U2Z Framework for Improving the Readability of Requirements of Safety Critical Systems

Monika Singh
College of Engineering and Technology,
Mody University of Science and Technology,
Lakshmangarh, Rajasthan, India

V. K. Jain
College of Engineering and Technology
Mody University of Science and Technology,
Lakshmangarh, Rajasthan, India

ABSTRACT
The aim of this paper is to present a framework which helps in accessing and improving the specification especially for Safety Critical System. This proposed framework takes use case diagram as input and produces a formal model of functional requirements as output. This formalization allows the developer to document a correct and complete specification which is the ultimate need for the reliable software. The more accurately the functional requirements are mentioned, the more reliable system will be implemented. In case of the Safety Critical System, correct and complete specifications are indeed. This paper discusses such an integrated framework. We rely on Z Notation for formalization. The further verification and validation of specification is done with Z/EVES.

Keywords
Critical systems, Formal specification, functional requirements, UML, Z Notation, Use case Diagram.

1. INTRODUCTION
Safety critical systems [1] are the one where a minor error may cause potential damage to either human life or to environment or may be to both. For example, control unit of brakes in railway system, air traffic control units, medical equipments, nuclear plants and many others [2-3]. Therefore, development of such systems required more attention. The root of reliable systems depends on how accurately the requirement specifications are documented as once the requirements are freeze, the next phase of software development process takes the input of previous phase. The main functionalities of systems are known as functional requirements [4] of the systems. The functional requirements are explicitly mentioned by the stakeholders. There are various methods and technologies for specifying the functional requirements. One of the ways is to use graphical modeling language. As a pictorial representation tends to get a better and clear understanding of what actually the customer wants. Unified Modeling Language (UML) [5] is a de-facto standard and used widely for visualizing and designing the software artifacts. It enhances the analysis and design of software system by allowing more cohesive relationships b/w objects. It has been observed that graphical representation of model is easily accessible and understandable to the user. The primary gap between the developer and the user has been easily fulfilled by the graphical description.UML composed of nine diagrams: Use case diagram, class diagram, sequence diagram, state diagram, activity diagram, interaction diagram, component diagram, deployment diagram and package diagram. Graphical representation always gives a better understanding of the proposed system. The UML- use case diagram defines the behaviour of a system i.e. the functionality of the system. Therefore one can get better understanding of system behaviour by making use case diagram of the system which further forms the root of Software Requirement Specification (SRS). Although UML has numerous good attributes yet not accepted for designing the safety critical system alone. One of the reasons is the lack of preciousness in semantic used in graphical model. Consequently, ambiguities are introduced. In case of safety critical system, even a minor ambiguity may cause serious hazards or even loss of life. Moreover, UML lacks features that would allow attaching non-semantic information to model.

There are famous examples of failure of safety critical system merely due to either incomplete or poor requirements such as:

-Mars Climate Orbiter ($125 million)

_Therac–25

_Bhopal (3–10K deaths, 500K injured)

One way of resolving these issues is to use a formal model in integration with UML [6]. Formal methods [7] use the discrete mathematics which includes set theory, first order predicates, logics and graphs. The lack of preciousness in UML semantics can be covered by the rigorous mathematics used by the formal methods by integrating UML with formal model, the expressiveness of graphical notation increases which ultimate enhance the modeling power of UML diagrams especially at analysis and designing part. The approach used for formalization is given in figure 1. Rest of the paper is organized as follow. Section 2 presents the ingredient of proposed framework. Section 3 validates the U2Z framework with a real –time safety critical system. Results and simulation are discussed in section 4. Section 5 unfolds the conclusion and discusses future directions.

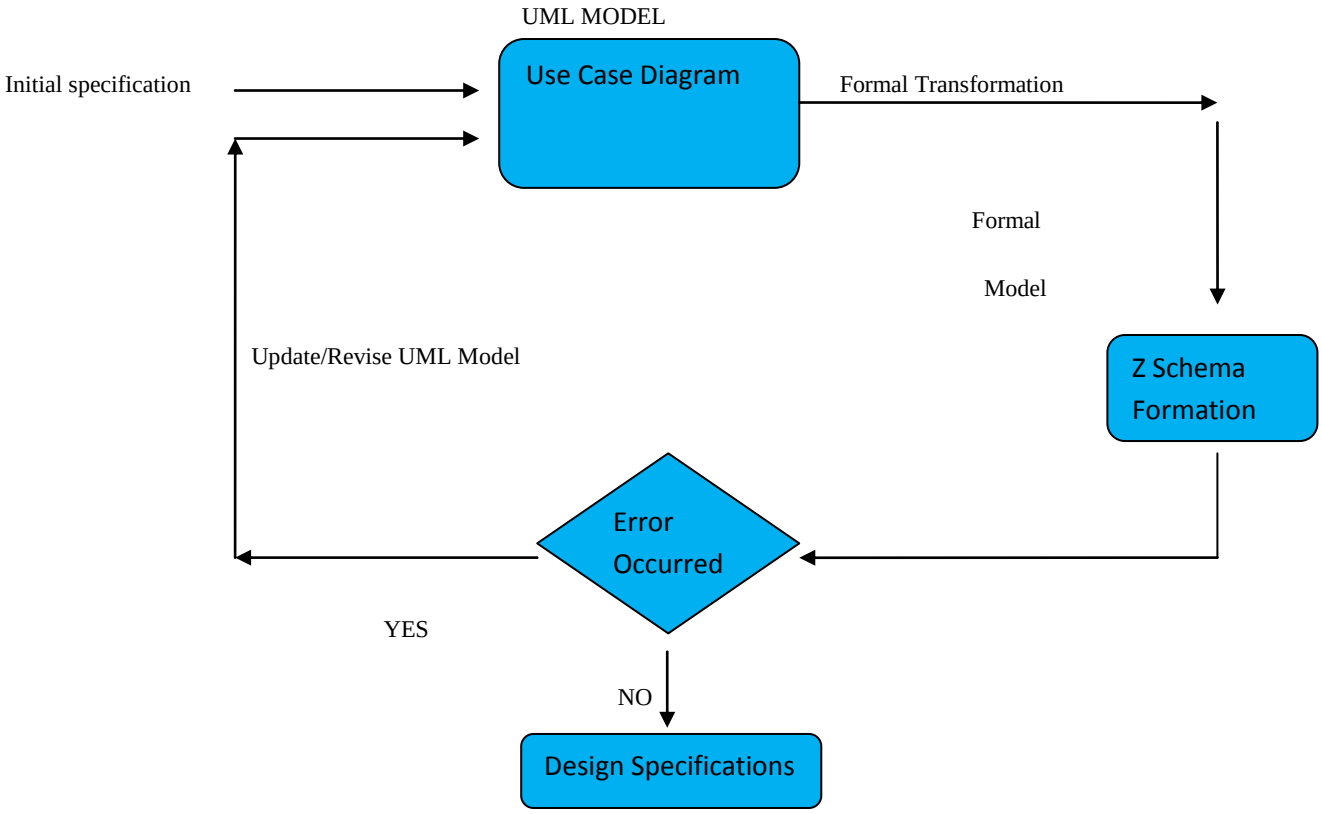

Figure 1: U2F Framework for Safety Critical System

2. COMPONENTS OF U2Z FRAMEWORK

This section describes briefly the ingredients of proposed framework. The main ingredients are: Unified modeling Language and Formal Methods.

2.1 Unified Modeling Language (UML)

Unified Modeling Language [3] is widely used for modeling the software system. Being rich in graphical notation UML regarded as a de-facto standard in visualizing and constructing the software system. UML diagram captures both the view of system i.e. static as well as dynamic by including its constituent diagrams, for instance Use case diagram are primarily used to maps the functional requirements of proposed system. The UML diagrams are: use case diagram, class diagrams, sequence diagrams, interaction diagrams, activity diagrams, state diagrams, component diagrams, object diagrams and deployment diagrams. However UML is a semi-formal language due to which it is prone to cause errors. Moreover, UML allow ambiguities at the design level due to its hidden semantics in computer software systems. There are nine diagrams in UML to model graphically any given system. Use case diagrams [5] are used to capture the interaction between the user and the system. In other words, Use case diagrams are used for capturing and improve the functional requirements of the system. The basic constituents of Use case diagram are: (a) Actors, (b) Use Cases, and (c) Relationships among the use cases and actors.

2.2 Formal Method

Formal methods [7] are the mathematical methodologies used to validate the software system by mathematical means. In context of Formal method, the verification and validation of specification is done by two means: Model checking [8] and Theorem proving [9].

Theorem proving This technique is used when the system is specified through mathematical definitions. Such system is verified using automatic/semi-automatic Theorem Prover, which are based on a library of axioms and a set of predefined inference rules. Automated theorem proving allows proving the properties of the system automatically, without human intervention. This technique is very expensive in terms of time and resources and is not practical for many complex specifications. Therefore, there exists interactive Theorem Prover which allows the designer to guide the proof. For example, Z/EVES [11] are a semi-automatic Theorem Prover, which allows proving theorems for verifying specifications written in Z notation [10]. Apart from these techniques, Formal methods can also be classified based on application area of method in two categories:

Model-oriented methods: The specification of system consists in defining a model of the system in terms of mathematical structures such as relations, functions, set and sequences. VDM [12], B [13], and Z notation [9], Communicating Sequential processes (CSP) [15], Calculus of Communicating Systems (CCS) [14], and I/O automata.

Property-oriented methods: The specification of the system consists in defining some properties, usually in terms of axioms that should be satisfied by the system. OBJ, LOTOS [16] is formal languages that belong to this category.

In the context of this paper, formalization is done by Z – Notation.

3. VALIDATION OF U2ZS FRAMEWORK

Functional requirements are defined as the statements of services the system should provide how the system should react to particular inputs, or how the system should behave in particular situations. In other words, behaviour or function will be specified by the functional requirements. They **describe what actually system will do**. Use case diagrams are used to capture the functional requirements of a system**.** Logically**,** uses cases are nothing but the system functionalities written in a systematized way. The components of use case diagrams are: (i) use case, (ii) Actors, and (iii) Relationships among the use cases and actors. To validate the Use case diagram and its components, Z notation [8] is used in proposed approach. In Z notation, Schema is the notion used to structure the specification written in Z notation. The generic structure of schema consists of three parts and presented in figure 2 as below:

- Schema Name
- Variables declaration
- Constraints

SchemaName

Variables declaration

---

constraints (*preconditions or postconditions*)

Figure 2: Basic Schema structure

Therefore the resultant of formal model of use case diagrams is the Z schema of all the three above listed components which includes following schemas:

- Figure 3 presents the formal model of Use case diagram.
- Figure 4 depicts the formal aspect of Actor schema
- Figure 5 represents the formal model of Use case Relationship

Use Case Diagram

Actors
UseCases
Relationship_of_Actors
Relationship_of_UseCase

Figure 3: Schema of Use case Diagram

The actor schema consists of basic data type as [Actor_name, Role] and three variables:

- Human, external or internal application which is the set of all possible Actor
- Act-Role is the set of actor role; the intended actor playing.
- aRole is the function which maps the actor name to its role**.**

Therefore the schema of Actor will be:

Actor-name

human, external application, internal application: $_1$ *Actor-name*
Act-Role: $_1$ *Role*
aRole: *Actor-name* → $_1$*Role*

⟨ *human, external application, internal application*⟩ ∈ $_1$*Actor-name*
dom aRole = human ∪ *external application* ∪ *internal application*
ran *a Role = Act-Role*

Figure 4: Schema of Actor

Now the constraints are:

- The actor name should be from a finite set and the finite set can be of human or non-human external, internal application.
- The domain of actor role should be the union of all finite sets of human, external and internal applications as possible actor's name.
- The range of aRole function is the finite set Act-Role; set of all defined roles by the actors in a given scenario.

The next schema in this series is UseCase schema. Figure 5 presents the UseCase schema with its variables and constraints as follow:

UseCase

Ucase: $_1$*Usecase-name*
Scenario: seq$_1$ *Events*
Usecase: $\mathbb{P}_1$ ◦ *Scenario*
Expoints: $_1$ *Extension-points*
Relation: *Usecase-name* ⇸ $_1$ *Extension-points*

---

dom Relation = Ucase
ran *Relation = Expoints*
∀ *uc* ∈ *ucase*; ∀ *eX*∈ *extension*(*uc*) *s.t.*
(*extension*(*uc*))>1 •
(*extension*(*uc*)\{*eX*}) ∩ {*eX*}= ∅

Figure 5: Schema for Use case

The invariants are:

- The domain of Relation is the set of all use cases for a particular scenario
- The range of Relation is the set of extension points and

For all use cases, there exist extension points for each use case such that the number of extension points for use case is greater than one. It implies that the extension points corresponding to a use case are distinct.

4. SIMULATIONS OF RESULTS

The functional requirements captured by Use case diagrams and written in Z notations are now verified by using semi-automated Theorem Prover tool .i.e. Z/EVES [9]. The schemas written in Z specifications are given as input file and test for syntax and type checking errors using Z/EVES tool.

The figure 6 shows the simulated results of Use case diagrams and the Actor schema respectively.

To enrich the significance of propose framework, a case study of road traffic management system has been considered in this paper. The Use case diagram of actor: vehicle owner for instance, is shown in figure 7.

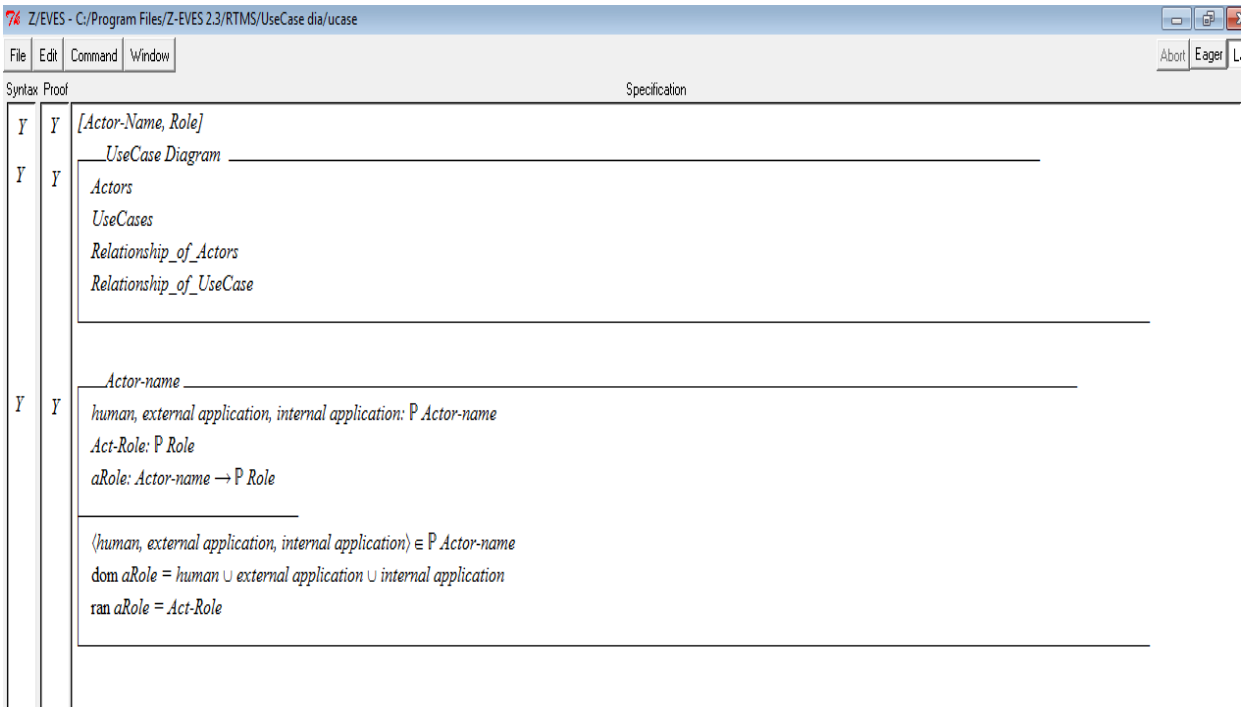

Figure 6: Simulation of Schemas

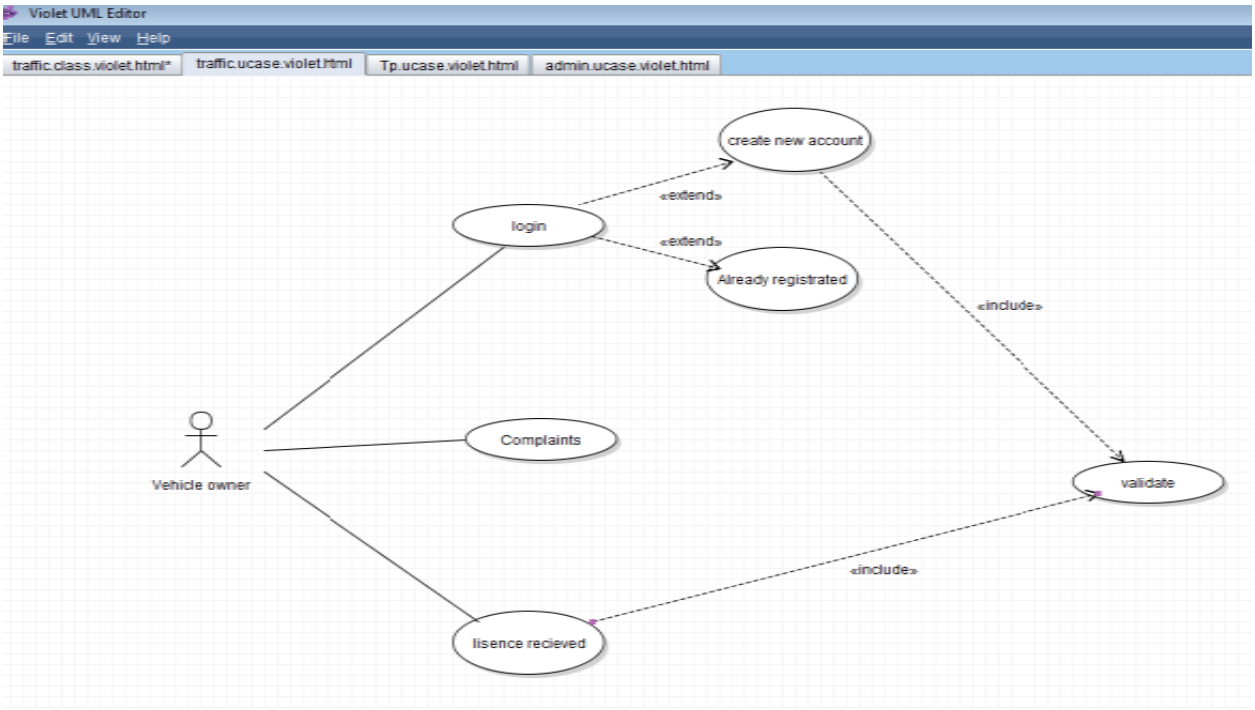

Figure 7: Use case diagram of Vehicle owner Actor

The simulation results of vehicle owner using Z/EVES are shown in figure 8 as follow:

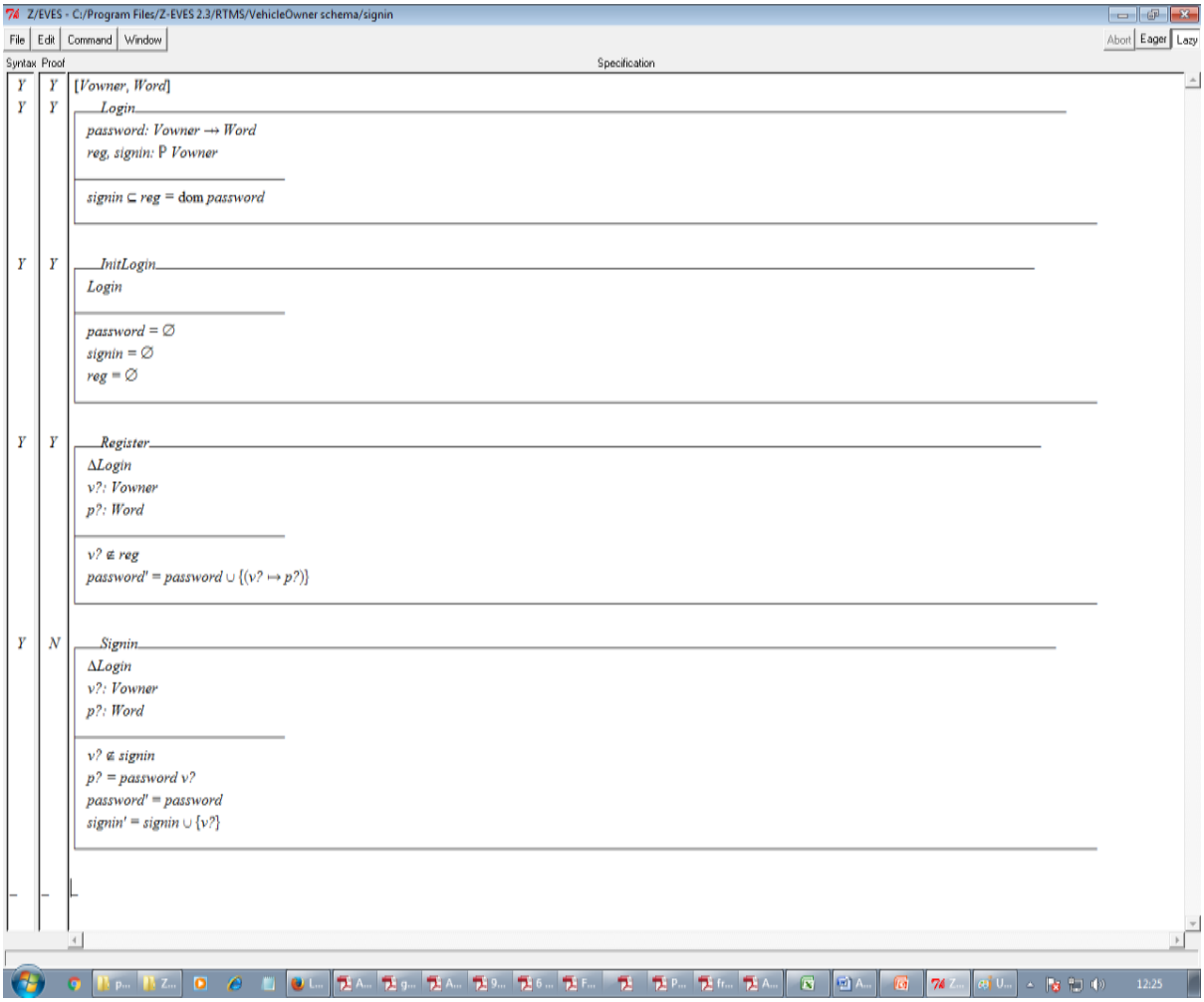

Figure 8: snapshot of Formal model of Use case diagrams of Vowner

Moreover, Table 1 depicts the model analysis results of formalization of use case diagrams for syntax, domain and type checking.

5. CONCLUSION

Being known as mature graphical modeling language, UML can't be used alone to specify the functional requirements of Safety Critical System as UML is semi-formal in nature and lack preciousness. This can't be tolerated in case of Safety critical system as a minor misinterpretation result into a tremendous consequence or even loss of human life too. The proposed framework presents a formal model for completeness, correctness and consistent functional requirements. The use of rigorous mathematics, the specification constructed by using Z notation are tends to more accurate and complete which is desired for constructing Safety critical application. To enrich the U2Z framework, a case of traffic system is discussed and result are analyzed with Z/EVES for syntax, domain, type checking and for modularity.

Table 1: Model Analysis Result

Schema Name	Syntax & Type checking	Domain Checking	Proof	Reduction
Use Case	Y	Y	Y	Y
Use Case Diagram	Y	Y	Y	Y*
Relation	Y	Y	Y	Y
Use Case Relationship	Y	Y	Y	Y*

6. REFERENCES

[1] W. R Dunn, 2002. Practical Design of Safety-Critical Computer Systems, USA: Reliability Press.

[2] J. Burcsuk, 2007, "Development of safety related systems," in Strategic Technology, 2007. IFOST 2007. International Forum on 3-6 Oct. 2007, pp.564,569.

[3] S. Yang, N. Sang, G. Xiong, 2004. Safety Testing of Safety Critical Software Based on Critical Mission Duration. In 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'04), pp. 97-102.

[4] K. E. Wiegers, 2003. Software Requirements Microsoft Press.

[5] Object Management Group (OMG), 2003. OMG Unified Modeling Language Specification, version 1.5.

[6] G. Booch, J. Rumbaugh, and I. Jacobson, 1999. The Unified Modeling Language User Guide. Addison-Wesley.

[7] Groote, J.F., Osaiweran, A.A.H. and Wesselius, J.H. (2011) Benefits of Applying Formal Methods to Industrial Control Software. 1-10.

[8] Fulara, J. and Jakubczyk, K. (2010) Practically Applicable Formal Methods. *Lecture Notes in Computer Science*, **5901**, 407-418.

[9] N. Amálio, S. Stepney, and F. Polack, 2004. "*Formal Proof from UML Models*", ICFEM, USA, pp 418-433, Springer.

[10] J. Michael Spivey, 2001. The Z Notation: A Reference Manual, Prentice Hall, Englewood Cliffs, NJ, 2nd Edition.

[11] Saaltink, M., 1999. *The Z/EVES 2.0 User's Guide*, Technical Report TR-99-5493-06a, ORA Canada, One Nicholas Street, Suite 1208 - Ottawa, Ontario K1N 7B7 – CANADA.

[12] C. B. Jones, 1990. Systematic Software Development using VDM, In Prentice Hall.

[13] S. Schneider, 2001. B Method- an Introduction Palgrave, Cornerstones of Computing series.

[14] J. Guttag and J. J. Horning, 1978. The algebraic specification of abstract abstract data types, Acta Inform., vol. 10, pp. 27-52.

[15] C. A. R. Hoare, 1985. Communicating Sequential Processes, In Prentice Hall.

[16] Howard Bowman, 1998. A LOTOS based tutorial on formal methods for object-oriented distributed systems, New Generation Computing, Volume 16, Issue 4, pp 343-372.