Journal of Software Engineering and Applications, 2014, 7, 299-310
Published Online April 2014 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2014.74029

Scientific Research

Integrating Formal Methods in XP—A Conceptual Solution

Shagufta Shafiq, Nasir Mehmood Minhas

UIIT-PMAS Arid Agriculture University, Rawalpindi, Pakistan
Email: shaguftashafiq786@yahoo.com, nasirminhas@uaar.edu.pk

Received 9 February 2014; revised 8 March 2014; accepted 15 March 2014

Copyright © 2014 by authors and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY).
http://creativecommons.org/licenses/by/4.0/
Open Access

Abstract

Formal methods can be used at any stage of product development process to improve the software quality and efficiency using mathematical models for analysis and verification. From last decade, researchers and practitioners are trying to establish successful transfer of practices of formal methods into industrial process development. In the last couple of years, numerous analysis approaches and formal methods have been applied in different settings to improve software quality. In today's highly competitive software development industry, companies are striving to deliver fast with low cost and improve quality solutions and agile methodologies have proved their efficiency in acquiring these. Here, we will present an integration of formal methods, specifications and verification practices in the most renowned process development methodology of agile *i.e.* extreme programming with a conceptual solution. That leads towards the development of a complete formalized XP process in future. This will help the practitioners to understand the effectiveness of formal methods using in agile methods that can be helpful in utilizing the benefits of formal methods in industry.

Keywords

Formal Methods, Specification, Verification, Agile, Extreme Programming

1. Introduction

Formal methods have proved as a powerful technique to ensure the correctness of software. The growth in their use has been slow but steady and FMs are typically applied in safety critical systems. Use of formal methods requires both expertise and efforts, but this is rewarded if they are applied wisely. It must be seen as good prudently news that Microsoft products increasingly use formal methods in key parts of their software development, particularly checking the interoperability of third-party software with Windows. We believe formal methods are here to stay and will gain further traction in the future [1].

How to cite this paper: Shafiq, S. and Minhas, N.M. (2014) Integrating Formal Methods in XP—A Conceptual Solution. *Journal of Software Engineering and Applications*, **7**, 299-310. http://dx.doi.org/10.4236/jsea.2014.74029

Formal methods are used for developing software/hardware systems by employing mathematical analysis and verification techniques and often supported by the tools [2]. Mathematical model's steadiness enables developers to analyse and verify these models in any phase of the development process *i.e.*, requirements engineering, specification, design and architecture, implementation, testing and maintenance [2].

Since their inception and use in the domain of real-time, critical systems, now these methods are finding their way to other widens area of industrial applications especially developing high quality software products [2].

Traditional software development process can be categorised into three phases: 1) requirement gathering. Sometime, specifications are also incorporating in requirement to get more precise and accurate requirements; 2) phase of design, modelling and implementation; 3) the late phase involves verification and validation process activities.

It can be suggest that formal methods can be effectively used in traditional software development process to get accurate system specifications using the formal specification methods like ASM, B, Z and VDM even these can be effectively used for representation and management of complex system specifications. Formal methods can also be used in software system design by defining formal models to refine the data, abstract function to represent system functionality [2] and to implement. Formal methods can be used for automated code generation and verification from formal models [2].

Formal methods are always perceived as highly mathematical based processes and can only be used by mathematicians and specialist software experts. This inclination leads towards the limited usage in industry-based software development processes. To change this misconception, a much wider industrial research has to be performed to get the true benefits of formal methods in industry [3].

In today's fast growing software industries, software industries make every effort to produce fast delivery, with better quality and low cost software solutions [2]. With Lightweight iterative approach with the focus on communication between client and developing team, family of agile methods has turned out as solution to achieve all these goals. Agile methods have a wide range of approaches from development process methods like extreme programming to complete project management process like scrum [4]. These methods have been effectively used in the software industry to develop systems on time and within budget with improved software quality and customer satisfaction [3].

A main reason of not using agile approaches for the development of safety critical systems is the lack of more formal evaluation techniques in agile methods where as safety critical systems require more rigorous development and evaluation techniques to ensure quality products [3].

As agile approaches less focus on documentation over processes with informal techniques which are often insufficient in determining the quality of safety critical systems [3], agile methods are still not effectively used to create systems which require more formal development and testing techniques for development [3].

It has been observed in literature that combination of agile and formal methods can bring best features of both the worlds [5] which can lead towards a better software development solution. In [6], authors present an evaluation of agile manifesto and agile development principles to show that how formal and agile approaches can be integrated and identify the challenges and issues in doing so. In [3], authors suggest that agile software development can used light weight formal analysis techniques effectively to bring potential difference in creating system, with formally verified techniques, on time and within budget.

Motivation

It has been observed through literature that application of formal techniques in early phases of software development improves the quality of software artefacts and as a result ensure precise and error free requirement details to the later phases of the development process. As a result the overall cost of a software project is significantly lower because of the minimized error rate. After that formal specifications transformed into concrete models to verify its consistency with the specification that lead towards the implementation.

Till date formal methods couldn't be effectively used in industry based product engineering but it has potential of widespread effectiveness for application development in different domains, whereas agile approaches lack precise techniques for planning and evaluation. A combination of formal methods and agile development processes can significantly encourage the use of formal techniques in industry base software development solutions [3].

Here in this article, in Section 2 we first describe the use of formal specification and verification techniques

with frequently used formal specification languages. We then present an overview of extreme programming in Section 3. Section 4 contains the related research work which shows the integration of formal methods with traditional software development and agile methodologies to support our main concept and the reason of choosing agile process method for our proposed approach. Section V describes our proposed approach.

2. Formal Methods in Practice

Formal methods can be generally categorized into two basic techniques and practices *i.e.* formal specifications and verification [7].

Formal Specifications can be described as the technique that uses a set of notations derived from formal logic to explicitly specify the requirements that the system is to achieve with the design to accomplish those requirements and also the context of the stated requirements with assumptions and constraints to specify system functions and desired behaviour explicitly [7].

In design specifications, a set of hierarchical specifications with a high-level abstract representation of the system to detailed implementation specifications are designed, **Figure 1** shows that hierarchy of specification levels [7].

Formal Verification is the use of verification methods from formal logic to examine the specifications for required consistency and completeness to ensure that the design will satisfy the requirements, assumptions and constraints that system required [7].

There are several techniques available for formal specifications with automated tool support. These automated tools can perform rigorous verification that can be a tedious step in formal methods [7].

There are many different types of formal methods techniques used in different settings; following are the most commonly used examples of formal specifications, *i.e.* VDM, B and Z [3].

2.1. VDM

VDM stands for "The Vienna Development Method" and consider as one of the oldest formal methods. VDM is a collection of practices for the formal specification and computational development [8]. It consists of a specification language called **VDM-SL**. Specifications in VDM-SL based on the mathematical models develop through simple data types like sets, lists and mappings, and the operations, causes the state change in the model [8].

2.2. B-Methods

B-method is another formal specification method consists of abstract notations and uses set theory for system modelling, and mathematical proof for consistency verification between the different refinement phases [7].

2.3. Z

Another most commonly used formal specification language for critical system development, using mathematical notations and schemas to provide exact descriptions of a system. System is described in a number of small Z modules called schemas which can cross refer each other as well as per system required. Each module is expected to have some descriptive informal language text to help users to understand it.

The selection of formal specification language made on the basis of developer's past experience with the selected method or the suitability of any model with respect to the system under develop and its application domain [3].

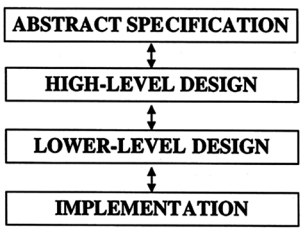

Figure 1. Hierarchy of formal specifications [7].

3. Extreme Programming an Agile Approach

An agile development methodology extreme programming can be define as light weight iterative approach for small and medium size development teams having incomplete or continuously changing requirements. XP works in small iterations with simple practices which focus on close collaboration, simple design to produce high quality products with continuous testing.

Extreme programming created by K. Beck in 1990's, is a set of twelve key practices [9] applied with four core values including communication, simplicity, courage and feedback.

Extreme programming [9] provides a complete solution for product development process and widely accepted for development of industry based and as well as object oriented software systems [10]. With the principles of agile methodology XP proves as novel approach in the family of agile methods that significantly increase productivity that produce high quality error free code [10]-[12]. **Figure 2** shows the complete XP development process in traditional settings.

Extreme programming is a Test driven approach. In TDD each user story converted to a test and at the time of release code developed during iteration will be verified though these pre develop tests. This regression testing technique provides a better test coverage at every phase of software development which involves writing of unit tests for each individual task in the system domain [11].

4. Agile Approaches towards Formal Methods

Many efforts have been made to integrate the rapidly growing agile practices, having wide industrial acceptance for producing quality products, with the formal methods having limited industrial acceptance but with strong background, distinctive features and benefits. **Table 1** shows the categorization of formal and agile methods.

Richard Kemmerer has made first attempt to investigate the integration approach for conventional development process using formal methods [15]. Kemmerer's work was related to the addition of formal techniques into the different stages of conventional waterfall model, our work is a step towards the integration of formal specification and verification within the agile software development process *i.e.* extreme programming.

Another study [16] proposed an integration approach for agile formal development approach with the name of XFun, proposed integration of formal notation using X-machine within the unified process.

In [17] author suggests an agile approach, combines the agile practice of tests driven development with formal approach design by contract within XP [17].

There is another study [18] that proposes the integration of formal method techniques into the traditional V-model for refinement of critical aspects of system [18]. The V-model representing the structure of activities providing guideline for software engineers to follow during the process development, while our study focuses on suggesting a complete solution as software development methodology which can be used by the system developers effectively.

In another study [19] authors have made an effort to develop a light weight approach using formal methods with the industry development standards, SOFL [19]. They have used a more graphical notation instead of pure mathematical syntax to define the high level architecture of the system. Later on author refine his proposed approach by developing agile based SOFL method [20].

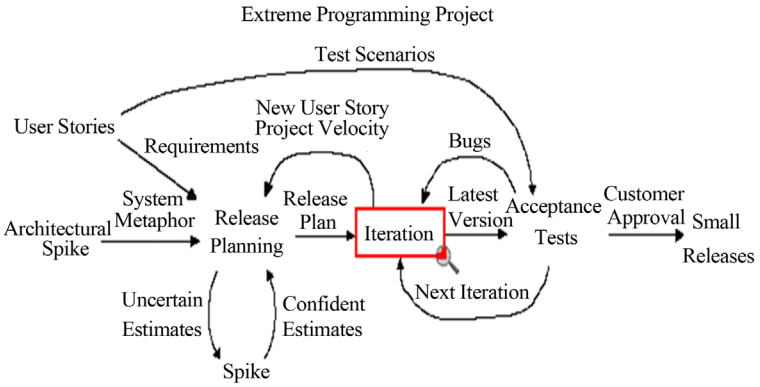

Figure 2. Extreme programming development process. [11]

Table 1. Agile and formal methods.

Characterizations of Formal and Agile Methods	
Agile Methods	**Formal Methods**
Validation	Verification
Pleasantness	Correctness
Refactoring	Refinement
Concrete	Abstract
Particular	General
Tests	Roofs
Design evolve with code	upfront design
Cowboy coding	Analysis paralysis
Team	Programmer
Beck [9] [10]	**Dijkstra** [13] [14]

In [21], authors proposed an extreme Formal Modeling (XFM) (agile formal methodology) to design the specifications from an informal description into a more formal language uses extreme programming approach.

Recently [3] presented an integration approach of formal specification and agile. Suggest a theoretical agile approach using scrum methodology and integrating formal specifications for safety-critical systems. In the proposed method formal specifications are applied within iteration phase and having a developing team that consists of both conventional as well as formal modelling engineers [3].

Most industrially accepted agile methods *i.e. extreme programming* [9] and *scrum* [22] have been used as emergent trends in dealing with core challenges and issues in software development process [23] such as: increase time, with low quality and increase cost at delivery time. [24]. although, it has been observed that agile software development practices are also effectively applied in different development settings for safety critical systems as well [25] [26]. In [26] author argued that Plan driven approaches are better suited for these types of systems. Whereas further studies suggested that the integration of agile approaches with the company's existing plan driven software development activities can be more effective and beneficial for producing safety critical systems [26]-[28]. Another study [29] suggests that integration of agile and CMMI can produce significant difference in developing quality softwares systems.

5. Formal Methods in XP: A Conceptual Solution

Formal methods are set of practices for specification and verification but are not constrained with any specific software development methodology. Mostly published reports focusing on improving the formal techniques for different domain and application development and lacks a complete methodology that can be followed for developing object oriented systems more effectively. Here in this account of literature we are suggesting a conceptual solution for software development industry with the integration of formal techniques into the extreme programming development methodology. Through this, companies will be able to get benefited aspects of both the integrated domains to develop high quality software systems.

Figure 3 shows our proposed approach for development process of XP with the integration of formal methods. Here we have suggesting a conceptual solution.

5.1. User Stories

User stories in XP serves as the basis for defining the functions of the system needs to perform, and to facilitate requirements managements described in an informal manner. In safety circle system use of formal specification techniques consider as the primary intention so the generated requirements can be error free hence using user stories in xp available in informal way needs to be describing through the formal specification techniques to make them more accurate and precise. And serve as the input for the release planning including system metaphor.

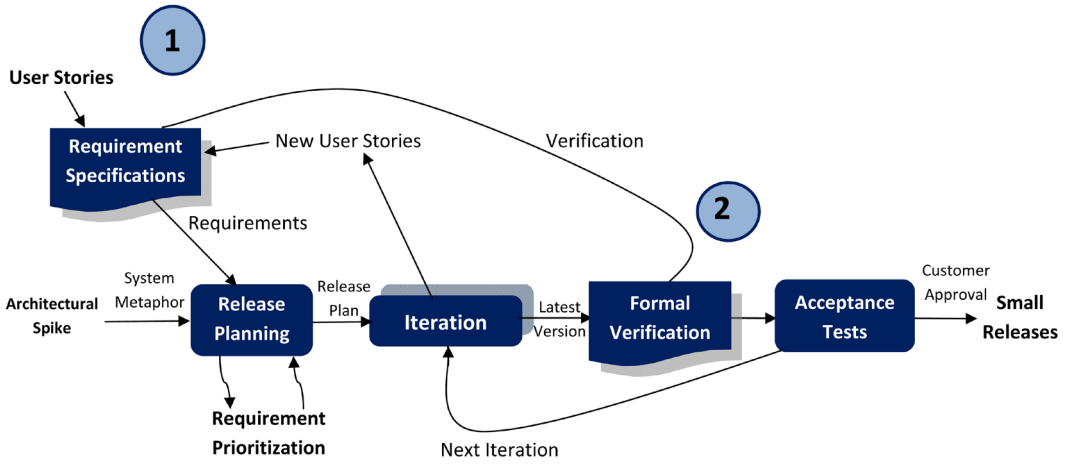

Figure 3. Proposed approach.

5.2. Requirement Specification

Formal specification is the description of a program's properties defined in alogic oriented mathematical language. Formal specification is not directly executable but capable of representing higher level of abstraction than a conventional programming language of the system. Here the focus of the formal specification tasks is generating abstract models from user stories and extracting requirements specification for development as well as for validation before the implementation in forthcoming development phases. **Figure 4** explains the process of proposed approach for requirement specification.

5.3. Release Planning

In our proposed approach, requirements will be extracted from the described formal specification in the earlier phase and then the requirement prioritization will be done through the spike. Once the requirement specification are generated, will be forwarded to release planning phase in which on the basis of each requirement programmers estimate their resources and efforts as per customer needs. At the end of the release planning phase a release plan will be developed for the forth coming iteration. **Figure 5** shows the inputs and outputs for the release planning phase.

5.4. Iteration

During release plan, iteration plan has been forwarded for each iteration phase of 2-4 weeks as per plan. This phase followed through the developing system's functionality incrementally with increasing complexity of the system model. Refactoring and pair programming are the core activities of development iteration in XP process representing described in **Figure 6**.

During iteration daily stand up meetings and close customer collaboration serve as a source to measure development progress are essential in XP. In addition pair programming and continuous integration supports frequent and automated integration of tests and ensure knowledge sharing for system consistency between the formal specification and the final implementation.

Figure 7 shows the development process with the integration of formal verification phase. In XP, TDD developers are required to write automated unit tests before the code is implemented this can be done by developing formal specifications defines at the earlier stage. And formal verification can be performed easily from the requirement specifications using automated code driven tests. This can be cost and time effective activity.

5.5. Continuous Integration

Another very effective practice for producing high quality software in *extreme programming* is continuous integration in which teams keep the system fully integrated after every coded functionality with passed acceptance test. Once the code has been verified from the formal verification techniques, new unit coded functionality inte-

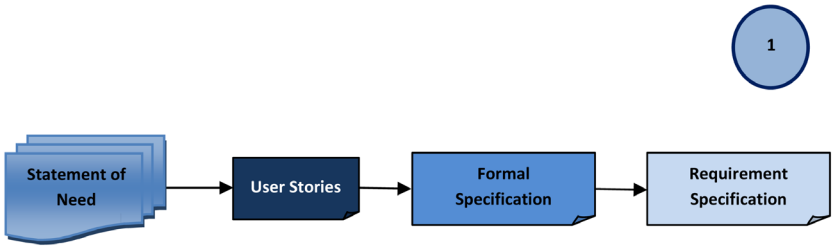

Figure 4. Formal specification in proposed approach.

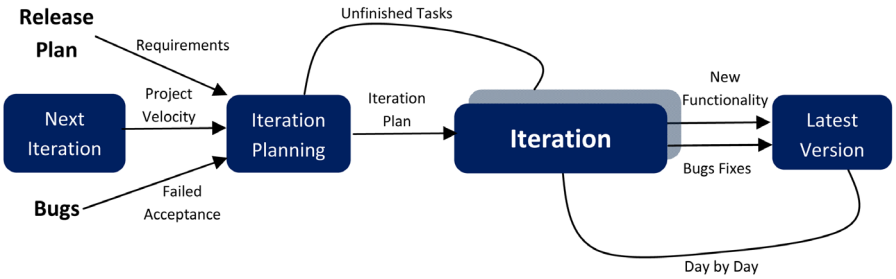

Figure 5. Planning game.

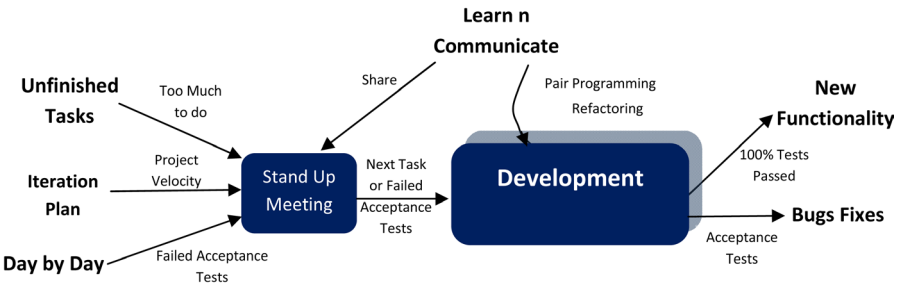

Figure 6. Iteration.

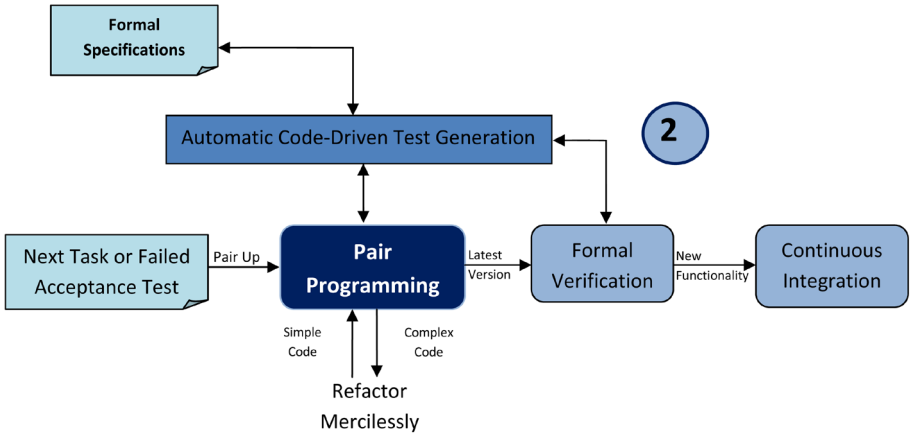

Figure 7. Formal verification.

grated into the system to increase the system quality and efficiency that reduces the system integration issues as well.

6. Evaluation of Proposed Solution

To get practical support for our proposed methodology we have conducted a control experiment. To conduct the

experiment we selected two groups of undergrad students having good understanding of XP with enough programming skills. Each group was comprised of five members; group-II has added knowledge of VDM and Z specifications as well. We have given a project titled police reporting system to both the groups, Group-I used the traditional XP, while Group-II followed proposed methodology for the system development. Groups were under continuous monitoring to get results with respect to time of system development phase, error rate and product quality. System details are eliminated here just for the sack of prise content and focusing only on the results.

Figure 8 represent the duration in days with the SDLC phases, because XP is iterative methodology and focuses more on development and implementation in contrast formal XP takes more time in planning and designing. Here we have presented cumulative time in days for each phase and implementation phase include development, testing and integration. Use of formal XP took initially longer time but reduces overall development time as compare to traditional XP that lead towards the higher productivity as result shows in **Figure 9**.

Following **Figure 10** present the number of unit functionalities developed in each iteration.

Product quality evaluated on the basis of number of passed acceptance tests after each iteration in **Figure 11** shows each iteration results.

Error rate evaluated during each unit development phase **Figure 12**.

7. Discussion and Conclusions

The work presented here is with the objective of devising a complete development method for the application of formal specification and verification within an agile system development approach *i.e.* extreme programming.

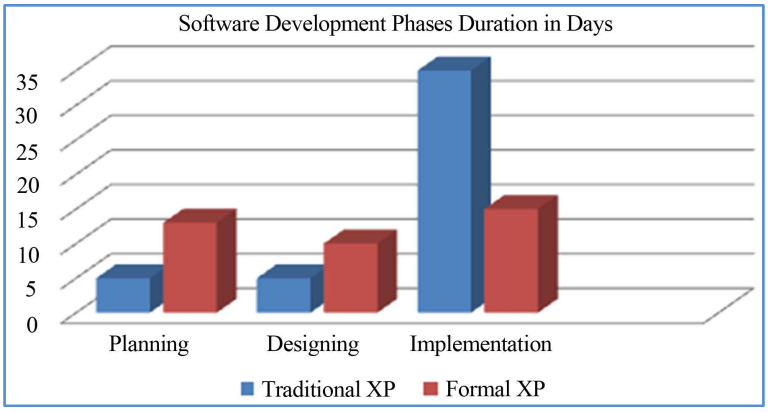

Figure 8. Cumulative number of days for each SDLC phase.

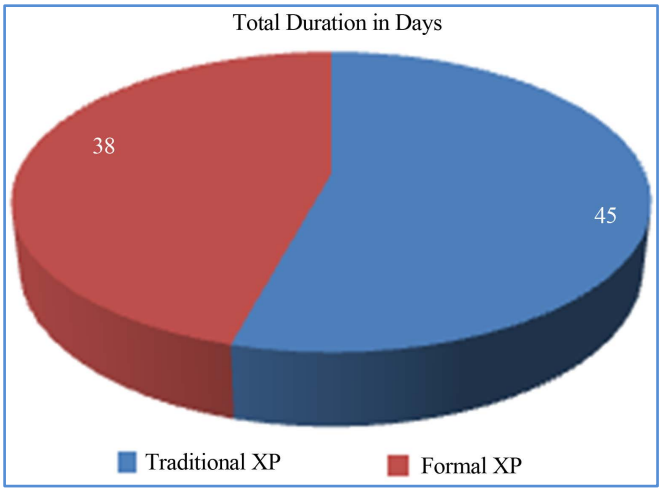

Figure 9. Project duration in days.

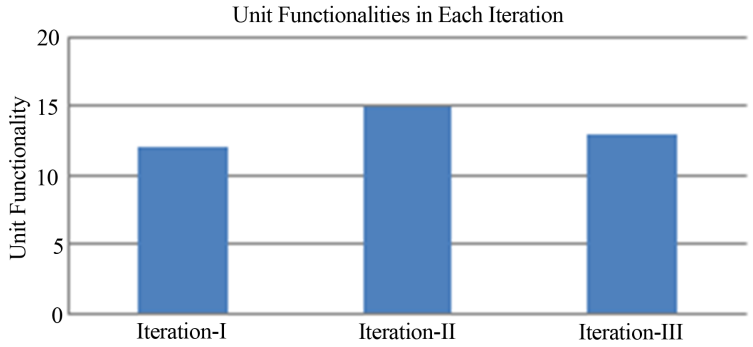

Figure 10. Number of unit functionalities developed during each iteration.

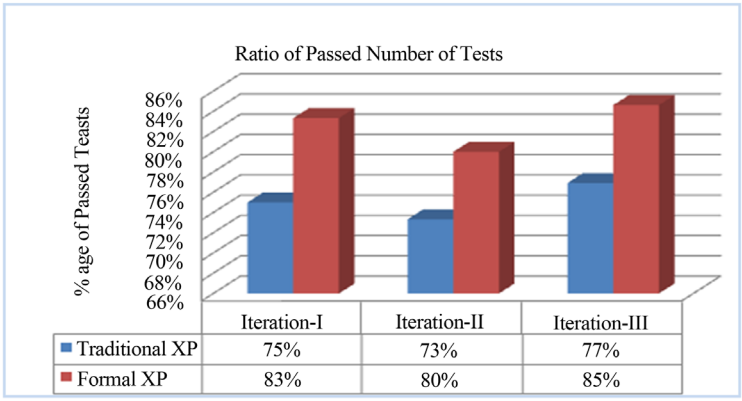

Figure 11. Number of passed acceptance tests after each iteration.

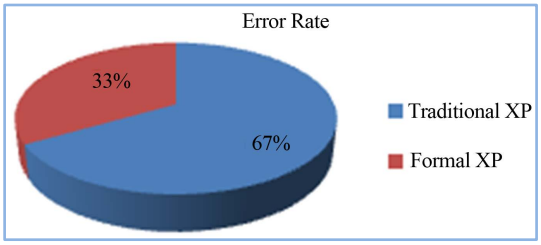

Figure 12. Error rate during development process.

Many literary and industrial based evidences show the effectiveness of XP in the traditional software development and the literary studies also report some evidences of the successful integration of XP practices in different domains for the system specification and verification, but it lacks a complete development process. In our proposed approach, we have suggested a complete process development for the extreme programming with the formal specification, formal verification techniques and the limited level validation process which supports our notion that formal XP can lead to a higher quality product with reduced error rate and improved time efficiency. **Table 2** represents the literature support for the proposed work.

Application of formal methods is believed as it improves system reliability at the cost of lower productivity whereas XP focuses on more productivity, So, in principle, using process development activities of FM and XP can improve its efficiency like pair programming, daily code development and integration, the simple design or metaphor and iterative development process. On the other hand, one criticism to XP is that it lacks formal or even semi-formal practices. So here in this paper we have tried to devise a XP process utilizing the formal method techniques and the result shows that the appropriate combination results in a more efficient and higher quality development method because each can be able to minimize others' issues.

Informal specification can have ambiguity and irrelevant details and self-contradictory and incomplete ab-

Table 2. Present the use of XP practices with FM to improve software quality.

Conceptual Model's Validation Support			
STUDY ID	**TITLE**	**YEAR**	**SUPPORTING CENCEPT**
STD-1 [30]	Formal Agility. How much of each?	2003	Studied XP practices from the prism of FM to show that how some XP practices can admit the integration of Formal Methods.
STD-2 [31]	Using a formal method to model software design in XP projects	2005	Successfully introduces X-Machine in XP for a succinct and accurate software system
STD-3 [32]	Applying XP Ideas Formally: The Story Card and Extreme X-Machines	2003	Present an approach of using XP story cards and transform those into formal specifications through X-Machine to produce high quality software products.
STD-4 [3]	Scrum Goes Formal: Agile Methods for Safety-Critical Systems	2012	Suggest that XP practices can successfully support the formal method and techniques
STD-5 [33]	Agile Specification Driven Development	2004	Present an approach of using TDD practice for specification driven development that leads towards quality software development.
STD-6 [34]	On the Use of XP in the Development of Safety-Oriented Hypermedia Systems	2003	Uses XP practices in the development of safety-oriented hypermedia systems with formal methods for exhaustive testing
STD-7 [35]	Formal Methods and Extreme Programming?	2003	Evaluated how formal methods overcome the lack of upfront specification and design practices in XP
STD-8 [36]	20 Years of Teaching and 7 Years of Research: Research When You Teach	2008	results from multiple experiments found that there was a measurable quality premium in using XP and uses extreme x-machines for producing high quality products
STD-9 [5]	Formal versus agile: Survival of the fittest?	2009	Suggest that XP practices can get benefit from formal methods
STD-10 [37]	Formal Extreme (and Extremely Formal) Programming	2003	Analyse how Formal Methods (FM) can interact with agile process XP, and suggest that XP practices can improved using FM. can

stractions which cannot be handled easily in traditional XP. By defining the requirement specification through the process of formal specification, these issues can be effectively minimized.

The role of manager in XP is to synchronize and manages the work of all team members, with the application of the formal specification and verification. It is required that all managers, trackers and coaches have the implementation knowledge of formal models and their synchronization in the software development process. To make this possible, developer's focus should be on the improvement of the formal specification technique which is easier to be read and understood by the people who don't have the strong mathematical background like the graphical notations used in SOFL or the more familiar C-like syntax for VDM.

The process of formal verification in our proposed approach can be successfully used in minimizing the manual unit tests and regression testing process in traditional XP and reduces the programmer's efforts of continuous testing with efficient time utilization. As suggested in the solution, formal requirement specifications at first step can be easily transformed into automated code driven test generation which leads towards the error free code generation of requirements. There are also many tools available for the system verification developed through formal specifications.

The method suggested in this paper can provide effective guidelines for companies looking for an effective development methodology for formal methods and applying formal specification and/or verification techniques for software development.

8. Limitations and Future Work

Here we have presented a theoretical model with a very limited evaluation process. But for the industrial applications, it should be verified from the industry. In future, we will try to develop complete specification process that includes how the user stories will be transformed into requirement specifications. In addition to the evaluation of the proposed conceptual solution, several things are needed in order to ensure higher acceptance of formal methods with industry and industrial practices.

References

[1] Boca, P., Bowen, J.P. and Siddiqi, J.I. (2010) Formal Methods: State of the Art and New Directions. Springer-Verlag London Limited, Berlin. http://dx.doi.org/10.1007/978-1-84882-736-3

[2] Woodcock, J., Larsen, P.G., Bicarregui, J. and Fitzgerald, J. (2009) Formal Methods: Practice and Experience. *ACM Computing Surveys*, **41**, 1-36. http://dx.doi.org/10.1145/1592434.1592436

[3] Wolff, S. (2012) Scrum Goes Formal: Agile Methods for Safety-Critical System. 2012 Formal Methods in Software Engineering: Rigorous and Agile Approaches (FormSERA), Zurich, 2 June 2012, 23-29. http://dx.doi.org/10.1109/MC.2009.284

[4] Schwaber, K. (2004) Agile Project Management with Scrum. Prentice Hall, Upper Saddle River.

[5] Black, S., Boca, P.P., Bowen, J.P., Gorman, J. and Hinchey, M. (2009) Formal versus Agile: Survival of the Fittest? *IEEE Computer*, **42**, 37-45.

[6] Larsen, P.G., Fitzgerald, J. and Wolff, S. (2010) Are Formal Methods Ready for Agility? A Reality Check. 2*nd International Workshop on Formal Methods and Agile Methods*, Pisa, 17 September 2010, 13 Pages.

[7] Johnson, S.C. and Butler, R.W. (2001) Formal Methods. CRC Press LLC, Boca Raton.

[8] Grunerand, S. and Rumpe, B. (2010) GI-Edition. Lecture Notes in Informatics. 2*nd International Workshop on Formal Methods and Agile Methods*, Vol. 179, 13-25.

[9] Beck, K. (1999) Extreme Programming Explained. Addison-Wesley, Boston.

[10] Beck, K. (2003) Test-Driven Development. Addison-Wesley, Boston.

[11] (2013) Extreme Programming: A Gentle Introduction. http://www.extremeprogramming.org/.

[12] Wood, W.A. and Kleb, W.L. (2003 Exploring XP for Scientific Research. *IEEE Software*, **20**, 30-36.

[13] Dijkstra, E.W. (1972) Notes on Structured Programming, Structured Programming. In: Dahl, O.-J., Hoare, C.A.R. and Dijkstra, E.W., Eds., *Structured Programming*, Academic Press, London, 1-82.

[14] Dijkstra, E.W. (1968) A Constructive Approach to the Problem of Program Correctness. *BIT Numerical Mathematics*, **8**, 174-186. http://dx.doi.org/10.1007/BF01933419

[15] Kemmerer, R.A. (1990) Integrating Formal Methods into the Development Process. *IEEE Software*, **7**, 37-50. http://dx.doi.org/10.1109/52.57891

[16] Eleftherakis, G. and Cowling, A.J. (2003) An Agile Formal Development Methodology. *Proceedings of* 1*st South-East European Workshop on Formal Methods*, *SEEFM*'03, Thessaloniki, 20 November 2003, 36-47.

[17] Ostroff, J.S., Makalsky, D. and Paige, R.F. (2004) Agile Specification-Driven Development. *Lecture Notes in Computer Science*, **3092**, 104-112.

[18] Broy, M. and Slotosch, O. (1998) Enriching the Software Development Process by Formal Methods. *Lecture Notes in Computer Science*, **1641**, 44-61.

[19] Liu, S. and Sun, Y. (1995) Structured Methodology + Object-Oriented Methodology + Formal Methods: Methodology of SOFL. *Proceedings of First IEEE International Conference on Engineering of Complex Computer Systems*, Ft. Landerdale, 6-10 November 1995, 137-144.

[20] Liu, S. (2009) An Approach to Applying SOFL for Agile Process and Its Application in Developing a Test Support Tool. *Innovations in Systems and Software Engineering*, **6**, 137-143. http://dx.doi.org/10.1007/s11334-009-0114-3

[21] Suhaib, S.M., Mathaikutty, D.A., Shukla, S.K. and Berner, D. (2005) XFM: An Incremental Methodology for Developing Formal Models. *ACM Transactions on Design Automation of Electronic Systems*, **10**, 589-609. http://dx.doi.org/10.1145/1109118.1109120

[22] Schwaber, K. and Beedle, M. (2002) Agile Software Development with Scrum. Prentice-Hall, Upper Saddle River.

[23] Karlström, D. (2002) Introducing Extreme Programming—An Experience Report. *Proceedings* 3*rd International Conference on Extreme Programming and Agile Processes in Software Engineering*, Alghero.

[24] Holström, H., Fixgerald, B., Agerfalk, P.J. and Conchuir, E.O. (2006) Agile Practices Reduce Distance in Global Software Development. *Information and Systems Management*, **23**, 7-18. http://dx.doi.org/10.1201/1078.10580530/46108.23.3.20060601/93703.2

[25] ISO TC 22 SC3 WG16 Functional Safety, Convenor Ch. Jung. Introduction in ISO WD26262 (EUROFORM-Seminar, April 2007).

[26] Drobka, J., Noftzd, D. and Raghu, R. (2004) Piloting XP on Four Mission Critical Projects. *IEEE Software*, **21**, 70-75. http://dx.doi.org/10.1109/MS.2004.47

[27] Wils, A., Baelen, S., Holvoet, T. and De Vlamincs, K. (2006) Agility in the Avionics Software World. 7*th Internation-*

al Conference, XP 2006, Oulu, 17-22 June 2006, 123-132.

[28] Boehm, B. and Turner, R. (2003) Balancing Agility and Discipline. Addison Wesley, Boston.

[29] Pikkarainen, M. and Mäntyniemi, A. (2006) An Approach for Using CMMI in Agile Software Development Assessments: Experiences of Three Case Studies. 6*th International SPICE Conference*, Luxembourg, 4-5 May 2006, 1-11.

[30] Herranz, Á. and Moreno-Navarro, J.J. (2003) Formal Agility, How Much of Each? Taller de Metodologías Ágiles en el Desar-Rollo del Software, VIII Jornadas de Ingeniería del Software Bases de Datos (JISBD 2003), Grupo ISSI, 47-51.

[31] Thomson, C. and Holcombe, M. (2005) Using a Formal Method to Model Software Design in XP Projects. *Annals of Mathematics, Computing and Tele-Informatics*, **1**, 44-53.

[32] Thomson, C. and Holcombe, W. (2003) Applying XP Ideas Formally: The Story Card and Extreme X-Machines. 1*st South-East European Workshop on Formal Methods*, Thessaloniki, 21-23 November 2003, 57-71.

[33] Ostroff, J.S., Makalsky, D. and Paige, R.F. (2004) Agile Specification-Driven Development. *Lecture Notes in Computer Science*, **3092**, 104-112.

[34] Canos, J., Jaen, J., Carsi, J. and Penades, M. (2003) On the Use of XP in the Development of Safety-Oriented Hypermedia Systems. *Proceedings of XP* 2003, Genova, 25-29 May 2003, 201-203.

[35] Baumeister, H. (2002) Formal Methods and Extreme Programming. *Proceedings of Workshop on Evolutionary Formal Software Development*, in Conjunction with FME, Copenhagen, 189-193, 1-2.

[36] Holcombe, M. and Thomson, C. (2007) 20 Years of Teaching and 7 Years of Research: Research When You Teach. *Proceedings of the* 3*rd South-East European Workshop on Formal Methods*, Thessaloniki, 30 November-1 December 2007, 1-13.

[37] Herranz, A. and Moreno-Navarro, J.J. (2003) Formal Extreme (and Extremely Formal) Programming. In: Marchesi, M. and Succi, G., Eds., 4*th International Conference on Extreme Programming and Agile Processes in Software Engineering, XP* 2003, LNCS, No. 2675, Genova, 88-96.