Modeling And Formal Specification Of Air Traffic Control System Using Z Notation

Maryam Jamal and Nazir Ahmad Zafar
SZABIST
Islamabad, Pakistan

Abstract: *In this paper, an abstract Air Traffic Control (ATC) System is modeled using Formal Methods, in terms of Z-notation. ATC system is a highly distributed and safety critical system. For modeling of distributed nature of ATC system, a separate queue of flying aircrafts is maintained at each controlled airspace. To ensure safety, it is mandated that each airspace and runway do not exceed its capacity limit in all state operations. Firstly, Requirements Analysis is done using UML diagrams and then the Formal ATC system Model is described by Z-notation. Finally, the Formal ATC system Model is checked and analyzed with Z/EVES tool-set.*

Keywords: *Advanced Software Engineering, Formal Methods, Air Traffic Control System, Z-notation, Safety properties*

1. INTRODUCTION

Air Traffic Control (ATC) system is one of the most challenging highly distributed and safety critical systems. Safety critical systems [9, 10], such as ATC system in which failure has severe consequences, the basic need is to identify and remove errors before system is deployed. Formal Methods make it possible to analyze and prove certain properties of the system that identify potential errors and inconsistencies at specification level whereas in informal approaches errors and inconsistencies are identified only during testing phase. Therefore, cost of removing errors is reduced because it increases rapidly as project progresses. Formal Methods ensure that system is correct and consistent with respect to its requirements giving high confidence in system to be built.

ATC system is also a highly distributed system. The task of safe journey of any aircraft is distributed amongst many controllers (computer-based system), which collaborate and direct an aircraft within their area of control. Each controller monitors and tracks the number of aircrafts in its area of control and ensures the safety.

ATC system has been an important research area. Many researchers have been contributing in this area using various approaches. The work presented in [8] is a case study of ATC system exploiting Layered Architecture. The research in [3] is focused on the detection and reduction of errors caused by a human operator in an ATC system. The work in [12] presents the complex ATC system as distributed cognitive representations, which are primarily visible and external to human actors and can be used as a source for conflict detection.

However, from the perspective of Formal Methods very little work has been done on complete ATC system. The work in [6] provides a case study at a very abstract level. It actually served as a starting point for our work but their work is in VDM. Similarly, to demonstrate the strength of a Model based language, an example of a simple hypothetical
ATC system using Sum Language, dialect of Z-Notation, and Cogito Methodology is presented in
[11]. There is also given the idea of distributed architecture of ATC system abstractly. Our work is in Z-notation because apart from other techniques, the rich mathematical notations offered by Z make it possible to reason rigorously and effectively about the behavior of specified system. Unlike the work done in [6], our work is more focused on distributed nature of ATC system and covers all phases of a flight i.e takeoff till landing ensuring safety at each phase. In [6, 11], safety in terms of exceeding capacity limitation of airspace is defined. Our work not only ensures capacity restriction in airspace but also on runways and defines safety properties in all phases of the flight from takeoff till landing.

The main objectives of this paper are: (i) applying formal methods to model critical systems, (ii) integration of formal and informal approaches, and (iii) proposing an abstract model ensuring correctness of formal specification of the system.

In Section 2, Formal Methods are introduced. In Section 3, Air Traffic Control system is described. Requirements Analysis of ATC System is done in Section 4 and Formal Model of ATC system is presented in Section 5. Finally concluding remarks are given in Section 6.

2. FORMAL METHODS

Formal Methods is an emerging technology that comprises of using mathematics for writing precise and unambiguous specifications. It provides the means for analyzing and proving certain properties of system to be built so that errors in specifications can be identified and removed. Using mathematical refinements, Formal Methods are used in every stage of development process, ensuring the development of high quality and correct system with respect to its requirements. There are more than 90 techniques of Formal Methods amongst them usage of Z-notation and Z/EVES tool-set is demonstrated in this paper.

3. AIR TRAFFIC CONTROL SYSTEM

ATC is the supervision of airborne and taxiing aircraft by ground-based controllers [4]. In many countries, ATC services are provided throughout the majority of airspace, and its services are available to all types of aircrafts. The Airspace is divided into Zones or Centers, and each Zone is divided into Sectors. An air traffic controller (computer-based system), which must collaborate with other controllers and with pilots, manages each airspace Sector. During the flight from source to destination, an aircraft is handed off from Sector to Sector and Center to Center. Through out the flight, important flight data of an aircraft is maintained. Figure 1 shows an aircraft during phases of its flight.

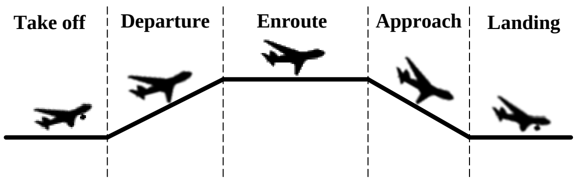

Figure 1. Phases of a Flight

4. REQUIREMENTS ANALYSIS

The basic requirement of an ATC system is to monitor and track aircraft from takeoff till landing. The safety of aircraft within air or on runway needs to be ensured. The information delivered to pilot about weather or navigation must be provided seamlessly and the maintenance of orderly and swift flow of air traffic must be guaranteed.

The domain being very large and complex and hence cannot be modeled completely in this research paper. Our work covers the following requirements:

- Modeling of airspace and runway controlled by a controller (Controlled Airspace and Runway).
- Modeling of controller currently on duty (On-duty Controller) and controller to which airspace or runway is assigned (Active Controller).
- Modeling of airspace to which controller is assigned (Activated Airspace) and airspace in which aircrafts are flying (Utilized Airspace).
- Monitoring of airspace so that the number of aircrafts flying within that airspace do not exceed its capacity limit.
- Modeling of aircrafts and maintenance of their important flight data like speed, heading and altitude.
- Takeoff – a vacant runway not being used by any other aircraft is searched and assigned to the aircraft ready to takeoff. Controller of assigned runway then controls the aircraft.
- Departure – after successful takeoff, a controlled airspace whose capacity is not exceeded is searched. Aircraft now ends its contact with ground control and is controlled by controller of assigned airspace.
- Enroute – depending on the route, aircraft is handed off from controller to controller provided capacity limit of receiving airspace is not violated. After each successful handoff the aircraft closes its communication with previous controller and communicates with the controller of receiving airspace.
- Approach – when an aircraft requests for its arrival, a vacant runway is searched. If it is found, the aircraft is connected to ground controller and contact with airspace controller goes to an end.
- Landing – during landing it is ensured that runway is vacant. After successful landing, the contact between aircraft and runway controller ends.

The set of requirements, described above, are modeled using Use-case Diagram of Unified Modeling Language (UML) [2] as shown in Figure 2. The ellipses inside system box represent system's functionality as viewed by external actor of ATC system (Controller).

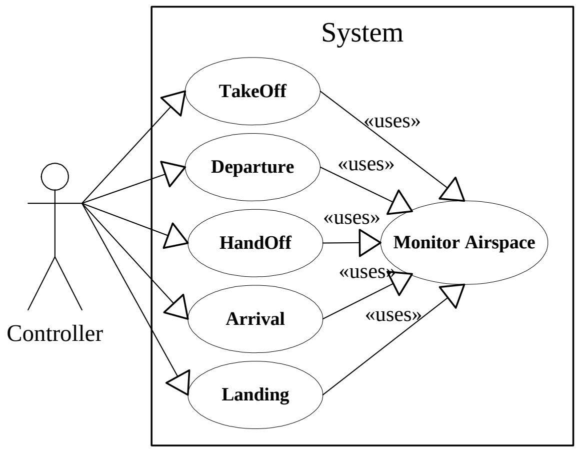

Figure 2. Use-case Diagram of ATC system

The Class Diagram of UML models the static components of a system represented as boxes and relationship between them. Figure 3 shows the Class Diagram of ATC System.

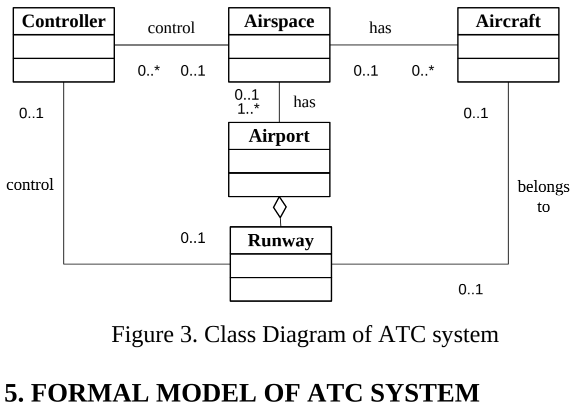

Figure 3. Class Diagram of ATC system

5. FORMAL MODEL OF ATC SYSTEM

The ATC system Model is formalized using Z-notation [7]. The Z-notation is based upon set theory and mathematical logic. Mathematical objects and their properties are collected together in schemas: patterns of declaration and constraints (invariants).

The following three abstract data types are used in formal model. The data types of Controller and Airspace have been taken from [6, 11].

- Controller – set of all controllers
- Airspace – set of all airspaces
- Runway – set of all runways

5.1 Core ATC Functionality

The core functionality of ATC system is specified in a schema named `ATCMain`.

```
∪_ ATCMain ____________________
→flightControl: Airspace ♥ Controller
→groundControl: Runway ♥ Controller
→flightQueue: Airspace ♥ seq Aircraft
→ground: Aircraft ♥ Runway
→on_duty: Π Controller
→capacity: Airspace ♣ N
→activeController: Controller
→activatedAirspace: Airspace
→utilizedAirspace: Airspace
∩_______________
→activeController ε ran flightControl
→            ϖ activeController ε ran groundControl
→            ƒ ran flightControl I ran groundControl = 0
→activatedAirspace ε dom flightControl
→utilizedAirspace ε dom flightQueue
→            ƒ ©flightQueue utilizedAirspace〉⌊ 0
→dom flightControl ζ dom capacity
→ran flightControl ζ on_duty
→ran groundControl ζ on_duty
→dom flightQueue ζ dom flightControl
→ran ground ζ dom groundControl
→As: Airspace
→   ∞ s ε dom flightQueue ƒ s ε dom flightControl
→     ƒ s ε dom capacityƒ ©flightQueue s〉 ε Φ (Z ξ Π (Z ξ
→        :altitudeLimit:Z; currentAltitude:Z; currentSpeed:Z;
→        destination: Airspace;heading: Z; source: Airspace;
→        speedLimit: Z∫)) ƒ  # ©(flightQueue s)〉 ⌡ capacity s
→As1, s2: Airspace
→   ∞ s1 ε dom flightControl ƒ s1 ε dom flightQueue
→     ƒ s1 ε dom capacity ƒ s2 ε dom flightControl
→     ƒ s2 ε dom capacityƒ s2 ε dom flightQueue ƒ s1 ⌊ s2
→     ⇒ ©flightQueue s1〉 I ©flightQueue s2〉 = 0
∠_________________________
```

The state variables defined in schema are

A partial injective function of Airspace and Controller (as in [6, 11]) is represented using the variable `flightControl`. It means, there can be zero or exactly one Controller for each Airspace. Similarly, a Controller can control zero or exactly one Airspace at a time.

A partial injective function of Runway and Controller is defined as variable `groundControl`. It means, there can be zero or exactly one Controller for each Runway. Similarly, a Controller can control zero or exactly one Runway at a time.

Unlike [11], the relation between Airspace and its queue of Aircrafts is modeled as a partial injective function in variable `flightQueue`. It means, there can be zero or exactly one queue of flying Aircrafts for each Airspace. Similarly, a sequence of Aircrafts is unique for each Airspace having no duplication.

A partial injective function of Aircraft and Runway is represented using the variable `ground`. It means, there can be zero or exactly one Aircraft on each Runway. Similarly, a Runway can have zero or one Aircraft at a time.

The set of Controllers currently on duty is represented as variable `on_duty` (as in [6, 11]).

Unlike [6, 11] the relation between Airspace and natural number is represented as a partial function, using the variable `capacity`. It means, an Airspace can have zero or exactly one capacity limit. Similarly, one or more Airspace can have same capacity limit.

Invariants

Some constraints specified in [6, 11] have been enhanced and translated into Z-Notation. Following invariants are defined in schema `ATCMain`.

1. A Controller is an Active Controller if it controls an Airspace or Runway and there is no Controller controlling an Airspace and a Runway simultaneously.
2. An Airspace is activated if it has a Controller.
3. An Airspace is utilized if it has a Controller and one or more Aircrafts are flying in it.
4. Each controlled Airspace must have a capacity limit.
5. The Controller controlling an Airspace must be on-duty.
6. The Controller controlling a Runway must be on-duty.
7. All Airspaces used during the flight must have a Controller.
8. All Runways used in ground phase of flight must have a Controller.
9. All controlled Airspaces with a capacity limit, having Aircrafts, must control finite set of Aircrafts according to its capacity.
10. All controlled Airspaces with a capacity limit, having queue of Aircrafts, each Aircraft belongs to exactly one Airspace at a time without duplication.

5.2 Aircraft

The schema `Aircraft` describes flight data of Aircraft utilizing ATC services. Each Aircraft is assigned a unique identification mark called `callsign`, the variable `Aircrafts` represents a total function. It means, each callsign is assigned to exactly one Aircraft and no two Aircrafts have the same callsign.

```
[callsign]
Aircrafts == callsign ϕ Aircraft
```

```
∪_ Aircraft _____________________
→source: Airspace
→destination: Airspace
→currentSpeed: N
→currentAltitude: N
→heading: N
```

```
→speedLimit: N
→altitudeLimit: N
∩_______________
→source ⎣ destination
→currentSpeed ⎠ speedLimit
→currentAltitude ⎠ altitudeLimit
→heading ⎠ 270
∠__________________________
```

The state variables defined in schema are

A Airspace from which aircraft has flown is represented by variable `source`.

A Airspace to which aircraft is destined to land is represented by variable `destination`.

The speed, altitude, heading, speed limit, and altitude limit of an Aircraft are represented as natural numbers in variables `currentSpeed`, `currentAltitude`, `heading`, `speedLimit`, `altitudeLimit` respectively.

Invariants

1. An Aircraft cannot have same source and destination.
2. The current speed should not exceed speed limit of Aircraft.
3. The current altitude should not exceed altitude limit of Aircraft.
4. The heading of an aircraft should not be greater than 360 degrees.

5.3 Monitoring of Airspace

This operation is specified in a schema named `MonitorAircraft`. The input airspace is monitored so that the number of Aircrafts flying in it does not exceed its capacity.

```
∪_ MonitorAirspace _________________
→ΞATCMain
→s?: Airspace
∩_______________
→s? ε dom flightControl
→s? ε dom capacity
→s? ε dom flightQueue
→©flightQueue s?〉ε Φ (Z ξ Π (Z ξ :altitudeLimit: Z;
→    currentAltitude: Z; currentSpeed: Z; heading: Z;
→    destination:Airspace;source:Airspace;speedLimit: Z∫))
→# ©(flightQueue s?)〉⎠ capacity s?
∠__________________________
```

Invariants

1. The input Airspace must have a Controller.
2. The input Airspace must have a capacity limit.
3. The input Airspace must have a queue of Aircrafts flying in it.
4. The input Airspace must have a finite set of Aircrafts flying in it.
5. The input Airspace must control Aircrafts within its capacity limit.

5.4 Takeoff an Aircraft

This operation is specified in schema named `TakeOff`. It takes an Aircraft as input that is ready for takeoff, after clearance of all ground activities. A vacant Runway not being used by any other Aircraft is searched and assigned to the Aircraft given as input. If a vacant controlled Runway is found, it is allocated to the Aircraft. The Controller of assigned Runway now controls the Aircraft.

```
∪_ TakeOff _____________________
→ΔATCMain
→a?: Aircraft
∩_______________
→As: Airspace
→   ∞ s ε dom flightControl ƒ s ε dom flightQueue
→     ƒ s ε dom capacity ƒ flightQueue s ε seq Aircraft
→     ƒ flightQueue s ℑ {a?} = ©〉
→a? ™ dom ground
→Er: Runway
→   ∞ r ε dom groundControl ƒ r ™ ran ground
→     ƒ (Ea1, a2: Aircraft
→           ∞ (a1 □ r ε ground
→            ƒ a2 □ r ε ground
→            ⇒ a1 = a2))
→     ⇒ ground' = ground Y {(a? □ r)}
∠__________________________
```

Invariants

1. The input Aircraft, ready for takeoff, should not be in flight. It means, all controlled Airspaces with a capacity limit having a finite set of aircrafts flying in it must not have a duplicated entry of input Aircraft.
2. The input Aircraft, ready for takeoff, should not have a duplicated entry on any Runway.
3. Each controlled vacant Runway must be assigned to exactly one Aircraft at a particular time.

5.5 Departure of an Aircraft

This operation is specified in a schema named `Departure`. During this phase Aircraft, given as input, enters the airspace after leaving the ground. After successful takeoff of Aircraft, given as input, a controlled Airspace whose capacity is not exceeded is searched. No criteria of selecting the Airspace and connectivity between them are specified. Aircraft now ends its contact with ground control and is controlled by Airspace controller.

```
∪_ Departure ____________________
→ΔATCMain
→a?: Aircraft
∩_______________
→As: Airspace
→   ∞ s ε dom flightControl ƒ s ε dom capacity
→     ƒ s ε dom flightQueueƒ flightQueue s ε seq Aircraft
→     ƒ flightQueue s ℑ {a?} = ©〉
→a? ε dom ground
→Es: Airspace
→   ∞ s ε dom flightControl ƒ s ε dom flightQueue'
→     ƒ s ε dom flightQueue ƒ s ε dom capacity
→     ƒ flightQueue s ε seq Aircraft
→     ƒ # ©(flightQueue s)〉< capacity s
→     ⇒ flightQueue' s = flightQueue s ⊥ ©a?〉
→ground' = {a?} ψ ground
∠__________________________
```

Invariants

1. All controlled Airspaces with a capacity limit having a finite set of Aircrafts flying in it must not have a duplicated entry of input Aircraft.
2. The Aircraft, given as input, must be successfully taken off.
3. A controlled Airspace, with a capacity limit having finite set of Aircrafts flying in it, is searched provided the number of Aircrafts in it is less then its capacity limit.

5.6 Handover an Aircraft

The concept of handover of an aircraft given in [11] has been enhanced and translated into Z-Notation as schema named `HandOff`. It is used in enroute phase of flight. Depending on the route, Aircraft (input) is handed off from Controller (input) to Controller (input) provided capacity of receiving Airspace is not violated. After successful handoff, the Aircraft closes its communication with previous Controller and communicates with receiving Airspace Controller.

```
∪_ HandOff ____________________
→ΔATCMain
→a?: Aircraft
→from?, to?: Airspace
∩______________
→from? ε dom flightControlƒ from? ε dom flightQueue
→ ƒ from? ε dom capacity ƒ flightQueue from? ε seq Aircraft
→to? ε dom flightControl ƒ to? ε dom flightQueue
→ ƒ to? ε dom capacity
→a? ™ dom ground
→flightQueue from? ℑ {a?} ⌊ ©〉
→©flightQueue to?〉ε Φ (Z ξ Π(Zξ:altitudeLimit:Z; heading: Z;
→  currentAltitude: Z; currentSpeed: Z; destination: Airspace;
→ source: Airspace; speedLimit: Z∫))
→# ©(flightQueue to?)〉 < capacity to?
→from? ε dom flightQueue'
→flightQueue' from? = flightQueue from? \ ©a?〉
→to? ε dom flightQueue' ƒ flightQueue to? ε seq Aircraft
→flightQueue' to? = flightQueue to? ⊥ ©a?〉
∠_________________________
```

Invariants

1. The Airspace, to which an Aircraft currently belongs, must be a controlled Airspace with a capacity limit having a queue of Aircrafts flying in it.
2. The Airspace, to which an Aircraft is desired to go, must be a controlled Airspace with a capacity limit having a queue of Aircrafts flying in it.
3. The input Aircraft, ready for handoff, should not have a duplicated entry on any Runway.
4. The input Aircraft must belong to Airspace from which it is handed off, given as input.
5. The Airspace, to which Aircraft is handed off, must have finite set of Aircrafts flying in it.
6. The Airspace, to which Aircraft is handed off, must have Aircrafts less than its capacity limit.

5.7 Descent of an Aircraft

This operation is specified in a schema named `Approach`. During this phase, the Aircraft leaves Airspace and enter the Airspace of destination Airport also known as Terminal Airspace. The Aircraft given as input requests arrival and a controlled vacant Runway is searched. If found Aircraft is connected to ground Controller and contact with airspace Controller ends.

```
∪_ Approach _____________________
→ΔATCMain
→a?: Aircraft
∩______________
→a? ™ dom ground
→Es: Airspace
→   ∞ s ε dom flightControl ƒ s ε dom flightQueue
→     ƒ s ε dom capacity ƒ flightQueue s ε seq Aircraft
→     ƒ s ε dom flightQueue' ƒ flightQueue s ℑ {a?} ⌊ ©〉
→     ⇒ flightQueue' s = flightQueue s \ ©a?〉
→Er: Runway
→   ∞ r ε dom groundControl ƒ r ™ ran ground
→  ⇒ ground' = ground Y {(a? □ r)}
∠__________________________
```

Invariants

1. The input Aircraft, ready for arrival, should not have a duplicated entry on any Runway.
2. The input Aircraft must belong to a controlled Airspaces with a capacity limit having a finite set of Aircrafts flying in it.
3. The Runway assigned to the Aircraft must be controlled and vacant.

5.8 Landing

This operation is specified in a schema named `Landing`. It deals with landing of Aircraft, given as input. During landing it is ensured that Runway is controlled and vacant. It is also made sure that it is assigned to exactly one Aircraft at a time. After successful landing, the contact of Aircraft and Runway controller ends.

```
∪_ Landing ______________________
→ΔATCMain
→a?: Aircraft
∩______________
→As: Airspace
→   ∞ s ε dom flightControl ƒ s ε dom capacity
→     ƒ s ε dom flightQueue ƒ flightQueue s ε seq Aircraft
→     ƒ flightQueue s ℑ {a?} = ©〉
→a? ε dom ground
→Er: Runway
→   ∞ r ε dom groundControl
→     ƒ a? □ r ε ground
→     ƒ (Ea1, a2: Aircraft
→      ∞ (a1 □ r ε ground ƒ a2 □ r ε ground
→      ⇒ a1 = a2))
→ground' = {a?} ψ ground
∠__________________________
```

Invariants

1. All controlled Airspaces with a capacity limit having a finite set of Aircrafts flying in it must not have a duplicated entry of input Aircraft.

2. There must be a Runway assigned to the input Aircraft.
3. The Runway assigned to Aircraft must be controlled and vacant and there must be exactly one aircraft on the Runway.

The model is verified and strengthened using Z/EVES Toolset [5]. 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 proving theorems. While proving our ATC system model in Z-EVES, some proof errors were identified and removed. Thus, our Model has been checked and strengthened by the Tool.

6. CONCLUSION

The power of applying Formal Methods in modeling of a complex, highly distributed and safety critical system is shown, which was one of the objectives of our research. The Informal Model of ATC system developed using UML Diagrams, has been formalized in terms of Z-notation. It shows informal approaches can be integrated into Formal Approaches, which was another objective of our research. By applying Formal Methods, a deeper insight of system to be built has been achieved. The errors and inconsistencies that were found while describing formal system specification of ATC system have been identified in Analysis phase those would have been detected in implementation or testing phase using Traditional Approaches. Therefore, the use of Formal Methods in this research has ensured making high quality, reliable and correct system specifications with respect to ATC system requirements specified in Requirements Analysis.

Another objective of the research was to apply Z-notation for modeling of ATC system because apart from other techniques, the rich mathematical notations offered by Z make it possible to reason rigorously and effectively about the behavior of specified system. Use of Z/EVES tool-set further analyzed the model giving high confidence in our ATC system.

Despite of all these advantages, many practitioners are reluctant to use Formal Methods because of many baseless myths and misconceptions prevailing in market [1]. But Formal Methods are very important for rigorous and concrete modeling of system. This has been observed in development of this ATC system in which we resolved the ambiguities and gave the complete and consistent definition of our system requirements.

REFERENCES

[1] Anthony Hall. “Seven Myths of Formal Methods” *IEEE Software*, 7(5):104–103, September 1990.

[2] C. Larmen. “Applying UML and Patterns”. *Prentice Hall PTR*, 0130925691, 2001.

[3] David Leadbetter, Peter Lindsay, Andrew Neal, and Mike Humphreys. “Integrating the Operator into Formal Models in the Air-Traffic Control Domain.” *Technical report 00-34*, November 2000.

[4] http://travel.howstuffworks.com/air-trafficcontrol.htm

[5] I. Meisels, and M. Saaltink. “The Z/EVES Reference Manual.” *TR-97-5493-03*, ORA Canada, 1997.

[6] J. C. Bicarregui, J. S. Fitzgerald, P. A. Lindsay, R. Moore, and B. Ritchie. “Proof in VDM: A Practitioner's Guide”. *FACIT Series. Springer-Verlag*, 3-540-19813-X , 1994.

[7] J. M. Spivey. “The Z Notation: A Reference Manual”. *Englewood Cliffs*, NJ, Prentice-Hall, 1992.

[8] Len Bass, Paul Clements and Rick Kazman. “Software Architecture in Practice”. *81-7808-546-1, Pearson Education Asia*, India, 2001.

[9] N. A. Zafar. “Formal Model for Moving Block Railway Interlocking System Based on Un-Directed Topology”. *ICET06* , pp. 217-223, Peshawar, 2006.

[10] 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.

[11] Peter A. Lindsay. “A Tutorial Introduction to Formal Methods”. *Proceedings 3rd Australian Workshop on Industrial Experience with Safety Critical Systems and Software*, Australian Computer Society, pp. 29-37, 1998.

[12] R.E. Fields, P.C. Wright and P. Marti. “Air Traffic Control as a Distributed Cognitive System: a study of external representations”. *ECCE9: Proceedings of the Ninth European Conference on Cognitive Ergonomics*, 1998.