International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 4, Number 7 (2014), pp. 687-692

http://www. irphouse.com

Implementing AGILE in Small Organizations

Ashish Agrawal[1], Sadhana Singh[2], Malay Tripathi[3], L.S. Maurya[4]

[1, 2, 3] *Software Engineering, SRMS CET (UPTU) Bareilly, INDIA*
[4]*CS/IT (HOD), SRMS CET (UPTU) Bareilly, INDIA*

ABSTRACT

To make quality centric software is ultimately the aim of every software organization either it is small or big. But sometimes, to achieve their aim becomes a white elephant for many small organizations because of some factors like a number of limited resources, small number of employees, time constraints, financial limitations etc. It has been seen that whenever a new concept develops, large organizations easily bear it but small organizations can't due to implementation cost and fear of unsuccessful. In the chain of software development models the current one is Agile Methodologies, which in turn more reliable, more dependable and more realistic in comparison to other ones. Agile methods not only provide agility but also transparency. This paper is an attempt to provide a way to implement this model in small organization so that these small organizations can also achieve the same level as of large organizations.

Keywords- Agile; Improvement; Quality; Agile Methods; Small organizations.

1. INTRODUCTION

In 2001, AGILE Alliance firstly coined the term agile manifesto which was a collection of principles and values related to agile software development methods. These methods are time boxed, iterative and incremental methods that works between self organizing, self motivated and cross functional teams. Since from the development of waterfall model many other models like iterative model prototype model, evolutionary model, spiral models and now agile are developed. The basic goal behind all these models was to make customer more satisfied, to increase ROI and to build high quality product. For example, in 2002, Mark Paulk, promoter of CMM stated (Paulk 2002), "Many of the practices in the agile methodologies are good practices that should be thoughtfully considered for any environment."

Table 1 is the brief one line description of some methods related to agile and it also shows pros and cons of these methods. Though, there are some cons but the use of these methods can reduce the number of cons presented with older software development approaches.

Table 1. *Agile Methods and their Pros and Cons*

AGILE Methods	One Line Description	Pros	Cons
Whole Team Development	Respect everyone's ideas and include every member.	P1-Improve quality planning P2-Gain commitment from everyone	C1-Require high level of communication from the entire team
Pair Programming	One coder and one tester on same module. 'Two minds are better than one.'	P1-Reduce bottleneck P2-Increases flexibility of making changes	C1-Lack of compatibility C2-Mismatched skills
SCRUM	Iterative and incremental approach that welcome changing requirements of customer.	P1-Team work together to improve quality	C1Purpose of meeting may lost
Extreme Programming(XP)	Continuous integration and testing and risk estimation at all levels.	P1-Customer interest & priorities P2-Lead to more useful products	C1-Communication gaps C2-Customer may become designer of system
KANBAN	Ensures that a particular activity is on time and will provide a product.	P1-Identifies build issues early and Risk reduction	C1-More work for developers
Planning Poker	Group activity that estimates project scope without influencing anyone's ideas.	P1- More accurate estimate	C1- Difficulty in making a consensus
Code Refactoring	Improve internal structure of code for getting better output.	P1- Reduced no. of errors P2- Quality Product	C1- More time is required

Due to fear of failure, small organizations are restricted to follow old methods and roadmaps and so take long time to reach at a better stage and even failed to better good products. But one point that should be forgotten here is that these small

organizations can be more enthusiastic, competitive and productive because they are keen to do much better.

But the question still remains, "what is a way to let them perform better"? The bottom line is that software process improvement should be done to help the business-not for its own sake. This is true for both large organizations and small [1]. Yet small organizations, just like large ones, will have problems with undocumented requirements, the mistakes of inexperienced managers, resource allocation, training, peer reviews, and documenting the product. Despite these challenges, small organizations can be extraordinarily innovative [1]. As Hoffman expresses it, "Don't require process that doesn't make sense." This paper is an attempt to find out a way to use agile methods in small organizations.

After giving introduction in the first section, the remaining part of the paper is organized as follows. In section 2 we present the background and the related work. In section 3 have described the proposed model. Finally we conclude the paper in section 4.

2. LITERATURE REVIEW

Agile methods are a subset of iterative and evolutionary methods and are based on iterative enhancement and opportunistic development processes. In all iterative products, each iteration is a self-contained, mini-project with activities that span requirements analysis, design, implementation, and test [7] [2].

Agile methodologies and principles place emphasis on incremental software development with short iterations, adaptation to changing requirements, close communication, self-organizing teams, and simplicity [8] [5]. A similar survey conducted by Version One [8] additionally reports enhanced ability to manage changing priorities and significantly improved project visibility. For this reason, agile methods are especially suitable for development of information systems with changing and emergent user requirements [6] [3]. A key difference between agile methods and past iterative methods is the length of each iteration. In the past, iterations might have been three or six months long. With agile methods, iteration lengths vary between one to four weeks [5] [2]. In this paper our aim is to look on the use of agile methods in small organizations. Yet the definition of 'small' may ambiguous. In 1998 SEPG conference panel on the CMM and small projects [9], small was defined as "3-4 months in duration with 5 or fewer staff". Brodman and Johnson defined a small organization as fewer than fifty software developers [4].Organization can be termed as small in terms of the kind of projects it takes for development, number of resources it have, number of employees, process and techniques used for the product development etc.

3. PROPOSED MODEL

As we have mentioned earlier, the objective of this paper is to provide a way for using new methods with old one, the proposed model is explained as follows-

3.1. Requirement analysis with Planning Poker

The first phase, requirement analysis is the most crucial phase because the whole project is developed on the basis of results obtained at this phase. Though, SRS (software requirement specification) provides a specification for most important requirements and gives a glance of project's feasibility. But sometimes the results get influenced by senior authorities or managers. Planning poker is the solution for these kinds of situations. Planning poker, a card activity in which every team member choose a card of his choice about the feasibility and effort estimation of the project. Then everyone shows their cards at the end without getting anchored by someone. Then on the basis of consensus, team decides they will go on with project or not. So, by using this activity, better results can be yield from this phase.

3.2. In house planning with whole team development

Before putting the project on the floor, some planning is required for the proper development and progress of the project. Let's have a look on this situation-

"Senior managers-we have decided everything and you have to follow our plans-coding technique…, time limit…..

Team members- If we do this in that way then…the output…..

Senior managers- Meeting is over and you all are supposed to follow this particular plan."

The result from this situation would be in terms of low quality product developed by uninterested people. But if every team member was involved, product would be in better quality state.

3.3. Design with scope for change and updation

Before developing the software its design is drawn in terms of software product entities and their real world relationships. Every design must have a scope for incorporating changing requirements without reflecting any negative impact. And this can be done if the process is based on iterative and incremental approach that let a software designer do changes even after late in software development. With this approach the advantage is, if customer request for any new feature then you don't have to design the whole product from scratch. This will not only save time but also the development cost.

3.4. Coding and Testing with Pair Programming

At this phase developers prepare actual working modules of software and then test them for finding bugs and errors. Normally it is a two way process –coding and testing. But for small organizations where resources are limited, pair programming can serve as a better approach. In pair programming, two team members do work o0n the same module, one is known as driver (who code the module) and the other one is known as navigator (who keep check the code being written). Due to this continuous checking, many defects get caught at the coding stage so result in small testing phase.

3.5. Production with iterative and incremental approach

Production should not be a one shot game. Instead it should be iterative so that we can

show that prepared modules to and can assure him about his product development. This will lead to a greater customer satisfaction and trust on organization.

3.6. Crystal clear communication

Communication is the soul of any organization and bad communication is the devil in any organization. If there is not proper communication present in organization then this may lead to a bad and unhealthy working environment. Crystal Clear approach focuses on communication with no manipulation. Communication exists between the employees, with customers or with the senior members. There should be good and direct communication to customers otherwise developers may interpret customer's requirements wrongly.

3.7. Documentation at every step

Proper documentation works as a good backup if any disaster happens and also works as a learning tool for new employees. This prevents developers to reinvent and reorganize the whole procedure for the common problem. Documentation needs to be systematic and free from grammatical mistakes but it should not be lengthy and complex. If graphical representations in the form of bar charts, Gantt charts and use cases were made during software development then they should also be the part of the documentation. Time duration, details and results of every meeting should be documented.

4. CONCLUSIONS AND FUTURE WORK

By incorporating new methodologies, small organizations can achieve their goal in a better way. This is possible only when senior management provide their support and is ready to take risk. Agile methods defined in previous sections are very much related to the basic steps of software development; the difference is just that they are more realistic. Small organizations are not supposed to or bound to replace their original methods but they should merge these new methods with old ones. So that not only their process will be improved but also their position will be improved.

This proposed model is a beginning to provide a modern yet simple way of software development to small organizations. In future, further research can be done on this model to make it better.

5. REFERENCES

[1] M.C. Paulk (1998), Using the CMM in Small Organizations, CMU, 1998.
[2] A. Agrawal, M. Tripathi, S. Singh (2013), AGILE: Boon for today's software industry-A Review, IJSRP, vol.3, issue12, ISSn-2250-3153.
[3] V. Manhic (2011), A Case Study on Agile Estimating and Planning using Scrum, ISSN 1392 – 1215.

[4] D. L. Johnson and J. G. Brodman (1998), "Applying the CMM to Small Organizations and Small Projects," Proceedings of the 1998 Software Engineering Process Group Conference, Chicago, IL, 9-12 March .

[5] E. Arisholm, member, IEEE, Hans Gallis, Tore Dyba (2007), Member, IEEE Computer Society, and Dag I.K. Sjoberg, Member, IEEE , IEEE Transactions On Software Engineering, Vol. 33, No. 2.

[6] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. A. Visaggio (2007), "Evaluating Performances Of Pair Designing In Industry," Journal Of Systems And Software, Vol. 80, No. 8, Pp. 1317–1327.

[7] L. Williams (2007), A Survey of Agile Development Methodologies.

[8] O. Salo, K. Kolehmainen, P. Kyllönen, J. Löthman, S. Salmijärvi, and P. Abrahamsson (2004), Self-Adaptability of Agile Software Processes: A Case Study on Post-Iteration Workshop, Springer Verlag.

[9] R. Hadden, "How Scalable is CMM Key Practices?" Crosstalk: The Journal of Defense Software Engineering, Vol. 11, No. 4, April 1998, pp. 18-20, 23.