International Journal of Engineering Research and Development
e-ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 8, Issue 3 (August 2013), PP. 20-25

Applying Agile Software Development Methodology in a Dynamic Business Environment

Fisnik Dalipi, Eip Rufati, Florim Idrizi
Department of IT, Faculty of Math-Natural Sciences, Tetovo State University, Macedonia

Abstract:- In a fast changing business environment, companies that can innovate better and faster, and respond quickly to customer's requirements, will win. One of the most important industries of knowledge-based economy, by all means, is the software industry. In order to achieve business success and to innovate in the market, almost every large organization relies on software. The emergence of Web 2.0 and web-based services have changed radically the way of conducting business as well as the way of building applications. In a web service environment, software applications are becoming services that can be easily linked with other software components, and can provide faster and more productive communication between businesses and customers. The software development companies now need to release products and services more frequently, involve the user participation even as co-developers, and improve the product continuously by receiving real time feedback from the users. In such a dynamic and networked business environment, many companies are adopting Agile software development. There are many reasons for this: agile methods require continuous dialog between business and IT. Fast respond to change and meeting customer satisfactions through early and continuous delivery of the software is also the main goal of agile methodology. The purpose of this work is to find out what are the business benefits of agile methods, and how suitable or valuable is agile development methodology in developing dynamic applications in Web 2.0 service environment. This challenge will be approached as followed: define the Software Development Life Cycle and illustrate some development methodologies, define the Agile Manifesto to clarify the concept of agile software development and Scrum in particular, and discover and analyze the business benefits, agile change and risk management of agile development methods.

Keywords:- agile, web 2.0, web services, agile manifesto, Scrum, agile change, risk management.

I. INTRODUCTION

Nowadays, application software is playing an indispensable and expanding role in business and can help companies to strengthen their competitive positions in a rapidly changing marketplace. it represents the support infrastructure that help companies change swiftly when adapting to new business environments.
Software development is a multistep process and therefore many reasons must be taken into consideration when developing sustainable and successful software to meet business opportunities. This process is called Systems Development Life Cycle (SDLC). The SDLC consists of the following stages: feasibility study, systems analysis, design, development, implementation, and maintenance.

- **Feasibility** study describes the way of addressing business opportunities and goals. This phase involves also establishing a project management plan and determining project goals.
- **Systems analysis** involves analysing the information needs of end-users. In order to meet the business requirements, IT experts and knowledge people brainstorm which action and tasks to take into considerations to make the system reliable and successful.
- **Design** refers to explicitly describing the desired features that the new system might have, and developing specifications for all resources (software, hardware, users) to produce desired results.
- **Development** considers taking all the designed documents from the design phase and executing or transforming them into a new physical system.
- **Implementation** means installing the system into production. If new system is purchased, following actions are required: converting data from the old to new system, and training employees to use the new system.
- **Maintenance** is the final phase of the cycle. This phase involves monitoring the system, implementing changes and a correction, fixing eventual bugs and upgrade the system to ensure that meets the business goals.

In developing the software, teams of computer professionals, business analysts, testers and users must work together to design qualitative software that will enhance the business success. For this reason, developers have created a number of development methodologies including: Agile development, Waterfall, Spiral, Rapid Application Development (RAD), etc.

II. PRINCIPLES OF AGILE DEVELOPMENT

In a dynamic and networked business environment, critical factor for software project teams to gain competitive advantage, to respond to change, and to consider the collaboration as main aspect is to remain flexible, feasible and agile. These teams should deliver useful software components which will satisfy customers by meeting their requirements even late in development and that the process is improved continuously. By using agile software development methods, this can become reality.

Agile project management methods have been widely accepted as the best way forward for managing these kinds of rapidly changing projects, while getting predictable, sustainable results [1].

The main difference between agile and non-agile traditional methods is that agile methods:

-produce less documentation;

-deliver software in short amounts of time, called iterations, which usually last one to four weeks;

-development process is based on face-to-face communication;

-deliver business value as the project starts;

-Agile teams are located in close proximity, very often in a single office.

The three most important success factors are culture, people, and communication. Agile Methods need cultural support, otherwise they will not succeed. Competent team members are crucial. Agile Methods use fewer, but more competent, people. Physically co-located teams and pair programming support rapid communication. Close interaction with the customer and frequent customer feedback are critical success factors [2].

Agile Methodolgies are based at the Agile Manifesto, which was created when seventeen experts of Agile Alliance (www.agilealliance.com) met to find solutions to rigorous, documentation driven development. t consists of four values and twelve principles. The value statements are:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

According to them, the highlighted items on the left are more valuable. The next section in the manifesto is the principles. They create an environment of innovation and creativity by specifying a simple set of rules (practices are rules in this case) that generate complex behaviour. The most widely used methods based on agile philosophy are: Scrum, XP, Crystal, Dynamic Systems, Development Method (DSDM), Adaptive Software Development (ASD), Feature Driven Development (FDD).

A. Overview of the Scrum philosophy

One of the most popular agile methods is Scrum. It's one of the main working framework used by hundreds of companies among which microsoft, yahoo!, and google. scrum as iterative development method has been developed in the 80’s and 90’s. according to[3], projects using small, cross-functional teams historically produce the best results. these teams are like the scrum formation in rugby. scrum has been practiced mostly by manifesto authors ken schwaber, jeff sutterland and mike beedle. scrum as a management process for managing product development is not used just in software development, but can be adapted for use for different types of projects. according to ken schwaber [4], scrum is used for complex work in which it is impossible to predict everything that will occur. accordingly, scrum simply offers a framework and set of practices that keep everything visible. This allows Scrum’s keep the project moving toward the desired goals.

Scrum helps in making things transparent by encouraging communication, embracing change and improving productivity. In order to work as Scrum suggests, the entire team must posses solid knowledge in analysis design, implementation, testing and documentation.

The primary goal of Scrum is to organize teams and deliver software that provides the highest business value. It concentrates on prioritizing work based on business value, amending the beneficiary of what is delivered, and enhancing revenue.

Scrum as a process is incremental. Each increment is called sprint, where each sprint lasts for four weeks. Prior to the sprint, there is a sprint planning meeting where the user determines what features should be developed and followed in the next sprint. During the sprint, the team meets daily at the same time at a short meeting called a scrum or the daily stand-up meeting.

Nowadays, many cutting edge and leading Silicon Valley companies are practicing Scrum method in applications development to reduce the time to the market and providing a stable platform for the web that should be able to take in new functionalities with as little changes as possible.

III. BUSINESS BENEFITS OF AGILE METHODS

In Agile, business users are actively involved in development efforts and there is continual focus on sustainable business value. Business and technical teams act as cohesive (see Fig. 1). Teams are comprised of business and technical team members and they work toward common goal or shared vision. Agile teams are able to deliver product to market within 6 months after start of development. The early delivery of the product has an additional benefit because the time value of that investment is increased. Cost or revenue savings can be figure out sooner and therefore greater ROI (Return On Investment) is available.

Agile methodologies emphasize delivering value early and continuously so that customers get the highest return on their investment. By including the value provided by our product features and the user roles that are interested in attaining that value, we are able to more easily prioritize to optimize this return (Sterling, 2008).

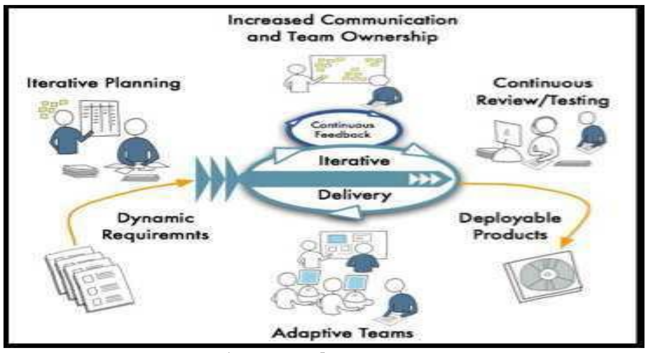

Fig. 1 Agile in action

business, managers, team members and stakeholders. We can identify two mechanisms, which can enable this: participation and confidence.

Participation: Many stakeholders are willing to be part of something that is progressing, to know what is going on with the project. Scrum does not specify any specific result or contribution to them. Rather, they might be looking to learn, making efforts to increase their influence, or just to be informed about the work being performed.

Agile approaches enable improved participation for the stakeholders by using: visibility and regularity of delivery. By visibility, we mean that stakeholders are allowed to participate to the demos, to analyze the burndown chart or to ask potential questions of the team.

Confidence: Predictable and accurate results can make some stakeholders to be concerned of. So we may ask the question as follows: What can push the stakeholder to trust the work of the team?

We can identify some practices that can be followed in order to build this trust. First, the effort of the work must be time-boxed in a consistent way and the same amount of time must be determined to each iteration. Secondly, there must be an adequate planning, estimation and tracking of tasks. Thirdly, the iterations used by the team must be short that the stakeholder does not get confused and loose control of the progress. Finally, the team should not allow any interruptions.

B. Continuous customer feedback

In today's and tomorrow's world, where software decisions increasingly drive system outcomes, this separation of concerns is increasingly harmful. Customers must be more closely related to the development process [6]. Agile methods give opportunities to customers for active involvement during the development process, for control and feedback through various processes, such as: backlog, demos, through engagement in the team’s conversation about the performance of the work and through user stories.

As we see that, the customers must have their contribution in what the software will look like, user stories are highly practiced during the work. They represent the simplest way to record the user requirements throughout the project. User story is a statement describing what user wants to do with a specific characteristic of the software; all this from a user’s side. Typically, these stories do not include description about technological aspects of the project and are compiled in an understandable business language. While handling with user stories, before the team members translate the stories into a code, the team should clearly understand the message of the story, or, what the user idea is for, and make sure no ambiguity takes place.

C. Developing products faster

Let us analyze two of the twelve principles of Agile Manifesto:

“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”

"Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale." [7].

Considering the fast changing business environment and reduced time to market, the developer team is able to deliver software faster in every development iteration, that software is called "valuable software". Since there is interaction and control, the alignment with the customer is much better allowing the customer's idea permanently to evolve.

According to [8], valuable software is software that supports the goal of the project as specified in the business case. Deliver valuable software for the customer is then translated to deliver software that maximizes the added value to the business case. Scrum iteration sessions are time-boxed in a consistent way, which ensures there are frequent releases of software to the customers. If the development work is quite complex, then one time-box will be not enough, so another time-box is needed. After each time-box, a software release is provided to the customers.

The interaction with the customer takes place to decide what is necessary. In addition, they can use the software release and can give feedback. Given this, agile methods enable requirements to change during the process, customers have flexibility to change their view on particular features of the product the team is yet to implement. By collaborating with customers and giving them demo/working versions, the teams can develop software, which will maximize the business value and be more aligned with the businesses strategy.

D. Managing the Agile change

In order to remain competitive on the business landscape, companies must control, adapt and manage change. Some authors link the change with the state of uncertainty. Among them, [9] defines agile environments as those that exhibit internal and/or external uncertainty, may require some unique expertise, and posses a high level of urgency.

Agile Environnements = [Uncertainty + Unique Expertise] x Speed

The change or uncertainty can come from two different directions, i.e. internal and external.

He identifies two types of **uncertainties**:

- internal – controlled by project manager; includes the project scope, schedule and cost of project, training of developers.
- external – outside of project manager's control; involves the industry's business environment, competition, business strategy decisions, advancement of new technology

Software projects base their roots in creativity and require the use of **unique expertise**. These individuals and groups with deep knowledge that drives the ideas and projects, represent the heart of every innovative company – the company's unique expertise.

The level of urgency or **speed** is an organic part of agile development. The basic principle of agile is to release software projects faster in every development cycle. In a time of changing customer requirements and dynamic business environments with increased use of Internet as an essential business medium, the amount of influence that external uncertainty has on internal uncertainties is becoming large. In the figure below, we can see that "changing customer requirements" is the first driver of such external uncertainties.

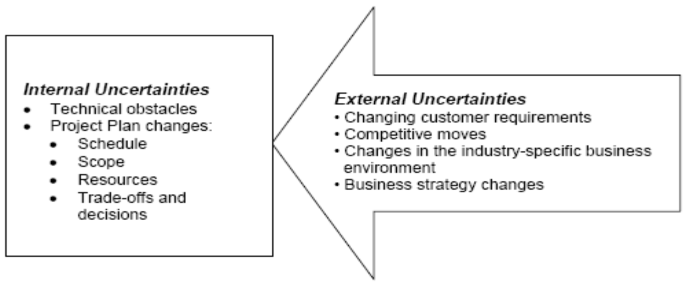
Internal Uncertainties
• Technical obstacles
• Project Plan changes:
• Schedule
• Scope
• Resources
• Trade-offs and decisions
External Uncertainties
• Changing customer requirements
• Competitive moves
• Changes in the industry-specific business environment
• Business strategy changes

Fig. 2 Project uncertainty is made up of both internal and external components

But how we will react to those changes while applying Scrum? At the stage of changing customer requirements, Scrum uses time-boxing to control schedule and is encouraging the customer to prioritize all product functionalities that will be delivered.

In Scrum, the person who prioritizes the tasks is called Product owner. Accordingly, [10] offers a flexible approach to changing customer requirements. He suggests building a stack (product backlog) of prioritized and estimated requirements, which are necessary to be implemented, as shown in Figure 3.

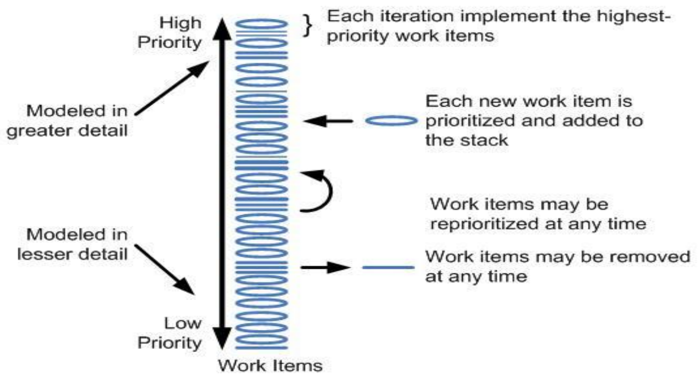
High Priority
Each iteration implement the highest-priority work items
Modeled in greater detail
Each new work item is prioritized and added to the stack
Work items may be reprioritized at any time
Modeled in lesser detail
Work items may be removed at any time
Low Priority
Work Items

Fig. 3 Agile requirement change management

The developers take the highest priority work items from the top of the stack (product backlog) which they can implement within the current Sprint. Any work item with low priority is modelled in lesser detail.
Scrum suggests that you freeze the requirements for the current iteration to provide a level of stability for the developers. If you do this then any change to a requirement you're currently implementing should be treated as just another new requirement [10].

It is important to remark that whatever the size of the time-box, the customer selects the most important requirements to do and the developers get as much of that done as they can in the allotted time [11]. By giving the product owner flexibility and control over changing, prioritizing and adding requirements, agile change management is becoming more and more adaptive to changing business and technological needs.

E. Continuous risk management

Every project as well as software projects are followed by uncertainties and are subject to risk. Agile development methods are business driven and risk driven. Because agile methods are incremental, this project risk should be the main concern of the team when choosing how to deal with high risk areas and how to proceed with the process without loosing time, money and market share. Though many books for project management offer several practices on dealing with risk in a certain projects and environments, most of these practices are very far from agile. So, the dilemma that arises here is: how to adapt these traditional risk management approaches meant for different projects into agile iterations that can last seven to 30 days?
According to [9], we identifie four parts to the risk management process:

- Identify potential risks,
- Assess the risks,
- Make plans to address the risks,
- Reassess the risks throughout the project.

These techniques are indispensable to risk management; they apply to traditional and agile projects. Many factors/characteristics make risk management process agile, such as, close interaction with the customer, collocation, and fast feedback on customer requirements. This is big advantage in making risk management agile.

Let us analyze the four steps of the processes listed above. Firstly, let's assume that we run a traditional project. Here, the team after identifying and brainstorming the potential risks, it can follow a long stage of assessments and analysis to define and prioritize the risks quantitatively, based on risk severity. As final step of the process is reassessing and reviewing the risks by keeping the top-priority risks visible to the team and management. Secondly, by running agile projects, the team does not encounter long hours of scrutinized quantitative analysis. As teams are collocated, and receive strong feedback, they use their knowledge to prioritize risks by using their personal judgments in determining the complexity, and severity of each risk. Even if their perception fails to be true, the short iteration, daily team meetings and the customer collaboration enables them to reassess and reprioritize the risks in the upcoming iteration.

As we can see, for agile risk management, we complete these four steps of the risk management process in much less time and with less effort. When we refer to severity, we mean the level of loss or the likelihood that the risk will happen and affect the project. As a result, the team and management have to develop proper action-plans to reduce or mitigate the likelihood of risk occurrence. Crafting a mitigation involves understanding the root cause of the risk, brainstorming potential ways to prevent the risk, and then breaking them down into WBS (Work Breakdown Structure) elements and individual tasks that can be added to the detailed project plan [9].

Taking into account the incremental nature of agile methods, they can identify risks in the current iteration and find solutions to mitigate them in the upcoming iteration. Therefore, agile can identify and recognize risks before they can influence the project. One of the challenges that developers and management may face in mitigating the risk is as follows: what if the items with low priority, which are not prioritized by the business, can contain risk.

Considering the priorities assigned by the stakeholders, the team selects the items with higher priority in the Product Backlog. But, if the remaining items contain any risk, then those items has to be moved up the stack of prioritized feature list and to be processed as soon as possible, as illustrated in Figure 4.

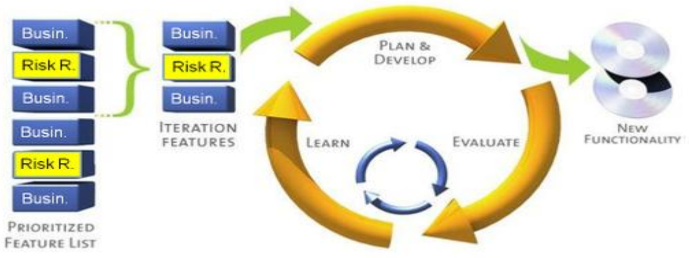

Fig. 4 Mitigating risk by balancing with business requirements

This can help teams to balance the risk and the business requirements. Despite the occurrence of not prioritized risk items in Product Backlog, the business side remains actively involved throughout the development process.

IV. CONCLUSIONS

Each day more and more enterprises are adopting agile development methods because they are characterized by rapid release of products, reduced documentation, face-to-face communication and flexibility when handling with risk issues.

We have seen that agile helps align IT with the business because developer teams perform only the priorities issued by the business. The collaboration with them makes the project more traceable, visible, and business needs and expectations are managed in more transparent way. Agile development supports and enables strong alignment between IT and business, as the business stakeholders are the focal point in the process. The short time between iterations and the faster time to market pushes software organizations to move toward web-based software development. Here, agile development can be considered as the most appropriate methodology to be used in creating such web-based services.

REFERENCES

[1]. CHURCHVILLE, David (2008), Agile Thinking: Leading successful software projects and teams Publisher: Extreme Planner Software
[2]. COHEN, David. LINDVALL, Mikael. COSTA, Patricia (2004) An Introduction to Agile Methods. URL:http://www.cs.umd.edu/~mvz/cmsc435-s05/pdf/agile-paper.pdf
[3]. TAKEUCHI, Hirotaka. NONAKA, Ikujiro (1986) The New New Product Development Game Publisher: Harvard Business Review
[4]. SCHWABER, Ken (2004) Agile Project Management with Scrum Publisher: Microsoft Press
[5]. STERLING, Chris (2008) Focus on Value: How to create value-driven user stories. URL: http://www.scrumalliance.org/articles/89-focus-on-value
[6]. TURNER, Richard. BOEHM, Barry (2003) People Factors in Software Management: Lessons From Comparing Agile and Plan-Driven Methods URL:http://www.stsc.hill.af.mil/crosstalk/2003/12/0312turner.html
[7]. COCKBURN, Alistair (2002), Agile Software Development Publisher: Addison Wesley
[8]. Van Amerongen , Robbrecht (2007). Agile Software development, the prprinciples. **URL: http://technology.amis.nl/blog/?p=1971**.
[9]. CHIN, Gary (2004), Agile Project Management: How to succeed in the face of changing project requirements Publisher: AMACOM
[10]. AMBLER, Scott (2007) Agile Requirements Best Practices URL: http://www.agilemodeling.com/essays/agileRequirementsBestPractices.htm
[11]. REED, Karl. DAMIANI, Ernesto. GIANINI, Gabrielle. COLOMBO, Alberto (2004) Agile management of uncertain requirements via generalizations: a case study.