KEYWORDS user satisfaction · user expectation · testing · agile approach · scrum · iteration planning

TOWARDS AGILE APROACH FOR BETTER USER SATISFACTION

Nizar Al Hawajreh
Master degree student at the Computer Science and Engineering department of Qatar University

Abdelaziz Bouras, Ph.D.
ictQATAR Chair Professor at Qatar University
abdelaziz.bouras@qu.edu.qa

Ashraf Abualia
Hanadi Al-Thani
Zohreh Fouroozesh
Kholoud Khalil
Computer Science Department, College of Engineering, Qatar University
Doha, Qatar
P.O Box 2713

Kholoud Mohammed
Muna Al Kuwari
Alanood Zainal

ABSTRACT

User satisfaction is a cornerstone indicator of any successful software project. Software projects are classified as successful only if the users are satisfied with the delivered software project result. Reducing the gap between users' expectations and the actual delivered software is one of the ultimate objectives of software project management. Therefore, it is essential to manage user expectations during the project, which is basically achieved by managing the user opinions related to the final performance of the software. However, this cannot be isolated from the adopted testing methodology, which is the way to guarantee the optimal user satisfaction. Furthermore, the stakeholder occupies a significant position in agile principles aligned with development teams. Flexibility to adapt stakeholders' late changes is also another gain in agility. Agility additionally focuses on the decentralized management approach to exploit any managing skills of the software project individuals. In this paper, we describe how these factors are relevant to the agile project management strategy. Our findings explain the hidden reasons behind the success of the agile strategy in software development projects.

INTRODUCTION

Several factors influence the success or failure of large software projects. Among these factors, user satisfaction can directly define the real success or failure of any project. In general, large projects have many tasks and phases that are required to be assembled together in order to deliver the project's objectives. In terms of executing software projects, more precise planning, in-depth analysis and design are needed which make the task of managing large projects more challenging. This is mainly the result of more required resources, time, and cost, and the plan is very complicated and might suffer imprecise blueprint. Adding to that, such projects have higher risk of failure because of the difficulty in controlling their long phases and deliverables.

Managing user expectations is one of the major tasks in software projects **[19]**. However, managing this task is considered challenging since it is not affiliated to a single phase of the project, instead, it must be carried out during the whole project. Based on that, we can confidently mention that it is extremely important to properly manage the user expectations in order to control the catastrophic consequences of increasing the gap between the delivered software and the user expectations. These facts leave few options other than building a user-oriented plan where the user is the key factor of the project success. This approach is eventually built on top of three main strategies that complement each other and lead to a successful management of the user expectations. These strategies are user involvement, leadership and trust. For example, the practical implementation of these three strategies is clearly considered in the testing phase, which should be carefully designed to cope with the decentralized and user-oriented approach of the project. It is very essential to develop a compatible testing plan to achieve the best satisfaction. The main purpose of the testing phase is to assure achieving users requirements, in addition to checking the correctness of the implemented functions and identifying any partial or complete failure which can be achieved through the early detection of bugs and errors.

Testing is usually laborious and detailed work. For example, testing a website requires a complete check of all links every time a change is performed in the website. From this point, creating templates, standards and documentations is a real time-saver. Unfortunately, testing is normally the phase that gets shortened if a project is late. When a task is delayed and is consuming more time than what was anticipated; the testing phase is compressed. However, it is crystal clear that compressing the testing phase leads to late prediction of bugs and eventually unsatisfied user. Late prediction also increases the cost of fixing bugs. Thus it is better to concurrently carry out testing after each phase to overcome these limitations.

The project lifecycle can be designed based on different development process models such as waterfall, spiral, incremental and iterative models. Waterfall and spiral approaches share one major limitation, where both of them require an exact comprehensive plan to reflect all stakeholder expectations. The stakeholder is unable to see the real software deliverables before the completion of development process which results in late changes in the project requirements. These changes may dramatically require a complete redesigning and reimplementation for multiple software deliverables. Moreover, in these approaches one project manager should be assigned to manage excessive number of tasks because of the centralized management approach. On the other hand, the incremental development approach divides the project into smaller increments that are easier to manage. During that time, stakeholder involvements become more frequent. This is actually to fill in the gap between the user expectations and how the project team understands the user requirements. However, this approach lacks a clear vision to leadership concepts. In contrast, the agile approach – one of iterative software development styles – is customer value oriented. Because of leadership and decentralized approach, project manager responsibilities are significantly reduced and some classical project manager roles are transferred to development teams. In agile, development team individuals have the ability to make decisions in their domains. Moreover, agile approach is designed to smoothly accept late changes which can save many efforts. This approach focuses on managing user expectations and the risk behind it. We show that testing is a very dependent phase to ensure the successful implementation of user expectations within a project lifecycle.

The paper is organized as follows. In section 1 we discuss user expectations. Testing will be presented in section 2. Section 3 presents an overview about classical development approaches.

The agile model is illustrated in section 4. Section 5 states the iteration planning. In section 6 a case study is addressed. Finally discussion and conclusion are demonstrated in sections 7 and 8 respectively.

1. User Expectations

Project management is summarized as shown in **Figure 1 [10]** into a diamond model. The diamond consists of time, cost, quality, scope and user expectations in the middle. To manage a software project these factors should be controlled. Addressing these constraints properly leads to a satisfactory deliverable software project. Where a successful project relies on user satisfaction **[17]**.

User satisfaction depends on service quality and product quality **[9]**. Service quality is a comparison between the user expectations and the perceptions of the service **[1]**. Whereas, product quality measures how the delivered system meets the requirements and satisfies the user during the product life cycle. Users evaluate service quality by evaluating both technical and functional quality. The technical quality is the quality of the delivered software such as performance, disaster recovery, high availability or response time. On the other hand, functional quality is the user interaction with the process of producing the outcome **[2]** that involves people, infrastructure and processes. Evaluation of product quality is based on defining external and internal system product attributes. External attributes involve the system functionality such as speed and safety and internal attributes involve the software projects architectural structure **[7]**.

User expectations in a software project can be defined as "a set of beliefs held by the targeted users of a system associated with the eventual performance using the system" **[15]**. Software development projects are associated with high failure rates. The users are not satisfied if the delivered software project does not meet their expectations. Therefore,

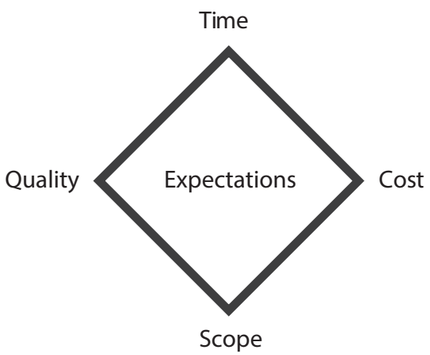

FIGURE 1. *Project Management Diamond*

bassisting users in generating their expectations, the failure rate can be decreased.

One of the studies related to user expectations is service quality where software project managers should ensure that the user develops a reasonable expectation while managing the project. One of the measurement instruments used to measure service quality is SERVQUAL **[3]**. It studies the difference between user expectations and perception. It has various dimensions including tangibles, reliability, responsiveness, assurance, and empathy **[3]**.That means quality insurance is an important factor to satisfy user expectations.

Additionally, identifying the requirements correctly from the beginning of the project is a key factor of satisfying user expectations. Requirements can be classified into three sets **[15]**. First, users change their requirements constantly, they may not know at the beginning what they want in the future. Their needs change based on certain reasons such as their situation, funding or even politics. Second, users have difficulty in information processing. Generating expectations depends on the user mental model which is subject to distortion. The user capacity in information process is limited. They may not be able to call all situations. They may focus only on some common requirements and ignore rare cases, which affect the overall correct functional requirements of the delivered system. Finally, analysis of requirements is not based on positivism. It is not a simple task to formulate the requirements and problem, as the user states literally. Since, what is in the user mental mode is different than what they say. These requirements should be addressed and managed cautiously.

As a result, satisfying user expectations requires ensuring a high quality product that meets their requirements. This in turn needs good management for the user expectations from the beginning of the project, when the user is expressing a set of requirements, to the end of the project life cycle. This can also be achieved by good testing throughout the life cycle. Poor testing and management of user expectations lead to unnecessary extra effort, money and time.

1.1 Managing User Expectations

Managing user expectations can be carried in two different scenarios, one when the user expectations are less than what is perceived. The other one is when the user expectations are more than what is delivered. In the first case the user would undervalue the system, while in the second scenario the user might be disappointed. Both cases negatively affect the delivered system and by then the stakeholders can consider it a failure. One aspect of software project failure can be defined as the gap between the user expectations and the delivered system outcome. In consequence, realistic level of user expectations may play a major goal towards project success. This raises the importance of managing user expectations which is defined as "the actions a project manager performs to ensure that the assumptions held by the user

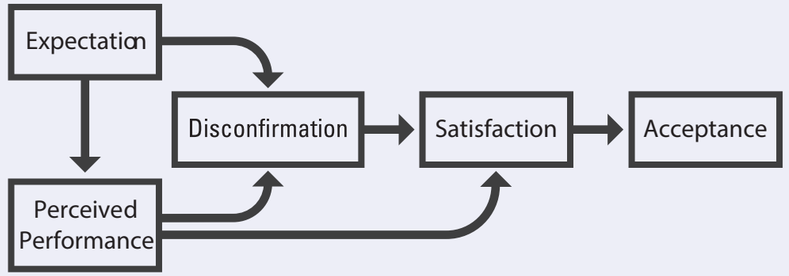

FIGURE 2. *Expectation-confirmation Theory*

for a software project are realistic and consistent with the software deliverable promised by the project team" **[18]**.

Another study related to managing user expectations is the expectation-confirmation theory which studies user behaviors and has a strong relationship between user expectation, performance, and satisfaction. **Figure 2**. illustrates the expectation-confirmation theory. A prior important stage before using software is to develop individual expectations. After using the software, the users have to develop opinions based on its performance. Consequently, the expectations are compared to the performance and the developed expectation can be whether it is confirmed or disconfirmed. If the expectations are disconfirmed, the user satisfaction is affected negatively and may change the user opinion of accepting the project **[4]**.

One technique of managing user expectations is to adopt the user centralized approach within the project lifecycle. The main goal of this approach is to satisfy the user, which means satisfying and meeting users requirement with their expectations. The user centralized approach involves three main strategies which complement each other and lead to a successful management of user expectations. These strategies are user involvement, leadership and trust **[18]**. These strategies were identified based on a real study aimed to address the risk of failing to manage user expectations. In this study, 12 software project managers from large IT and management companies were interviewed. They were asked to recall two different experiences they faced previously. In the first experience, they successfully managed user expectations and in the other they failed to manage user expectations. Analysis and comparison were conducted on the entire situation of 24 experiences. User involvement is about making the user interact in the process of the project lifecycle. Leaderships consist of two types: a project leader and a champion leader. While the first deals with the software development team, the other deals with users. Finally, trust is used by managers to make the users feel that they are working with them not on or over them.

Any software development life cycle passes through different phases. User requirements are gathered at the beginning of the life cycle. In this phase it is crucial to understand and analyze these requirements in order to build a high quality product that satisfies the user when the product is delivered. If the delivered product meets these requirements, this certainly satisfies the user. User satisfaction is also the main goal of the testing phase where the user requirements are verified. Thus, among all phases of the life cycle, the testing phase is the most crucial phase which focuses on ensuring high quality of the software product that meets user requirements. Testers in all testing levels check whether user requirements are achieved which leads to satisfaction. The next section will present the software testing phase in more details.

2. Software Testing

Software testing is an essential task of software quality assurance that leads to users' satisfaction. Thus many software companies and organizations spend most of their resources on testing **[14]**. Software testing has many definitions, briefly it is defined as the process with the goal to find bugs and errors in a software product before it is delivered to the end user **[21]**. It also aims to make sure that all customer requirements have been achieved. The testing process includes all dynamic or statistical activities that are carried out to ensure that the product satisfies the end user's requirements and specifications.

2.1 Software Testing Life Cycle (STLC)

Software testing life cycle is a crucial concept that presents the different phases of testing. Every organization has its own STLC that is affected by the management's policies. In this section, we will shed light on these different testing phases.

2.1.1 Requirement Analysis Phase

In this phase the test team tries to understand and analyze the users' requirements by interacting with them. These requirements could be either functional where it defines the function of the system or non-functional such as security

availability **[23]**. It is important to understand these requirements in order to achieve satisfying the end user.

2.1.2 Test Planning Phase

Planning the testing is a crucial phase of testing life cycle to ensure that the project is delivered with good quality on time and within reasonable cost **[14]**. Test plans consist of testing methodologies, testing environment, testing strategies and availability of hardware and software resources. All test phases should be included in the test plan. Subsequent separate test plans are developed for each phase with specialized teams. This phase helps to estimate the project cost, time and effort **[8]**.

2.1.3 Test Case Development Phase

Test Case development begins when test planning tasks finishes. This phase involves creating test cases which include test conditions and test input as well as procedures that will be followed while testing the software. Moreover, test scripts are created including a set of instructions to be performed on the system being tested, it could be manually or automated. Test data are also created, reviewed and reworked here **[23]**.

2.1.4 Test Environment Setup

It can be done in parallel with the test case development phase. In this phase the software and hardware conditions as well as the environment of software product testing are set up. This environment could be a work place or in a laboratory. Efficiency of test efforts depends very much on setting up a high level environment where the testing is conducted, thus preparing such environment is needed before testing is executed **[24]**.

2.1.5 Test Execution

Once setting up of the testing environment has finished and test cases have been defined, the test execution phase can start. Testers in this phase conduct testing based on the prepared test plans and test cases. Different methods are used here in order to find the errors and bugs. Then reports are written about these bugs to the development team to resolve these bugs and the product is retested again to ensure that it is free of errors and bugs.

2.1.6 Test Closure

The software testing life cycle reaches the closure stage when all bugs are fixed and the product meets the user's requirements. Lessons about strategies and best practices are taken from current testing life cycle for future works **[23, 24]**.

2.2 Testing Levels

Good testing can be achieved by conducting different levels of testing which helps to easily identify the bugs as each piece is tested separately in coordination with other pieces of the system. It also helps to verify at each level whether the software product with all its components is done according to user's requirements. The main four levels are shown in **Table 3**.

2.3 Testing Now or Later

Testing within all classical Software Development models such as traditional V model, spiral model and waterfall model, is exercised at the end of life cycle **[21]**. Therefore earlier phases require revisiting for bugs to be fixed if they occur, which could be more expensive and cause the product to be delivered late.

Many modern studies prove that it is better to spread testing throughout the development lifecycle to have better results that meets user's expectation. According to Huq's simulation study, it was found that concurrent testing after each phase is less expensive than testing after coding. Though it might require the same or more efforts and time that are spent in testing after each phase, later it will save time and efforts in the maintenance phase. His study also suggests that better performance of the software product that achieves user satisfaction can be done through testing simultaneously **[13]**.

Agile approach supports this idea where better user satisfaction can be achieved by simultaneous testing. Also to have the user involved through the life cycle to be sure that the product is not far from user's expectation. This approach will be discussed in detail in the following section after showing some limitations of the classical models that lead to transferring to the agile approach.

3. Classical Software Lifecycle Development Approaches

Large software projects have many challenges to succeed. This is due to complexities in software applications and hardware infrastructure **[20]**. Large size software projects are executed in long periods. During such long periods, many changes might happen worldwide as well in the same organization. Changes can be in emerging technologies, obstacles in hardware, gaps between stakeholders and development teams, and redirecting of resources. Additionally, it is very difficult to predict costs and required resources. In fact, this is a primary reason behind exploding costs that might accelerate a project failure. As a result, it is hard to build a precise project plan for future. Furthermore, multi-national organizations may have multiple software development teams located in different locations around the world. These types of distributed teams are hard to manage and interact with project managers. So there could be a delay in receiving directions, especially if there is a centralized management approach. One of the software challenges in most software development projects is the change of requirements while progressing to the project execution. These late changes can dramatically extend the project duration and explode costs and resources, due to the complexity of software projects and hardware infrastructure. These late changes are not planned, and it requires modifying all development cycles steps, such as analysis, design, implementation and testing.

3.1 One-Shot Software Development Models

One Shot models are software development models in which the stakeholder cannot see the product and test it until the completion of the product. There are two models addressed in this paper as one shot models, namely: waterfall model and Spiral model.

3.1.1 Waterfall Model

Waterfall model is a common model for software development projects **[20]**. This model is divided into six steps, namely: planning, requirements analysis, design, implementation, and testing. The sequence of these steps is consecutive. In other words it moves from one step to another. A step is not preceded until the completion of the previous step. Such model does not accept late changes. Additionally, the stakeholders are not involved except in the early steps such as planning and requirement analysis. Such model is seriously affected by software project challenges, hence resulting in failure. **Figure 3.** depicts the waterfall model.

3.1.2 Spiral Model

On the other hand, spiral model involves important stakeholders to review each step's outputs **[5]**. Still, it is an advanced representation of the Waterfall model. During the software development process, the stakeholders can see incremental prototypes. Furthermore, a spiral model introduces the risk analysis for each phase of software development life cycle. However, stakeholders are unable to see any real component of the product until the entire project is complete, as a result, this model is inflexible with the late changes that can appear after the development process is complete.

3.2 Incremental Software Development Models

Incremental software development models are based on breaking down the developed software application into smaller components **[12]**. These smaller components can be developed in sequence or in parallel according to available resources and dependencies. Each of these components is developed in short-term unit time increment. There are different models adapting this idea and these models have different names for an increment such iteration or sprint. Such increment is easier to plan and predict. Additionally, early feedback can be forwarded to the development team by concerned stakeholders. Late changes could be feasible during an iteration because it requires a given component. Moreover, any future integration can be taken into account in this change as it already becomes a part of a component. Furthermore, a customer might be able to start using such component. **Figure 4.** illustrates the incremental model.

Testing level	Function	Responsibility
Unit testing	Testing the smallest piece of the product separate from other parts to check its functionality and correctness of the codes and correctness of outputs based on inputs **[8]**.	Programmers
Integration testing	Testing different parts of the product in combination to ensure they are working together without contradiction.	Special testers
System testing	Testing the entire system to ensure that it is aligned with user requirements, and checking its functionality.Testing includes security testing, volume testing, usability testing, and performance testing **[8]**.	Tester (must have knowledge about the components of the system and their functionalities **[23, 16]**.)
Acceptance testing	Testing the entire system to ensure it meets the actual requirements.	End user

TABLE 3. *Testing Levels*

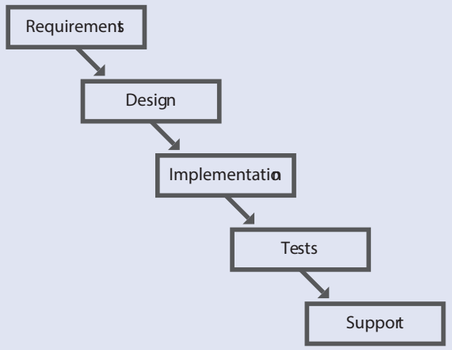

FIGURE 3. *Waterfall Model*

4. Agile Software Lifecycle Development Approach

In addition to previous experiences of software models, agile methodologies have also emerged. Agile models are flexible, incremental, human-elaborated, self-organized teams, software development methodology **[20]**. Agile principles mainly stem from three primary sources, namely, lean production, agile manifesto and incremental software model. Agile Manifesto is a group of people who introduced 12 principles for agile methodologies **[26]**. These principles recommend several aspects in software development process. These aspects are customer involvement and satisfaction from the beginning to the end. Late changes should be taken into account at any point. Breaking down a software project into smaller increments and each increment duration should be short, approximately weeks and months. Shorter durations are also preferred over longer ones. Moreover, in a team the collaboration of individuals is a crucial aspect in the agile approach in terms of decision making, trust, and motivation. Frequent team meetings should be encouraged in all organization hierarchal levels to be able to overcome outstanding obstacles and adopt changes. Teams should be authorized to organize themselves as they know the skills and talents of the internal team members. Finally simplicity is an important aspect of the agile approach. This aspect apparently appears in decentralized management approach, and smaller component iterations. Team members can evaluate their assigned task situation and their ability to make the correct decision about a given problem in their scope. For example, the project manager whose scope should focus on high level goals instead of spending so much time and effort on details. There are many software development models that are considered to be under the agile umbrella, such as lean software development, Scrum, test-driven development, extreme programming, and rational unified process. In the following section we are going to discuss "Scrum" which is the most common model.

4.1 Scrum Software Development

Scrum is a specific iterative, predefined-roles agile model. In Scrum, the development life cycle is divided into sprints or iterations. Each sprint is usually accomplished in two weeks. This model has three roles, namely: product owner, team, and Scrum master. Additionally, there are three documents in this model: product backlog, sprint backlog, and sprint results. Finally this model requires three kinds of meetings: sprint planning meeting, daily Scrum meeting, and sprint review. **Figure 5** is a diagram that illustrates the Scrum software development lifecycle.

Product owner is the representative of stakeholders and is authorized to express all stakeholders' requirements, approve the deliverables, and release payments of the project. Team on the other hand is responsible for developing and testing sprint tasks. Teams also have the responsibility of organizing themselves and distributing tasks among their members. The third role is the Scrum master, responsible for attaining the sprint process goals, sprint quality and progress. In addition, he or she is responsible for overcoming outstanding issues and eliminating any obstacles for a given sprint. Scrum master can also modify the Scrum process to best fit the requirements of the project and organization. The most important responsibility of Scrum master is to preserve the high quality of sprints and not leave any pending bugs in order to prevent any future accumulated bugs. The added value in Scrum is that the Scrum master is not responsible for managing any team tasks. His role is to motivate the team members, share responsibility, and let them take their own decisions in their scope which is the assigned sprint.

Scrum monitors product documentation in three documents respectively; product backlog, sprint backlog, and sprint results. Product backlog is the master log file that contains all product requirements. The sprint backlog contains all required user stories in a given sprint. A user story is provided by product owner and stakeholders which contains the full description of an end-to-end process. Additionally, it records the interactions between team members, Scrum master, and product owner. Once a sprint user story is accomplished by a team, verified by Scrum master, and approved by product owner, it is then recorded in the sprint results. Otherwise it is returned back to the product backlog to be reworked in the future.

A recommended activity in agile is face-to-face interaction. Scrum model determines three kinds of meetings, first, sprint planning meeting, in which, new sprint is planned in collaborating with all primary Scrum participants. Team, Scrum master, and product owner negotiate the sprint plan. Then, the product owner selects some product user stories that can put together a product component and can be accomplished during one sprint. Moreover, the product owner prioritizes these user stories. The Team in turn estimates the required time. Second, daily sprint meetings, 15 minutes long, guided by the Scrum master are held with team members to follow up what was accomplished the day before and what is going to be done the next day for each team member. Additionally, the barriers are evaluated and sorted out by the Scrum master in daily sprint meetings. The final objective of the daily sprint meetings is for the Scrum master to evaluate the Scrum process and enhance it, if required. In the last sprint review meeting all sprint participants review actual results with stakeholders and verify its completion or stakeholders express their feedback, and the user story is sent back to product backlog to be worked on later.

Scrum agile approach covers the three aspects of the user-oriented approach which are user involvement, leadership and trust. User involvement is the main criteria in the agile approach which can be exploited as follows: effective user involvement during the development process of a software project is crucial for the overall success of the project. The involvement must be interactive such that the product owner should focus on listening to user's concerns and questions. In addition, the Scrum master has to let the users be part of the project lifecycles by letting them make their choices, when there may be some conflicts or tradeoffs. For instance, they may give higher priority to budget over functionality or schedule over performance. For a large group, it is useful to break down the group into smaller groups and exchange ideas with them. It is also important to train the helpdesk on communicating with users to make them feel comfortable with the team during the lifecycle of the project. Informing the users with the progress is essential to make them feel involved and a part of the project **[18]**. Failing to interactively involve the user means having to deal with requirements uncertainty and failure to meet the business goals.

Also, leadership is adopted in the agile process with the following: the product owner can act as project champion for the users. The project champion is responsible for communicating with users in order to manage their expectation. Successful management of user expectation starts by promoting the vision and purpose of the project, and also educating the users on the value of the software and ensuring they are engaged in the different phases. One important characteristic

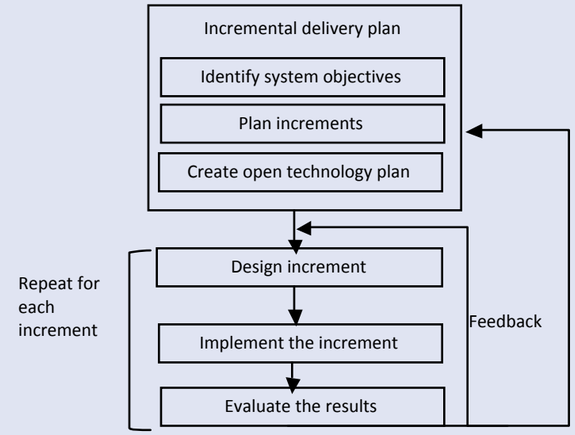

FIGURE 4. *Incremental Model*

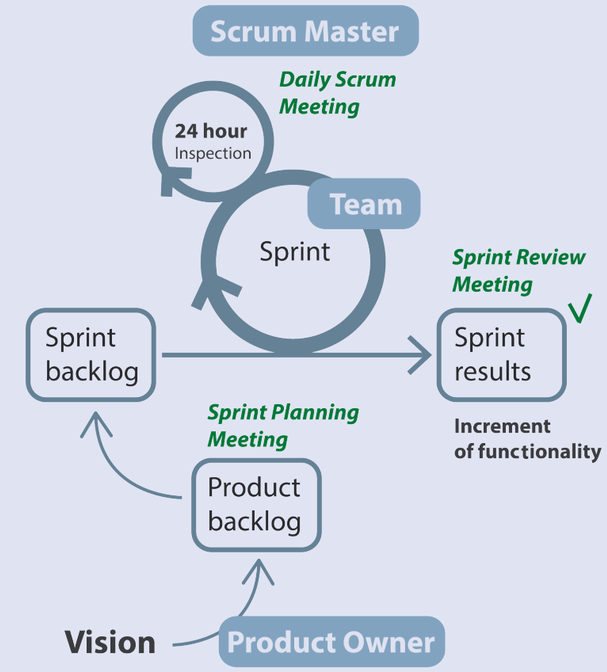

TABLE 2. *Scrum Software Development Lifecycle*

of project champion is to have a clear and strong vision of the project goal. Moreover, the champion should be strong in leadership to ensure the successful outcome of high-risk projects. On the other hand, the Scrum master acts as a project manager who oversees the completion of all tasks during the development process. The manager responsibilities include leading all users on the right path and ensuring that his team work is done effectively. The manager should take strong management control over the scope of the project. He needs to keep motivating his team by not requiring from the members to do more than what they are willing to do **[18]**.

Moreover, trust is maintained by the Scrum master to build strong relation with product owners. The trust exists within any situation where relationships are involved. That is, the Scrum master has to talk with all product owners about any problem faced during the project development. It is important for product owners to hear about any problem from the Scrum master first rather than from someone else. A strong Scrum master makes all product owners feel that they are not over committing to something. The Scrum master should be transparent with the product owners and tell the truth about the state of the project. Sharing good and bad aspects with the product owners and offering precise time for deliverables helps to build strong users trust. Finally, the Scrum master should totally avoid the general saying, "Fake it till you make it" to not lose users trust. Losing users trust leads to failure in managing user expectation, hence leading to project failure **[18]**.

5. Iteration planning

According to the Scrum model, it is apparently important to achieve two prerequisite tasks before starting any sprint. The first task is grouping a number of development resources to form a sprint team, which is done by task assignment. The second task is to define all sprints and their dependencies. One of the suggested approaches that proved to be an optimized approach of iteration planning is the semi-automated planning of iterations **[11]**. This context release and increment are used interchangeably. Additionally, iteration and sprint are used interchangeably.

5.1 Release Planning

Important decisions are made during the release planning phase **[6]**. After requirement analysis and software product specifications are complete. Requirement dependencies enforces rigid order of implementation, these are determined based on components prerequisites and technical requirements. It is the responsibility of the project manager and technical team to analyze such kind of dependencies. These dependencies are constraints that cannot be overridden by priorities and this order must be followed. After dependencies are analyzed, it is possible to assign the requirements a priority, which is based on the stakeholders importance and priorities.

5.2 Task Assignment

Most approaches used in task assignment involve machine learning methods, such as bug tracking and version controlling systems. The Project manager analyzes all project tasks **[6]**. Then, all related tasks are grouped to accomplish an iteration within a given release. Then each task is given an estimate to specify the time it needs to be performed. Finally, the project manager can assign it to the developer to implement it according to the plan in a certain iteration.

5.3 Semi-automated planning of iterations approach

In conventional agile processes, task assignment and release planning is done separately. One of the suggested approaches that proved to be an optimized approach of iteration planning is the semi-automated planning of iterations **[11]**.

Iterations are normally a collection of implemented requirement in a predefined time period with a planned outcome, which is called a release. Release represents a piece of deliverable software to a customer.

The two dimensions of a problem affecting the decision on the planning iterations are release planning and task assignment. Stakeholders are divided into two types, external and internal stakeholders. External stakeholders are interested in the application and their implemented requirements. Internal stakeholders are interested in the implementation aspects. Prioritizing the requirements is based on their interests. The Project manager sets the task based on the requirements identified and assigns it to the sprint to be implemented according to the plan in an iteration. The main resource in this approach is the developer, where a developer bears two main constrains: workload and expertise. The workload of the developer is defined by time availability and the expertise regards how familiar and skilled the developer is with the task assigned.

The objective of the semi-automated iteration planning approach is to optimize iteration planning for the two criteria given above: release planning and the task assignment. The result of the following approach will be a plan with a pre-defined number of iterations detailing every task that should be accomplished, in a given iteration, by the developer.

The approach suggested in **[11]** of semi-automatic planning iterations is summarized in the following four steps. It starts with the preparation step which includes modeling all the required information. This basically involves modeling requirements, setting the priorities for each requirement, defining the estimated task and defining the available resources. The second step regards determining expertise, where expertise in the identified tasks of every developer should be determined. This is done to optimize task assignment in planning iterations. Measuring expertise is done using the existing data about the developer saved from previous tasks performed. Such information can be found in a task management tool, called UNICASE or other task management tools. The third step is the actual iteration planning done by using a genetic algorithm. It optimizes the iteration plan according to the priorities and dependencies provided by the requirements. It also helps optimizing the developers' workload and expertise.

Using a genetic algorithm scheme helps identifying the best solution to the iteration planning problem. Initially a random population is generated then compared to the evaluation function until the final best solution is generated. The evaluation function will evaluate individuals *(solutions)* in every generated population of the genetic algorithm. The evaluation considers dependencies between: requirements, requirement priorities, developer's expertise and the availability of resources, where each solution will be evaluated by an evaluation function. For each Solution S, an evaluation that first calculates four tuples with these elements: dependency, priority, expertise, and overload. The result of the generated plan can be finalized by the manual modification and review done by the project manager.

6. Agile Project Management for Government Case Study: FBI Sentinel Project

In 2001, the FBI realized the usage of separate and obsolete technologies to manage electronic case documents and digital media relevant to evidence and intelligence information **[22]**. These are several ad-hoc modules which are not integrated to each other. Additionally they are used to manage these pieces of information. Management and search of such files are not efficient and are unreliable. Moreover, there are difficulties in exchanging these files and usability of existing ad-hoc processes. These requirements motivated the FBI to have a new system that enhances existing systems and replaced them with new Virtual Case File system *(VCF)*.

The first attempt to develop this system was based on the classical waterfall model. So, it required building a complete plan and comprehensive specifications to meet all user requirements. Such complete plan was impossible to be built in this large software project. Additionally, it had to take into account all user requirements and expectations. The system then had to be designed, implemented, and tested completely in sequence. After its completion, the entire system could be demonstrated to the stakeholders. It costs time and money to have a complete plan, and once it is finished the user might not accept the resulting system, because the users were not involved in the development process and no earlier feedback had been provided. Technical barriers were raised too late, hence having to rebuild the system from scratch. Additionally the stakeholders did not accept the new system.

This attempt cost $475 million and spent three years developing it. Traditional audit reports were about incomplete and incomprehensive plans. Furthermore, the design was not precise, not meeting the stakeholder requirements, and no specific milestones. Oversight of the project was also inadequate. Finally, there were no penalties imposed on the suppliers.

The second attempt was also on the same waterfall model but a stricter model to have a more precise plan, correct design, and milestones. Moreover, these milestones regarded one attempt and stakeholders were not involved until the end of each milestone. In 2005 it was suggested to access the old system in enhanced web-interface. 25% of the budget was planned to be paid for oversight on the contractor. Extra efforts were assigned to the contractor by the oversight team. In this attempt the initial estimates are not justified. At the end of phase 1 with extended two months, some functionalities were working and 57 critical functionalities were not. It required to access new and old systems. As a result, it was not efficient to use an old and system at the same time. Then the new system was stopped. Some reasons for the second attempt failure regard the non-technical background of the project management office staff. Plans were not accurate as usual. After three years, users completely rejected all the system's deliverables phases due to the system's poor usability interfaces, poor performance, and other quality problems. According to these reasons the second attempt also failed.

In September 2010, the FBI made a decision to alternatively use the Scrum agile software development process instead of the waterfall process. Furthermore, they decided to replace traditional contractors with product experts. The waterfall requirement document was converted to Scrum product backlog. This requirement document resulted in 670 user stories and grouped into 21sprints. The project manager role also was replaced by a Scrum master. The Scrum master's role is not to manage the project; the role is to lead self-organized teams and sort out any possible barriers encountered by any team.

The teams started working by prioritizing user stories. Additionally, they created a relative measure to weigh each task based on complexity and size, and its name is the story point. The story point metric helped each team to self-evaluate. Each sprint is given two weeks to complete. At the end of each sprint, different tests were performed against each user story. A user story could not be considered as complete until it passes all required tests and is approved by the stakeholders. In case of test failure for a given user story, it is transferred to the product backlog to be reworked again in the next sprint or any future sprint. After the completion of high priority sprints, it is possible to have an idea about the overall view of the entire project, and the time expected for its completion. There were actually two primary increments

to complete the whole project, and each one consists of the half of all sprints.

Eventually, the Scrum agile model had decreased the planned project budget into 52% to accomplish 88% of the required functionalities. In the projects' 18 months, two product releases were completed to be used. Eventually, the stakeholders accepted the new system to be used and most feature capabilities were operated.

7. Discussion

A successful project manager should know how to manage user expectation successfully to avoid system failure. For example, a manager should not overwhelm the user with problems he is facing, which is exhausting to the user. Furthermore, some users frequently change their ideas and occasionally with unrealistic requirements. Hence, it is desirable not to involve such users very often as in the Agile Approach. An important issue to satisfy user expectations is to carry out a solid and continuous testing from the beginning. As previously mentioned, testing after coding can be very expensive and might not meet the user's expectations. This may cause the whole product to be cancelled, which will certainly be expensive as will cost a lot of effort, money and time. This also causes delay in delivering the project and can eventually lose the users' trust. In addition, if the project is not well managed, there may not be sufficient time for testing at the end, which may lead to poor testing. Therefore we support the idea of performing the test through the software development life cycle to have better results and better user satisfaction.

The Agile Scrum model is a practical model to overcome long running projects and the main problems related to large projects, such as an early perfect plan. And involving the stakeholder has yielded successful stories to receive the stakeholder's early feedback. It is the main advantage of user satisfaction. As a result, these feedback points are addressed earlier as the stakeholder heavily participates in the development process. Additionally, the stakeholder is in direct contact with developers to reflect the actual requirements and eliminate early any conflict detected between the developer and the stakeholder. Technical issues that could be predicted are also detected early and the development team can sort them out accordingly.

However significant work must be done before starting any sprint. Sprints usually implement atomic components. In the long run, some of these components need to be integrated with some other components. If some of these completed components start operating, it is difficult to modify them for an integration requirement during operation. If only the stakeholders' satisfaction is taken into consideration, such integration issues will not be discovered until the product is fully completed. So, it is the role of the Scrum master to realize such important aspects. In my view, to overcome such issue, there should be a dedicated team who has a comprehensive view of the system. This team is the integration team and its role is to ensure the sprints adopt the integration. The integration team can ask the development teams to previously prepare a standard integration interface. This interface should be previously tested and verified for each sprint. If such component is in operation, future integration should not be a problem, because the integration can be done while the end users continue working without any interruption.

User involvement should undergo a policy to control his requirements. User satisfaction differs from one person to another. Thus, there must be an average satisfaction level. For example one user can accept many things because he is uncertain about his or her requirement or is not familiar with all requirements. Consequently, he or she might give an incomplete user story. Another one might require user stories that are not completely applicable. User stories must be controlled by some criteria or policy. Product backlog is an excellent tool to document such user requirements. Additionally these user requirements should be controlled by project constraints which are scope, time, and cost. It is the role of the Scrum master and product owner to manage such problems.

8. Conclusion

In conclusion, the development process of any software project should have a managerial focus in satisfying user expectations. Delivering a software project that satisfies all user requirements will save money, time and efforts. Therefore, successful management of user expectation is vital towards successful project. It is also essential to carry out good testing throughout the development life cycle to achieve user satisfaction. Furthermore, 'agile' prioritizes user satisfaction and recommends the customer or stakeholder to be an essential party in the development team, thus filling the gap of understanding requirement earlier. Any additional inquiries could be promptly answered and clarified by the stakeholder since the stakeholder is frequently in touch with the concerned development team. Furthermore, the development team can offer better enhancements or innovative ideas to the stakeholder, as a result, more trust is gained from the customer representative. The change for agile adopted have increasing the user satisfaction as the user feels more comfortable with late changes, because any late change may be considered a major change as it obviously appears in classical one-shot software development models.

authors

Nizar Al Hawajreh is a senior application engineer with Aljazeera Media Network since more than ten years. His experience is related to the development of archiving systems, integration of heterogeneous media systems and also design and development of automated Human-Resources business processes. His primary research is related to project mangement and formal verification of automated workflows (model checking techniques). He is currently finalizing his master thesis on these subjects at the Computer Science and Engineering department of Qatar Unniversity.

Abdelaziz Bouras is ictQATAR Chair Professor at Qatar University, QU. He is currently the Chair of the IFIP WG5.1 on « Global Product development for the whole life-cycle". His current research interests focus on distributed systems for lifecycle engineering, including ontologies and lifecycle modeling for intelligent products. He teaches Software Project Management and Simulation in the Department of Computer Science and Engineering of QU. Also was professor at the University of Lyon (France) where he leads a research team of the LIESP laboratory. He has been conferred the HONORIS-CAUSA honorary Doctoral Degree in Science of the Chiang Mai University (Thailand) from Her Royal Highness Princess Maha Chakri Sirindhorn of Thailand. Prof. Bouras is Editor-In-Chief and founder of the International Journal of Product Lifecycle Management (IJPLM), Associate-Editor and co-founder of the International Journal of Product Development (IJPD), and Editorial Board Member of several International Journals related to knowledge management and supply chain management. He is also co-founder of the new International Federation of Information Processing IFIP WG5.1 Group on Product Lifecycle Management (PLM) for which he acts as Vice-Chair in charge of Europe and Africa; and is member of the IFIP WG 5.7 Group on Integration in Production Management. He is also the co- founder of the international WG-PLM and its annual international doctoral workshop on PLM.

Ashraf Abualia, Hanadi Al-Thani, Zohreh Fouroozesh, Kholoud Khalil, Kholoud Mohammed, Muna Al Kuwari, Alanood Zainal Computer Science Department, College of Engineering, Qatar University, Doha, Qatar, P.O Box 2713

references

[1] Agbor, J. The Relationship between Customer Satisfaction and Service Quality: a study of three Service sectors in Umeå. Umeå School of Business, http://umu.diva-portal.org/smash/get/diva2:448657/FULLTEXT02.html, *(accessed on Dec. 2013).*

[2] Andersson, M.; Liedman, G. *(2013)* Managing Customer Expectations: How Customer Expectations are Formed and Identified during a Project Delivery, Chalmers University of Technology, Göteborg, Sweden, Report No. E 2013:061.

[3] Berry, L.; Zeithaml, V.; Parasuraman, A. *(1990)* Five imperatives for improving service quality. Sloan Manage Rev, 31*(4)*: p. 29–38.

[4] Bhattacherjee, A. *(2001)* Understanding information systems continuance: an expectation-disconfirmation model, Management Information Systems Research Center, Vol. 25, No. 3, p. 351-370.

[5] Boem, B. W. *(1988)* A Spiral Model of Software Development and Enhancement, ACM, Vol. 11, No 4, p.14-24.

[6] Bruegge, B.; Creighton, O.; Helming, J.; Kögel, M. *(2008)* Unicase – an Ecosystem for Unified Software Engineering Research Tools, Third IEEE International Conference on Global Software Engineering, ICGSE 2007.

[7] Elliott, J. Achieving Customer Satisfaction through Requirements Understanding, Defence Evaluation and Research Agency, Malvern, UK, http://www.iscn.com/select_newspaper/requirements/dera.html, *(accessed on Dec. 2013).*

[8] Everett, G. D.; Jr, R. M. *(2007)* Software Testing: Testing across the Entire Software Development Life Cycle, ISBN 978-0-471-79371-7, p. 69-91.

[9] Gopalakrishnan, D.; Sharma, M.; Kumar, S. Managing Customer Satisfaction at a Maruti authorized service station Service Marketing Management, Indian institute of management, Bangalore, http://www.slideshare.net/gopal_capricorn/service-marketing-management-group-project-customer-satisfaction, *(accessed on Dec. 2013).*

[10] Hamil, D. *(2005)* Expectation Management: A "Gateway Key" to Project Success – Client Satisfaction. A 2005 Professional Development Symposium White Paper, p. 2-3.

[11] Helming, J.; Koegeland, M.; Hodaie, Z. *(2009)* Towards Automation of Iteration Planning, OOPSLA '09 Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, ACM 978-1-60558-768-4/09/10, p. 965-971.

[12] Hughes, B.; Cotterell, M. *(1999)* Software Project Management, McGraw-Hill.

[13] Huq, F. *(2000)* Testing in the software development life-cycle: now or later, International Journal of Project Management, Vol. 18, No. 4, p. 243-250.

[14] Jovanovic, M. *(2008)* Software Testing Methods and Techniques, p. 30-40.

[15] Lai, L. *(2012)* Managing user expectation in information system development, World Academy of science, Engineering and Technology, Vol: 72 2012-12-23.

[16] Li, E. Y. *(1990)* Software Testing in a System Development Process: A Life Cycle Perspective, In Journal of Systems Management, Vol. 41, No. 8, p. 23-31.

[17] Munns, A.; Bjeirmi, B. *(1996)* The role of project management in achieving project success, International Journal of Project Management Vol. 14, No. 2, p. 81-87.

[18] Petter, S. *(2008)* Managing user expectations on software projects: Lessons from the trenches, International Journal of project Management, Vol: 26.7, p. 700-712.

[19] Schmidt, R.; Lyytinen, K.; Keil, M.; Cule, P. Identifying software project risks: an international Delphi study, http://sydney.edu.au/engineering/it/~isys3207/readingsondesign/identifyingprojectrisk.pdf, *(accessed on Dec. 2013).*

[20] Stober, T.; Hansmann, U. *(2010)* Agile Software Development: Best Practices for Large Software Development Projects, Springer.

[21] Tsai, B.; Stobart, S.; Parrington, N.; Thompson, B. *(1997)* Iterative Design and Testing within the Software Development Life Cycle, Software Quality Journal, Vol. 6, Issue 4, p. 295-310.

[22] Wernham, B. *(2012)* Agile Project Management for Government Case study: The Success of the FBI Sentinel Project, Agile Business Conference *(ABC2012).*

[23] Introducing Software Testing, http://hiromia.blogspot.com/2013/07/introducingsoftware-test.html, *(accessed on Dec.2013).*

[24] Software Testing Life Cycle http://qualitytestified.blogspot.com/2013/02/software-testing-life-cycle.html, *(accessed on Dec. 2013).*

[25] Testing at different phase of software development life cycle, http://www.kostcare.com.html *(accessed on Dec. 2013).*

[26] Principles behind the Agile Manifesto, http://agilemanifesto.org/principles.html, *(accessed on Dec. 2013).*