bullet Restricted\ c = FALSE \wedge Retries\ c = 3$

Invariants:

- At initial set state the atm machine is idle and money cassette is full.
- If the card is not already restricted, its retries are three. It mean you can maximum of three times allowed to enter wrong pin number.

4.3 Dynamic Model

CreditAmountSufficient

The schema CreditAmountSufficient is using the schema bank.

It takes the desired amount to be withdrawn from user. This amount is declared as natural number.

CreditAmountSufficient

$$\Xi Bank$$
$$amount?: \mathbb{N}$$

$$FundStatus = \{adequate\}$$
$$\wedge\ amount? \leq LoanFunds$$
$$\wedge\ amount? \leq AtmFunds$$
$$\wedge\ amount? \leq 20000$$

Invariants:

- Fund status should be adequate.
- Input amount should be less than or equal to loan funds.
- Input amount should be less than or equal to atm funds.
- It should be less than or equal to 20000 which is the maximum transaction limit per day through atm.

InsufficientAtmFunds

The following schema uses the schema.

InsufficientAtmFunds

$$\Xi Bank$$

$$AtmFunds < AtmCriticalLevel$$

Invariants:

- If atm funds are less than atm critical level, it means funds are sufficient.

InsufficientLoanFunds

This schema uses the schema bank.

InsufficientLoanFunds

$\Xi Bank$

$LoanFunds < LoanCriticalLevel$

Invariants:

- If loan funds are less than loan critical level, that is 5000 defined in schema bank, then it means that loan funds are not sufficient for transaction.

InsufficientAtmFundsReport

This schema uses the output variable report! Of type REPORT, which is a free type already defined.

InsufficientAtmFundsReport

report!: REPORT

---

report! = AtmFundsInsufficient

Invariants:

- In case of insufficient funds in ATM machine, the report particular to this problem will be displayed.

InsufficientLoanFundsReport

This schema also uses the variable report! of type REPORT.

InsufficientLoanFundsReport

report!: REPORT

---

report! = LoanFundsInsufficient

Invariants:

- If there are insufficient loan funds in credit card department, then the message particular to this problem will be displayed to tell the user about the exact problem.

CreditAmountTooHigh

The schema uses the Bank schema and it takes the input amount of type natural number from user.

CreditAmountTooHigh

$\Xi Bank$
amount?: $\mathbb{N}$

---

amount? > 20000

Invariants:

- If the input amount is greater than 20000, which is maximum limit of withdrawal per day from atm, then it will be considered too high to withdraw.

AmountTooHighReport

It uses the variable report of type REPORT.

AmountTooHighReport

report!: REPORT

---

report! = Amount exceeds the limit

Invariants:

- This schema displays the report, your requested amount is too high to be withdrawn.

CardAuthenticOk

The schema uses another schema CardDatabase.

It takes the input card of type CARD.

CardAuthenticOk

Ξ*CardDatabase*
card?: CARD

$\forall$*a: alreadypresentCard* • $\exists$*b: alreadypresentCard* • *b = card?*

Invariants:

- The input card will be compared with already present cards in database and if found, the authentication will be ok.

CardAuthenticNotOk

This schema takes the same variables as above schema.

CardAuthenticNotOk

$$\Xi CardDatabase$$
$$card?: CARD$$

$$\neg (\forall a: alreadypresentCard \bullet (\exists b: alreadypresentCard \bullet b = card?))$$

Invariants:

- The card authentication will not ok if the input card did not match with the already present card set in data base, which has been defined in carddatabase schema.

CardAuthenticNotOkReport

CardAuthenticNotOkReport

$$report!: REPORT$$

$$report! = InvalidCard$$

Invariants:

- If card is not authentic, then the message invalid card will be displayed.

CardRetained

This schema uses another schema named ATM.

Action! is of type ACTION.

C? is of type CARD.

Userpin1? And so on are the four digit pin numbers of natural number entered by user.

CardRetained

ΞATM
$action!: ACTION$
$c?: CARD$
$userpin1?, userpin2?, userpin3?: \mathbb{N}$

$userpin1? \neq Pin\_no\ c? \land userpin2? \neq Pin\_no\ c?$
$\land\ userpin3? \neq Pin\_no\ c?$
$\Rightarrow action! = RetainCard$

Invariants:

- If three times the user enter a wrong pin number, his card will be retained by machine.

CardEjected

State? of type TRANSACTION, which has already defined as free type.

Action! is of type ACTION.

CardEjected

$state?: TRANSACTIONSTATE$
$action!: ACTION$

$state? = Complete \lor state? = Cancelled$
$\lor state? = AwaitingForLong$
$\Rightarrow action! = EjectCard$

Invariants:

- If transaction get complete or the transaction has been cancelled at any time during its processing or machine is awaiting for user response for a long time, then the card will be ejected.

TransactionLog

DATE is a user defined data type.

Transaction date is of type DATE.

WithdrawAmount is of type natural number.

AccountNo is of type AccountID, already defined globally.

Transaction is a sequence of account number, date and withdraw amount.

[*DATE*]

TransactionLog

TransactionDate: DATE
WithdrawAmount: ℕ
AccountNo: AccountID
Transaction, Transaction': seq (*AccountID* × *DATE* × ℕ)

Transaction'
= Transaction ⌢ ⟨(*AccountNo, TransactionDate, WithdrawAmount*)⟩

Invariants:

- When ever a new transaction will occur, its record comprising of account number, transaction date and with draw amount will be saved in transaction log.

TransactionLimitExceed

This schema uses two more schemas named TransactionLog and ATMmain.

Today? is a variable of type date.

ThisAccount? is of type AccountID.

TransactionLimitExceed

Ξ*TransactionLog*
Ξ*ATM*
Today?: DATE
ThisAccount?: AccountID

Today? = TransactionDate
∧ *ThisAccount? = AccountNo*
∧ *WithdrawAmount = TransactionLimit ThisAccount?*

Invariants:

- The schema by taking the current date and account number checks whether the transaction limit is equal to withdraw amount?. If that is the case the the maximum transaction have been made for the day.

TransactionLimitExceedReport

TransactionLimitExceedReport

$$report! : REPORT$$

$$report! = HaveMadeMaxWithdrawToday$$

Invariants:

- The output report is 'HaveMadeMaxWithdrawToday'.

AlreadyMadeSomeWithdrawal

This schema uses two other schemas TransactionLog and ATMmain.

Variable Today? is of type DATE.

ThisAccount? is of type AccountID.

RemainingAmount is of type natural number.

AlreadyMadeSomeWithdrawal

$$\Xi TransactionLog$$
$$\Xi ATM$$
$$Today? : DATE$$
$$ThisAccount? : AccountID$$
$$RemainingAmount : \mathbb{N}$$

$$Today? = TransactionDate$$
$$\wedge\ ThisAccount? = AccountNo$$
$$\wedge\ WithdrawAmount < TransactionLimit\ ThisAccount?$$

Invariants:

- If of today's date the user have made withdraw transaction which is less than the transaction limit of that account, then it means that user have already made some withdrawal.

AlreadyMadeSomeWithdrawalReport

AlreadyMadeSomeWithdrawalReport

$$report! : REPORT$$

$$report! = HaveAlradyMadeSomeWithdrawalToday$$

Invariants:

- The report will be printed if the user have already made some withdrawal.

DebitAmountTooHigh

This schema uses the schema Bank.

It takes the input amount? Of type natural number.

It takes the AccountNo? Of type AccountID as input.

DebitAmountTooHigh

$\Xi Bank$
$amount?: \mathbb{N}$
$AccountNo?: AccountID$

$amount? > AccountBalance\ AccountNo? \lor amount? > AtmFunds$

Invariants:

- The requested amount will be considered too high to withdraw if it is greater than the account balance or it is greater than the atm funds kept in atm machine.

DontAllowTransaction

This schema concatenates the other schema to show a meaning full behaviour.

$$DontAllowTransaction \triangleq CardAuthenticNotOk \land CardAuthenticNotOkReport$$

Invariants:

- The transaction will also be not allowed if card authentication is not ok. In that case too, CardAuthenticationNotOk report will be displayed.

AllowTransaction

This schema also combines other schemas ThumbAuthenticOk and CardAuthenticOk.

$$AllowTransaction \triangleq CardAuthenticOk$$

Invariants:

- The transaction will be allowed if card authentication is ok.

RefuseWithdrawByCreditCard

This schema is made up of seven schemas given below

$RefuseWithdrawByCreditCard \triangleq$
$AllowTransaction \wedge (CreditAmountTooHigh \wedge AmountTooHighReport)$
$\vee\ InsufficientAtmFunds \wedge InsufficientAtmFundsReport$
$\vee\ InsufficientLoanFunds \wedge InsufficientLoanFundsReport$

Invariants:

- The withdraw by credit card will be refused if amount requested is too high or there is insufficient amount of money in ATM or there is insufficient money in loan funds of credit card department of bank.

RefuseWithdrawByDebitCard

This schema is also combination of seven schemas.

$RefuseWithdrawByDebitCard \triangleq$
$AllowTransaction \wedge (DebitAmountTooHigh \wedge AmountTooHighReport)$
$\vee\ InsufficientAtmFunds \wedge InsufficientAtmFundsReport$
$\vee\ TransactionLimitExceed \wedge TransactionLimitExceedReport$

Invariants:

- In spite of permissions for making withdrawal, the debit card holder will not be able to make withdrawal if his requested amount is greater than his account balance or there are insufficient amount in atm or he has already made the maximum withdrawal in the same day.

PermitWithdrawByDebitCard

This schema like above schemas is the combination of other schemas.

$PermitWithdrawByDebitCard \triangleq$
$AllowTransaction$
$\wedge \neg (DebitAmountTooHigh \wedge AmountTooHighReport)$
$\wedge \neg (InsufficientAtmFunds \wedge InsufficientAtmFundsReport)$
$\wedge \neg (TransactionLimitExceed \wedge TransactionLimitExceedReport)$
$\vee\ AlreadyMadeSomeWithdrawal \wedge AlreadyMadeSomeWithdrawalReport$

Invariants:

- If debit card holder is allowed to proceed the transaction and requested amount is not exceeding the maximum limit and there are sufficient amount of money in atm and the person has not taken the maximum amount of money in the same day or he has taken the money in the same day but not to the maximum limit, in all these cases, he will have permission to take money through atm.

PermitWithdrawByCreditCard

The schema below is the combination of other schemas.

$PermitWithdrawByCreditCard \cong$
$AllowTransaction$
$\wedge \neg (CreditAmountTooHigh \wedge AmountTooHighReport)$
$\wedge \neg (InsufficientAtmFunds \wedge InsufficientAtmFundsReport)$
$\wedge \neg (InsufficientLoanFunds \wedge InsufficientLoanFundsReport)$

Invariants:

- If the credit card holder is allowed to proceed the transaction and the requested amount is not exceeding the maximum limit and there are sufficient amount of money in atm machine and there is sufficient loan available for the person, then he will be given permission to make withdrawal.

Chapter 5:

Model Checking

Chapter 5

Model Checking

The resultant formal model is checked by the Z/EVES toolset. Z/EVES is a tool for analysing Z specifications. It can be used for parsing, type checking, domain checking, schema expansion, precondition calculation, refinement proofs, and proving theorems [31]. Here is the brief introduction of the Z/EVES.

5.1 Z/EVES Environment

5.1.1. Getting Starting with Z/EVES

There are two versions of Z/EVES. We have used its GUI version. The User has to double click the *Z-EVES GUI* icon to get start.

5.1.2. Z/EVES server

At first after clicking the GUI icon a small window will open shown below. This is Z/EVES server. One can either ignore it or minimize to continue.

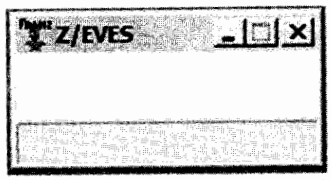


Figure 5.1. Z/EVES server window

5.2. Z/EVES Displays

Following are the Z/EVES main displays.

5.2.1. The Specification Window

The other window that will open with Z/EVES server will be the specification window. This is the window where the actual Z specifications are written and checked.

The sequence of paragraphs is important in Z notation. Z notation is a sequence of paragraphs, so they should be declared before use. When Z notation is presented in documents the order some times is not maintained. Such specification when imported

There are two operation modes of Z/EVES: 'Eager' and 'Lazy'. The checking rules are strict in Eager mode. For example this mode does not allow the checking of later paragraphs without checking the previous paragraphs. In Lazy mode the checking rules are not strict. It allows the paragraph checking with out checking the previous paragraphs. This mode is helpful if specifications are on experimental stage. By default the Lazy mode is active in Z/EVES.

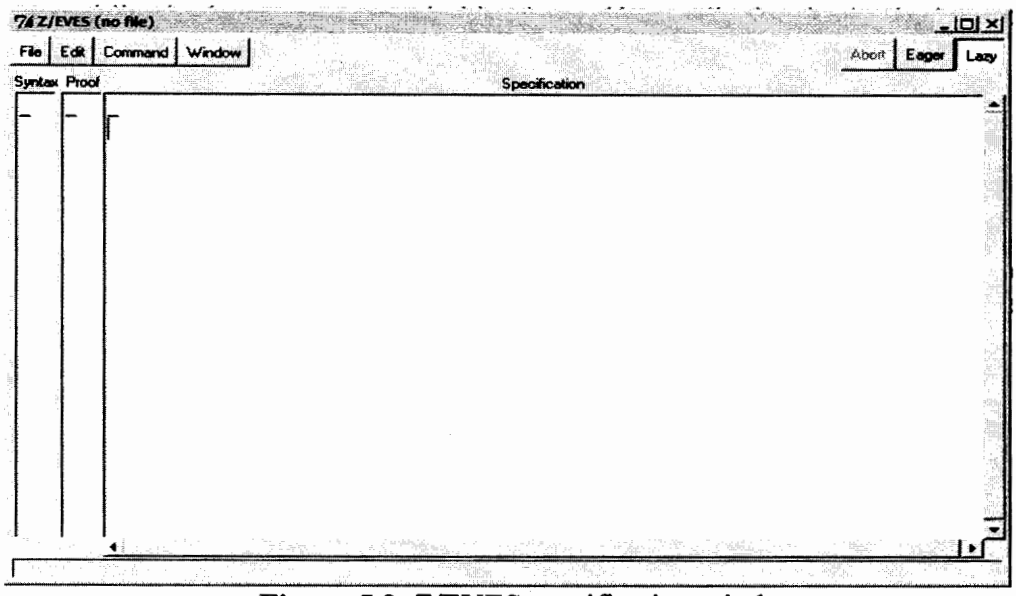


Figure 5.2. Z/EVES specification window

The window below is also a specification window with some written specification.

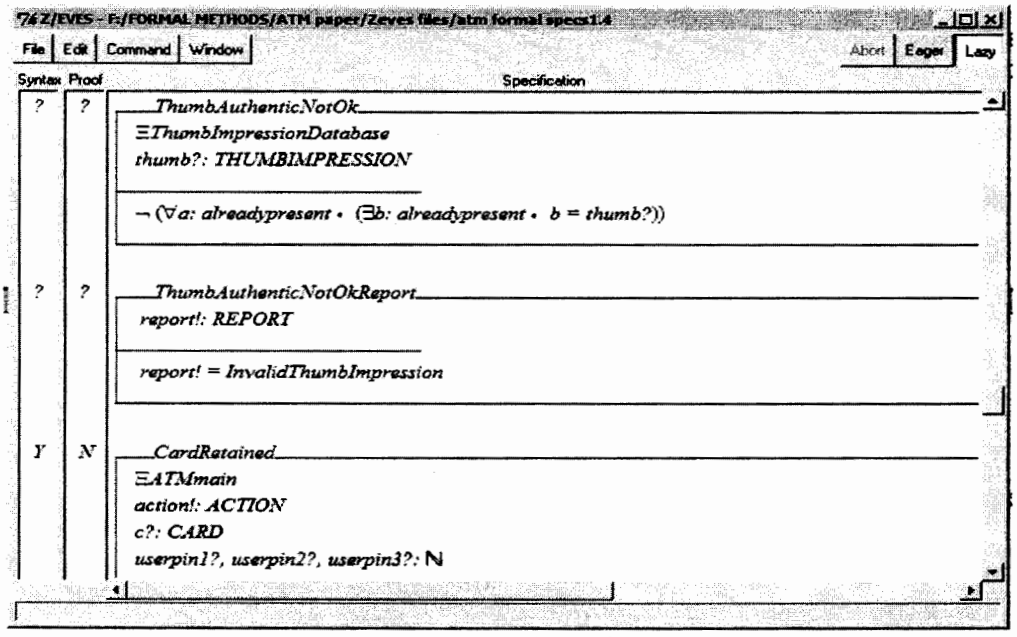


Figure 5.3. Z/EVES specification window with specification

The specifications are checked by right clicking on them and selecting the check option or if some one wants to check all the previous paragraphs he can opt for check

up to here. There are two status columns at the left side of specification window named 'syntax' and 'proof'. Besides each paragraph there is one of these symbols.

- '?' indicates that the paragraph have not been checked yet.
- 'Y' indicates that the paragraph has been checked and has no syntax or type error and also the logical proof of specification is existing.
- 'N' indicates that the paragraph has been checked and has error in it.

5.2.2 The Editor Window

Below is the edit window for Z specification. Specifications are written in the form of paragraphs. The new paragraphs can be added or existing paragraphs can be modified. This window provides the basic editor capabilities like copy, paste, insertion and deletion etc. Below edit area it provides a palette of symbols. These symbols can be inserted at insertion point just by clicking on them.

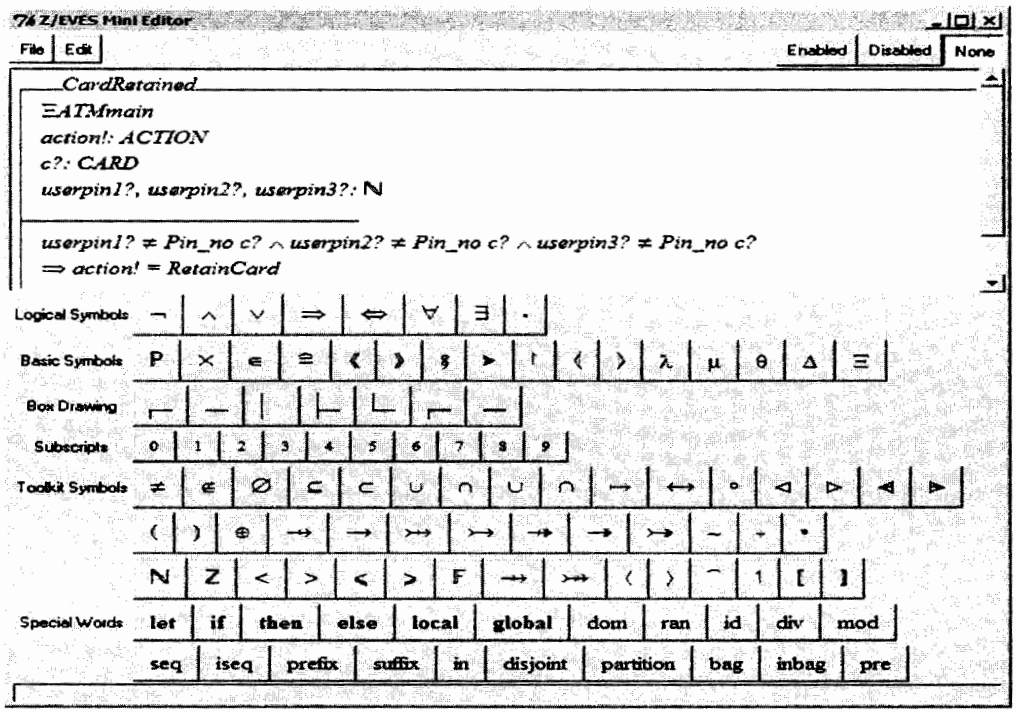


Figure 5.4. Z/EVES editor window

Boxed paragraphs can be added by inserting them. They are in the edit menu of edit window. There are three type of boxes named:

- Axiomatic box
- Generic box
- Schema box

When ever a box is inserted it replaces the existing specification. After inserting the box, its name, declaration and predicates can be edited.

5.2.3 The Proof Window

The proof window provides the following three functions:

- Inspection and modification of a proof script
- Interactive construction of a proof
- Proof browsing

Below is the snap shot of proof window.

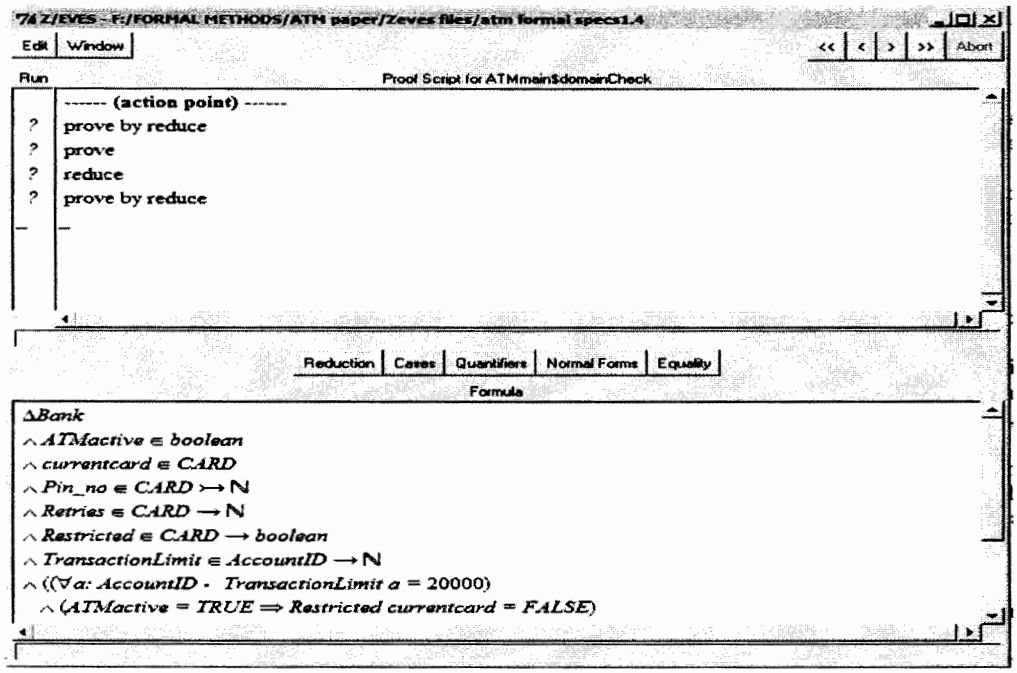


Figure 5.5. Z/EVES proof window

A proof script is a sequence of Z/EVES prover commands. These commands are displayed with their status. There are different ways of providing proves. One is simple prove and the other is prove by reduce.

5.3 Analyzing Z specifications

The use of Z, or any other formal notation, in a specification is a great step forward: natural language specifications are notorious for their ambiguity. Making formal statements about a system can in itself be beneficial, as it can lead to a careful consideration of the important aspects of the system and to the development of a consistent terminology [33].

However, if we have developed the formal specification of a system, it does not mean that the specifications are correct and consistent. There can be different type of errors ranging from trivial spelling errors to inconsistencies in meaning.

5.3.1. Checking for errors

Syntax and type checking

Z language has a quite complex syntax. If is very easy for inexperienced person to make mistakes while developing the specifications. Z/EVES provides the way to detect and correct these errors. One can use the incremental approach for error checking, which means after finishing paragraph, it can be checked at that time.

Domain checking

Domain checking does checking which is beyond the scope of syntax and type checking. In Z notation, one can write the expressions which are not necessarily meaningful. There are two types of such expressions.

- A function can be applied outside its domain.

For example in *1 div 0, max Z, or #N.*

- A definite description (μ-term) is not meaningful if there is not a single value satisfying the predicate.

Example 1: $\mu\ x: Z \mid x \neq x$, for which there is no possible value of x satisfying the predicate.

Example 2: $\mu\ x: Z \mid x > 0$, for which there are many possible values.

These are the meaningless expressions. Mathematicians have proposed different ways to deal with such meaning less expressions. The way Z/EVES uses to cater such problems is to show that every expression is meaningful. Using domain analysis technique Z/EVES examines each paragraph as it is entered, and checks each function application and definite description for meaningfulness [31].

5.4 Exploration of Formal ATM Model

The formal ATM Model is checked and analyzed against the Z/EVES toolset. The outcome of analyzed formal ATM model is described in table 5.1 and 5.2. The schemas are analyzed with major four techniques of Z/EVES toolset (syntax and type checking,

domain checking, reduction and prove by reduce).The outcome of analyzing formal model is classified into two types. Firstly, some schemas were well written and proved automatically without any prove assistance of the tool (indicated with a ✓ mark in the table 5.1). Secondly, some schemas were proved using the prove assistance of the tool (indicated by ✓* symbol).

Static Model	Syntax & type Checking	Domain Checking	Reduction	Prove by Reduce
Card Database	✓	✓	✓	✓
Bank	✓	✓	✓	✓
ATM	✓	✓	✓*	✓
InitialStateATM	✓	✓	✓	✓

Table 5.1: Results of exploration of Static Model of ATM system

Dynamic Model	Syntax & type Checking	Domain Checking	Reduction	Prove by Reduce
CreditAmountSufficient	✓	✓	✓	✓
InsufficientAtmFunds	✓	✓	✓	✓
InsufficientLoanFunds	✓	✓	✓	✓
InsufficientAtmFundsReport	✓	✓	✓	✓
InsufficientLoanFundsReport	✓	✓	✓	✓
CreditAmountTooHigh	✓	✓	✓	✓
AmountTooHighReport	✓	✓	✓	✓
CardAuthenticOk	✓	✓	✓	✓
CardAuthenticNotOk	✓	✓	✓	✓
CardAuthenticNotOkReport	✓	✓	✓	✓
CardRetained	✓	✓	✓*	✓
CardEjected	✓	✓	✓	✓
TransactionLog	✓	✓	✓*	✓
TransactionLimitExceed	✓	✓	✓	✓
TransactionLimitExceedReport	✓	✓	✓	✓
AlreadyMadeSomeWithdrawal	✓	✓	✓	✓
AlreadyMadeSomeWithdrawal Report	✓	✓	✓	✓
DebitAmountTooHigh	✓	✓	✓	✓

DontAllowTransaction	✓	✓	✓	✓
AllowTransaction	✓	✓	✓	✓
RefuseWithdrawByCreditCard	✓	✓	✓	✓
RefuseWithdrawByDebitCard	✓	✓	✓	✓
PermitWithdrawByDebitCard	✓	✓	✓	✓
PermitWithdrawByCreditCard	✓	✓	✓	✓

Table 5.2: Results of exploration of Dynamic Model of ATM system

Chapter 6:

Conclusion

Chapter 6

Conclusion and Future Work

The main objective of this research was applying formal methods in safety critical systems to show the strength of formal methods and to introduce a change in classical software development life cycle. Safety critical systems are complex and sensitive enough to reason about the use of formal methods. As complex systems demonstrate the power of formal methods that is why ATM system has been selected for modelling and formal specification using formal methods. ATM system is widely used safety critical system all over the world. Although the specification done here is at abstract level and has not done for implementation purpose but it will be useful for researchers interested in the application of formal methods in safety critical systems.

By applying formal method in terms of Z notation, in this research, it is observed that it does not require a high level of mathematics rather it requires knowledge of basic set theory and first order logic for the description and analysis of a complete system. A person who does not have deep understanding of mathematics can use the full capabilities of this specification notation. Z notation has got wide tool support and automatic checking of specification, so using it has now become easier.

Firstly informal description of system using use cases has been presented and then system is described formally using Z notation. Furthermore description of formal specification has also been given to increase the understandability. By this way, gap between informal and formal approaches have been minimized. Formal methods are emerging technology and at the current stage of development in formal methods, it is not possible to use them alone because they have not yet got full maturity. Hence formal methods are applied in this research along with the existing approaches such as UML. In this way it will provide ease both to developer and the common stakeholder or user to adopt the change. As formal methods are at the very initial stage of development so using them along with old approaches is looking reasonable.

Now the question is what benefit we get by using this new approach. The answer is self explanatory. We have used the Z notation in this research at specification level which is the stage of analysis. By using Z notation, we have modelled the system and analyzed in such depth and breath that we have even thought about the types of variable and functions that we will use further. We have divided the whole informal specification

into chunks called schemas and given them mathematical symbols which are quite clear to understand and comprehend and which are not ambiguous.

Here much of the system logic at specification level has been captured. It was observed that the implementation after formal specification might be done in easier way than using existing software Engineering approaches. It was also observed that Z has expressed its power by giving type definitions even at specification level. The work is under progress and the complete model of this system will be presented in future work. Finally the specifications have been analyzed using Z/EVES toolset [21].

References

1. A. Avizienis, J. C. Laprie and B. Randell, "Fundamental Concepts of Dependability", Research Report No. 1145, LAAS-CNRS, 2001.

2. A. Canning, "Assessement at the Requirements Stage of a Project", 2nd Safety Critical System Club Meeting, Beaconsfield, UK, Publisher: Advanced Software Department, ERA Technology Ltd, 1991.

3. Anthony Hall, "Seven Myths of Formal Methods", IEEE software, Vol. 07, No. 5, pp. 11-19, 1990.

4. B. R. Ladeau and C. Freeman, "Using Formal Specification for Product Development", Hewlett-Packard Journal, pp. 62-66, 1991.

5. B. S. Lee and B. R. Bryant, "Contextual Knowledge Representation for Requirements Documents in Natural Language", URL: http://citeseer.ist.psu.edu/lee02contextual.html , Proc. of FLAIRS 2002, 2002.

6. C. A. R. Hoare, "Communicating Sequential Processes", Prentice-Hall International, ISBN-10: 0131532715, ISBN-13: 978-0131532717, 1985.

7. C. B. Jones, "Systematic Software Development Using VDM", Englewood Cliffs, NJ, Prentice-Hall, ISBN: 0-13-880725-6, 1986.

8. C. Michael Holloway, "Why Engineers Should Consider Formal Methods", 16th Digital Avionics Systems Conference, Publisher: NASA Langley Technical Report Server, 1997.

9. Constance Haitmeyer, "On the Need for Practical Formal Methods", LNCS, Publisher: Springer-Verlag, Vol.1486, pp.18-26, 1998.

10. D. Harel and A. Pnueli, "On the Development of Reactive Systems in Logic and Models of Concurrent Systems", NATO ASI Series, Vol. 133, pp. 477-498, 1985.

11. D. Harel, "Statecharts: A Visual Formalism for Complex Systems", Series of Computer Programming, Publisher: Elsevier North-Holland, Inc. Vol. 8, pp. 231-274, 1987.

12. D. L. Parnas and J. Madey, "Functional Documentation for Computer Systems Engineering", Version 2, Report No. 237, TRIO, Communications Research Laboratory, Faculty of Engineering, McMaster University, Hamilton, Ontario, Canada, 1991.

13. D. W. Bustard and A. C. Winstanley, "Making Changes to Formal Specifications: Requirements and an Example", IEEE Press Piscataway, NJ, USA, ISSN: 0098-5589, Vol. 20, pp. 562-568, 1994.

14. Daniel Jackson, "Alloy: A Lightweight Object Modelling Notation", ACM Transactions on Software Engineering and Methodology, Vol. 11, Issue 2, pp. 256-290, 2002.

15. Daniel M. Berry, "Formal Methods: The Very Idea_Some Thoughts About Why They Work When They Work", Science of Computer Programming, Publisher: Elsevier, Vol. 42, No.1, 2002.

16. E. W. Dijkstra and C. S. Scholten, "Predicate Calculus and Program Semantics", Springer-Verlag, 1990.

17. Ferenc Dosa Racz and Kai Koskimies, "Tool-Supported Compression of UML Class Diagram", 2nd Int'l Conference on the Unified Modelling Language, 1999.

18. Heitmeyer and Constance, "SCR: A Practical Method for Requirements Specification", URL: http://handle.dtic.mil/100.2/ADA465445 , Report Date: 1998.

19. http://vl.fmnet.info/b/, Date of access: 13/08/2007.

20. http://www.cs.utexas.edu/users/csed/formal-methods/docs/johnson.html, Date of access: 03/03/2007.

21. http://www.ifxforum.org/home, Date of access: 11/08/2007.

22. I. Meisels and M. Saaltink, "The Z/EVES Reference Manual", TR-97-5493-03, ORA Canada, CANADA, 1997.

23. ISO, "Information Systems Processing-Open Systems Interconnection-LOTOS", Technical Report, 1987.

24. J. Bowen and V. Stavridou, "Safety Critical Systems, Formal Methods and Standards", Software Engineering Journal, Vol. 8, Issue. 4, pp. 189-209, 1993.

25. J. Hooman, S. Ramesh and W. P. Roever, "A Compositional Semantics for Statecharts", Technical Report, Netherland, 1989.

26. J. M. Spivay, "Understanding Z: A Specification Language and its Formal Semantics", Cambridge University Press, 1988.

27. J. M. Spivay, "An Introduction to Z and Formal Specification", Software Engineering Journal, Vol. 4, Issue 1, ISSN: 0268-6961, pp. 40-50, 1989.

28. J. M. Spivey, "The Z Notation: A Reference Manual", Englewood Cliffs, NJ, Prentice-Hall, 1989.

29. J. M. Wing, "A Specifier's Introduction to Formal Methods", IEEE Computer, Vol.23, No.9, pp.8-24, 1990.

30. J. V. Guttage, J. J. Horning and J. M. Wing, "The Larch Family of Specification Languages", IEEE Software, Vol. 2, No. 5, pp. 24-36, 1985.

31. J. Wensley et al., "SIFT: Design and Analysis of a Fault Tolerant Computer for Aircraft Control", Proc. of IEEE, 1978, pp. 1240-1254, 1978.

32. Jean-Francois Monin, "Understanding Formal Methods", Springer Verlag, ISBN: 1852332476, 2003.

33. Jeffrey Douglas and Richard A. Kemmerer, "Aslantest: A Symbolic Execution Tool for Testing Aslan Formal Specifications", Proc. of the 1994 ACM SIGSOFT International Symposium on Software Testing and Analysis, pp: 15-27, 1994.

34. Jonathan Bowen, Mike Hinchey and David Till, "The Z Formal Specification Notation", 10th Int'l Conference of Z Users Reading, Springer-Verlag, LNCS: 1212, 1997.

35. Jonathan P. Bowen and Victoria Stavridou, "The Industrial Take-up of Formal Methods in Safety-Critical and Other Areas: A Perspective", book title: "Industrial Strength Formal Methods", Springer-Verlag, Vol. 670, pp. 183-195, 1993.

36. K. Lano, "The B Language and Method: A Guide to Practical Formal Development", Springer-Verlag, ISBN: 3-540-76033-4, 1996.

37. M. Buchi, "The B Bank: A Complete Case Study", 2nd Int'l Conference on Formal Engineering Methods, IEEE Press, No.0, ISBN: 0-8186-9198-0, pp.190–199, 1998.

38. M. C. Thomas, "The Future of Formal Methods", Proc. Of 3rd Annual Z Users Meeting', Oxford University Computing Laboratory, UK, pp. 1-3, 1988.

39. Maritta Heisel and Jeanine Souquieres, "A Method for Requirements Elicitation and Formal Specification", Springer-Verlag, ISBN: 3-540-66686-9, Vol. 1728, pp. 309 – 324, 1999.

40. Martin Giese and Fogardt Heldal, "From Informal to Formal Specification in UML", Proc. of UML2004, 2004.

41. N. A. Zafar, "Formal Model for Moving Block Railway Interlocking System Based on Un-Directed Topology", ICET06, pp. 217-223, 2006.

42. N. Rescher and A. Urquhart, "Temporal Logic", Library of Exact Philosophy, 1971.

43. N.A. Zafar and K. Araki, "Formalizing Moving Block Railway Interlocking System for Directed Network", Research Reports, Department of Computer Science and Communication Engg., Kyushu University, Japan, 2003.

44. Neumann, "Illustrative Risks to the Public in the Use of Computer Systems and Related Technology", ACM SIGSOFT Software Engineering Notes, pp. 23-33, 1992.

45. P. Chapront, "Vital Coded Processor and Safety Related Software Design", Proc. of IFAC Symposium, Zurich, Switzerland, pp. 141-145, 1992.

46. R. Barden, S. Stepney, and D cooper, "The Use of Z", Z User Workshop, York 1991, Springer-Verlag, Workshops in Computing, pp. 99-124, 1992.

47. R. Tuok and L. Logrippo, "Formal Specification and Usecase Generation for a Mobile Telephony System", Computer Networks and ISDN System, Publisher: Elsevier, Vol.30, pp.1045-1063, 1998.

48. R. W. Butler, "What is Formal Methods?", 2001.

49. Richard Denney, "Succeeding with Usecases: Working Smart to Deliver Quality", Addison-Wesley Professional Publishing, ISBN: 0-321-31643-6, 2005.

50. Sofia Kanwal and N.A Zafar, "Modeling of Automated Teller Machine System using Discrete Structures", Proc. of 8th IPMC, 2007.

51. Sofia Kanwal and N. A. Zafar, "Formal Model of Automated Teller Machine System using Z Notation", Proc. of 3rd ICET, IEEE Catalog No. 07EX1885, ISBN: 1-4244-1493-8, 2007.

52. Sten Agerholm and Peter G. Larsen, "A Lightweight Approach to Formal Methods", Proc. of the Int'l Workshop on Current Trends in Applied Formal Methods, Boppard, Ger... Springer-Verlag, 1998.

53. W. Reisig, "Petri Nets: An Introduction", Springer-Verlag, Berlin, 1985.

54. X. Liu, H. Yang and H. Zedan, "Formal Methods for the Re-engineering of Computing Systems: A Comparison", Computer Software and Applications Conference, 1997.

55. X. Liu, H. Yang and H. Zedan, "Formal Methods for the Re-engineering of Computing Systems", Proc. of the 21st Int'l Conference on Computer Software and Application", IEEE Computer Society, pp. 409-414, 1997.

56. Y. Wang and Y. Zhang, "Formal Description of an ATM System by RTPA", Proc. of CCECE'03, IEEE CS Press, pp. 1255-1258, 2003.