Effectiveness and Efficiency of Software Development Methodologies

Submitted To: Software Development Product Managers

Submitted By: Rusheel Shah

Date of Submission: December 6, 2015

Abstract

This project will focus on the field of software development in the current corporate world. The purpose of this project is to determine the most efficient and effective software development methodology used in the corporate world. Software development teams use these methodologies to clearly and concisely organize the work that must be completed for a software development. Efficiency and effectiveness will be evaluated using three methods. First, the validity of a given software development methodology will be tested to determine what consists of a software development methodology. Next, the method for evaluating efficiency for a given software development methodology will be determined. Lastly, three different software development methodologies will be evaluated and compared. These three methodologies are the Agile, Waterfall, and Spiral development processes. The results of these findings will enable all software development teams to use the most efficient software development methodology. In today's hyper-competitive software market, emerging and established companies alike will need this information to build a framework of success.

Table of Contents

Project Description

This project proposal focuses on the field of software development, specifically the software development process in today's enterprising world. The goal of this project is to ultimately determine the most efficient and effective software development methodology and to present these findings to companies to improve their product efficiency. A software development methodology is a process that is used by software development teams to clearly and concisely organize the work that has been completed and the work that must be completed for a given software development project. The results of this project will be very beneficial to companies that are already established, like Google or Microsoft, and companies that are just emerging, like small businesses and startups. This information will result in established companies re-evaluating the software development methodology or methodologies they use and determining if there is a more efficient and effective methodology out there so that they can get their products out quicker and at a higher quality while also reducing costs. This will also result in the emerging companies being able to pick one of the methodologies analyzed in the project to provide a framework for their new company. The work plan for conducting this research is divided up into three parts. First, I will analyze what exactly makes up a software development methodology and the need for it. Is there evidence of teams being more efficient when a methodology is in place as opposed to when there is no specific methodology? Next, I will determine the method of evaluating efficiency for a given software development methodology. How do we know if one methodology is more effective than another? This question is especially important for emerging companies like startups because startups are faced with the most competition, tightest schedules, and most evolving markets out of any type of company. This requires startups to have effective and efficient features embedded in the company right at the point of the company's creation. Lastly, I will compare and contrast the three most prominent software development methodologies. These three methodologies were found through the literature review to be the most common methodologies in the current corporate world.

Literature Review

The need for effective methodology in software development

Defining a software development methodology

A software development methodology (or a software process model as defined by Boehm) servs as the main framework for the workflow of the software development process for a product until its release (Boehm, 1988). Software development methodologies first ask if the current state of development is complete. If the current state is complete, then the next state should be determined and planned. This process is acted upon recursively until the release of the software (Boehm, 1988). In short, the three main functions of a software development methodology are to organize, plan, and direct. There are different stages that are set to complete these functions including planning, analyzing, designing, implementing, and maintaining (Verma, Bansal, and Pandey, 2014). Without these methodologies, there is no

organized route from creation to release for a given product. The methodologies give some sort of template for not only the overall software design but also make the tasks within the overall design easier to manage (Boehm, 1988). The methodologies in place today are a result of a course of evolution within the software development field. For example, one of the first "methodologies" was the code-and-fix solution (Boehm, 1988). This model followed the path of writing code and then fixing those problems. This became problematic because the product's requirements, maintenance, and overall design was only taken into account after the code was written. To put into perspective, this is like trying to build a car by starting with the wheels and headlights; they are necessary components but there is no framework. Obviously this model was not considered the most effective methodology (Boehm, 1988). This methodology was quickly made obsolete by the methodologies that will be discussed in the following sections. The question then arises on how to determine the effectiveness of the various software methodologies.

Determining effectiveness of software development methodologies

In Verma, Bansal, and Pandey's publication, some methodologies are compared to determine the best development methodology. To determine effectiveness, the authors compare various parameters relating to each of the methodologies. These parameters, among others, include cost, complexity of implementation of the methodology, and size of project that would follow this methodology (Verma, Bansal, and Pandey, 2014). Based on the qualitative data associated with these parameters and responses to the parameters using the Likert scale, the effectiveness of each methodology were determined. This procedure is comparable to the one that will be used in this project because the parameters in this project are roughly similar to those used in this publication. Based on these parameters, Verma, Bansal, and Pandey were able to create a blueprint on how to select one of three methodologies in certain situations (2014). An opposing view to this method is Schaeffer's analysis on the efficiency of software processes. Schaeffer believes that the main indicator of efficient software development methodology should be the cost (2013). This perspective on efficiency is vastly different to Verma, Bansal, and Pandey's perspective. Schaeffer argues that regardless of the quality of software, if it is good or bad, a company can go bankrupt either way if the costs are too high. He adds that the cost of doing business should dissociate the cost of development (2013). Determining effectiveness of these methodologies is very important for all companies but crucial for startups (Paternoster et al., 2014). Startups are faced with the most competition, tightest schedules, and most evolving markets out of any type of company. Startups are requires to to have effective and efficient features embedded in the company because of these obstacles. As a result, there is much research that goes into determining courses of action that will lead to success and avoiding those that will bring failure (2014).

Evaluating various Software Development Methodologies

Agile

The first software development methodology that will be evaluated is the agile development process. The agile process's biggest advantage is that it is flexible to changes in the

requirements or specifics of a given product (Williams and Cockburn 2003). Another advantage of this methodology is that there is always working code that can be implemented and tested on its own so that the developers can see what they currently have and how it works so that it can be either used, changed, or deleted (Highsmith and Cockburn 2001). With other methodologies, where the whole project is planned at the beginning, developers can only speculate as to what the code will look like or do, whereas with the agile methodology there is clear evidence of what the code does throughout the design phase (2001). In Yu and Petter's publication, they argue that a hidden advantage of the agile methodology that has been overlooked by most researchers and companies is the cognitive processes that are shared by a team that follows agile (2014). By using the shared mental models theory, the agile approach clearly encourages team interaction and holistic understanding of each task by the whole team (2014). Given these advantages there are a couple clear disadvantages when using the agile methodology. First, since agile is a fairly new methodology there is not much research that has been done on its effectiveness versus other methodologies. Critics of agile even argue that it is a step back from previous methodologies because the creators of agile acted on their frustrations with the old methodologies (Yu and Petter, 2014). Another disadvantage with the agile methodology is that although the communication within the team is high and there is a shared understanding, there are cases where the increase in meetings become redundant and actually counter-productive (Bindrees et al., 2015). Developers have found that they lose a lot of time to meetings that become redundant and repetitive when the time could instead be used to code and design (2015).

Waterfall

The waterfall software methodology is much older than agile and was commonplace in companies around 1970 before the creation of agile (Boehm 1988). This methodology ensures that software is developed in discrete stages such as planning, requirements, coding, testing, and evaluating. One advantage of this methodology is that there are feedback loops after each stage that would allow for reviewing each stage upon completion (1988). This way, the next stage could be started knowing that the previous stage was completed. Another advantage of this type of methodology is the low cost associated with its implementation (Verma, Bansal, and Pandey 2014). This methodology has such a low cost because most of the variables are known such as the requirements, technology, and the end product. The last advantage to this methodology is that if there is excellent communication in the requirements and planning stages, then the other stages that follow are very clear and the development teams know exactly what to expect (Bindrees et al. 2015). A disadvantage of this methodology is that the software must be developed in the specific order (Boehm, 1988). This eliminates the possibility of any flexibility after the initiation of the project.

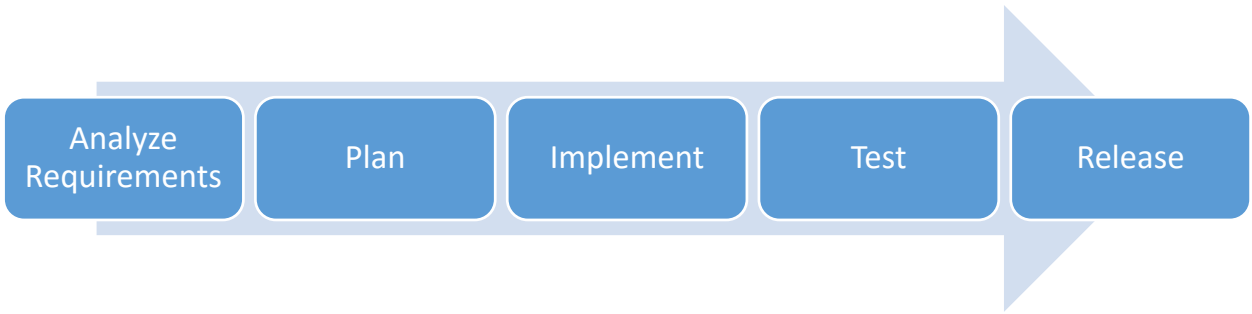

Figure 1: Shows the workflow of a project when the development team is following Waterfall

Spiral

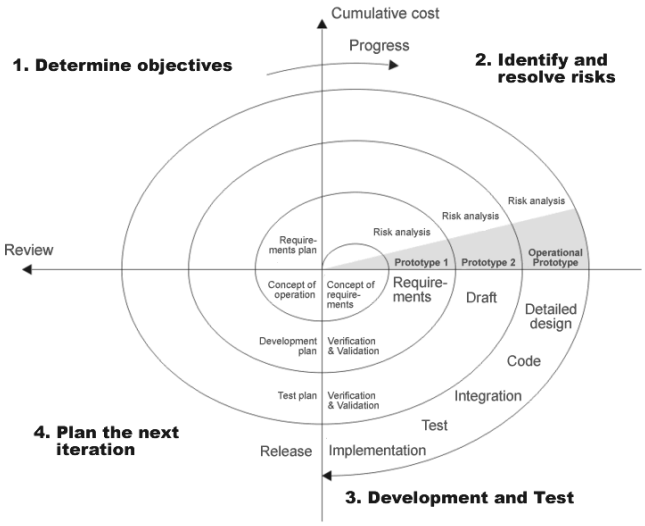

Figure 2: Shows the workflow of a Spiral development methodology
Image from: http://leansoftwareengineering.com/

The last software development methodology that will be evaluated is the spiral development process. A major advantage of the spiral process is that it is largely risk driven (Verma, Bansal, and Pandey 2014). As a result of the risks being the driving force, the spiral model allows multiple options to be formulated to create a specific software (Boehm 1988). In fact, the alternative means of implementation are an integral part of the process (1988). This accommodates the cases where the requirements are frequently changed, something that the waterfall method cannot do. A disadvantage that is associated with this property and methodology overall is the high cost associated with it (Verma, Bansal, and Pandey 2014) which, again, cannot be said about the waterfall method. Another advantage of the spiral methodology is that there is not much documentation required in relation to the waterfall or agile methods, which is also why it is easy to make changes to the overall plan and design of the project (Boehm 1988). The spiral model will either mimic the good traits of existing methodologies (Verma, Bansal, and Pandey 2014) or it will provide a framework of which

methodologies are the best to use in specific situations by evaluating all risks (Boehm, 1988). A disadvantage associated with this advantage is that developers and development teams using this methodology must identify all risks associated with the project (1988). Sometimes it is not so simple to identify all risks which in turn reduces the effectiveness of the spiral process.

Research Plan of work and estimated schedule

The research method that will aim to determine the most efficient and effective software development methodology is divided into 3 phases which are estimated to take a total of **9-13 weeks**:

Phase I: Defining a software development methodology and the need for these methodologies ***(3-4 weeks)***

The first phase of research will aim to convince the audience that there is a need for software development teams to follow an efficient and effective software development methodology. In order to complete this phase, first I will explore what specifically consists of a software development methodology. This will be done using the literature review and the articles by Boehm and Verma, Bansal, and Pandey **(1 week)**. Once the key features of a software development methodology have been determined, I will then analyze the need for these methodologies and if production is improved as a result. This will be done by obtaining data about a software development team's product quality, workflow efficiency, team dynamic, and team communication **(2-3 weeks)**. The data will be obtained by surveying customers, team members, and project managers using the Likert scale. The surveys will be given to a sample of software development teams, half of which follow a specific software development methodology while the other half of the sample does not follow any methodology.

Phase II: Determining what makes a software development methodology efficient and effective ***(3-5 weeks)***

The second phase of research will present how to determine if one software development methodology is more effective and efficient than another. In order to complete this phase, first I will determine the specific qualitative variables that will be tested for each methodology. This will be done by using variables determined to be vital throughout the production process using past research analyzed in the literature review **(1-2 weeks)**. I predict these variables will include cost, product quality, team chemistry, and timeliness of release among others. Using these variables, I will again obtain data from a sample of software development teams. The sample will be selected such that there will be different software development methodologies used by different teams. The data will be obtained by again surveying the customers, team members, and project managers to see how these variables are addressed quantitatively using the Likert scale **(2-3 weeks)**.

Phase III: Evaluating Agile, Waterfall, and Spiral development processes ***(3-4 weeks)***

The third and final phase of research will evaluate the three most used software development methodologies according to the literature review. The goal of this phase is to determine why these three methodologies are the most common in software development teams and to determine if these methodologies are in fact effective and efficient using the variables defined in Phase II. In order to complete this phase, I will use the same procedure for all three methodologies. First, I will use the literature review to define the methodologies and the properties associated with each one **(1 week)**. Next, I will evaluate the advantages and disadvantages associated with each methodology using the literature review **(1 week)**. Next, I will use the data from Phase II to determine how effective and efficient each of these three methodologies are **(1-2 weeks)**.

Anticipated Involvement of the Product Managers:

For this project, I request access to interview and survey each manager's team and customers. The data that results from these interviews and surveys will be used solely in the research methods outlined above. In return, the product managers will benefit from this research because it will gauge how efficiently and effectively their teams are producing. Based on my research, each product manager can go back to their respective teams and either implement a new software development methodology or try to improve their already existing methodology.

Researcher Qualifications:

I am qualified to do the project that I am proposing based on my academic background and experience in the field of software development. I am currently a candidate to receive my Bachelor of Science in the field of Computer Science with a concentration in software engineering. Additionally, through my experience as an intern, I have been part of a software development team that followed a software development methodology. In fact, during my internship period, the team was transitioning from one methodology to another. I had firsthand exposure to the importance of software development methodologies and how they directly influence the team's success. This experience also allows me with the knowledge of various companies that use certain software development methodologies that I can contact when I am conducting my research and potentially use their development teams in my sample.

Budget:

This project will need a significant amount of funds in order to be completed, in total about $9,000. The costs of this project will be mainly costs associated with travelling. As outlined by the table below, the cost for travel include airfare, hotel, car rental, and food costs. These are necessary costs because I will be meeting with product managers from companies across the country to obtain the data I need from each software development team for my research. I will first reach out to them via phone or e-mail and then set up a day or two where I can come to their company and get the data that I have elaborated on above.

Cost	**Amount**
Airfare	$5,000
Hotel	$1,000
Car Rental	$1,000
Food	$1,000
Total	*$9,000*

References

M. Bindrees, R. Pooley, I. Ibrahim and N. Taylor, "Re-Evaluating Media Richness Theory in Software Development Settings", *JCC*, vol. 02, no. 14, pp. 37-51, 2014. Available: http://file.scirp.org/Html/4-1730145_52338.htm [Nov. 15, 2015].

B. Boehm, "A spiral model of software development and enhancement", *Computer*, vol. 21, no. 5, pp. 61-72, 1988. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=59 [Nov. 15, 2015].

J. Highsmith and A. Cockburn, "Agile software development: the business of innovation", *Computer*, vol. 34, no. 9, pp. 120-127, 2001. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=947100 [Nov. 15, 2015].

N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek and P. Abrahamsson, "Software development in startup companies: A systematic mapping study", *Information and Software Technology*, vol. 56, no. 10, pp. 1200-1218, 2014. Available: http://ac.els-cdn.com [Nov. 15, 2015].

R. Schaefer, "Business-efficient software development processes", *SIGSOFT Softw. Eng. Notes*, vol. 38, no. 4, p. 7, 2013. Available: http://delivery.acm.org/ [Nov. 15, 2015].

J. Verma, S. Bansal and H. Pandey, "Develop Framework for Selecting Best Software Development Methodology", *International Journal of Scientific & Engineering Research*, vol. 5, no. 4, pp. 1067-1070, 2014. Available: http://www.ijser.org/researchpaper/Develop-Framework-for-Selecting-Best-Software-Development.pdf [Nov. 15, 2015].

L. Williams and A. Cockburn, "Agile Software Development: It's about Feedback and Change", *IEEE Computer Society*, vol. 36, no. 6, pp. 39-43, 2003. Available: http://www.computer.org/csdl/mags/co/2003/06/r6039.pdf [Nov. 15, 2015].

X. Yu and S. Petter, "Understanding agile software development practices using shared mental models theory", *Information and Software Technology*, vol. 56, no. 8, pp. 911-921, 2014. Available: http://ac.els-cdn.com/ [Nov. 15, 20