How does agility ensure quality?

Ming Huo, June Verner, Muhammad Ali Babar, Liming Zhu
National ICT Australia Ltd. and University of New South Wales, Australia
{mhuo, jverner, malibaba, limingz}@cse.unsw.edu.au

Abstract

Software quality is one of our most important software concerns. Agile methods may produce software faster but we also need to know how they meet our quality requirements. In this paper we compare the waterfall model with agile processes to show how agile methods achieve software quality. We also show how agile methods attain quality under time pressure and in an unstable requirements environment, i.e. we analyze agile software quality assurance. We present a detailed waterfall model showing its software quality support processes. We then show the quality practices that agile methods have integrated into their processes. This allows us to answer the question can agile methods ensure the quality even though they develop software faster and can handle unstable requirements?

1 Introduction

Since Kent Beck introduced Extreme Programming [1], agile software development has become a controversial software engineering topic. Some practitioners and researchers vociferously argue about the benefits of it, others are forcefully against agile methods, while others suggest a mix of agility and plan-driven practices [2]. However, the reality is that agile methods have gained tremendous acceptance in the commercial arena since late 90s because they accommodate volatile requirements, focus on collaboration between developers and customers, and support early product delivery.

Two of the most significant characteristics of the agile approaches are: 1) they can handle unstable requirements throughout the development lifecycle 2) they can deliver products with shorter timeframes and under budget constraint when compared with traditional development methods [3-6]. Many published reports support the above advantages of agile methods. However, proponents of agile methods have not yet provided a convincing answer to the question "what is the quality of the software produced?" Does agility provide enough rigors to ensure quality, as do the traditional development methods, e.g., waterfall model, and if these methods do provide the same level of quality then how is it achieved.

We now compare the quality assurance techniques of agile and traditional software development processes. Our approach consists of three steps: 1) build a complete outline of the traditional waterfall model including its supporting processes, 2) Identify those practices within agile methods that purport to ensure software quality when compared with software quality assurance techniques found in traditional methods, 3) determine the similarities and differences between agile and traditional software quality assurance techniques. By applying such an approach, we believe we can systematically investigate how agile methods integrate the support for software quality within their life cycle.

The rest of the paper is organized as follows. Section 2 presents a short description of waterfall and agile methods to highlight the disadvantages of the former and the reasons why the latter has become popular. Section 3 gives a brief introduction to software quality assurance techniques. Section 4 explains why we chose a waterfall approach in order to perform our comparison. In this section we perform a comparison with respect to software quality. Section 5 closes the paper by identifying future work required to substantiate our approach.

2 Waterfall model vs. Agile Methods

Since the late 60s, different software development methods (such as waterfall model, evolutionary development method, spiral development model etc.) have been developed and widely used by the software engineering community [7] . Over the years, the

developers and users of these methods have invested significant amounts of time and energy to improve and refine them. Owning to continuous improvement efforts and being practiced for such a long time, most of the above mentioned methods have become quite mature and stable level. That is why they are usually referred as traditional software development methods

Each of the traditional development methods attempts to address quite different development issues and implementation conditions. Among the traditional development approaches, the waterfall model is the oldest the software development process model. (Royce 1970). Waterfall model has been widely used in both large and small software intensive projects. It has been reported as a successful development approach especially for large and complex engineering projects [7]. The waterfall model divides the software development lifecycle into five distinct and linear stages. Because the waterfall model is the oldest and the most mature software development model we have chosen it to investigate its QA process [8].

Despite the success of Waterfall model with large and complex systems, it has a number drawbacks, like linearity, inflexibility in the face of changing requirements, highly ceremonious processes irrespective of the nature and size of the project etc [7]. Such drawbacks can also be found in other traditional development approaches. However, agile methods were developed to address a number of the drawbacks inherent in the Waterfall model.

Agile methods deal with unstable and volatile requirements by using a number of techniques of which most notable are: 1) simple planning, 2) short iteration, 3) earlier release, and 4) frequent customer feedback. These characteristics enable agile methods to deliver product releases in a much short period of time compared to the waterfall approach.

This brief comparison of the waterfall and agile methods brings this discussion to our research question, how can agile methods ensure product quality with such short time periods? **Our research hypothesis is that in agile methods, to a certain degree, some of their practices include traditional QA supporting process within their development life cycle.**

Before we continue future, we analyze various quality assurance techniques, a general description of these techniques and their associated supporting processes.

3 Quality assurance techniques

Since we are concerned with the quality of the software produced with both the Waterfall model and the agile approach, we investigate quality-centric supporting processes in software development. We concentrate on two of the most widely used general quality-focused processes, Software Quality Assurance (SQA) and Verification and Validation (V&V) to examine software product quality.

"SQA governs the procedures meant to build the desired quality into the products" and V&V is aimed more directly at product quality including intermediate products [8]. These two supporting processes are normally used to support the waterfall model in order to provide a full complete process model.

Quality assurance techniques can be categorized into two types, static and dynamic. Static and dynamic techniques are both used in SQA processes. The selection, objectives, and organization of a particular technique depend on the requirements and nature of the project. A Waterfall development method selects these techniques according to very different criteria, such as some people-intensive techniques chosen in waterfall model [8].

Unlike dynamic techniques, Static techniques do not involve the execution of code. Static techniques involve examination of documentation by individuals or groups, this examination maybe be assisted by software tools, for example, inspection of the requirements specification and technical reviews of the code. Testing and simulation are dynamic techniques. Sometimes static techniques are used to support dynamic techniques and vice versa.

The waterfall model uses both static and dynamic techniques. However, agile methods mostly use dynamic techniques. We will compare and contrast the quality assurance techniques used by these two approaches later in this paper.

4 Agile methods quality techniques assessing methods

In this section, we build a complete model of waterfall with QA supporting process. Figure 1 shows the diagram form. In the second half of this section, we address some of the quality assurance practices of agile methods. In this paper, we only have the space to discuss a few of these practices. In the future, we plan to provide a comprehensive list of these practices in near future.

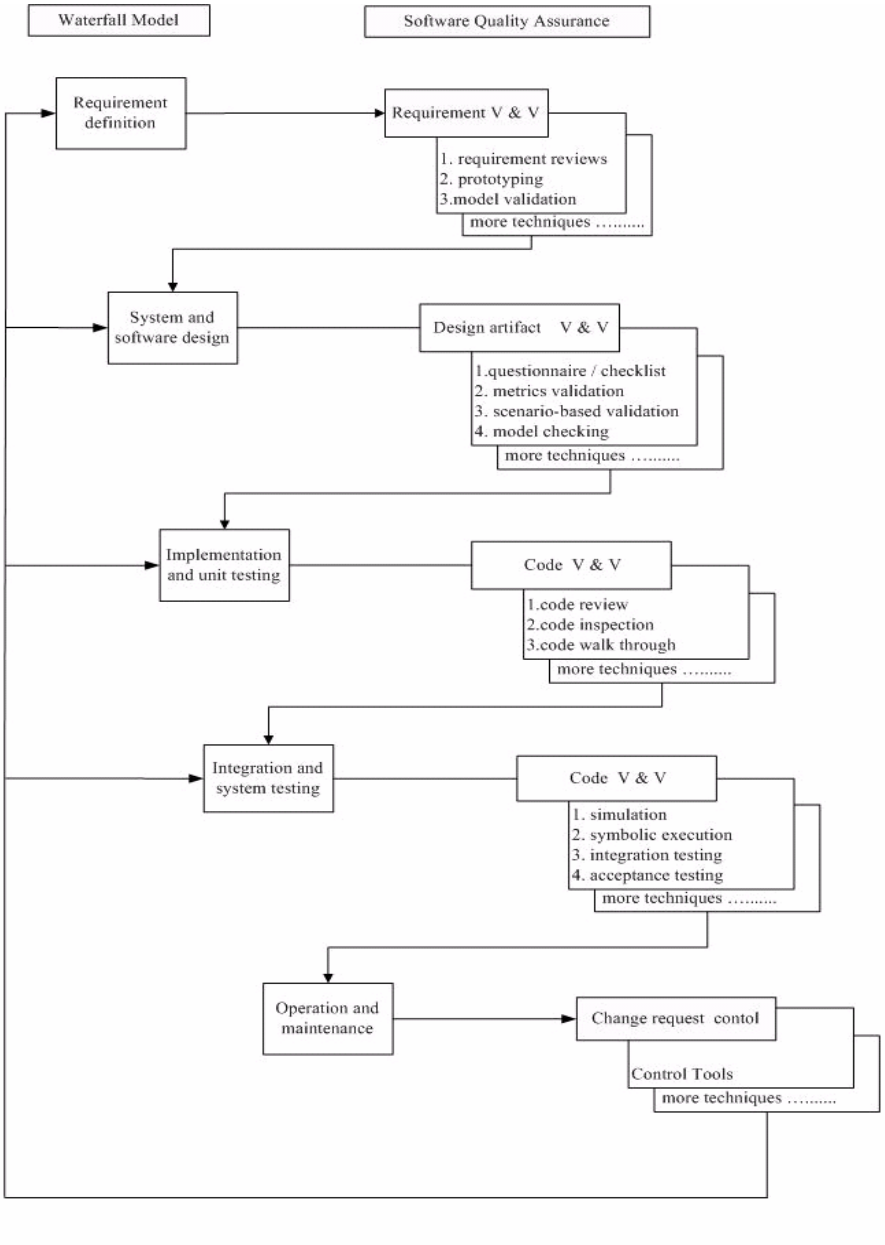

Figure 1. Completed process model

4.1 Waterfall model with SQA and V&V

The fundamental development activities of in the Waterfall model include: 1) requirements definition 2) system and software design 3) implementation and unit testing 3) integration and system testing 4) operation and maintenance [7]. Each activity supported by V&V techniques is supposed to produce of well-defined deliverables. Since the deliverables of one activity are the input for the subsequent activity, no subsequent phase can begin until the predecessor phase finishes and all of its deliverables are signed off as satisfactory. To complete the output must be approved by these QA activities. Figure 1 shows the development process.

We will use this complete model as a base for comparison with the quality assurance techniques of the agile methods. We will explain the comparison method in section 4.3.

4.2 Agile Methods: quality techniques

Agile methods include many practices that have the potential ability for quality assurance. By identifying these practices and comparing with quality techniques used in waterfall model, we can analyze the quality assurance status of agile methods. We list some agile practices, which have been recognized, as quality techniques below, and we believe there is a number of other techniques have not been explicitly identified yet.

Having an *On-site customer* is a general practice in most of the agile methods. Customer helps developer to refine and correct the requirements. The customer should support the development team throughout the whole development process. There is no such activity in the traditional methods. In waterfall, customers are normally involved in requirement definition and possibly system and software design but not involved as much and contributes as much as they are expected in agile methods. Consequently the customer involvement in agile methods is much heavily than waterfall development.

Pair programming means two programmers continuously work on the same code. Cockburn and Williams found pair programming could improve design quality and reduce defects [9]. Its effect shows that pair programming includes code V&V techniques. Its shoulder-to-shoulder technique serves as a continual design and code review process, and result reducing defect rates. This action has been wildly recognized as continuous code inspection [9].

Continuous integration is also a popular practice among agile methods. Continuous integration means the team does not integrate the code once or twice. The team needs to keep the system fully integrated at all times. Integration may happen several times pre day. Martin has pointed out that, "The key point is that continuous integration catches enough bugs to be worth the cost." Continuous integration also reduces the time that people spend on searching bugs and allows detection of compatibility problems early. This practice can be treated as a code V&V technique and is an example of dynamic techniques. Waterfall model development also requires integration, but this is much later and its frequency is much lower than agile methods [10].

Acceptance testing is carried out after all unit test cases have passed. This activity is a dynamic quality assurance technique [8]. Waterfall approach has acceptance testing but the difference between agile acceptance testing and traditional acceptance testing is as followings. Acceptance testing happens much earlier and much more frequently and not only done once.

This only provides a sample of agile quality assurance techniques. We are going to identify all of this kind of agile activities in our full paper.

However, if we compare the difference between agile quality assurance activities and waterfall SQA from three aspects: 1) many of the agile activities occur much earlier than they do in waterfall development 2) the frequency of these activities is much greater than in waterfall model 3) agile methods have fewer static quality assurance techniques when compared with waterfall development.

5 Future work

In this section, we discuss future work that needs to be done in this area. This work comprises two parts: 1) agile practices identification 2) quality techniques comparison. We discuss these further below.

1. Agile practices identification As noted in section 4.2, there are many agile practices that have a quality assurance potential. These practices include more than those listed in section 4.2. Further work need to be done to identify and classify them as static or dynamic techniques.

After identification, clarification of what agile practices has certain SQA support techniques and at which stage these agile practices occurs is necessary.

2. Quality techniques comparison There are two major differences between waterfall development and agile methods quality assurance: 1) Agile methods include fewer static techniques than waterfall development. This can be explained by their background. Many of the static techniques in waterfall are people-intensive; these cost time and resource [8]. Agile methods are used when we have market pressure and budget limitation so people-intensive techniques are not acceptable. The second reason is that waterfall normally begins with static requirement documents. Hence static techniques are suitable. Agile methods, on the other hand, begin with poor and violate requirements. This makes static methods unsuitable at this stage.

2) Quality assurance activities start earlier and more frequent than waterfall development. The agile process has many small releases and each release can be considered to be similar to a tiny waterfall release. Clearly analyzing how quality can be achieved in each

agile release will help us understand how agile processes achieve quality.

This will allow us to identify which parts of agile development add the most quality to our software.

6 Conclusion

Even though some of agile practices are not new, the agile methods are recent. Because of the advantages they bring, they become very popular in industry. Experience reports detail how these methods solve problems such as development time limitation and unstable requirement [5]. There is an important need for developers to know more about the quality of the software produced. Developers also need to know how to revise or tailor their agile methods in order to attain the level of quality they required. Our research is going to shed the light on this issue.

7 Reference

[1] K. Beck, *extreme programming eXplained : embrace change*. Reading, MA: Addison-Wesley, 2000.

[2] B. Boehm and R. Turner, "Using risk to balance agile and plan-driven methods" *Computer*, vol. 36, pp. 57-66, 2003.

[3] J. Grenning, "Launching extreme programming at a process-intensive company," *Software, IEEE*, vol. 18, pp. 27-33, 2001.

[4] O. Murru, R. Deias, and G. Mugheddue, "Assessing XP at a European Internet company," *Software, IEEE*, vol. 20, pp. 37-43, 2003.

[5] J. Rasmussen, "Introducing XP into Greenfield Projects: lessons learned," *Software, IEEE*, vol. 20, pp. 21-28, 2003.

[6] P. Schuh, "Recovery, redemption, and extreme programming," *Software, IEEE*, vol. 18, pp. 34-41, 2001.

[7] I. Sommerville, *Software engineering*, 6th ed. Harlow, England ; New York: Addison-Wesley, 2000.

[8] A. Abran and J. W. Moore, "Guide to the software engineering body of knowledge : trial version (version 0.95)." Los Alamitos, CA: IEEE Computer Society, 2001.

[9] A. Cockburn and L. Williams, "The Costs and Benefits of Pair Programming," in *Extreme Programming examined*, G. Succi and M. Marchesi, Eds. Boston: Addison-Wesley, 2001, pp. xv, 569 p.

[10] Continuous Integration, http://www.martinfowler.com/articles/continuousIntegration.html.