Agile Approach to Requirement Engineering: How Agile Processes Can Help in Time-Constrained Requirements Engineering

Monika Singh[1], Ruhi Saxena[2]

[1,2]*Mody University of Science and Technology (MUST), Lakshmangarh, Rajasthan, India*

***Abstract*--Increasing competition and complex business processes are pushing IT firms gain competitive advantage by squeezing delivery timelines. IT firms are thus increasingly seeking ways to become more responsive in their methodology and requirements.**

This paper focus on a modified Requirements Engineering (RE) technique by using Agile methodology which would help in developing quality software products faster and stamping out problems associated with traditional RE technique.

Keywords-- Requirement engineering, Agile model, Waterfall model, DSDM, Scrum, XP.

I. WATERFALL MODEL

The simplest process model is the **waterfall model**, which states that the phases are organized in a linear order. In this model, a project begins with feasibility analysis. Upon successfully demonstrating the feasibility of a project, the requirements analysis and project planning begins. The design starts after the requirements analysis is complete, and coding begins after the design is complete. Once the programming is completed, the code is integrated and testing is done. Upon successful completion of testing, the system is installed. After this, the regular operation and maintenance of the system takes place. It is very simple to understand and use.

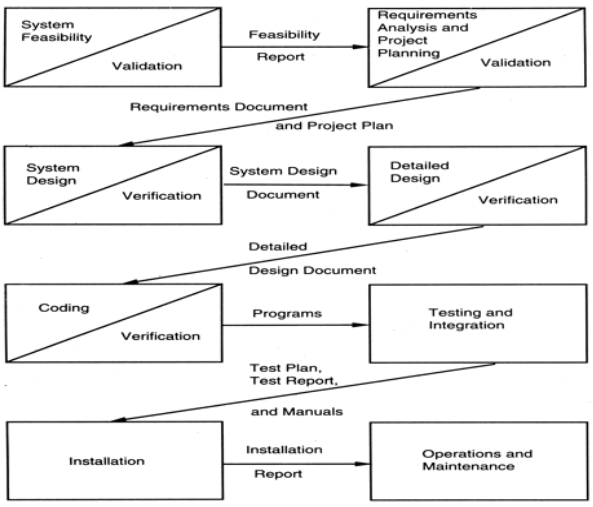

Fig1.1: Waterfall Model

In a waterfall model, each phase must be completed fully before the next phase can begin. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In waterfall model phases do not overlap [1].

II. AGILE MODEL

Agile is an iterative, team-based approach to development. This approach emphasizes the rapid delivery of an application in complete functional components. Rather than creating tasks and schedules, all time is "time-boxed" into phases called "sprints." Each sprint has a defined duration (usually in weeks) with a running list of deliverables, planned one sprint in advance. Deliverables are prioritized by business value as determined by the customer. If all planned work for the sprint cannot be completed, work is reprioritized and the information is used for future sprint planning. As work is completed during each sprint, it is continuously reviewed and evaluated by the customer, who may be considered the most critical member of the Agile team. As a result, Agile relies on a very high level of customer involvement throughout the project [5].

There are various methodologies that are collectively known as agile, as they promote the values of the agile manifesto and they are consistent with the above principles. The most popular ones are:

DSDM (Dynamic System Development Method) is probably the original agile development method. DSDM was around before the term 'agile' was even invented, but is absolutely based on all the principles we've come to know as agile. DSDM seems to be much less well-known outside of the UK [9].

When to use waterfall model	When to use Agile model
--Requirements are very well known, clear and fixed. --Product definition is stable. --Technology is understood. --There are no ambiguous requirements --Ample resources with required expertise are available freely --The project is short.	--When new changes are needed to be implemented. New changes can be implemented at very little cost because of the frequency of new increments that are produced. --To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it. --Very limited planning is required to get started with the project. --Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.

Scrum is also an agile development method, which concentrates particularly on how to manage tasks within a team-based development environment. Scrum is the most popular and widely adopted agile method because it is relatively simple to implement and addresses many of the management issues that have plagued IT development teams for decades [4].

XP (Extreme Programming) is a more radical agile methodology, focusing more on the software engineering process and addressing the analysis, development and test phases with novel approaches that make a substantial difference to the quality of the end product [6].

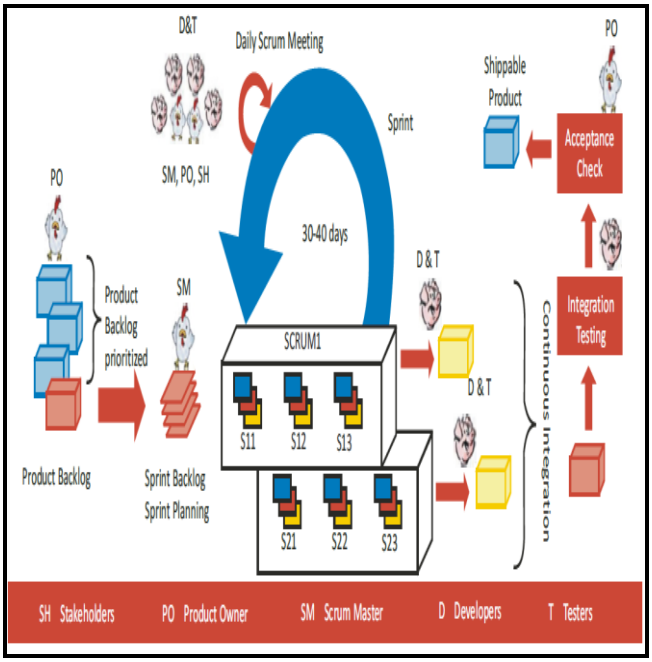

Fig 2.1: Scrum --- Agile Development Methodology

III. OVERVIEW OF RE USING AGILE

Agile software processes provide very efficient RE in that they narrow the gap between customer's needs and the software product being developed [3]. The main aspects of Agile are:

Onsite and Co-located Customer: This principle suggests that at least one person from the customer's group physically joins the team of developers for the predominant part of the project. He/she selects and sorts the requirements to implement and set priorities by doing so. Development is therefore driven by business interests that help define what is useful to the customer.

Table1: Factors to consider when Extracting Requirements from Agile

More Detail	Less Detail
New System	Simple to Moderate Addition to Existing System
Offshore or Distributed Teams	Co-located Team
Large Complex Solution	Small Simple Solution
A Newly Formed Team	Team that has been Working together for a while

Frequent Short Releases: Frequently releasing pieces of software product provides the ability to deliver faster and expected results to the clients. Also, shorter delivery cycle means sharper crystallization of client's needs. During the incipient stages of the project, clients are usually clueless on the system's requirements. Short releases help them form a mental map of how the future system would look like. Also, each release takes into account the experiences of previous releases, thus leading to a continuous improvement in the subsequent releases.

Facilitating Extraction: High level (abstract) and detailed level (concrete) end-user requirements can be extracted using Agile. User stories or use-cases form the high level requirement. Usually clients write user-stories because they know best about the desired functionality. Each user-story is written in a language that enables the customer to prioritize such stories according to their business value. One of the first things one wants to decide on is how much detail to include in the requirements artifacts. Table 1 lists factors that may be considered to decide what level of requirements should be extracted using Agile.

Acceptance Test Criteria During RE: Clients write acceptance tests that are transformed into unit tests by the developers before any other development activity. The system has to pass all acceptance tests on every release. Acceptance test descriptions are also very suitable as a contractual base. The above aspects allow for a lot of knowledge to be transferred from customer to developer in a very short time with few bureaucratic overheads [3] [7].

IV. AGILE VS WATERFALL MODEL

Agile methods grew out of the real-life project experiences of leading software professionals who had experienced the challenges and limitations of traditional Waterfall model on various projects. By delivering working, tested, deployable software on an incremental basis, Agile methodology delivers increased value, visibility and adaptability much earlier in the lifecycle, significantly reducing project risk. Figure 1 depicts the difference between traditional model and Agile model. In traditional Waterfall project, we release version 1 (i.e., RL1) to meet the business target as shown in Figure 4.1. But with time the value of that software/ product decreases. This is expected because of the changing business environment. Thus, the software is not in sync with the pace of change as every new release in Waterfall model involves planning, design, development, testing and deployment -- a rather time consuming process[2].

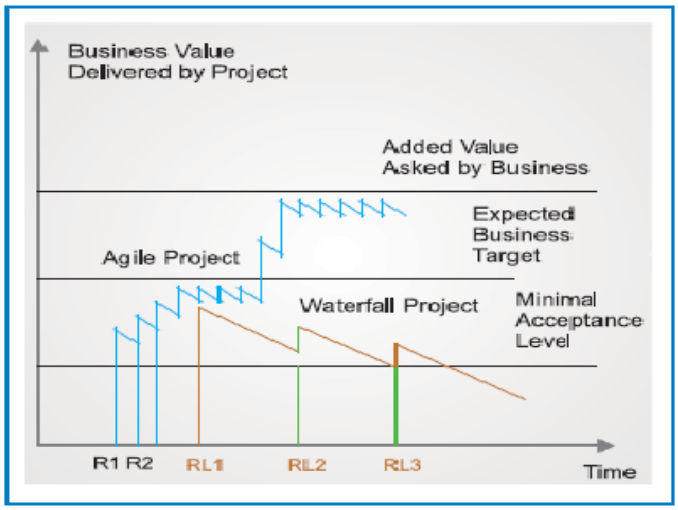

Fig 4.1: Traditional Model vs. Agile Model

Above all, second and subsequent release(s) most often fall short of meeting the business goal or in other words, the gap between business need and value that the software delivers is not fully recovered. In most cases, the software fails to deliver to the business' requirements [8].

The value delivered by Agile project is therefore expected to be much higher than the value delivered by Waterfall project over this whole period of time. The benefits of Agile methodology can be identified as:

*Higher Visibility on Project Progress***:** In Agile methodology, measuring and evaluating the status based on the working and testing software provides much more accurate visibility into the actual progress of projects. In traditional projects, the visibility is high during the requirements gathering phase but the customer remains almost in the dark post the requirements gathering phase till the implementation.

Higher Adaptability to Changes: In the course of building business software requirements, changes are the norm. A late change in requirement is often a competitive advantage for the customer. As one cannot avoid it, the question is what cans one do about it. Waterfall method is a predictable methodology and using a predictable process in an unpredictable environment almost always leads to trouble.
RE with Agile enables the customer to include the latest changes that need to be incorporated in the requirements that add to the market advantage of the customer. As a result of the planning and feedback loop, teams are able to continuously align the delivered software with desired business needs, easily adapting to changing requirements throughout the process.

Lower Overall Risk: Agile methodology maintains a focus on the rapid delivery of business benefit. As a result of this

focus and its associated benefits, organizations are capable of significantly reducing the overall risk associated with software development.

*Effective Requirement Elicitation by Feedback Method***:** It is often difficult for customers to clearly articulate what they need, in the first go. At times they are far from having a complete, concise and unambiguous model in mind that has to be transferred to the developer. Seeing and feeling the running system, in iterations in Agile, enables the customers to provide requirements in iterative feedbacks.

V. AGILE APPROACH TO REQUIREMENT ENGINEERING

Agile focuses very strongly on customer interaction. However, most of this interaction is done by prototyping. It is sometimes not possible to have all the requirements from just one person. So there is still the need to consider elicitation as a process to gather requirements from viewpoints of different business stakeholders.

The project using Agile should also learn from various interviewing techniques that have been in use in the RE of Waterfall model. The importance of context-free questions, open questions and meta-questions is also valid in an Agile environment. If there are conflicts between stakeholder requirements, the use of Joint Application Development (JAD) can help promote the use of a professional facilitator who can help resolve conflicts [3].

Agile methods rely on the use of validation (testing), not only as a way of achieving quality by getting rid of errors, but also as a way of identifying requirements. Agile processes do not address aspects of verification and they would benefit if they additionally include verification together with validation. Simple tools to check early requirements descriptions associated with effective management practices for applying inspections can improve the quality of Agile processes [11].

Sometimes, software systems fail to deliver the intended objective due to the lack of consideration of non-functional requirements (NFRs). As Agile considers non-functional requirements at implementation level, it may cause some problems. There is a need for Agile methods to include techniques that make it possible to identify non-functional requirements early and describe them in a manner to indicate that an analysis may happen before implementation. Since Agile develops software in small releases and uses refactoring as a frequent practice, change is intrinsic to Agile methods. Agile methods should not rely only on configuration management practices but also adapt the requirements management practices to provide traceability from customers' needs to actual implementation [10].

In the first release of the project using Agile, environment setup should not be mixed with the actual product development. Even though the first release is small in Agile, environment set up should be made ready before the release. Also, projects using Agile should have similar detailed project planning for each release like that of projects using Waterfall model.

CONCLUSION

Agile effectively manages requirements and focuses on generating value for the customer by reducing irrelevant elements. Therefore, the involvement of the customer is of paramount importance to achieve this goal. Also, as customers give continuous feedback throughout the lifecycle of the project, any late changes become easy to incorporate. Agile is a valuable approach to software development for some types of projects, but their limits are not well defined yet. Since these methods are new, the subject is still evolving and many techniques are under investigation.

REFERENCES

[1] Pankaj Jalote,"An Integrated Approach to Software Engineering", second edition, 2001.

[2] http://www.infosys.com/infosys-labs/publications/infosyslabs-briefings/Pages/agile- methodology.aspx

[3] http://www.infosys.com/infosys-labs/publications/Documents/agile-requirements-engineering.pdf

[4] http://www.allaboutagile.com/what-is-agile-10-key-principles/#sthash.4OEnczZs.dpuf

[5] http://www.seguetech.com/blog/2013/07/05/waterfall-vs-agile-right-development-methodology

[6] Beck, K, Extreme Programming Explained: Embrace Change, 2001, Addison Wesley.

[7] Cockburn, A., Agile Software Development, 2002, Pearson Education.

[8] Ambler S (2002) When does (n't) Agile modeling make sense? Accessed on December 5, 2004, http://www.agilemodeling.com/essays/whenDoesAMWork.htm.

[9] Stapleton J (1995) DSDM −Dynamic system development method. Addison-Wesley, UK.

[10] Tomayko JE (2002) Engineering of unstable requirements using Agile methods. In: Proceedings of International Conference on Time-Constrained Requirements Engineering, Essen, Germany, 9-13 September.

[11] Paetsch F, Eberlein A, Maurer F (2003) Requirements engineering and Agile software development. In Proceedings of 8th International Workshop on Enterprise Security, Linz, Austria, 9-11 June