International Journal of Computer and Communication Technology

Volume 3 | Issue 3 Article 15

July 2012

AN ALTERNATIVE APPROACH TO AGILE USING REQUIREMENT ENGINEERING

SATHYA NARAYANAN H
Computer Science and Engineering, Government College of Engineering, Salem, India,
sathya.compsci@gmail.com

MEENAKSHI S
Computer Science and Engineering, Velammal College of Engineering, Madurai, India,
mena.sathya@gmail.com

AN ALTERNATIVE APPROACH TO AGILE USING REQUIREMENT ENGINEERING

[1]SATHYA NARAYANAN H, [2]MEENAKSHI S

[1]Computer Science and Engineering, Government College of Engineering, Salem, India
[2]Computer Science and Engineering, Velammal College of Engineering, Madurai, India
E-mail: sathya.compsci@gmail.com, mena.sathya@gmail.com

Abstract- Many small-scale developers have shifted from a traditional, waterfall method for developing software to lighter weight, agile methods. Though the agile method is quite prevalent among small scale industries, there are several shortcomings in it. In this paper we describe the shortcomings in existing agile methodologies and the methods to overcome some impediments using Requirement Engineering. The best features of Agile and Requirement Engineering is combined and a tool is being created which acts as a repository of data.

***Keywords*: -** Agile, RGM, repository,

I. INTRODUCTION

Software development is a complex process that can be accomplished in many different ways with varying results. Ideally, to be a successful software developer one must be able to build a quality project in a reasonable time frame and budget for the client company. There have been many different approaches engineered for various situations. Some methodologies that have been becoming more common in practice are those known as agile software development. These methods are showing convincing benefits to small to medium scale companies and should not be ignored by global companies developing large-scale systems [5]. This paper analyzes the issues present in traditional models and the existing agile methodologies and suggests way to overcome some of them.

II. LITERATURE REVIEW

There are many software development approaches available today, but only a few are found to be efficient and reliable. Traditionally Waterfall model was used, later we switched over to many conventional models like agile. Before describing the challenges of agile approach, we will describe and contrast the traditional waterfall and agile philosophy.

A. Waterfall Model

The waterfall method is a sequential design process in which each stage is completed before proceeding to the next one. An implementation of this process includes five phases: requirements specification, design, implementation or coding, testing and debugging, and maintenance [11]. It is similar to construction of a building. The impediment that stumbles agile is the customer can see the output only at the end and moreover if a customer is not satisfied with the output the entire process goes waste and the process has to be started again which will be more time consuming. Hence, it is not suitable when a software model has to be developed in a short span of time.

B. AGILE DEVELOPMENT

The more programming methods evolve to suit the environments of software development, the less they resemble the traditional waterfall methods [11]. Agile development is a way of thinking about development. It is not a method in itself, but rather a philosophy [8]. This philosophy is focused on a set of 4 basic values and 12 principles, as stated in the Agile Manifesto [2].

C. AGILE MANIFESTO

The Agile Manifesto is a document written by 17 software developers in the quest to find a lightweight, effective development method. The Agile Manifesto's writers include representatives from many of what are now known as agile methods or practices. Using a collective knowledge of software development and seeing a need to change from heavyweight methods such as waterfall, they wrote the Agile Manifesto. The Agile Manifesto values read as follows [2]:
We are uncovering better ways of developing software by doing it and helping others to do it. Through this work we have come to value:
1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan. That is, while there is value in the items on the right, we value the items on the left more. Principles behind the Agile Manifesto: We follow these principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time-scale.

- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity {(the art of maximizing the amount of work not done)} is essential.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly."

This philosophy influenced the creation of many different agile methods seen today. The most notable one that we will describe is Extreme Programming (XP).

D. Extreme Programming

Extreme programming is one of the most popular of today's agile methods. Focusing its values on communication, simplicity and feedback to improve the speed of development and quality of code, it eliminates the requirements, design and testing phases, and all the extensive documentation as separate phases, but not entirely [5]. Rather, XP suggests integrating all of these steps at the same time in short iterations of about one to two weeks (see Figure 1). To do this, XP suggests that developers keep constant communication with the client-company or customer, thus allowing flexibility that is impossible in rigid waterfall-like processes. The customer is considered a part of the team, and works with the developers creating user stories, developing tests, and prioritizing features to point the project in the right direction [8]. Each iteration consists of a little of each of what is normally seen in traditional phase-based waterfall methods (see Figure 1). Each iteration includes a little planning, analysis, design, coding, and testing, followed by deployment. Each iteration deployment is a point that the customer may choose to release the project for use by the client company. Another important part of XP is the numerous practices that bring the agile values together. Some of the most notable of these practices include pair programming, and frequent testing and refactoring. For example pair programming is where two programmers work together on one workstation collectively writing and reviewing each other's code and guiding each other. Studies have shown this to result in fast production of simple clean code that requires less refactoring later [3]. The figure 1.(a) and 1.(b) compares the waterfall and XP life cycle models.

E. Benefits of Being Agile

Heavily structured plan-driven methods such as waterfall:

- do not adapt easily to changing requirements,
- rely heavily on the quality of initial plans and estimations, which are often unreliable, and lack continuous customer involvement, which can lead to misunderstandings and wasted time.

In the design of software systems, features and functions that seem great to the developers may not always be needed or understood by the customer, and projects may need changes at later stages, when the costs of these changes are the highest. Agile software development aims to remedy the deficiencies of heavyweight methods. With short iterations and regular customer involvement, project changes can be handled at any stage [10]. Also, coding standards, pair programming, and extensive testing seen in most agile methods allow for development of potentially cleaner code, and cleaner code requires less refactoring and documentation.

F. Soft-Structured Agile Framework

One hybrid method is Soundararajan's Soft-Structured Agile Framework. This framework consists of two main parts: the Agile Requirement Generation Model (Agile RGM) and the Development Process. This particular framework's main objective is to accommodate change in both large-scale or small-scale projects [9]. We will describe the two parts and then summarize the benefits to the soft-structured agile framework. The figure 2 shows the phases involved in Agile RGM.

G. Agile RGM

The Agile RGM is a set of well-defined activities that provide a more structured approach compared to XP. As shown in Figure 2, the Agile RGM consists of three phases to help capture requirements: Education, Feature Development and Story Development. All of thesis phases incorporate agile principles and practices such as customer involvement, iterative life cycle, and minimal documentation. The Agile RGM is employed at the beginning of a project that will later follow either an agile or a conventional development process.

1. Education phase

The education phase is essentially a meeting among the development company, the client company and potentially other various customers to help create a better understanding of the project. This is necessary to help build and plan a set of objectives and goals that are to be achieved in later phases and in the finished product. This is different from the usual XP

approach, which employs smaller sessions at the beginning of each iteration.

2. Feature Development phase

At this stage, the customer works with the development team to iteratively identify expected system features. A feature is a small set of functionality that is valuable to the customer. In creating a feature, the development team will give an estimation of how long it will take or if it is possible. Then after a feature is accepted by both customer and developers, the customer will prioritizes each feature according to its business value to the client company. “Business value is something that delivers profit to the organization paying for the software in the form of an Increase in Revenue, an Avoidance of Cost, or an Improvement in Service" [6]. For example, consider development of a website for an e-commerce company. The “Online Payment" feature has a high business value.

3. Story Development phase

Using the features created in the previous phase, developers require additional details before proceeding. In this phase, features are decomposed into stories. A story is a refined user- or customer-expected feature that will be used in the development process. If the development team is made of multiple teams, each team may independently work towards decomposing one or more features into stories. Here, the feature “Online Payment" is decomposed to “As a user, I can pay by credit card".

H. Development Process

In this part of the soft-structured agile framework, the developers may take two alternative approaches depending on the scale of the system (see Figure 3). For small-scale systems, the development team may follow an iterative structure like XP, and make each story an iteration. For large-scale systems, the development team may require a more structured approach and can choose to follow a more conventional waterfall-like approach. For large-scale systems, a waterfall-like approach is usually deployed. First, subsets of stories are transformed into one or more requirements. For example the story “As a user, I can pay by credit card" will be given much more detail and transformed to “the system shall use Advanced Encryption Standard(AES) to encode all credit card information to be transmitted over the internet." The requirements produced from this stage will each be developed in a waterfall-like approach similar to the example seen in Figure 1(a). Although a conventional approach is used here, it still fits within an agile environment as it is guided by the features and stories produced from the Agile RGM process. With the requirements, this stage will also still be able to adapt to changes more easily than conventional waterfall methodologies because requirements are created from stories [9]. Figure 3 shows the spectrum of software approaches.

III. PROPOSED SYSTEM

Though the agile has several advantages, when the customer wants the same project to be developed again, it is quite time consuming process. Since, the developers have to develop it again. So, the proposal which is been given here is a tool is being created where the developers can store the requirements which they had used to develop that product. They can also edit it at times whenever the customer changes his mind. Hence this tool acts as a repository of data. Thus, it does not mean that it involves documentation. It is just a repository of data and acts as a reference for developers for future use.

1.(a)Waterfall Life cycle

1.(b) XP Life cycle

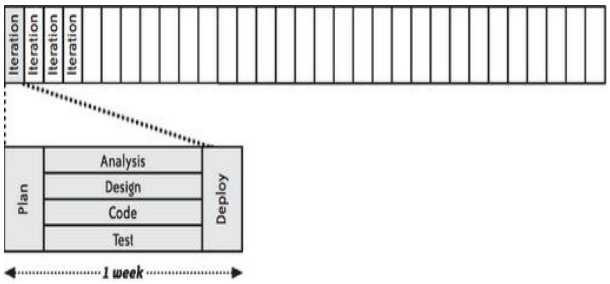

Figure 1: This figure compares the waterfall and XP life cycles [8].

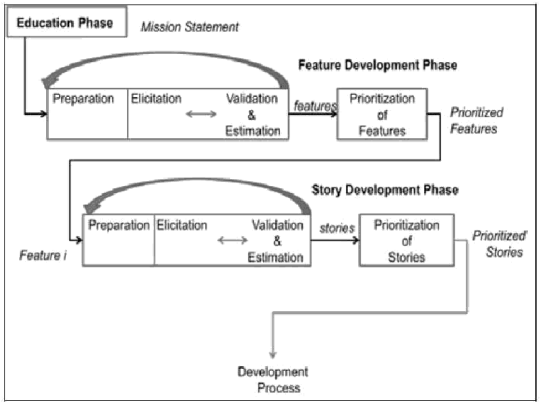

Figure 2: Agile RGM [9].

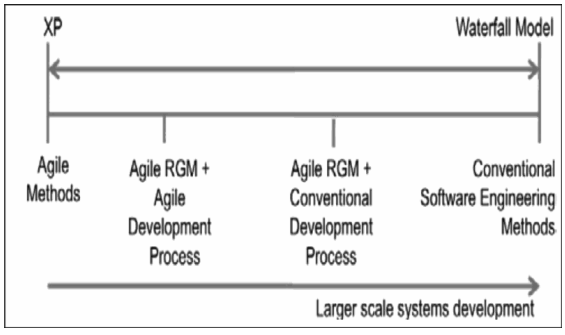

Figure 3: Spectrum of Software Development Approaches [9].

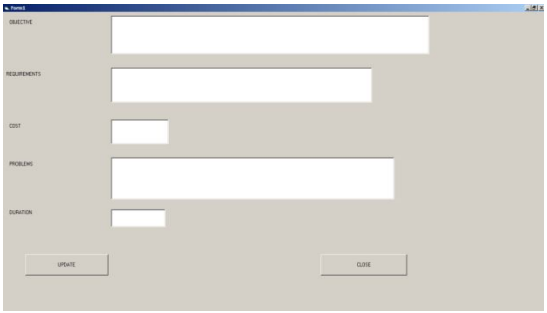

Figure 4: The tool for customer entry.

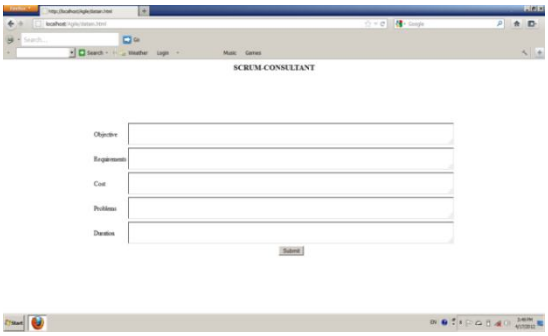

Figure 5: A web application for online storage.

IV. RESULTS AND CONCLUSIONS

The idea which is proposed and a tool which is created can mould the agile software model. It can considerably improve the efficiency of agile prototype. Hence the lack of structure in Agile model is also overcome with the help of this tool. So far an online as well as an offline tool is proposed which will act as repository of data. (Fig 4, Fig 5).

V. FUTURE WORK

Agile RGM can be extended to accommodate large sale systems and MLC systems. Validate the Agile RGM by using it to an organization.

REFERENCES

[1] Barnett, L.,"Best Practices for Large-Scale Agile Development". Agile Journal, pp 39-45. Jul 2006.

[2] Beck, K. et al. (2001): Manifesto for Agile Software Development. Snowbird, Utah, 2001.

[3] A. Cockburn and L. Williams. The costs and benefits of pair programming. In eXtreme Programming and Flexible Processes in Software Engineering XP2000, pages 223-247. Addison-Wesley, 2000.

[4] Conboy, K,Fitzgerald, B, "'Method and Developer Characteristics for Effective Agile Method Tailoring: A Study of XP Expert Opinion", Acm Transactions On Software Engineering And Methodology, pp 214 – 221, 2010.

[5] T. Hildenbrand, M. Geisser, T. Kude, D. Bruch, T. Acker, "Agile Methodologies for Distributed Collaborative Development of Enterprise Applications", In workshop on Engineering Complex Distributed Systems (ECDS), Complex, Intelligent, and Software Intensive Systems (CISIS), pp. 540-545, 2008.

[6] J. Patton, "Ambiguous Business Value Harms Software Products,". IEEE Software, vol. 25, no. 1, pp. 50-51, January/February 2008.

[7] A. Qumer and B. Henderson-Sellers, "A framework to support the evaluation, adoption and improvement of agile methods in practice", Journal of Systems and Software, vol. 81, no. 11, pp. 1899-1919, 2008.

[8] J. Shore and S. Warden " The Art of Agile Development", O'Reilly Media, 2007

[9] S. Soundararajan and J. Arthur, "A soft-structured agile framework for larger scale systems development", In Engineering of Computer Based Systems, ECBS 2009. 16th Annual IEEE International Conference and Workshop on the, pages 187 -195, April 2009.

[10] http://en.wikipedia.org/w/index.php?title=Agile_software_development&oldid=420668857/,2012.Online; accessed 02-July-2012.

[11] http://en.wikipedia.org/w/indexphp?title=Waterfall_model&oldid=420124129/, 2012. Online; accessed 02-July-2012.