Estimation of Agile Functionality in Software Development

Bashir Nasr-Azadani, Reza MohammadDoost, *IAENG*

***Abstract*— An introduction to agile principals through agile manifesto has been shown. To prove the functionality of agile methods, qualitative (or quantitative) comparisons have been carried out with the other regular methods. It's always a complicated task for a project analyst to analyze the development of a software project; especially when it comes to evaluation of the development methodology and estimation of the project's success considering every related aspect. The goal of this paper is to simplify this task for the analyst and clarify the important aspects of a project which develop with agile methodologies.**

***Index Terms*—**Software Engineering, Software Development Methodologies**,** Agile Methods, Evaluation of Functionality

INTRODUCTION

There's a misconception due to comparison of "plan-driven" software development methodologies to agile methodologies while it can be concluded that agile methodologies are "unplanned" which is not true. In fact, the concept within this comparison is agile methodologies are "adaptive" rather than "predictive".

Predictive methods focus on formation of a programmed plan to perform the project. In contrast, adaptive methods focus on analyzing the current situation and finding the best solution in every step of performing the project.

This paper focuses on how to comprehend if Agile methodologies are useful for a specific project.

Section 1 discusses about the principles of Agile methods through the Agile community's manifesto. Section 2 explains the major differences among Agile methodologies and some other software development methodologies. Section 3 discusses about a method that can be used to evaluate the functionality of Agile methodologies in a software project. Answering some specific questions will lead the analyst to determine the factors and data that can be used as input to the estimation formulas. Those factors and data will be put in couple of statistical formulas and then analyst has to interpret the outcome. Section 4 talks about a software that can do these calculations.

I. AGILE COMMUNITY'S MANIFESTO

Some of the principles behind the Agile Manifesto [1] are:

- Customer satisfaction by rapid, continuous delivery of useful software
- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress.
- Even late changes in requirements are welcomed.
- Close, daily and cooperation between business people and developers
- Face-to-face conversation is the best form of communication.
- Projects are built around motivated individuals, who should be trusted
- Continuous attention to technical excellence and good design.
- Simplicity
- Self-organizing teams
- Regular adaptation to changing circumstances

The publication of the manifesto back in 2001 spawned a movement in the software industry known as agile software development.

The Manifesto has become an important piece of the Agile Movement, in that it characterizes the values of Agile methods and how Agile distinguishes itself from traditional methods.[2],[3]

II. COMPARISON WITH OTHER METHODS

Agile methods are often defined as being at the opposite end of the spectrum from "plan-driven" or "disciplined" methodologies. This misconception is misleading, as it implies that agile methods are "unplanned" or "undisciplined." In fact, the true distinction is to say that methods exist on a continuum from "adaptive" to "predictive". [4] Agile methods exist on the "adaptive" side of this continuum.

Adaptive methods focus on adapting quickly to changing realities. When the needs of a project change, an adaptive team changes as well. An adaptive team will have difficulty describing exactly what will happen in the future. The further away a date is, the vaguer an adaptive method will be about what will happen on that date. An adaptive team can report exactly which tasks are being done next week, but only which features are planned for next month. When asked about a release six months from now, an adaptive team may only be able to report the mission statement for the release, or a statement of expected value vs. cost.

Predictive methods, in contrast, focus on planning the future in detail. A predictive team can report exactly what features and tasks are planned for the entire length of the development process. Predictive teams have difficulty changing direction. The plan is typically optimized for the original destination and changing direction can cause completed work to be thrown away and done over differently. Predictive teams will often institute a change

control board to ensure that only the most valuable changes are considered.

Agile methods have much in common with the "Rapid Application Development" techniques from the 1980's as espoused by James Martin and others.[5]

II.I Contrasted with the waterfall model

Agile development has less in common with the waterfall model. In some eyes the waterfall is discredited, but as of 2004, this model is still in common use.[6]

The waterfall model is the most predictive of the methodologies, stepping through requirements capture, analysis, design, coding, and testing in a strict, pre-planned sequence. The main problem of the waterfall model is the inflexible nature of the division of a project into separate stages, so that commitments are made early on, and it is difficult to react to changes in requirements. Iterations are expensive. This means that the waterfall model is likely to be unsuitable if requirements are not well understood or are likely to change radically in the course of the project.[7]

Agile methods, in contrast, produce completely developed and tested features (but a very small subset of the whole) every few weeks or months. The emphasis is on obtaining the smallest workable piece of functionality to deliver business value early, and continually improving it/adding further functionality throughout the life of the project.

Some agile teams use the waterfall model on a small scale, repeating the entire waterfall cycle in every iteration.[8] Other teams, most notably Extreme Programming teams, work on activities simultaneously.

II.II Contrasted with "cowboy coding"

Cowboy coding is the absence of a defined method: team members do whatever they feel is right. Agile development's frequent reevaluation of plans, emphasis on face-to-face communication, and relatively sparse use of documents sometimes causes people to confuse it with cowboy coding. Agile teams, however, do follow defined (and often very disciplined and rigorous) processes.

As with all methodologies, the skill and experience of the users define the degree of success and/or abuse of such activity. The more rigid controls systematically embedded within a process offer stronger levels of accountability of the users. The degradation of well-intended procedures can lead to activities often categorized as cowboy coding.

III Preparing the input data for the formulas

In order to estimate more accurate, we should compare Agile methodologies with other methodologies in different projects. In order to calculate a value that can help the analyst choose the best methodology to manage a project , some questions should be answered. Those answers will be the data the analyst needs to put into the formulas. These questions derived from the comparison of Agile methodologies with other methodologies and the Agile manifesto and the features of Agile methodologies.

Agile methods encourage businesses to be accountable for the value produced by software development efforts. The key metrics we use should allow us to monitor this accountability. Metrics should help validate businesses that make smart software investments and teams that deliver business value quickly.[9]

An answer to each question is a value between 1 to 10. The value shows the importance of each factor asked in the question. (1 indicates the least importance,10 indicates the most.)

The questions have been asked based on different side of each project for example customer's point of view or project manager's point of view and etc.

The major questions concerns:

Customer

1) How often developers can be in touch with the customer?
2) How fast the customer wants the project to be delivered?
3) How important is the number of deliveries during the project development period for the customer?
4) How fast the customer wants each delivery?
5) How much does the customer know about his/her requirements and his/her project?
6) How much does the customer support and collaborates with developers during the whole development operations?

Project development process

1) How important the documenting process is?
2) How important is the time limits?
3) If the project (major problem) can be divided into smaller parts , how difficult is to make the parts?
4) How dynamic the project is?
5) How important is testing the project by the time each iteration would be completed?
6) How complicated the project is?
7) How big is the size of the project?
8) How important is the risk management for this project?

Production Team

1) How much facility the team has for communication among developers?
2) How many senior developers in the team are?
3) How democratic the culture of the team's organization is?
4) How dense the team is?
5) How close relationships in the team are?
6) How honest the team members are?
7) How important security of information in the team is?
8) How many programmers and developers are in the team?
9) How adaptive the team is?

Product

1) How clear major problems of the project are?
2) How familiar this kind of project is to the team members?
3) How important is the quality of the product?

III.I Statistical calculations

Each answer is a value named *K* between 1 to 10. it also has an error factor named a. a can be $-a$ or $+a$ or both values. $+a$ means the answer to the question can be more and $-a$ means the answer can be less, $+a$ and $-a$ both means the

answer can be *a* value more or *a* value less.

After all questions being answered and the error factors being given analyst do the additions:

1) Addition of all *K*s.
2) Addition of all (*K*+*a*)s.

The mathematical definition of *U* will be (1) and (2).

$$U_{(\min)} = \sum_{i=1}^{m} [K_i - a_{(\min)_i}] \qquad (1)$$

$$U_{(\max)} = \sum_{i=1}^{m} [K_i - a_{(\max)_i}] \qquad (2)$$

Here *m* indicates the number of questions and K_i indicates the value of each answer to the question number *I*, respectively. Refer to (1) and (2) it can be concluded that:

$$\frac{U_{(\min)}}{10*m} \le \frac{\sum_{i=1}^{m} [K_i - a_{(\min)_i}]}{10*m} \le \frac{U_{(\max)}}{10*m} \qquad (3)$$

Note that, in (3), the term 10**m* means all the questions has been answered with 10 value (which indicates the best possibility for using Agile methodologies). Thus, the division of $\sum K_i$s to 10**m* results a value named *U*.

Multiplication of *U* to 100 results in a percentage quality named *S*. (4)

$$S = \frac{U}{10*m} * 100 \qquad (4)$$

The quality *S* indicates the percentage of Agile methodologies functionality for the software project according to the answers which has been given to the questions.

Therefore, using $U_{\min}$ and $U_{\max}$ to calculate $S_{\min}$ and $S_{\max}$, it will be clear that:

$$S_{(\min)} \le S \le S_{(\max)} \qquad (5)$$

However the value of *S* should be interpreted and the best conclusion should be drawn by analysts. It means which value scopes explain how useful Agile methodologies can be for this specific project. By entering the data as inputs into the software which will be discussed later, the final answer will be calculated.

Here are some solutions to improve the estimation:

1) If analyst has more factors in mind regarding the project, he/she simply can add them into the software as questions. Hence, the software can recalculate *S* value concerning the new questions. In contrast, analyst can remove any needless questions in his/her point of view.
2) Normally, all questions have the same importance factor named *B* and its value is 1. However, if some factors will be more important to the analyst he/she can change the importance factor manually.

By entering the importance factor (*B*) into the calculations, we have:

$$U_{(\min)} = \sum_{i=1}^{m} B_i * [K_i - a_{(\min)_i}] \qquad (6)$$

$$U_{(\max)} = \sum_{i=1}^{m} B_i * [K_i - a_{(\max)_i}] \qquad (7)$$

Therefore, *S* formula modifies into (8):

$$S = \left[\frac{U}{10 * \sum_{i=1}^{m} B_i} \right] * 100 \qquad (8)$$

IV. Developing the Software for Other Methodologies

The software can calculate all those formulas which has been discussed above automatically. However, the input data is important.

The software has the ability to add and remove the questions, due to this fact analyst can add questions regarding the factors of other methodologies; so the calculations works for that methodology too. However, It is important to interpret the final output and estimate the best methodology through that outcome.

References

[1] Beck K., Cockburn A., Jeffries R., Highsmith J., "Agile manifesto", http://www.agilemanifesto.org, 2001.

[2] Glass R., "*Agile versus traditional: Make love, not war*", Cutter IT Journal, *http://www.cutter.com/content/itjournal/fulltext/2001/12/itj0112c.html*, 2001, pp. 12-18

[3] Cohen D., Lindvall M., & Costa P, "An introduction to agile methods". Available: http://www.cs.umd.edu/~mvz/cmsc435-s05/pdf/agile-paper.pdf, 2004, pp 8-12.

[4] Boehm, B., R. Turner, "Balancing Agility and Discipline: A Guide for the Perplexed", Boston, MA: Addison-Wesley, 2004, pp. 165-194

[5] Martin J, "Rapid Application Development", Macmillan Coll Div, 1991

[6] Laplante, P.A., C.J. Neill, ""*The Demise of the Waterfall Model I6 Imminent" and Other Urban Myths*", http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=110, 2004.

[7] Sommerville I, "4.1.1. The waterfall model", Software engineering, 8th edition, Harlow: Addison Wesley, 2007, pp. 66f.

[8] As *reported* by *HeavyLogic*, *http://heavylogic.com/agile.php*

[9] Hartmann D, Dymond R, "Appropriate Agile Measurement". Available: http://www.berteigconsulting.com/AppropriateAgileMeasurement.pdf, 2006, pp. 3-6.