International Journal of Natural and Engineering Sciences 8 (3): 35-42, 2014
ISSN: 1307-1149, E-ISSN: 2146-0086, www.nobel.gen.tr

Mapping Formal Methods to Extreme Programming (XP) –A Futuristic Approach

Tasmia SAEED[1*] Syed Shah MUHAMMAD[1] Muhammad Abuzar FAHIEM[1]
Sarfaraz AHMAD[2] Muhammad Tariq PERVEZ[1] Abdul Basit DOGAR[1]

[1] Department of Computer Sciences, Virtual University of Pakistan, Lahore, Pakistan
[2] Departement of Computer Science, Faculty of Engineering and Technology, Lahore College for Women University, Lahore, Pakistan

*Corresponding author:
Email: tasmiach@live.com

Received: September 16, 2014
Accepted: October 30, 2014

Abstract

The Agile methodologies have been gaining popularity for last few years. This approach satisfies the variety of customer needs in a better way and requirements as compared to the traditional software development methodologies. It also accepts changes in requirements conveniently. Agile Extreme Programming (XP) is frequently used as it provides an effective project engineering ability. However, there are some drawbacks of XP. Some projects can suffer from the lack of documentation. Sometimes it poses risks for life critical systems and a general de-emphasis on architecture may be observed. In the practical field, there is a need to improve XP while developing safety critical systems. For such situations, a futuristic approach is introduced in this research work by combining Formal Methods (FMs) with XP. It is a promising contract to combine both these techniques. However if combined in a fruitful way they can give the best of both XP and FMs. By using this approach XP can be used for the safety critical systems without any risk and at the same time cost is much reduced. In this paper a Formal Extreme Programming (FXP) model is introduced. In this model "Software Cost Reduction" (SCR) is applied at the initial stage to specify the requirements formally. A high level FM, Algebraic Specification will be written before coding. Algebraic specification gives formal description of requirements in a mathematical way. The Design by Contract (DbC) is mapped conditionally at the testing phase for complex systems. A survey through questionnaire is conducted by professionals of software industry.

Keyword: Extreme Programming (XP), Formal Methods, Formal Extreme Programming (FXP) Model, Software Cost Reduction (SCR), Design by Contract (DbC).

INTRODUCTION

Development Models e.g. Water Fall, Fountain, and Spiral etc., were introduced for software development. Later on some of those traditional models were revised e.g., Revised Waterfall Model, primarily to improve the value & quality of developed software. During development those traditional models do not support the changing user requirements. To cope with this issue Iterative and Incremental Agile models were introduced. In 1999, Beck introduced a light weight, flexible and predictable methodology, Extreme Programming (XP) that reduces the cost of change. The Agile models were popular enough after the publication of research paper "Extreme Programming Explained," [1].

We find that XP gives better results as compared to other agile models because of its four values / principles i.e., "communication" with customer, "simplicity" of design, getting "feedback" to solve occupational hazards and "courage" with other three values, [2]. An ideal XP project passes through following six phases for the production of a successful XP product, [3].

Exploration

At this phase the customer gives his requirements in the form of story cards, and then a priority is assigned to each story. Later on the XP team will spend about one to two weeks to design system architecture. If we have no idea about how to implement a user story then architectural experiments can be used to let it easy for us. The programmers will estimate each task in this phase and when they have done with a task, they will mention time required to implement a task in the calendar.

Planning

At this phase team size (number of members required for development), code ownership (who will be allowed to change the code?), working hours, sitting arrangements, schedule for development and programming pairs are planned.

Iteration to release

In this phase the planned schedule is further broken into iterations. Iteration is a set of functions required to be developed. The pair of programmers takes an iteration plan and start the development process, by developing the iteration with highest priority first. The functional test cases are also produced for stories in the same iteration. The time period of iteration should be smaller than 6 months because the more the time you take for iteration the more risky it will be. At the end of iteration let the customer check it and sign the story cards of the completed stories.

Productionizing

When iteration is developed it is released for production. At this phase it is executed to record feedback. At this stage it is also evolved that what are the changes required in this release, for this purpose one should have enough knowledge about the design to justify it. If developers are unable to justify some of the ideas then they will make there a list about their position and decide when the process goes in production. XP suggest daily standup meeting for better production, in this way everybody get a chance to know what others are doing on the iterations.

Maintenance

This phase deals with the changes in the iteration in development, production and sometimes dead project, either by adding new functionalities or by changing existing functionalities. Maintenance phase also deals with the change in the development team, like changing the position of the programmers, managing help desk, and adjusting new programmers in team. It is challenging to program in this phase as compared to the development phase, so when new members are required to program in this phase, for first 2 to 3 iterations let them paired with an experienced software developer.

Death

When the development and testing are completed and the user has no new story and happy with your work, then it's the time to let the system "dead". At this stage we develop a document to describe a tour of the system.

XP gain acceptance from the software industry since 1999, for small projects and rarely for larger projects. Paulk [4] in his study about XP doesn't support the use of XP for life critical project, as well as safety critical systems, because XP can suffer from the lack of documentation, risky for life critical systems and de-emphasis on architecture. Considering the advantages of XP there seems a need to improve the XP model to deal with these issues such as how the XP model is to be improved to let it work perfectly for life critical and safety critical systems? How to reduce the risk of failure of Safety Critical Systems? We know Formal Methods (FMs) can be a better solution to these problems. The FMs are optimal to be used in the Safety Critical System even though some software developers consider it tough and costly [5].

The FMs are the generic methods to verify, describe and develop a system based on mathematics. These methods are used to describe the requirements, design and test in a mathematical way. So there is a need to integrate suitable FMs to the XP model. The FMs refers to pertaining structural relationship between the elements of the system, by using formal logic and discrete mathematics, in this way the software development industry is provided with a more concrete and versatile framework.

In this paper we will introduce a Formal Extreme Programming (FXP) model. In this model we map FMs (SCR, Algebraic Specification and Design by Contract) on different phases of the XP model. This mapping provides us a formal documentation in the form of Software Cost Reduction (SCR) tables and formal specification using algebraic specification. This reduces risk, we can say up to 99%. This FXP model can efficiently be used for Safety Critical Systems, life critical systems and complex Object Oriented Systems (OOS).

Software Cost Reduction

The Software Cost Reduction (SCR), is mostly applied to the requirement specification phase**.** It manages four tables to specify the requirements. Condition table specify post conditions and functioning of a process. It shows all the conditions implied on a mode.

i- There should be one condition table for each mode of the system.

ii- Event transition table will be designed as a function of mode and event. It describes the transition between two events.

iii- Linkage table shows the relationship between two modes, and this table is designed in such a way that it shows if we are at mode Mi and an event E occurs it will lead us to mode Mj. This table shows the transition between all modes of the system in tabular form.

iv. This table in SCR contains the record of each data type to be used in the development.

Algebraic Specifications

The algebraic specifications are used to specify the subsystem interface in this way operations are specified while specifying their relationship between operations for an abstract data type. The algebraic specification is written using a formal language. The basic format of algebraic specification is shown in figure 1. This figure shows that there are five parts of algebraic specification, i- specification name, ii- imports, list of specification names, iii- informal description including return type, iv- operation signatures, describes the syntax of the interface and v- axioms describe the semantics of operations to show the behavior of the abstract data type.

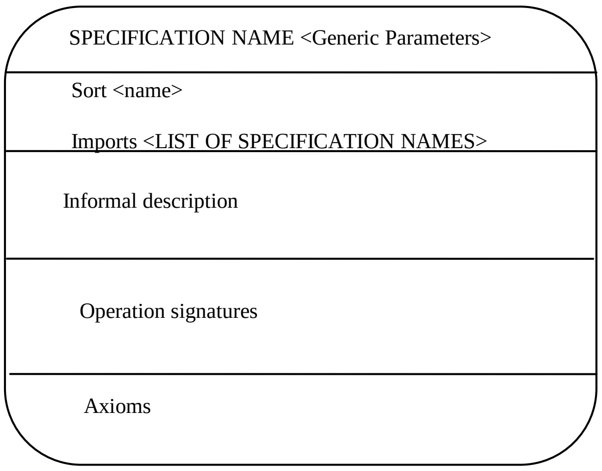

SPECIFICATION NAME <Generic Parameters>
Sort <name>
Imports <LIST OF SPECIFICATION NAMES>
Informal description
Operation signatures
Axioms

Figure 1. Algebraic Specification

Design by Contract (DbC)

Design by Contract is a pragmatic technique introduced by Bertrand Meyer [6]. The DbC can be called a defensive technique for design of software. A technique named as defensive programming was used before DbC, but that requires the routine to be more general, it will be dangerous otherwise. DbC is a contract between the working software and the user. It defines the limitations on the functions, by specifying the pre-conditions and invariants; the desired result is defined by post-conditions. Those limitations should require to be fulfilled during execution. The DbC is not supported in most of the previously designed languages like C# and java etc., but latter on extensions to some languages are introduced to support the DbC like in Java, JML and .net etc.

The Literature study about XP, FMs and combining XP and FMs is explained in section 2. All phases of proposed "Formal Extreme Programming" (FXP) model, along with the purpose of mapping FMs, are explained in section 3. To verify our proposed model a survey through questionnaire was conducted form different software houses. The opinions of professionals are analyzed in section 4. Section 5 comprises the conclusions.

Related Work

The agile development gives quick response to satisfy the customer by being iterative, incremental, adaptive, and convergent. The traditional software development have heavy software documents and less response to the changing requirements, while the agile development methods reduces the heaviness of the documentation, and provide a quick response to the changing requirements [7]. All aspects of XP are explained and it is described that the use of XP will render a guarantee to the programmers that they will work for the routine problems and will not face any disaster alone. Also for the customer and the project managers, who will get the best and valuable response after every programming week [3]. XP development methodologies were characterized as iterative, incremental and interactive with customer and developing team. It is also termeded as "people centric approach" with adaptive behavior [8].

Paulk [4] stated that the XP is used to provide system perspectives for programming and that the Capability Maturity Model (CMM) demonstrates system perspectives for process improvement. He did not support the use of XP for life critical projects because there are some drawbacks of XP such as it provides poor and less documentation, risky for life critical systems and de-emphasis on architecture.

Using FMs

FMs can be used to overcome this problem of XP. The FMs were studied in 1970's in Europe, and mostly applied to the upper stage of development. These methods are used to describe the requirements, design and test in a mathematical way. A general framework, end of proof, data refinement, algorithmic refinement and hiding, is described and mapped on computation paradigm. A set theoretical models for computation was explained, both forward and backward approaches were described for this purpose [9]. Victor [10] adopted a Model Driven Engineering approach to present a way to FM tools to rely on agile methodologies. He proved that the mathematics of FMs can be applied to modelling, examination, and inspection at many different phases, e.g. the Z notation is used for documentation. Wolff [11] used the formal specification techniques as a part of the agile development process scrum. Two teams are used which work parallel. One with conventional development method and the second uses FMs. He explained that combining these two methodologies can bring the best of both the agile and FMs. Zuo et al. [12] applied the refinement of Object and Component System (rCOS) FM to the agile software development. They divided the agile process in the four stages and introduced the formal foundation of the agile development process in rCOS framework. Larsen et al. [13] performed a reality check that the agile methods can be combined with the FMs. They explained the four value statements and showed how FMs can be applied on the agile methodologies. It is proved that the formal system modeling can support the agile development, and provides tools for correct transformation of complete running systems, and suggest that some standardization are needed to support the Object Oriented Modeling [14]. Black et al. [15] explained that the FMs can add values to the testing, requirements, documentation, verification and refactoring by acting as a safety net. They explained that both the agile and FMs are friends not foes, because both have the aim to produce reliable software, FMs by assurance and solid documentation, and agile methodology while satisfying customer needs, and allow flexibility. According to them the combination of these two will be a promising contract. They jointly can succeed and offer useful interchange within both. FMs can improve the quality of the system while using Design by Contract, to measure the vigilance and diagnosable [16]. The ten commandments of FMs were restudied 10 years later, because some peoples did not strongly agree with them. It was believed that FMs facilitates passing trends and will always have their importance in the software industry [17]. Tretmans et al. [18] discussed seven myths of FMs (Guarantee correctness, proving program, use for safety critical systems, require mathematical training, increase cost of development, unaccepted by user, and not used on real software). They analyze and affirmed that the use of FMs can increase the quality of software and are quite usable at industrial context, but in addition, they require learning process. The authors suggested a need to make the FMs applicable, and integrate them in the software engineering practices.

Proposed Model

The relationship between two areas of software engineering paradigm, agile methodologies and Formal Methods were focused in this research. XP and FMs are combined to introduce a new model of software development, named as "Formal Extreme Programming" (FXP) model. This model adds the spice of FMs to the sweetness of XP. In this model SCR, algebraic specification and DbC are mapped on the XP model. The flow diagram of this proposed FXP model is given in figure 2. The phases of FXP model are explained in this section.

User Stories

There are many requirement gathering strategies that can be used to gather the requirements efficiently, depending on the design situation and nature of the project. To collect most appropriate requirements it is better to provide our customer the easiest and comfortable way to describe his/her requirements. In any of the design situation the first problem faced by a software engineer is to discover the real problem. It is considered that the use of user stories/story cards is best option. At this phase of the FXP model, to get the user's requirements, let the customer write as many different stories as he/she can. Greater number of different user stories means more detailed explanation of almost all the functionalities to be performed. This story cards help us to read the customer's mind easily. Once the story cards are filled by customer these cards are passed to the development team. The team will read and analyze the stories. If there is something found missing in a story they do ask the customer to add

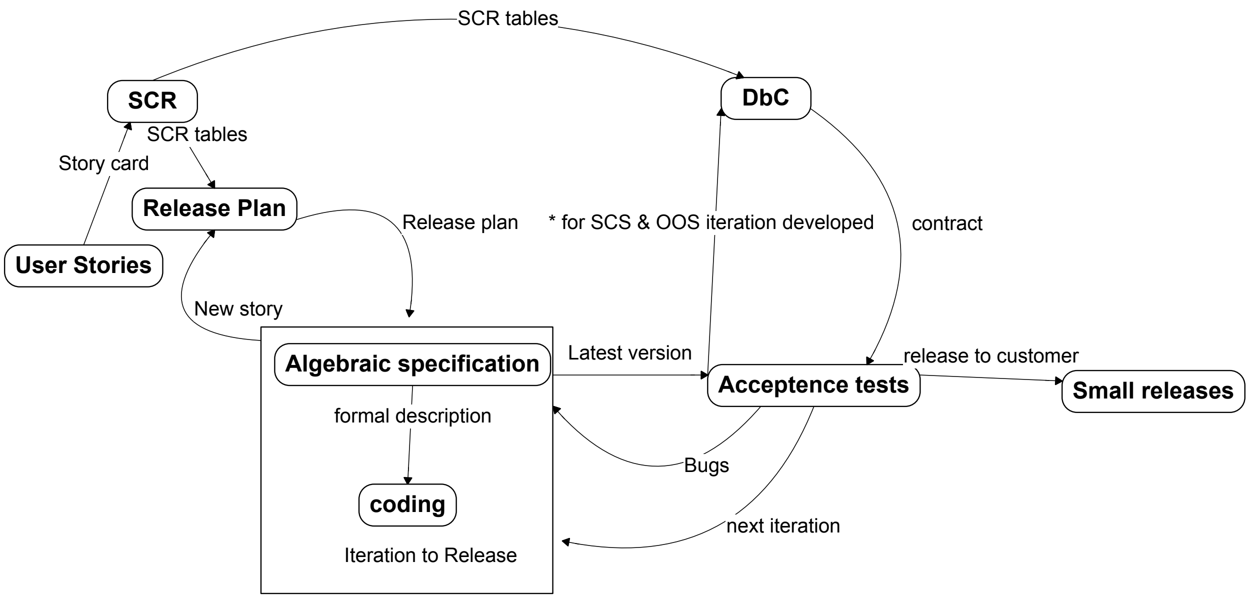


Figure 2. Formal Extreme Programming Model

more detail about that. When the customers mention enough detail in a story card for a particular requirement, the story card is reviewed by the development team and then saved with the priority given by user.

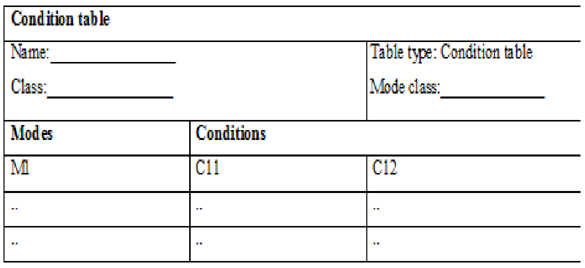
Software Cost Reduction (SCR)

Once all the user stories are collected the project move to SCR phase. The purpose of this phase is to formally describe the user requirements. It is good practice to take a story, analyze it and formulates it to index cards. If by chance there are many basic requirements described in a single story, and unfortunately the team missed any of them, it can lead to failure and disgrace the team. It can be more hazardous especially when we are working for a safety critical system. For this situation, a phase is introduced to formalize the XP model. At this phase a FM named "Software Cost Reduction" is applied to formally describe the requirements from the given user stories.

For SCR, four tables were constructed, as described in introduction section, to specify the requirements, completely by describing all required aspects, concretely and concisely. These SCR tables must completely describe the requirements, conditions and their linkage unambiguously.

i. Condition Table

The condition table is designed by using task description of the story card. In this table all the conditions implying on a mode are described for each mode separately. At least one condition table is required for each controlled variable. In this way the condition tables describe the complete functionalities of the system. This condition table is shown in figure 3.

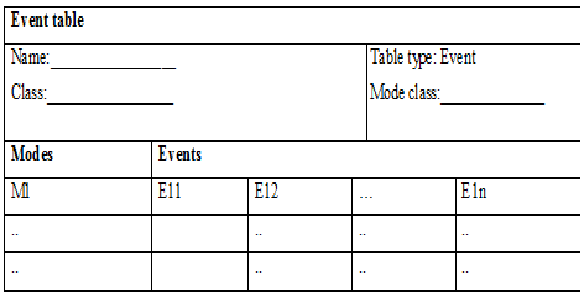


Condition table		
Name:____________ Class:____________		Table type: Condition table Mode class:__________
Modes	Conditions	
MI	C11	C12
..	..	..
..	..	..

Figure 3. Format of Condition Table

ii. Event Transition Table

In event transition table the designer describe the output variable as relation to mode, events and values. It describes the transition between a mode and an event, as shown in figure 4.

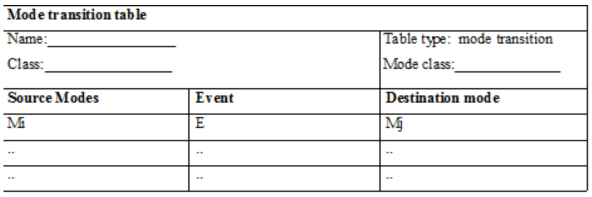


Event table				
Name:____________ Class:____________			Table type: Event Mode class:__________	
Modes	Events			
MI	E11	E12	...	E1n
..		..	..	..
..		..	..	..

Figure 4. Event Transition Table

iii. Linkage Table

This can also be known as mode transition table. This table is designed to show relationship between modes as shown in figure 5.



Mode transition table		
Name:_______________		Table type: mode transition
Class:_______________		Mode class:____________
Source Modes	**Event**	**Destination mode**
Mi	E	Mj
..	..	..
..	..	..

Figure 5. Limkage Table

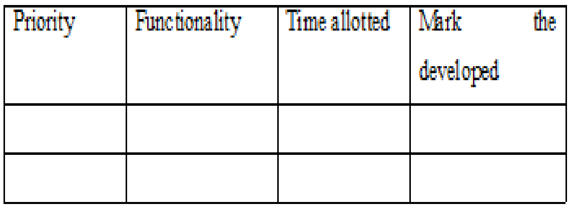
iv. Directory

This table in SCR comprises the record of each data type to be used in the development. There are four types of directories maintained. Content directory is used to give values to constants, type directory to define the user defined types, mode class directory to store list of all class modes and variable directory is used to store list of variable

A newer FM Graph Based Hoare Logic (GBHL) can be mapped at this phase for complex Object Oriented System. The same was introduced by Zhao et al. (2013). But latter on it was realized that the mapping of GBHL at this phase costs more than its benefits. So that is why it is decided that the GBHL should not be mapped at this stage.

Release Plan

At this phase the development process of the project will be planned, by using formalized requirements (SCR tables). For that purpose a priority table of requirements is designed, shown in figure 6. Take condition table from the SCR tables and assign a priority to all functionalities, by keeping in view the relationship between two events and user assigned priority of each of the functionality. The priority table consists of four columns; priority, functionality, time allotted and mark the developed.



Priority	Functionality	Time allotted	Mark the developed

Figure 6. Priority Table

After designing priority table the next step at this phase is to draw a Gantt chart to show the schedule of the project. The Gantt chart shows graphically the time period required for the development of a particular task. It also shows that which task can be done in parallel to the other. In addition to priority table and Gantt chart the size of team, code ownership, working hours, sitting arrangement and programming pair are also planned.

i. Team Size

It is required to plan that how many members will be in your team. In FXP we ideally suggest to choose small teams.

ii. Code Ownership

The code ownership is also required to be planned at this phase. The FXP can support both ownerships. The owner of the code can change the code, or Collective ownership may be desired and provided. When owner pair is only allowed to change, the details of the code there are less chances of disaster situation. When we allow the collective ownership, one of the member in each pair knows about FM, we are using. So they can understand the code and consequently there may be more disaster situations.

iii. Working Hours

We require our developers to be relaxed during the working hours. For that purpose ideal working hours suggested for FXP are 35-40 hours in a week. We advise that do not plan more than 40 hour week because a tired developer produces more and more errors.

iv. Sitting Arrangement

Plan a comfortable sitting arrangement. As we support pair programming so the arrangement will have two chairs on single PC.

v. Planning Pair

For FXP model we need a programming pair with at least one of the member as FM expert, and the other can be a programmer or FM expert. The expert is supposed to have enough knowledge about FMs, especially about SCR, algebraic specification and DbC. If in any case the expert is not that much familiar with the FMs, short training will be arranged.

Spike Solution

It is mandatory to discuss “spike solutions” introduced in the XP model. The spike solutions are the solution to the problems considered tough from technical and design point of view. In XP it is a solution to a particular problem only, while all other aspects of system are ignored. For most of the time the spike is not good to have. For FXP we do not strictly suggest the spike solution to be implemented for difficult problems, because they solve a design problem from one perspective and ignore all other aspects of the system as a whole. That is why it is not used by FXP model. Spike solutions are considered good when developers are knowledge limited, and not when time limited [19].

Iteration to Release

At this phase the set of tasks from release plan are grouped into iteration (the set of related functionalities needed to be developed together) on the basis of their priority. Ideally iteration does not take more than 3 weeks. At this stage the functionalities with failed acceptance tests are also reviewed. Afterwards programming pair writes all possible test cases for all functionalities. When all possible test cases are written algebraic specification will be written for each mode of the system. In XP the selected user stories are translated to task cards/index cards (the story cards written in developer’s language). While writing these index cards if any task is found duplicated then it can be removed.

Algebraic Specification

In FXP Model we replace the index/task cards by algebraic specification. These specifications provide a more formal view of a task, and data type. The algebraic specification describes the requirements of a task and functions of a data type more clearly and concisely, to make the programmer’s job easier as compared to the index cards. The FM experts write the algebraic specification for a system. We prefer the algebraic specification to be written before coding because in this way we can get a very concrete, formal and to the point description of the requirements and functionalities to be implemented, in a mathematical way. It specifies all the functions performed, their signatures, return values of each function, and axioms to evaluate the functions. The axioms describe all the possible conditions of a function. Afterwards the team decides that which functionality is required to be developed first. This can be viewed from the priority table, the developers sign up the task, and estimate the time required to develop.

Coding

Once algebraic specification is written for iteration it is passed on to the programming pairs to translate it to code. The algebraic specification also supports coding standards, the one followed in the specifications. At this phase a working iteration will be developed. One thing to note here is that in the pair of programmers at least one of the members should have to have training about formal methods. The FM expert will analyze the algebraic specification and help his partner for implementation/coding.

Acceptance Test

At the completion of coding the latest version of running iteration is passed on to the acceptance test phase. At this phase the running iteration is checked against each requirement specified by the user and if any error/bug is founded it will returned back to the iteration planning or coding phase.

In FXP model we associate the Design by Contract (DbC) conditionally with this phase. The purpose of associating DbC at this phase is to secure the development of safety critical system or a complex OO system.

Design by Contract

The DbC is applied at the acceptance phase only for the Safety Critical Systems (SCS), and for complicated Object Oriented (OO) systems. For the design of a SCS the developer is required to have balance of high quality skills with the aspects discussed by [20]. DbC is applied on these systems to form bugs-free SCS and OO systems, because designing and reviewing OO system is annoying, burdensome, and costly. In the case of SCS, we have spent lots of cost on their implementation for example in case of an aircraft system; a single lapse is enough to destroy the whole system. So in both of these cases it is better to design a contract rather than destroying whole system. We do not suggest it mandatory for simple and smaller projects because it does nothing except increasing the cost of the system. So by following the key point of designing software “do not spend money on unneeded features” we does not recommend it for simple and small projects.

DbC is a contract between the running iteration and the user. This does not allow the user to violate the contract. This contract is implemented by languages like JML and Eiffel etc. DbC include a pre-condition, invariant and a post condition. These three components of DbC will be written using JML, java or Eiffel etc. The DbC is used to blame the faulty section of the program. If the implementation of the program breaks the contract then it is informed and will be fixed later on. The DbC add the checks for efficiency of the program, and also prevent it from inefficient defensive checks.

Small Release

If the iteration passed the acceptance test it will be released. Now the user checks it from each perspective and approve. If the customer found any problems in the release or want to change any requirement then user will give feedback and communicate with development team/team leader. If any change in requirements is requested it will be implemented by passing through all the phases of FXP model.

RESULTS AND DISCUSSIONS

In this section the validity and acceptance of FXP model is explained. A survey through questionnaire is conducted from the professionals of software industry working in different software houses.

The use of formal methods is reduced because of two main reasons;

i. The developers are generally less inclined to change their old way out, to avoid difficulties.

ii. They considered the same as difficult and time consuming, and eventually more costly, because of required level of mathematical formulations.

However for the design of SCS and complex systems it is required to have good quality assurance abilities, like FMs and quality assurance tools. The main hazard in the acceptance of FXP model is cost of FMs. The professional’s opinion about the cost of FMs used in FXP model is illustrated by figure 7. The survey concluded that the FMs are not as much costly and awful as they are considered in the past years because total of 53% professionals disagree with the statement that the formal methods are costly.

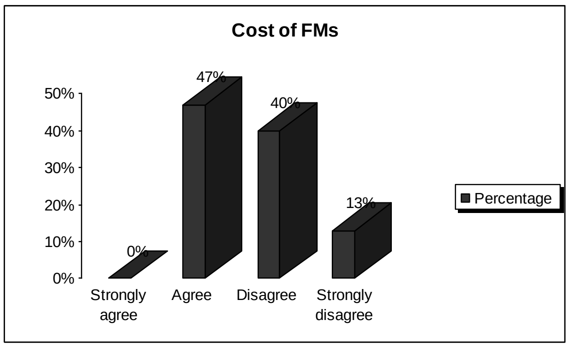


Figure 7. Cost of Formal Methods

About 80% of the professionals support the formal specification of the requirements. At the initial stage of FXP model the SCR is used to specify requirements/user stories in tabular form. The purpose of specifying requirements in SCR tables is to considerably reduce the ambiguity in requirements and their specifications. The survey results show that it is good practice to describe the requirements formally by using SCR. The results shown in figure 8 illustrate that the professionals appreciate the use of SCR. This shows that 86% of the professionals support the use of SCR tables for requirements specifications.

It is studied that the GBHL can be applied to represent the requirements formally and graphically by the construction of “class graph” and “state graph”. We get positive response by the professionals about that, as shown in figure 9. However it is not suggested for FXP model, because when cost of applying GBHL is analyzed it is recognized that this mapping costs more than its relative benefits.

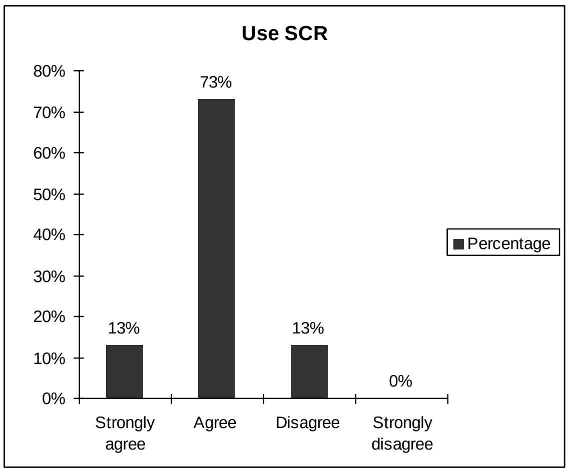


Figure 8. SCR For Requirements Specification

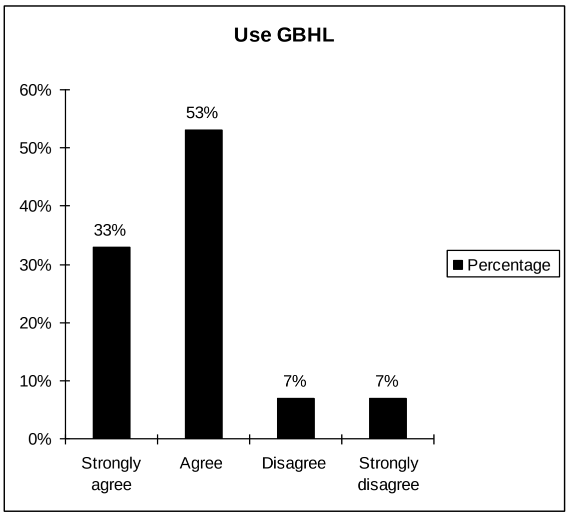


Figure 9. Using GBHL

Algebraic specification is written at the iteration development phase before coding. Figure 10 demonstrates that the cost of failure for larger project is more than the cost of representing the tasks/functionalities in algebraic specification. 73% of the professionals agreed that the cost of failure is more than the cost of applying algebraic specification.

In FXP model the need of spike solution is reduced because of algebraic specifications and use of formal methods at initial phases. The professional's opinion about the creation of spike solution is shown in figure 11. About 80% of the professionals agreed that if we use FMs at initial stages of development, there remains no need for spike solution.

Applying Design by Contract (DbC) is not suggested for simple projects while for complex Object Oriented systems, life critical systems and SCS, it is strongly recommended to write a contract, because the cost of failure of these systems is more than the cost of applying DbC. The figure 12 shows that 87% of the professionals agreed that the cost of applying DbC is less than the cost of failure.

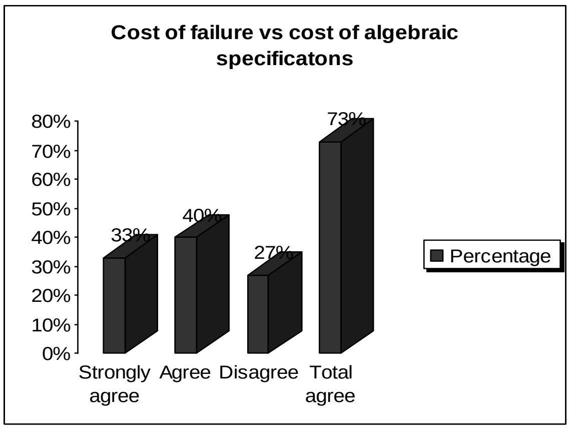


Figure 10. Cost of Algebraic Specification

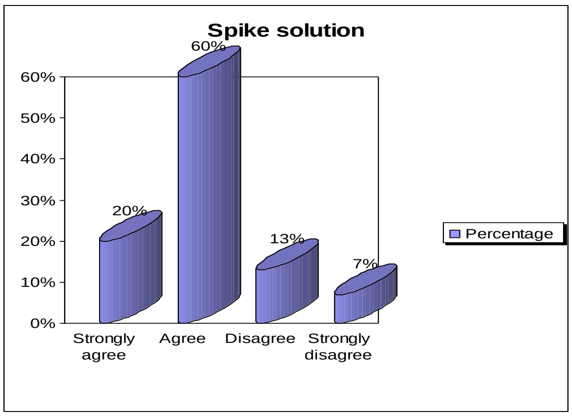


Figure 11. Spike Solution

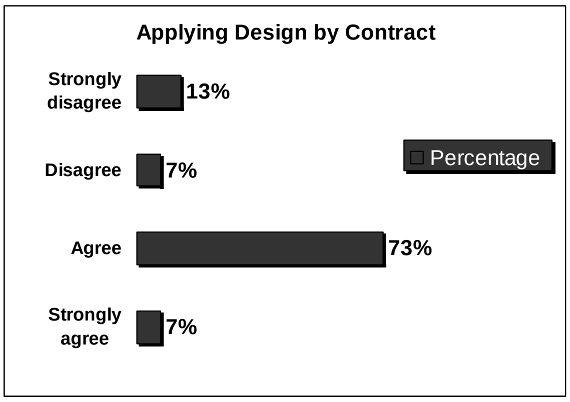


Figure 12. Cost of Applying DbC

Some of the professional's opinions about the acceptance of this proposed FXP model are given in table 1.

CONCLUSION

In this research a new formal agile model FXP was introduced which is preferred to use for safety critical systems and complex OO systems. In this regard XP is selected from agile methodologies, and SCR, algebraic specification and DbC from FMs. The FMs are mapped on XP to introduce a new FXP model. This model gives us best quality software for 99% of the times. No doubt the FMs increase the cost but in this research work we concluded that the quality they provide is more than its cost. The survey results showed that this model can bring good change in the traditional XP model in future.

Table 1. Professional's Comments on FXP Model

Name	Comment
Anonymous	"It is a good step towards the betterment of software industry. It would surely help making new innovations in this industry."
Muhammad Yousaf	"The Extreme Programming discipline will change entirely the SDLC, and hence will improve the software development practices in the coming times."
Anonymous	Yes! This will bring a closed coordination and bridging gaps in SDLC.
Anonymous	"Mapping exercise is very important and will not improve the software delivery life cycle but customer satisfaction will also improve."
Khalid Rasheed	"The mapping of Formal Methods on Extreme Programming (XP) can bring a good change resulting better development in software industry."
M. Adnan Khadim	"Hope so"
Abu Sufyan	"It's all depends upon the nature of project, resource availability and their capacity as well as budget and project timelines."
Nadeem Aamir	"Management's decision prevails."

Future Work

It is required to automate the use of FMs and a practical research is required to prove the efficiency, and correctness of the proposed model. The FMs should be mapped to other agile development models as well as traditional development models.

REFERENCES

[1] Beck, K. 1999. Embracing Change with Extreme Programming. IEEE Computer. (CA, USA). 32(10): 70-77. ISSN: 0018-9162.

[2] Lindstrom, L. and R. Jeffries, 2004. Extreme Programming and Agile Software Development Methodologies. Information Systems Management. (Florida, USA). 21(3): 41-52.

[3] Beck, K. and C. Andres, 2004. Extreme Programming Explained: Embrace Change. 2nd Ed. Addison-Wesley Professional. (MA, USA). ISBN: 0321278658.

[4] Paulk, M. C., 2001. Extreme Programming from a CMM Perspective. Software, IEEE. (PA, USA). 18(6): 19-26.

[5] Bowen, J., and V. Stravridou, 1993. Safety Critical Systems, Formal Methods and Standards. IEEE Software Engineering Journal. (NJ, USA). 8(4): 189-209. ISSN: 0268-6961.

[6] Meyer, B., 1992. Applying Design by Contract. IEEE Computer Society. (CA, USA). 25(10): 40-51.

[7] Erickson, J., K. Lyytinen and K. Siau, 2005. Agile Modeling, Agile Software Development, and Extreme Programming. The State of Research. Journal of Database Management (JDM). (MO, USA). 16(4): 88-100.

[8] Abrahamsson, P., O. Salo, J. Ronkainen and J. Warsta. 2002. Agile Software Development Methods Review and Analysis. VTT Publications 478. (Oulu, Finland). ISBN: 951-38-6009-4:107 p.

[9] Abrial, J. R. 2013. "Set-Theoretic Models of Computations." Springer-Verlag Berlin Heidelberg. (Marseille, France). pp. 1–22. LNCS: 8051.

[10] Victor, K. R., 2013. Adaptable Model-Driven Engineering for Formal Methods Integration with Agile Techniques for Design of Software Systems. Academic Research International. (Lodhran, Pakistan). 4(1): 446-456. ISSN: 2223-9553.

[11] Wolff, S., 2012. Scrum Goes Formal: Agile Methods for Safety-Critical Systems. In Proc. of IEEE Conf. on Formal Methods in Software Engineering: Rigorous and Agile Approaches. (Zurich, Switzerland). pp. 23-29.

[12] Zuo, A., J. Yang and X. Chen, 2010. Research of Agile Software Development Based on Formal Methods. In IEEE Conf. on Multimedia Information Networking and Security. (Nanjing, China). pp. 762-766.

[13] Larsen, P. G., J. S. Fitzgerald and S. Wolff, 2010. Are Formal Methods Ready for Agility? A Reality Check. In Proc. of 2nd Int. Workshop on Formal Methods and Agile Methods. (Pisa, Italy).Vol. P-179. pp.13-25. ISBN: 9783885792734.

[14] Lowe, M., 2010. Formal Methods in Agile Development. Electronic Communications of European Association of Software Science and Technology. (Berlin, Germany). 30: 1-6. ISSN: 1863-2122.

[15] Black, S., P. P. Boca, J. P. Bowen, J. Gormanand and M. Hinchey, 2009. Formal Versus Agile: Survival of the Fittest. IEEE Computer Society. (CA, USA). 42(9): 37-45. ISSN: 0018-9162.

[16] Le Traon, Y., B. Baudry and J. M. Jezequel, 2006. Design by Contract to Improve Software Vigilance. IEEE Transactions on Software Engineering. (CA, USA). 32(8): 571-586.

[17] Bowen, J. P. and M. G. Hinchey, 1995. Seven More Myths of Formal Methods. IEEE Software. (Oxford, UK). 12(4): 34-41.

[18] Tretmans, J., K. Wijbrans and M. Chaudron. 2001. Software Engineering with Formal Methods: The Development of a Storm Surge Barrier Control System Revisiting Seven Myths of Formal Methods. Journal of Formal Methods in System Design. (MA, USA). 19(2): 195-215.

[19] Pressman, R. S., 2010. Software Engineering: A Practitioner's Approach. 7th Edition. McGraw-Hill Higher Education. (Singapore, Asia). pp. 65-93 and 557-582. ISSN: 0071267824.

[20] Bowen, J., 2000. The Ethics of Safety-Critical Systems. Communications of the ACM. (NY, USA). 43(4): 91-97.