MAPPING OF TRADITIONAL SOFTWARE DEVELOPMENT METHODS TO AGILE METHODOLOGY

Rashmi Popli[1], Anita[2] and Naresh Chauhan[3]

[1]Assistant Professor, [3]Professor , Computer Engineering, YMCA University of Science & Technology, Faridabad, Haryana, India
rashmimukhija@hgmail.com, nareshchauhan19@gmail.com
[2]Research Scholar, Computer Engineering, YMCA University of Science & Technology, Faridabad, Haryana, India
anitaarora_20@rediffmail.com

ABSTRACT

Agility is bringing in responsibility and ownership in individuals, which will eventually bring out effectiveness and efficiency in deliverables. Companies are drifting from traditional Software Development Life Cycle models to Agile Environment for the purpose of attaining quality and for the sake of saving cost and time. In Traditional models, life cycle is properly defined and also phases are elaborated by specifying needed input and output parameters. On the other hand, in Agile environment, phases are specific to methodologies of Agile - Extreme Programming etc. In this paper a common life cycle approach is proposed that is applicable for different kinds of methods. This paper also aims to describe a mapping function for mapping of traditional methods to Agile methods.

KEYWORDS

Agile Software Development, Pair programming, Mapping function.

1. INTRODUCTION

Agile software development life cycle appeared as a reaction to traditional approaches of developing software and it defines the need for an alternate approaches to traditional documentation based, heavyweight processes. Lifecycle of any model is the time span between activities that comprises of release of first version to last version. Software effort needed for development follows one lifecycle. The aim of software development [4], [15] is to utilize the resources and time to its fullest but not at the cost of sacrificing quality. Lifecycle models provide a starting point for defining what will be done.

The traditional software life cycle models, like the Waterfall model, spiral model and prototype model, are based primarily on heavy documentation. The teams in traditional approaches are role based. The allocation of work is done initially and it specifies "what is to be done, how it is to be done and also the exact time allotted for doing that work". This shifts the focus from individuals and their creative abilities to the processes themselves. The traditional software development methods are often referred to as "plan-driven" or "documentation based" or heavyweight methods

Jan Zizka (Eds) : CCSIT, SIPP, AISC, PDCTA - 2013
pp. 117–123, 2013. © CS & IT-CSCP 2013 DOI : 10.5121/csit.2013.3612

Agile software development [10] is adopted for attaining quality and quality is respected by each and every customer. To ensure this transition, software industry requires many concerns to be discussed namely top level management interest, infrastructure needed, resources, attitude and many more. For accommodating Agile in the software industry a mapping is required that need to be discussed so that transition can take place between two software development life cycles in proper manner. In this paper, a mapping function has been presented so that transition task can be achieved with convenience of team members and top management.

2. SOFTWARE DEVELOPMENT LIFE CYCLE

Traditional methodologies are predictive because in these a schedule is made at the beginning of a project. Complex software systems can be built in a sequential, phase-wise manner where all of the requirements are framed at the beginning, design is completed next, and finally the master design is implemented in producing quality software. A brief about traditional model is described below.

2.1. Waterfall/Linear Sequential Model

In it, development flows steadily through requirements analysis, design implementation, coding, testing, integration, and maintenance [4]. With every phase, one deliverable is compulsory. This is basically document driven model in which proper sequence is maintained. Problem of this classic approach is mainly inflexibility which is related with change in customer mind set by seeing changing needs of time. Also, bugs keeps on propagating from one phase to another.

2.2. Prototyping Model

The process of this model involves many small activities viz identification of basic requirements, development of initial prototype, review and enhancing of prototype. There are two types of prototyping including close-ended and breadboard. In the former case, prototype of the requirement is created that will eventually be discarded rather than becoming part of the final delivered software.

In Agile way of software development, customer interactions are more important and either customer or one of the representative of customer is always present with the team members so that feedback can be received for the improvement at any time and requirements can be modified by changing trends of market.

2.3. Iterative Incremental Model

Incremental model is an evolution of waterfall model. The product is designed, implemented, and tested as a series of incremental iterations. The Incremental software development model may be applicable to projects where the basic software functionality is required early. But in agile context, builds are gradually created. Then review is done with the help of demonstrations to the customer. Also, review is possible within the existing team members or product owners.

3. RELATED WORK

Research work in Agile Estimation was done by Abrahamsson et al. [19], William et al. [20], and Erickson et al.[21]. Their work describes the various agile methods and how these methods can be applied in industry and why these methods are beneficial than traditional.

Williams et al. [21] investigated the usage of a subset of XP [6] practices at a group in IBM. The product developed at IBM using XP was found to have significantly better pre-release and post-release quality compared to an older release. The teams using XP reported an improvement in productivity, schedule, cost and effort estimation.

4. AGILE SOFTWARE DEVELOPMENT LIFE CYCLE

Agile SDLC contains the six phases as shown in Figure 1: Pre project planning, Start, Construction, Release, Production and Retirement. In Pre-project planning phase, firstly the goals of the project and market aspects are defined. In the Start phase, the requirement modeling is done. In it, active participation of stakeholders is needed to identify the initial requirement modeling or high-level requirements for the system. Main goal of Start phase is to understand the problem and solution domain.

During Construction iterations, high-quality working software is delivered incrementally, which meets the changing needs of the user or customer. In Agile Software Development, change in the requirements is allowed to meet the exact needs of the customers. At the end of each development cycle or iteration there is a partial, working system to show it to the customer. Pre-production testing can be done like system integration testing.

During the Release iteration phase, also known as the "end game", the system is transit into production. The goal of the Production Phase is to keep systems useful and productive even after the product has been deployed to the user community. The Retirement Phase is also known as system decommissioning phase.

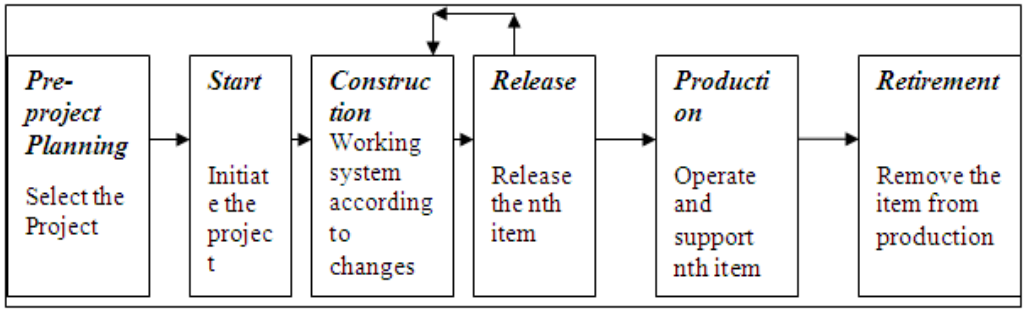

Figure 1. Agile Life Cycle

5. PROPOSED AGILE MODEL

Accepting change is the mandatory requirement for the agile teams. Without accepting change, agile cannot exist in industry. Issue is how to do the transition when one of the traditional models is the heart of the company and everybody has expertise in that. One thing is for sure that if somebody is ready for accepting change then in the beginning things would seem to be complex but with the support of the organization / management, team and proper coach agile can be implemented with good success rate. The main components (Refer Figure 3) of the life cycle are:
Team Formation by good recruitment policy (TFR)

Goal Building cycle with Quality Analyst, business Analyst and Customer. (GBC)
Effort and Budget estimation (EBE)
Coding & Testing activities with Communication (CTC)
Demonstrations in Review meeting with feedback (DRF)

Risk evaluation and correction (REC)
Satisfaction for all parties (SFP)
These components are the base of an agile culture. Description of each and every component is given below:

TFR- In Agile environment, Team can be formed by devoting time by trainers in upgrading the technical and managerial skills, polishing the right kind of attitude, embracing the change from time to time and by following good recruitment policies to identify the right person. In the team, there can be experienced members as well as fresher but attitude is the biggest factor while doing recruitment.

GBC- Stories [8] are identified, evaluated and approved by customer, quality analyst and business analyst by considering the market demand and return on investment value.

EBE- Effort and budget are estimated by considering the resource and tool requirements for each story from time to time by product owner or customer. After identification of iteration or sprint stories two weeks cycle starts. Estimation related with effort can be done by planning poker or any other famous technique. Estimation is possible at three scales namely iteration plan, release plan and project estimation. Estimation units are story points and ideal time.

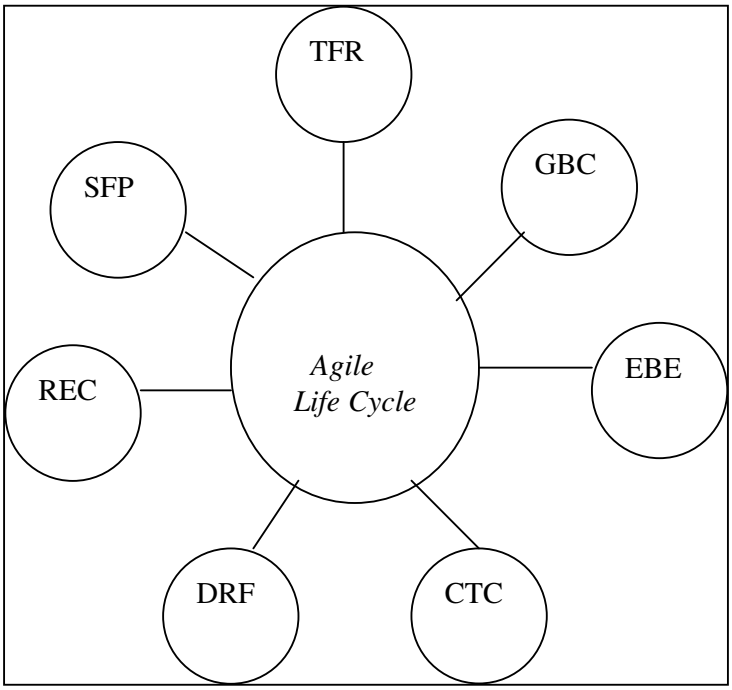

Figure 2. Proposed Agile Model

CTC- Pair programming approach [1], [2], [11] is used while doing coding and testing in which one person is the leader or executor and second person is the reviewer by working on the single terminal. This immediate feedback helps in reducing number of bugs which may otherwise keep on propagating. Also, test driven development [3, 7, 12, 13, 14] (TDD) approach is used in which test cases are written before writing the code for the story.

DRF- At the time of review meeting, team members, management and customers sit together for the purpose of demonstrations of the software product. One of the representatives of team gives the demo for the product. Then, goal matching action is performed. This review meeting is informal kind of meeting.

REC- Risk evaluation is done for the future stories so that things can be improved or taken to the next level of quality. Actually customer is not just end user rather she is bothered about money, quality, [16]. So, there is a need of early detection of high risk stories so that risky outcomes are in mind before finishing the current stories.

SFP- Ultimately, all parties are satisfied namely customer, team and management as product can be delivered on time by following continuous delivery, continuous feedback, continuous integration, continuous testing and continuous ROI.

Now, question of mapping arises, when there is a need to do transition from existing model to agile. Issues that may come during mapping are:

What is the reason for the transition?
Is management interested or customer?
Whether team is of that much calibre or not?
Whether infrastructure requirements are sufficient like open workspace or not?
Whether automated tool knowledge is needed?
Whether requirements are fixed or dynamic?

After resolving all these issues, work can be started for the mapping function from one model to agile. In Mapping equation (1), mapping function MF is important. Its role is to map the large teams into small teams (T), large tasks into small stories (J), long iteration into small sprint (I), long feedback cycle into instant feedback (F), late delivery into fast small delivery (D), long meetings into daily small meetings (M), late testing into test driven testing (TG), two monitors into one terminal for pair programming (MO), estimation in lines of code into story points (E), project manager into no boss approach (B) and co-ordination effectiveness(CE).The CE (Refer Figure 3) depends upon some implicit and explicit factors (Refer equation 2).

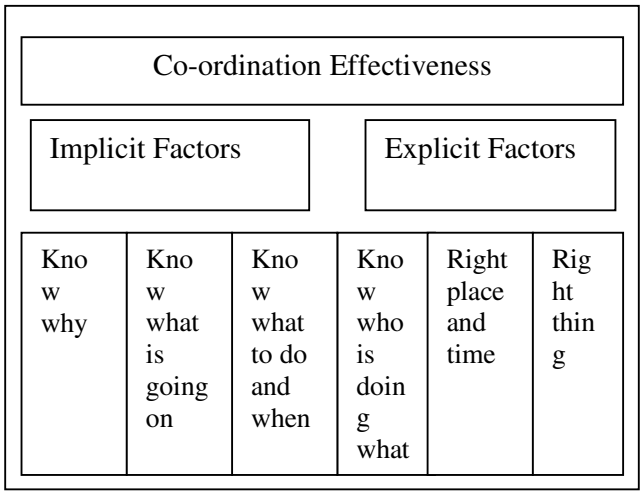

Figure 3. Factors of Coordination Effectiveness

$$MF = (T, J, I, F, D, M, TG, MO, E, B, CE). \quad (1)$$

$$CE = \text{Implicit factors} + \text{Explicit factors}. \quad (2)$$

By using this mapping function, any of the traditional models can be converted to the agile model. In this function, mainly ten parameters are there which are needed for the complete agile environment in an organization. Along with it at hardware ground, cubicles can be converted into open work environment, many documents can be converted into one story board, many automated tools can be converted into specific tool for specific technology or domain, and more overtime is converted into 40hrs/ week of effective work. It means, in short, agile is "less is more" approach which is more beneficial with less cost and time. But interactions are more through face to face communication in collocated culture and through video conference in distributed environment. The major benefits of mapping function are that the time consumption would be less. It is very simple to implement it and everybody would be happy (team members, client, and management).

6. CONCLUSION & FUTURE WORK

In this paper, an agile model is proposed for the purpose of adopting a new process in the organization. Also, a mapping function is presented for the sake of doing transformation from one traditional model to new agile model. This mapping function is the backbone of the agile culture in the organization and success rate of any agile project can scale up by matching all the mapping parameters.

In future, on the basis of this mapping function, a tool can be designed which can identify the existing parameters in the existing traditional model and finally can do the transformation in the form of new parameters namely team size, sprint size, effort size, and how many terminals are needed and how frequently switching would take place among the team members in pair programming scenario. Also, a real project can be projected where transition takes place easily and with less effort.

ACKNOWLEDGEMENTS

The authors would like to thank their employers/institutions for providing facilities for doing research.

REFERENCES

[1] Duque, R. and Bravo, C., Analyzing work productivity and program quality in collaborative programming, The 3rd International Conference on Software Engineering Advances, 2008, pp.270-276
[2] Preston,D., Using collaborative learning research to enhance pair programming pedagogy, ACM SIGITE Newsletter, Vol.3, No.1, January 2006, pp.16-21
[3] "Test Driven Development" by Kent Beck. Addison Wesley, 2002.
[4] Abran, A., Moore, J. W., Bourque, P., & Dupuis, R. (2004). Guide to the software engineering body of knowledge. Los Alamitos, CA: IEEE Computer Society.
[5] Agile Manifesto. (2001). Manifesto for agile software development. Retrieved January 1, 2009, from http://www.agilemanifesto.org
[6] Beck, K. (2001). Extreme programming: Embrace change. Upper Saddle River, NJ: Addison-Wesley.
[7] Briggs, T., & Girard, C. D. (2007). Tools and techniques for test driven learning in CS1. Journal of Computing Sciences in Colleges, 22(3), 37-43.
[8] Cohn, M. (2004). User stories applied: For agile software development. Boston, MA: Addison-Wesley.

[9] Hedin, G., Bendix, L., & Magnusson, B. (2003). Introducing software engineering by means of extreme programming. Proceedings of the 25th International Conference on Software Engineering (ICSE 2003), Portland, Oregon, 586-593.
[10] Highsmith, J. A. (2002). Agile software development ecosystems. Boston, MA: Addison Wesley.
[11] Jacobson, N., & Schaefer, S. K. (2008). Pair programming in CS1: Overcoming objections to its adoption. SIGCSE Bulletin, 40(2), 93-96.
[12] Janzen, D. S., & Saiedian, H. (2006). Test driven learning: Intrinsic integration of testing into the CS/SE curriculum. Proceedings of the 37th ACM Technical Symposium on Computer Science Education (SIGCSE 2006), Houston, Texas, USA, 254-258.
[13] Janzen, D. S., & Saiedian, H. (2008). Test driven learning in early programming courses. Proceedings of the 39th ACM Technical Symposium on Computer Science Education (SIGCSE 2008), Portland, Oregon, USA, 532-536.
[14] Kollanus, S., & Isomottonen, V. (2008). Test driven development in education: Experiences with critical viewpoints. Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education (ITICSE 2008), Madrid, Spain, 124-127.
[15] Mead, N., Carter, D., & Lutz, M. (1997). The state of software engineering education and training. IEEE Software, 14(6), 22-25.
[16] Rico, D. F. (2008). What is the ROI of agile vs. traditional methods? An analysis of extreme programming, testdriven development, pair programming, and scrum (using real options). TickIT International, 10(4), 9-18.
[17] Koch, Agile Software Development - Evaluating the methods for your Organization. Artech House Inc., Norwood, Massachusetts, 2005
[18] C.Naresh, Anita, Testing in an agile environment: A new Approach, International Conference on Next Generation Communication and Computing System(ICNGCCS-2010)
[19] P. Abrahamsson, Koskela, J., "Extreme Programming: A Survey of Empirical Data from a Controlled Case Study", Proceedings of International Symposium on Empirical Software Engineering, pp. 73-82, 2004.
[20] L. Layman, L. Williams, and L. Cunningham, "Motivations and Measurements in an Agile Case Study", Proceedings of ACM SIGSOFT Foundation in Software Engineering Workshop Quantitative Techniques for Software Agile Processes (QTE-SWAP), Newport Beach, CA, 2004.
[21] F. Maurer and S. Martel, "Extreme Programming: Rapid Development for Web-Based Applications", IEEE Internet Computing, 6(1), pp. 86-91, Jan/Feb 2002.

Authors

Rashmi Popli is pursuing her Ph.D in Computer Engineering from YMCA University of Science & Technology, M.Tech(CE) from M.D University in year 2008,,B.Tech(IT) from M.D University in the year 2004.She has 9 years of experience in teaching. Presently she is working as a Assistant Professor in department of Computer Engineering in YMCA University of Science &Technology, Faridabad, Haryana, India. Her research areas include Software Engineering, Software Testing and Software Quality.

Anita is pursuing her Ph.D in Computer Engineering from YMCA University of Science & Technology, M.Tech(CE) from M.D University in year 2009, B.Tech(CSE) from M.D University in the year 2004. She has 11 months of industry experience and 7 years of teaching experience. She is lifetime member of Computer Society of India and Agile Software Community of India. Her research includes Software Engineering, Software Testing and Software Quality.

Naresh Chauhan received his Ph.D in Computer Engineering in 2008 from M.D University, M.Tech(IT) form GGSIT,Delhi in year 2004,B.Tech(CE) from NIT Kurukshetra in 1992.He has 20 years of experience in teaching as well as in industries like Bharat Electronics and Motorola India Pvt. Ltd. Presently he is working as a Professor in the department of Computer Engineering ,YMCA University of Science and Technology, Faridabad, Haryana, India.. His research areas include Internet Technologies, Software Engineering, Software Testing and Real Time Systems.