Software Quality and Assurance in Waterfall model and XP - A Comparative Study

Sana'a Jawdat Khalaf Sana_j_11@hotmail.com

Mohamed Noor Al-Jedaiah m_aljedaiah@ammanu.edu.jo

Abstract: -Dealing with an increasingly volatile organizational environment is a serious challenge for managers of any software development .Traditional formal software development methodologies can be characterized as reflecting linear, sequential processes ,and the related management approaches ,and be effective in development software with stable ,known ,consistent requirements .Yet most real-world development efforts are much more likely to be conducted in more volatile environments, as organizations adapt to changing technology, markets, and social conditions. Requirements for systems must be able to change right along with them, often at "Internet speed" [1]. Project management approaches based on the traditional linear development methodologies are mismatched with such dynamic systems. The support of software quality in a software development process may be regarded under two aspects: first, by providing techniques, which support the development of high quality software and second, by providing techniques, which assure the required quality attributes in existing artifacts. Both approaches have to be combined to achieve effective and successful software engineering [2]. 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 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 quality even though they develop software faster and can handle unstable requirements?"[3]

Key-Words: - Agile processes, Extreme Programming, Waterfall model, Software development, Software quality, Customer Satisfactions, Customer needs.

1 Introduction

A Successful software engineering strongly depends on the delivery of high quality software. High quality is typically defined by quality attributes like customer satisfaction (which is mainly determined by being on budget and time). Besides skilled people, an appropriate and well-working software development process is a key success factor for

achieving the demanded high quality software. Traditional software development process models four core process steps: analysis, design, implementation and test. These processes guarantee the correct analysis of requirements and their valid implementation. Some process models include deployment and maintenance activities after test. These processes support the stability of the running system. Others mainly support the quality attributes customer satisfaction by keeping the project on time and budget. However, most of them do not explicitly define software quality development and software quality assurance as a part of the software development model itself. Yet, these quality support processes are vital for achieving high quality software [2] .The usage of software assurance techniques can be minimized due the extensive usage of software quality development techniques. Thus, the total cost of software quality may be reduced.

In this paper, we examine two of the most known software development processes regarding practices for supporting software quality development and software quality assurance in the software development process itself. The findings can be used as basis for selecting a suitable process for high quality software development or for adopting running processes to well-working practices found in this study.

Section 2 gives an overview of the related work on why software quality development and software quality assurance in software development process models?

Section 3 gives a short introduction in the Waterfall model and Extreme Programming (XP).

Section 4 describes the principles and techniques, which we have derived form analyzing the two models compared here, and evaluates based on the principles and techniques presented in this section . Finally, we summarize our results and suggest to add some standard for software quality support. This standard may be applied to the definition, adoption, or selection of software development models.

2 Related work

It's not enough to talk and talk by saying that software quality is important .You have to

1-Explicitly define what is meant when you say" software quality".

2-Create a set of activities that will help ensure that every software engineering work product exhibits high quality.

3-Perform quality control and assurance activities on every software project.

4-Use metrics to develop strategies for improving your software process and, as a consequence, the quality of the end product [4].

Every one involved in the software engineering process is responsible for quality .If a software team stresses quality in all software engineering activities ,it reduces the amount of rework that it must do. That results in lower costs and more importantly , improved time-to-market.

Software quality is achieved by three approaches: testing, static analysis and development approaches [2]. The integration of all three approaches is the most desirable

approach. However, there is no consensus about the details of such an integrated framework. A different categorization of approaches towards software quality regards four ways to establish software quality: software quality via better quality evaluation, better measurement, better processes, and better tools [2].

2.1 Software quality via better quality evaluation

Almost every organization has its own internal standards that provide a guideline for measuring and monitoring quality. Standards increase the level of understanding of the process by the project members, thereby promoting better communication. In addition to standards, organizations need clearly defined quality models to effectively meet the demands from customers. It is important for such organizations to explicitly develop a quality model that best suits their interests and implement it. One of the earliest quality models was proposed by McCall Model (McCall, Richards, and Walters 1977); this model describes quality as being made up of a hierarchical relationship between the quality factors, quality criteria, and quality metrics.

ISO 9126 recently, a new standard for software product evaluation, ISO 9126, has been developed by the ISO (1992). This standard has identified six basic quality characteristics that must be present in a quality software product. The standard also provides a sample decomposition of these basic characteristics into sub characteristics [4].

3 Two major Software Process Models

The software development processes discussed in this paper represent the most common and widely used process models throughout the industry. Even though, on an abstract level, the waterfall model and agile methods like XP are very different process methods, their actions within the development sequence share some similarities. In this section, we provide a short description of both the waterfall model and agile methods.

3.1 Waterfall model

Since the late 60s, many different software development methodologies have been introduced and used by the software engineering community [3]. Over the years, developers and users of these methods have invested significant amounts of time and energy in improving and refining them. Owning to continuous improvement efforts, most of the methodologies have reached a mature and stable level. Hence, they are referred as traditional software development methods. Each of the traditional development methods attempts to address different development issues and implementation conditions. Among the traditional development approaches, the waterfall model is the oldest the software development process model. It has been widely used in both large and small software intensive projects and has been reported as a successful development approach especially for large and complex engineering projects [5]. The waterfall model divides the

software development lifecycle into five distinct and linear stages. Because it is the oldest and the most mature software development model we have chosen it to investigate its QA process [3]. In addition we chose the waterfall model because the phases in a waterfall development are more linear than other models. This provides us the opportunity to clearly present the quality assurance (QA) processes. In practice, the waterfall development model can be followed in a linear way. However, some stages can also be overlapped. An iteration in an agile method can also be treated as a miniature waterfall life cycle. Despite the success of the waterfall model with large and complex systems, it has a number drawbacks, such as inflexibility in the face of changing requirements, and a highly ceremonious processes irrespective of the nature and size of the project .Such drawbacks can also be found in other traditional development approaches. However, agile methods were developed to address a number of the problems inherent in the Waterfall model [1].

3.2 ExtremeProgramming(XP)

Extreme Programming was created by Kent Beck, Ward Cunningham ,and Ron Jeffries during their work on the C3 project in March 1996 and in October 1999, Extreme Programming Explained was published. XP is a discipline of software development based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.In Extreme Programming, every contributor to the project is an integral part of the whole team . The team forms around a business representative called "the Customer", who sits with the team and works with them daily. Extreme Programming teams use a simple form of planning and tracking to decide what should be done next and to predict when the project will be done. Focused on business value, the team produces the software in a series of small fully-integrated releases that pass all the tests the Customer has defined.[7] XP planning addresses two key questions in software development: predicting what will be accomplished by the due date, and determining what to do next. The emphasis is on steering the project, which is quite straightforward, rather than on exact prediction of what will be needed and how long it will take, which is quite difficult. hand, with so much visibility, the Customer is in a position to cancel the project if progress is not sufficient. On the other hand, progress is so visible, and the ability to decide what will be done next is so complete, that XP projects tend to deliver more of what is needed, with less pressure and stress.

4. Finding common software quality development practices

The best practices presented in this section, which support software quality development and software quality assurance, have been extracted applying an analytical bottom-up

approach. This approach is based on the analysis of the chosen process models (Waterfall and XP) adding all principles and techniques defined in the processes to the criteria catalogue, which apply to the basic goal: support of software quality development and software quality assurance. The criteria descriptions are based on the definitions of Waterfall, and XP and on expertise derived from the local software engineering community. The bottom-up approach ensures that no criteria are selected which are not implemented in at least one industrial process model. Therefore the criteria catalogue will be rather industrially practicable than scientifically optimized.

4.1 Iterative software development

To establish higher software quality, a software development process has to use an iterative and incremental development approach. Iterative software development is characterized by

refining and improving artifacts in several iteration cycles. Incremental software development means to start with a draft version of an artifact in the initial iteration and to extend and refine it through the following iterations. Iteration cycles include all development activities analysis, design, implementation, testing and finally deployment. Quality assurance can be applied more effectively during the overall process. By using an iterative approach, a process gains more flexibility in dealing with changing requirements or scope. The product releases of the product force early feedback from the customer and the stakeholders, which is vital for improving the overall quality of the software. However, iterative development must be supported by risk management and early involvement of the end users to achieve its full potential. XP builds on a very strict iterative approach, demanding a daily build of all components. This limits the time needed to encounter errors and forces developers to fix a problem as soon as possible. Of course, incomplete components or single methods are excluded from the daily build. The work breakdown structure must consider these issues to allow an integration of smaller components every day. Using this approach requires a lot planning, but definitely enables high software quality [3].

4.2 Quality as an objective

A software development process needs to define quality as a major objective to improve overall software quality. Quality targets have to be defined and documented by involving the project team and the customer. This ensures that the quality goals become achievable and measurable.

4.3 Continuously verification of quality

A set of procedures that document every change during the project is required to finally ensure quality. Not only project status reports, but also assessments of the current activities and possible changes are needed to identify problems as soon as possible. To support these procedures, every project needs a defined process to managed changes. All these actions

can be implemented as meetings or as supporting workflows. Continuously verifying quality includes extensive testing. Besides internal testing, external acceptance tests with the customer are needed too in order to verify that the product fulfils the needs and requirements of the customer. A software development process must therefore include a testing workflow throughout the complete process, including external tests with the end users to ensure high software quality.

4.4 Customer requirements

A software development process builds on clear structure and methodology to elicit and document customer requirements. It also has to integrate these requirements into the complete process. The elicitation of requirements is one of the most complex software engineering disciplines. The needs and wishes of the customer, who normally does not have a deep technical knowledge, have to be documented so that developers are able tobuild an application based on that information. Thus, it is necessary that the project team understands the customer and his business. Otherwise it is not possible to correctly implement the customer needs. A software development process has to focus on the requirements throughout the entire project. Eliciting and documenting the requirements at the beginning is not sufficient. The implementation of the requirements has to be traced during development. Furthermore, the software development process has to ensure thatnot only the customer, but also the end users are involved in the requirements process. Success of the project strongly depends on these two groups: the first buying the product and the second using it. A software development process has to define procedures to train the end users to use the final product.

4.5 Architecture driven

In modern software development, the architecture of a system has a major impact on the overall quality of the product. One reason for this is the integration into existing systems and environments as a major part of today's software development. Re-use has become increasingly important due to increasing time and cost pressure. Using a well-designed architecture allows easy integration and re-use, so a software development process has to be architecture driven.

4.6 Focus on teams

A team has to be seen as a set of equal persons, who together are responsible for the quality and success of a project. When responsibility for failure can be assigned to a single person, the project success is not guaranteed anymore. Focusing on teamwork also improves motivation of the project members, as everyone is seen as an equally important part of the project. This finally leads to a high identification of team members with the product. It is obvious that motivated team members contribute to high quality, as they work more concentrated and conscientious. A software development process has to include a well-defined team structure, including an

efficient task assignment and clear communication guidelines.

4.7 Pair programming

Pair programming is closely linked to the focus on teams, but was picked as another assessment criterion since it has been underrated for its contribution to high quality in the past. XP demonstrates how two developers can complement each other rather than inhibiting each other. One developer implements the current method while the other is working on integration issues. This approach saves time, and minimizes the number of errors. Better solutions are more likely since two persons most likely have different perspectives of the same problem and therefore, complement each other in solving it.

5. Evaluation of the quality support in Waterfall model and XP.

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 [3]. The development activities 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 [6]. Each activity produces well-defined deliverables. Since the deliverables of one activity are input for a subsequent activity, from the theory point of view, no subsequent phase can begin until the predecessor phase finishes and all of its deliverables are signed off as satisfactory. The output from each phase is input to the corresponding supporting phase and will be verified or validated by its supporting process; this output is then sent to the next stage as input.

In the waterfall model, customers are typically involved in requirements definition and possibly system and software design but are not involved as much and do not contribute as much as they are expected to in XP. In practice, in a waterfall development, some milestone reviews might be set up and customers will participate, but this kind of customer involvement is less intense than it is in XP. Waterfall model development integration is done much later and its frequency is much lower than in an agile method development.

5.2 Agile Methods: quality techniques

Agile methods include many practices that have QA potential. By identifying these practices and comparing them with QA techniques used in the waterfall model, we can analyze agile methods QA practices.

System metaphor is used instead of a formal architecture. It presents a simple shared story of how the system works; this story typically involves a handful of classes and patterns that shape the core flow of the system being built. There are two main purposes for the metaphor. The first is communication. It bridges the gap between developers and users to ensure an easier time in discussion and in providing

examples. The second purpose is that the metaphor contributes to the team's development of a software architecture [5]. This practice helps the team in architecture evaluation by increasing communication between team members and users.

Having an On-site customer is a general practice in most agile methods. Customers help developers refine and correct requirements. The customer should support the development team throughout the whole development process. Consequently customer involvement in agile methods is much heavier than in waterfall development.

Pair programming means two programmers continuously working on the same code. Pair programming can improve design quality and reduce defects [3]. This shoulder-to-shoulder technique serves as a continual design and code review process, and as a result defect rates are reduced. This action has been widely recognized as continuous code inspection [3]. Refactoring can reduce the chances that a system can get seriously broken during the restructuring [5]. During refactoring developers reconstruct the code and this action provides code inspection functionality. This activity reduces the probability of generating errors during development. Continuous integration, a popular practice among agile methods means the team does not integrate the code once or twice. Instead the team needs to keep the system fully integrated at all times. Integration may occur several times a day. Continuous integration catches enough bugs and reduces time that people spend on searching for bugs and allows detection of compatibility problems early. This practice is an example of a dynamic QA technique. Acceptance testing is carried out after all unit test cases have passed. This activity is a dynamic QA technique [8]. A Waterfall approach includes acceptance testing but the difference between agile acceptance testing and traditional acceptance testing is that acceptance testing occurs much earlier and more frequently in an agile development; it is not only done once. Early Customer feedback is one of the most valuable characteristics of agile methods. The short release and moving quickly to a development phase enables a team to get customer feedback as early as possible, which provides very valuable information for the development team. Although this kind of development style renders most separate static techniques on early phase artifact unsuitable, code makes dynamic techniques useful and available very early. Also developers are more responsible for quality assurance compared with having a separate QA team and process. This allows more integration of QA into the development phase. Small releases also bring customer feedback for product validation frequently and requirements verification. The QA techniques for agile methods are based on: Applying dynamic QA techniques as early as possible (e.g. TDD, acceptance testing) .Moving more QA responsibility on to the developer (e.g. code inspection in peer/pair programming, refactoring, collective code ownership, coding standards).Early product validation [7] (e.g. customer on site, acceptance testing, small release, continuous integration).

6. Conclusion

To sum up, 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 require. In this paper we have analyzed and compared the differences between the SQA from three aspects:

1) many of the XP activities occur much earlier than they do in waterfall development, 2) the frequency of these activities is much greater than in the waterfall model; most of these activities will be included in each iteration and the iterations are frequently repeated during development, 3) XP have fewer static quality assurance techniques, move into the development phase very quickly. The approach that the customer is on site and involved in the iteration planning process strengthens quality control from the customer site. The short iterations force the project team to develop functional releases at the end of each iteration to pass acceptance testing. Requirements in XP uses so-called "user stories" to capture the requirements but defines no ongoing process for requirements management.

References :

[1] W.H.Morkel Theunissen, Derrick G. Kourie and Bruce W, Standards and Agile Software Development,Watson, SAICSIT 2003, Pages 178–188.

[2] Sven Heiberg, ,Methods for Improving Software Quality:a Case Study, Master's Thesis, University of Tartu,2002.

[3] J. E. Gaffney, Metrics In Software Quality Assurance, ACM Journal, November 9-11-1981.

[4] Scott Ambler, Quality in an Agile World , SQP vol. 7, 2005.

[5]Jim Highsmith, Retiring Lifecycle Dinosaurs, Software Testing & Quality Engineering(STQE)magazine, July/August 2000.

[6] Pressman, Roger S.,Product Metrics for Software Engineering, McGraw Hill; 6/e, 2005, ch.15.

[7] Ambler, S. W. Agile Modeling, John Wiley and Sons, (2002).

[8]Cockburn,A.,AgileSoftware Development, Massachusetts, Addison Wesley Longman, 2001.

[9] Highsmith, J, Adaptive SoftwareDevelopment", Dorset House., (1999):

[10] Highsmith, J., Agile Software Development Ecosystems, Boston, MA, Addison- Wesley.2002.

[11]A.Cockburn,L.Williams, TheCosts and Benefits of Pair programming,.International Conference .