Use of Formal Methods in SE (Software Engineering) with Examples

Muhammad Ijaz Ijaz
Virtual University of Pakistan, Lahore
Pakistan
ms140400204@vu.edu.pk

Mustafa Hameed Ghulam Mustafa
Government of Punjab, TEVTA
Pakistan
ghulammustafahameed@gmail.com

ABSTRACT: *In this paper formal methods are discussed for software engineering. Formal methods are very important for critical systems development where system properties such as security, reliability, and safety are very important. Critical systems have very high validation costs and the costs of system failure are large and increasing. Formal methods can reduce these costs. They are used to improve the software quality, reliability, usability, dependability and maintainability.*

Formal methods based on mathematical models; two basically tools are used to build the mathematical model that are logics and set theory. Formal techniques use the discrete mathematics and mathematical logics. Formal methods also provide the correctness proofs.

The formal method's area of application is requirement analysis, specification, design & implementation and verification & validation that are briefly discussed.

In this paper the some important formal methods, techniques and their importance is elaborated with example. Especially propositional logic and Calculate assignment are also discussed with examples.

Keywords: Formal Methods, Critical Systems, Dependability, Reliability, Usability, Intensive Systems, Formal Semantics, Colleting Semantics, Propositional Logics, Calculating Assignment, UML (Unified Modelling Language), OCL (Object Constraint language)

Received: 13 April 2016, Revised 20 May 2016, Accepted 28 May 2016

1. Introduction

All Formal methods are effectively used in transport, nuclear, financial, defence, healthcare, consumer electronics, telecom, offices & administration and others applications and software systems [1].

Formal methods base on formal specifications and formal models, formal techniques base to prove the refinement or equivalence.

For the formal methods, the software engineers should have excellent experience to demonstrate and solve the discrete mathematics and mathematical logics.

The formal methods are significance for software systems and hardware systems (hybrid systems).Our lives mostly depending on computer based systems such as media player, cooking control system, banking systems, airplane control system, airplane ticket booking, cooling systems, GPS navigation systems, train control systems, traffic light control systems, air craft and many others are available in the world for human facility. If failure of any system means result of system in accident and failure, the lives could be lost so the software should be intensive, reliability, and dependability to save the cost, human effort and lives when such systems or embedded systems are used as daily routine in ours lives [2]. We can develop the intensive, reliable and dependable software by using the formal methods and techniques.

The scope of formal method is that: the use of formal methods is increasing in the area of critical systems development, where emergent system properties such as safety, reliability and security are very important. The high cost of failure in these systems means that companies are willing to accept the high introductory costs of formal methods to ensure that their software is as dependable as possible. Critical systems have very high validation costs and the costs of system failure are large and increasing. Formal methods can reduce these costs The requirement engineering is a 1st step for high quality development process. Formal methods are very useful for the requirements elicitation, requirement articulation and requirement representative [3].

Formal method tools have good approach to check the quality assurance activities such as completeness, reusability and traceability and verifiability. Formal method tools and techniques support the requirements evolution, the management inconsistency and the requirements diverse as diverse viewpoint [4].

Formal methods are used for software specifications for example ASM [5], B [6], and VDM [7]. A technical contract is provided as specification between the software engineers or programmers and client such as design by contract. Both client and programmer provide the benefits and obligations for each other that is a contract [8].

State machine specification, abstraction functions and simulation are involved in the data refinement, where formal methods give the proofs when that is used in software design [9].

The formal methods are used at the implementation level for code verification. Code or program verification uses the inductive assertion method for code or program correctness that was invented by the Floyd and Hoare [10] and the mathematical assertions is used for annotating program where formal model automatically generate code for example the B-method [11] and SCADE [12].

The formal methods give the support for the software maintenance [16]. The formal methods give the support for software evaluation [17]. Many applications of formal methods are available to maintain the software legacy code in which mostly are Microsoft applications, every 10 lines is an assertions [18].

The UML (Unified Modeling Language) is used. For the (OO) Object Oriented as the use of full standard notations, UML is a complete set of notation for (OO) Object Oriented paradigm but the problem of informal semantic. Here the need to use the formal semantic to formalize the concepts overlapping (diagrams for example class vs actor), to complete the inconsistencies in the diagrams or relationships between the diagrams. Formal semantics is also used to formalize the ambiguous concepts such as semantics (informal semantics) of aggregation relationship between the classes. For this purpose some specification languages are used for example; the use of some languages that are used for formal specification such as Algebraic specification languages, Model-based specification languages, OCL (Object Constraint Language is used for the first order functional language) and state based language (like Z) [13].

On yearly basis hardware dependability is continually increasing that mean time to failure; dependability measurement is failure in this regards the software design is very important. Design should be able to adopt the changes with control, corrective operation, validity and verifiability that means where need to meet the changing requirements the design should be extensible with open and close principles. For this purpose, the formal methods are used very effectively in (OO) Object Oriented paradigms as design semantics to produce the intensive, dependable, verifiable, reliable, maintainable and qualitative software applications and systems [13].

2. Related Work

Formal methods base on formal specifications and formal models, formal techniques base to prove the refinement or equivalence.

Another word the formal methods are used to developing the computer based software and hardware products or systems on based mathematical techniques and tools. When we need to develop such computer based software and hardware products or systems that give the guarantee perfection where need to use the formal methods.

The formal methods perform such activities to develop the computer based software and hardware products or systems that give the guarantee perfection such as to write the formal specification, to prove the properties about written specification, to construct a program code by using the mathematical manipulate the specification, to verify the program code using the mathematical tools and arguments.

The formal methods increase the cost of software development product but in fact that is opposite means the formal methods decrease the cost of software development product.

In the safety critical project; the use of formal planned test and formal specification is better to achieving the high quality productivity. IBM has shown that the cost of learning of formal methods is extremely important as a onetime cost.

Formal methods provide the considerable improvements in V and V (Verification and Validation). The use of formal methods is limited but mostly use in high risk areas likely security and safety. Formal methods have four level of rigour (formal methods are not use on Level 0, concepts and notations of discrete mathematics are used on Level 1, formalized specification language is used on Level 2 and full formal specification language with comprehensive support environment, including the mechanized theorem proving or proof checking is used on Level3).

The formal method's area of application is Requirement analysis, specification, design & implementation and verification & validation as followings:

2.1 Requirement Analysis

This area has a limited value and high level communication with customers, because at time of product delivered the customer should be satisfied; otherwise you are failing in the field of software engineering. The high level communication is very important to get the correct, meaningful and unambiguous requirements from the customer.

Some areas of the applications are very significant such as defence systems, medicine, energy industry, aerospace, airplane, railway control systems and transportation for the regulatory authorities. The safety standard is very important for regulatory boards; the regulatory boards enforce the some certification criteria that provide the satisfaction according to the safety standard for different parts before the system put into the service. Computer involvement in such critical systems is increasing on daily basis, there is urgently need new efficient and effective methods for development of such critical systems: likely formal specification and verification with other methods of software development such as software fault tolerance and software testing are conjunctively used in development process. Mostly software engineers suppose the formal methods is only way to achieve the high level safety, security and correctness for critical applications or systems. By using the formal methods separately from causative we can improve the understanding of the whole specification, the refinement consistency, the include unambiguity and giving the way to check the completeness (in a way such as with respect to a predefine set of questions as key and inference based on the information specified). Here an efficient and effective systematic development method is needed during requirement analysis to give the efficient and effective analyst way to concentration on the realities of such systems. Separation of Mission and critical requirement is a good technique for any safety critical systems. In the mission requirements security, availability and reliability are attributes of dependable requirements. Timeliness and to consider getting the term of function are concentrated in mission requirements, and what are. Some giving the benefits by using the formal methods in the requirement analysis such as resolve the potential conflicts, to detect the omission error and inconsistencies between the mission and critical issues, to simplify the safety certification and to create the ability to attention on the safety and critical issues.

For the critical requirements two phases are very important: in the first phase to be identified real world properties and the system hazards, in the second phase to be mapped the real world state into the system. We can say the concerned laws and rules to directive the behaviour of the system and how use of these concerned laws and rules for perceiving and handling by system. Use of formal methods in requirement analysis, we can exploit and accommodate the differences in the demands and characteristics of different safety critical application areas, for example different level of risk are involve in turn demand cost related solution.

In the generally structured the safety critical systems in different phase such as process control systems and three distinct components (the operator, the controller and the physical process) to usefully guide the requirement analysis such as in Figure A.I.

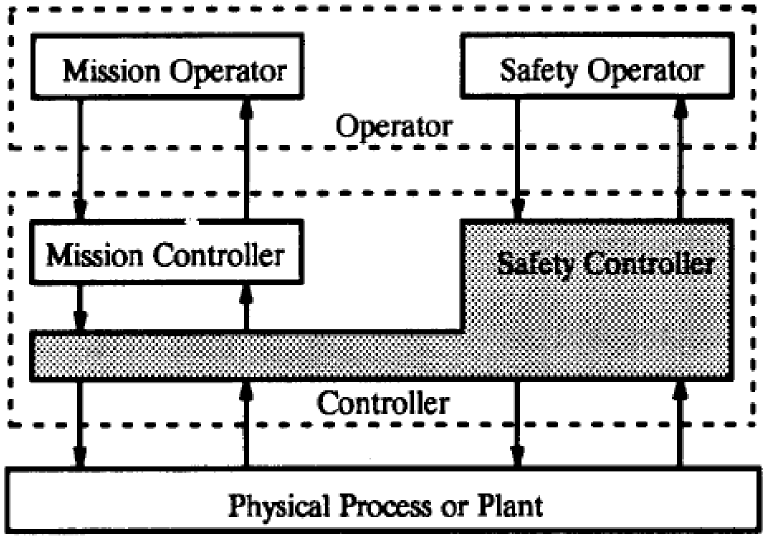

Figure 1. System Framework

This is a general framework that is used for the formal specification and verification of the critical requirements in the development of safety-critical system that is also called general structure of safety-critical system [14].

2.2 Specification

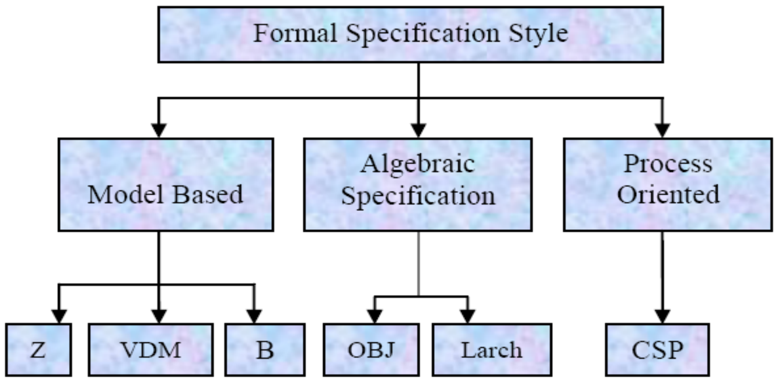

Figure 2. Specification Style

This area of application has the use of some languages that are used for formal specification such as Algebraic specification languages, Model-based specification languages, Set based approaches and Formal methods for concurrent systems. These formal method techniques provide the general benefits when the formal language to make more concisely and explicitly specifications.

The specification style Figure 2.

Algebraic specification languages (level2)

- OBJ
    - Executable specification language
    - Behaviour is expressed by rules (equations)

Model-based specification languages (level 2)

- Vienna Development Method (VDM)
    - VDM has its own 3-valued logic, which allows treatment of undefinedness not explicitly treated in Z or B
    - Operations, states
    - Pre- and post-conditions
    - Programming constructs: while, if. Then else,
    - No time concept, but extensions to cope with time possible
    - Only sequential systems

Set based approaches (Level 2)

- Z (University of Oxford)
    - Based on typed set theory
    - Variables and axioms
    - No time concept, but extensions to cope with time possible
    - Only sequential systems
- Object-Z
    - Extension of Z
    - Object-Z introduces class structure (which encapsulates operations that can affect that state)
    - Class represented by event histories (i.e., sequences of events)
    - Allows specification of properties by restricting set of allowable histories

Formal methods for concurrent systems (level 2)

- Temporal logics (CTL, LTL)
- Process algebras (LOTOS, CCS)

Formal Specification Languages (3/3)
Level 3 approaches:

- Higher Order Logic (HOL)
- PVS
- Boyer-Moore
- B (level 3)
    - The B Method synthesizes many approaches to formal methods; it is also based on set theory
    - Predicate-transformer style of specification; you write the state change as a substitution
    - Explicit preconditions and invariant

• Includes imperative programming constructs as part of the notation

• Refinement down to code within the same semantic framework

• Object-based; information hiding of various kinds is enforced, so that encapsulation of state by operations is the order of the day.

• There is no proper concept of inheritance or polymorphism, so it stops short of being object-oriented

When the specification is informal, the specification will be free form and in natural language with ambiguity and lack of organization, for such reasons the specification can be incompleteness, misunderstandings and inconsistency.

Formal specification methods consist on formal specification, formal proofs, model checking and abstraction.

• Formal Specification:

In the formal specification, non-mathematical description of intensive systems such as diagrams, tables, written text is translated into a formal specification language. High level behavioural description and system properties are concise and some semantics language (well defined) supports the formal assumption about specification.

The more efforts are required for the formal specification involvement in the primary phases of software development. The formal specification reduces the system's requirements errors by using the requirements analysis application area that give the result in discovered and resolved the incompleteness, misunderstandings and inconsistence of systems. This gives the benefits as savings expanse of rework and reduced the requirement problems. See the following see in following Figure 2.2, for formal specification in the software process [15].

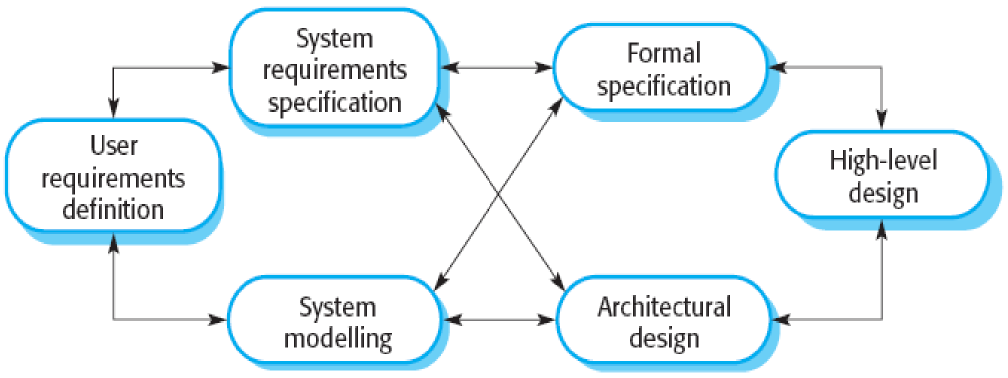

Figure B.II

Formal Specification in the Software Process

Figure 3. Software Development Cost with Specification

The information of the system is vice versa between the specification and design process.

Cost profile of a project changes can be reduced (for implementation and validation costs) by using the formal specification such as more effort and time (greater up-front costs) are spent developing the specification. This process can reduce the requirement errors and ambiguity. Approximately 50% about of development costs is validation cost and about double the costs of specification are implementation cost and design cost in the conventional process, but when we use the formal specification, the specification and implementation costs are comparable and the systems validation costs are expressively reduced (with formal specification methods).

The software development cost with the formal specification is presented in Figure 3.

• Formal proofs:

By using the formal specification, we can get the formal proofs such as completeness and convincing argument that shows the validation of some properties of the described system. For this purpose some sequences of steps are constructed that are justified by using a small set of rules such as when illustration the informal conclusions, to eliminate the ambiguity and subjectivity characteristics, this constructed with automated assistance some time may be manually.

• **Model Checking:**
In model checking, emphasis the operational specification rather than the analytical specification. Basically it is used to identify the all reachable paths in the computational tree that are derived from the State Machine Model. It is very important determination by using the model checker when the requirements expressed in a given logic as formula are satisfied by given the finite state machine.

• **Abstraction:**
In the abstraction, simplified the requirements specification and ignored the irrelevant details, generalized the major vital properties and characteristics and avoid early commitments with design and implementation choice.

2.3 Design and Implementation

There is need for formal methods which should be able, abstract and model system at a suitable level of presentation of an intensive system. That is an abstraction interpretation which is used to develop solid design principles, developing the complex systems, proving the properties of more complexes, large and intensive system such as important concurrency and among the component interaction of those systems. Abstraction interpretation is very easy to prove the said properties of the systems. A big issue for software engineering community is a good design, because hardware dependability is increasing continually day by day. It is not software engineering goal to produce the good testable code; the good design is a most important than code that should be simpler, extensible and flexible with good design principles, criteria and techniques. Software must be able to changeable means intended to change.

Formal verification methods are used to verify the specification, development and software hardware systems based on mathematically techniques. It identifies the required property such as specification for satisfaction by using the formal model of the behaviour of the system that model is called the semantics. All such models that are used to check the satisfaction of behaviour of the system are formal model and also called semantic domain.

The strongest property of the system is a proof that satisfies the property of the system as equivalent to the defined the strongest property of the system by using the set of semantic models. It is also called collecting semantics. Beside of these described techniques some other techniques are also used to create the good design such as trace semantics of a programming language, using the theorem prover and a model checker or a static analyzer with formal verification methods and without great cost for computational to define the specification and the semantics of complex system that is very difficult and hard to define some more formal design techniques are following that are used to create the software design as simpler, extensible and flexible with good design principles, criteria and techniques. Software must be able to changeable means intended to change:

• Abstract Interpretation

• Abstract Domain

• Concretization function

• Incomplete

• False Alarms

• Abstraction Function

• Reasoning methods and effective algorithms

• Verification by Static Analysis. Static code analysis

• Dynamic analysis

The specification is transformed into an architectural design, a detailed design and an implementation in the Figure 4:

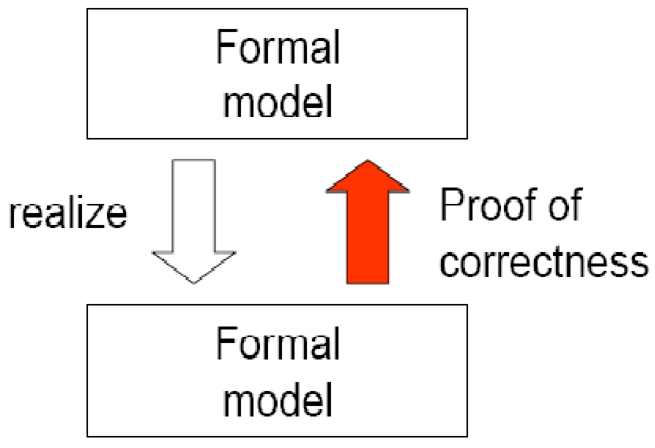

Figure C.I

Figure 4. Step-wise Refinement

Some steps are following in Figure 4. for Stepwise Refinement:

• The more detailed model that is **Realize**
The model fulfills the model or specification that is **Proof**

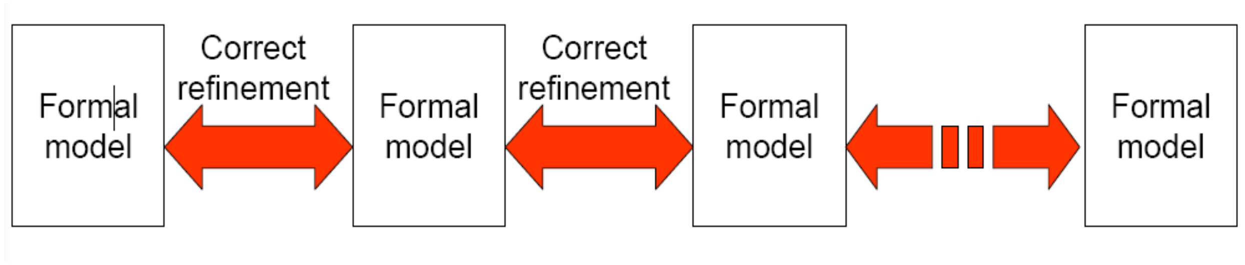

Figure C.II :- Some Steps for Stepwise Refinement:

Figure 4.1 Step-wise Refinement

2.4 Verification & validation
In the formal code verification, the formal methods and techniques are used to prove the code correctness of a given program against the formal specification of its requirements. All possible given program code executions are identified that is infeasible by dynamic testing alone.

Pre-conditions and post conditions are analyzed. Safety and security properties are validated. Formal methods give the proof of absence of run time errors and identify the loops and recursion termination

Formal methods, formal techniques and tools perform great role in the verification and validation.

Problem statement 2.4.1:
Consider the following code segment:

```
if ((input1 > input2) and not (( input1 > input2) and (input3 < 500)))
        do something here;
```

```
else
        do something else here;
```

Convert this if statement into propositions, simplify them using propositional rules and then re-write the above code segment using simplified version of if statement. Show all steps.

Solution:-
Let suppose:

p= (input1 > input2) and q=(input3 < 500)

Now we Convert the if statement into propositions as:

$$If\ (p \wedge \sim(p \wedge q))$$

By simplifying we get:

$$p \wedge \sim(p \wedge q)$$

$$p \wedge (\sim p \vee \sim q) \qquad \text{by applying negation}$$

$$(p \wedge \sim p) \vee (p \wedge \sim q) \qquad \text{by using distributive law}$$

$$(false) \wedge (p \wedge \sim q)$$

$$(p \wedge \sim q)$$

Re-write the above code segment

```
if ((input1 > input2) and not (input3 < 500)))
        do something here;
else
        do something else here;
```

Problem statement 2.4.2:
The factorial function m! Satisfies the equations 0! = 1 and, for all natural numbers m, (m+1)! = (m+1) x m!

Suppose program variables f and m satisfy the property f = m! And it is required to maintain this property by a suitable assignment to f whilst simultaneously incrementing m by 1. Calculate the assignment.

Solution:-
Suppose program variables f and m satisfy the property f = m!
We want to increment the 'm' by 1 to maintain the relationship between 'f' and 'm'.
Now we will calculate the suitable assignment for 'f'
{ f = m! } m, f :=m+1, f * m { f = m! }
By applying assignment axiom we get:
f*m= (m+1)!
f= m! $\rightarrow$ f*m= (m+1)!
As (m+1)!= (m+1)m!
so =(m+1)f :. f= m!
Hence,
f= m! $\rightarrow$ (m+1)f = (m+1)!
So
{ f = m! } m, f := m+1, (m+1)f { f = m! }

Problem statement 2.4.3:
For this question, suppose you are on the island of knights and knaves. Remember that knights always speak truth while knaves always tell a lie.

Part 2.3.1
You would like to determine whether an odd number of A, B and C is a knight. You may ask one yes/no question to any one of them. What is the question you should ask?

Part 2.32
Suppose you come across two of the natives. You ask this question "whether the other one is a knight?" from each of them. Will you get the same answer in each case? Justify.

Solution of Part 2.3.2
The question will be:

"Are the other two knights?"

Let suppose A is a knight and we are going to ask from A Table ***Part 2.3.1***

A	**Q**	**B**	**C**
T	T	T	T
T	F	F T F	F F T
F	T	F T F	F F T
F	F	F T F	F F T

Solution Part of Part 2.3.2
Suppose the two natives are A and B.
A= A is a knight
B= B is a knight
Q= whether the other one is a knight

As we not know that these two natives are knight or knave but we know that if A is a knight than he will speak truth similarly if B is knight he will also speak truth and in case of knave both will tell a lie.

Now we can say that when we ask the question from A then he will be true if and only if A is knight. Similarly for B we will apply iff.

For Native A		
A	Q	A ⇔ Q
T	T	T
T	F	F
F	T	F
F	F	T

For Native B		
B	Q	B ⇔ Q
T	T	T
T	F	F
F	T	F
F	F	T

3. Conclusion

In this paper formal methods and their importance is discussed. The formal methods are significance for software systems and hardware systems (hybrid systems).Our lives mostly depending on computer based systems such as media player, cooking control system, banking systems, airplane control system, airplane ticket booking, cooling systems, GPS navigation systems, train control systems, traffic light control systems, air craft and many others are available in the world for human facility. If failure of any system means result of system in accident and failure, the lives could be lost so the software should be intensive, reliability, and dependability to save the cost, human effort and lives when such systems or embedded systems are used as daily routine in our lives .We can develop the intensive, reliable and dependable software by using the formal methods and techniques. The formal methods are used in the requirement analysis, specification, design & implementation, and verification & validation areas for developing the intensive and critical systems and also discussed the specification style in Figure 2.2.1

In the generally structured the safety critical systems in different phase such as process control systems and three distinct components (the operator, the controller and the physical process) to usefully guide the requirement analysis ; see in Figure 2.2.1.

The more efforts are required for the formal specification involvement in the primary phases of software development. The formal specification reduces the system's requirements errors by using the requirements analysis application area that give the result in discovered and resolved the incompleteness, misunderstandings and inconsistence of systems. This gives the benefits as savings expanse of rework and reduced the requirement problems. See in Figure 2.2.2 for formal specification in the software process.

Cost profile of a project changes can be reduced (for implementation and validation costs) by using the formal specification such as more effort and time (greater up-front costs) are spent developing the specification.

Formal verification methods are used to verify the specification, development and software hardware systems based on mathematically techniques. It identifies the required property such as specification for satisfaction by using the formal model of the behaviour of the system that model is called the semantics. All such models that are used to check the satisfaction of behaviour of the system are formal model and also called semantic domain. Software must be able to changeable means intended to change some formal techniques such as abstract Interpretation, Abstract Domain, Concretization function, Incomplete, False Alarms, Abstraction Function, Reasoning methods and effective algorithms, Verification by Static Analysis. Static code analysis, Dynamic analysis are used for design the intensive and critical system.

Formal methods, formal techniques and tools perform great role in the verification and validation. Pre-conditions and post conditions are analyzed. Safety and security properties are validated. Formal methods give the proof of absence of run time errors and identify the loops and recursion termination.

In the formal code verification, the formal methods and techniques are used to prove the code correctness of a given program against the formal specification of its requirements. All possible given program code executions are identified that is infeasible by dynamic testing alone.

The formal methods are used to developing the computer based software and hardware products or systems on based mathematical and logical techniques and tools. When we need to develop such computer based software and hardware products or systems that give the guarantee perfection where need to use the formal methods.

4. Acknowledgement

I am grateful to my Supervisor of CS709 – Formal Methods for Software Engineering. He taught basic concepts, techniques and methodologies of this subject that gave me the great help to do the research in best way. He gave the respectable comments on my research proposal and work to improve the qualitative research in successful way.

I am thankful to the Virtual University of Pakistan for giving me the chance to learning the great door step education on MS level with great research work and course work based on excellent and qualitative techniques and methodologies.

I am also thankful to HOD of Computer System Department, UCET, IUB who motivated and gave me the enough time to complete my study activities with due date such as assignments, preparation midterm and final term papers and especially in my research work like term paper.

References

[1] Joseph, Mathai. (2005). Formal techniques in large scale software engineering. Keynote at IFIP Working Conference on Verified Software: Theories, Tools and Experiments (VSTTE), Zurich (October 10-13, 2005), http://vstte. ethz. ch. 2005.

[2] Hinchey, Mike, (2008). Software engineering and formal methods, *Communications of the ACM* 51.9, 54-59.

[3] George, Vinu., Rayford Vaughn. (2003). Application of lightweight formal methods in requirement engineering. CrossTALK-The Journal of Defense Software Engineering (Jan 2003) (2003).

[4] Ghose, Aditya K. (2000). Formal tools for managing inconsistency and change in RE. *In*: Proceedings of the 10th International Workshop on Software Specification and Design. IEEE Computer Society, 2000.

[5] Börger, Egon, (2005). A high-level modular definition of the semantics of Co &.*Theoretical Computer Science* 336.2 (2005): 235-284.

[6] Abrial, Jean-Raymond.,Börger, Egon., Langmaack, Hans(1996). Formal methods for industrial Applications: Specifying and programming the steam boiler control. V. 9. Springer Science & Business Media, 1996.

[7] Jones, C. B. (1990). Systematic Software using VDM. (1990).

[8] Meyer, Bertrand. (1992). Applying'design by contract' *Computer* 25.10. 40-51.

[9] Neumann, Peter G. (1975). Toward a methodology for designing large systems and verifying their properties. Gl-4. Jahrestagung. Springer Berlin Heidelberg, 1975. 52-67.

[10] Harel, David. Dynamic logic. Springer Netherlands, 1984.

[11] Abrial, Jean-Raymond. (1996). Extending B without changing it (for developing distributed systems). 1st Conference on the B method. Vol. 11. 1996.

[12] Berry, Gérard. (2008). Synchronous design and verification of critical embedded systems using SCADE and Esterel, Lecture Notes in Computer Science 4916. 2-2.

[13] André, Pascal, et al. (2000). Checking the Consistency of UML Class Diagrams Using Larch Prover, Rigorous Object-Oriented Methods. 2000.

[14] Lutz, Robyn R. (1993). Analyzing software requirements errors in safety-critical, embedded systems, Requirements Engineering, 1993 Proceedings of IEEE International Symposium on. IEEE, 1993.

[15] Sommerville, Ian., Sawyer, Pete (1997). Requirements engineering: a good practice guide. John Wiley & Sons, Inc., 1997.

[16] Woodcock, Jim, (2009). Formal methods: Practice and experience." ACM Computing Surveys (CSUR) 41.4 (2009): 19.

[17] Mens, Tom., Tourwé, Tom (2004). A survey of software refactoring. *Software Engineering, IEEE Transactions on* 30.2 (2004): 126-139.

[18] Hoare, Charles., Richard, Antony (2002). Assertions: A personal perspective, Software pioneers. Springer Berlin Heidelberg, 356-366.