SQrum: An Improved Method of Scrum

Proposed Metamodel for SQrum Method

Najihi Soukaina, Merzouk Soukaina, Marzak Abdelaziz
Department of Mathematics and Computer Sciences, Hassan II University- Casablanca
Faculty of Sciences Ben M'sik, Casablanca, Morocco

***Abstract*—Software systems are having a major impact on many aspects of personal and professional life. Safety-critical applications, such as production line controls, automotive operations, and process industry controls, rely significantly on software systems. In these applications, software failure may result in bodily damage or death. The proper operation of software is essential to the safety and well-being of individuals and businesses. Therefore, software quality assurance is of paramount relevance in the software business today. In recent years, Agile Project Management and particularly Scrum, have gained popularity as a method of dealing with "vuca" business environments, which are characterized by rising Volatility, Uncertainty, Complexity, and Ambiguity. This paper contributes to the software development body of knowledge by proposing a metamodel of Scrum quality assurance, named SQrum ('SQ' of Software Quality and 'rum' of Scrum). Our objective is to make Scrum more efficient and reliable and to assist enterprises in undertaking quality assurance activities while considering agile practices and values.**

Keywords*—Agile project management; IT; OMG; meta-object facility; MOF; metamodel; scrum; SQrum; quality assurance; QA; quality management; QM; software development project

I. Introduction

The capacity to successfully execute Information Technology (IT) projects has become a crucial and strategic need for modern organizations. When expenses for field updates, recalls, repairs, downtime, etc. are included, the release of a product with problems may be extremely costly. Damage to a company's reputation is less measurable but equally significant. In addition, a failed software project might damage the competitive position of an organization. Value and quality delivery are essential variables for determining the success of a software development project; they are crucial assets for any firm seeking to remain competitive in the marketplace. Despite extensive efforts to find methods for maintaining software quality, software projects continue to fail [1].

There is a growing demand for the deployment of software development methods that are both flexible enough to keep up with the rapid rate of change and the competitive market and rigorous enough to prevent defects and assure product quality. However, humans are fallible. Even with the most advanced and conscientious design processes, erroneous outcomes cannot be avoided beforehand. As a result, software products, like the outcomes of any engineering effort, must be validated against their requirements throughout their development. Agile software development has arisen as an alternative to planning and managing complicated projects by offering methods to accommodate frequent project changes.

Agile is a method characterized by continuous iterations and testing throughout a product's Software Development Life Cycle. Scrum is the most popular agile approach [2]. Scrum is a lightweight, agile framework that provides processes for managing and controlling the software and product development process. Although Scrum offers a number of benefits, such as incremental deliverables at the end of each iteration, stakeholders and product owners can modify requirements throughout the process, and Scrum can swiftly adapt to these modifications. Process and product quality remain Scrum's principal problems.

One of the most significant contrasts between agile and traditional development is the agile "whole-team" approach [3], in which quality is the responsibility of the entire team and quality assurance is incorporated into the process itself, without an explicit Quality Manager (QM) role. Quality Assurance (QA) activities are integrated into the team's day-to-day operations in order to improve product quality through a smooth process. Large organizations frequently face the issue of combining the Agile requirements of adaptability, transparency, and collaboration while also assuring product quality and adhering to required QA processes. With these changing issues in this area, businesses struggle to identify the best method to integrate QA into Agile environments, particularly Scrum.

Within an Agile team, each team member is responsible for testing and product quality and participates in test-related activities. Each member of the team may see quality from a unique perspective and mindset. All of them are acceptable contributions to quality, but Scrum projects are still facing quality challenges due to the lack of defined rules in Scrum. This necessitates an explicit QM position to incorporate non-functional requirements (NFR), assure process improvement, and provide shared ownership of quality. Just as the Product Owner is responsible for maximizing the value, the QM is responsible for increasing the quality.

Every member of an agile team is a tester, but a QM is more than just a tester. A QM on the Agile team is able to give an overview perspective on all team contributions in order to establish the product quality strategy. Instead of criticizing, QM provides proactive ways to improve productivity, promote software quality within the team, and give software testing and quality coaching.

Quality assurance is like a ship; everyone participates in making it move forward, but only one person can steer it in the right direction. A QM is the one who gives the "course" to follow for the whole team to reach a satisfactory quality level. He is the one who carries the global vision of the product and process quality.

The aim of this paper is to propose a new agile approach called "SQrum", whose central concept is the addition of a new quality role and all of the artifacts it will require. We will build a metamodel using the OMG Meta-Object Facility (MOF) that provides an abstract view of the new agile development process. The Meta Object Facility (MOF [4]) standard was created by the Object Management Group (OMG) to facilitate the development of modeling formalisms in the form of metamodels. This consists of a set of meta-classes connected by meta-association [11]. The rest of this paper is organized as follows: Section II details the related work of this study. Section III provides an overview of Scrum. Section IV describes the disadvantages of Scrum. SQrum is presented in Section V. Finally, Section VI concludes the study.

II. Related Work

Hanssen et al. [5] found that Quality Assurance processes in Scrum are becoming inadequate. Consequently, they suggested a new method called SafeScrum, which is a Scrum variant with some supplementary XP methods that can be used to develop safety-critical software and IEC 61508 certified software. They evaluated the standard, sought out an impartial evaluator, and collaborated with the Scrum team to identify the required additional tasks to be included in the Safe Scrum process for an internal quality assurance component. They did not cover all aspects of quality in a scrum project with the proposed role.

Jeon et al. [6] explain that Scrum does not place enough focus on non-functional requirements, whereas the key success factor of software projects is not only the satisfaction of functionalities, but also of quality attributes; therefor, they propose the ACRUM method for the analysis and incorporation of quality attributes into software projects. Jan Bosch concurs with Jeon et al. and suggests an approach that explicitly takes nonfunctional criteria into account during design.

Timperi [7] explains the weakness of scrum is the lack of concrete guidance and instructions about quality assurance activities and that the focus has been on the development activities while quality assurance practices of different agile methodologies have received less attention and an overall picture is missing. The author recommends combining quality assurance practices of different methodologies, like Scrum and XP, in order to get good enough software delivered to the customer.

Aamir et al. [8] assert that due to the rapid delivery process of sprints, quality is not considered in the scrum framework, and the majority of Quality Control (QC) operations are overlooked. To address this issue, the authors offer an enhanced scrum model for implementing QC activities and evaluating the product's quality. They also provide a new concept of "test backlog" for documenting test cases within the scrum.

Bajnaid et al. [9] addressed the limitations of agile practices that do not include quality assurance in their process to guarantee that the quality assurance procedure has been followed and quality assurance criteria have been satisfied. To overcome the drawbacks, they suggested a process-driven e-learning system that senses developers' activities and guides them through necessary software quality assurance methods during software development.

III. Scrum Overview

Scrum is a management process that was initially mentioned in 1986 by Takeuchi and Nonaka in their paper "The New New Product Development Game," in which they describe a flexible, fast, and self-organizing product development process .Sutherland and Schwaber [10] used these discoveries and the word "Scrum" to create the currently known framework, which was initially introduced in 1995.

Scrum is a lightweight development methodology that allows IT organizations to handle complicated adaptive challenges while delivering solutions of the highest possible quality. Scrum was developed based on the premise that software development is too complicated and unpredictable to be meticulously planned at the start of a project [11]. Therefore, a progressive, iterative method is used to maximize predictability and limit risk. Given that change cannot be avoided, it must be managed. Scrum handles change by developing software in iterations as opposed to a one-shot method.

Scrum is based around three roles (Product Owner, Scrum Master, and Development Team), four meetings (Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective) and three artifacts (Product Backlog, Sprint Backlog, Product Increment) [10] (see Fig. 1.)

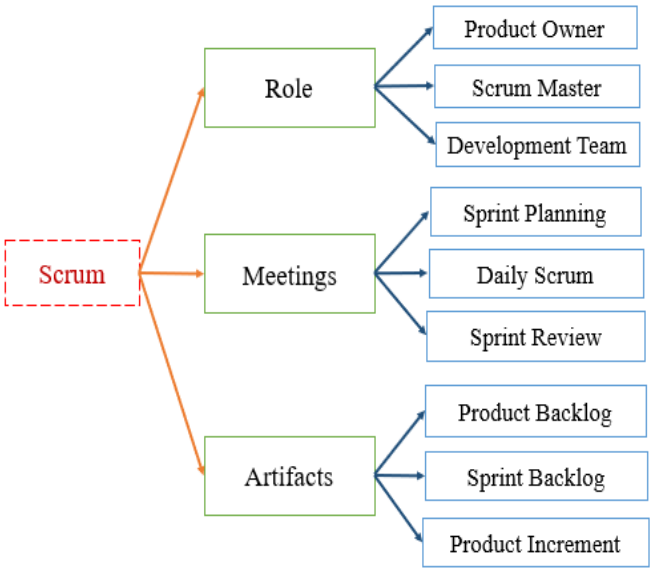

Fig. 1. Roles, Artifacts, and Activities in Scrum [12].

Each project has a Product Backlog, which is a list of a product's requirements ordered by business value. It is a constantly evolving document based on changing requirements, changing problem understanding, and changing contexts. Each Product Backlog Item (PBI) is estimated using an abstract effort measure based on "Story Points" and has a set of acceptance criteria.

Scrum teams are self-organized, multi-skilled, and capable of producing products iteratively and incrementally, thus increasing opportunities for ongoing feedback. A Scrum team consists of a product owner, who acts on behalf of the client and is charged with maximizing the value of the developed product, and a development team, which is responsible for creating the product. The development team is made up of developers and a Scrum master. The Scrum master is a facilitator who ensures that the development team is supplied with an appropriate environment to finish the project effectively, removes impediments for the team, and guarantees adherence to Scrum practices.

The Scrum development process is carried out by cross-functional teams of individuals with diverse skill sets[10]. The teams often possess a variety of specializations, including programming, testing, analysis, database administration, user experience, and infrastructure. All of these skills are required to provide the product, and Agile projects employ a whole-team approach to execute it. Advantages of employing a whole-team approach include the fact that quality is everyone's responsibility. Scrum focuses on developing high-quality software within a timeframe that optimizes its business value. This is everyone's responsibility, not just the testers. Every member of a scrum team is a tester. Tests, from the unit level on up, drive the code, teach the team how the program should function, and indicate when a task or story is "done." A Scrum team must have all the competences necessary to generate high-quality code that provides the organization's requested features. This means that the team is responsible for all testing activities, including test automation and manual exploratory testing. It also implies that the entire team continuously considers testability while creating code.

Scrum divides a project into iterations known as "Sprints". A Sprint is a time-boxed, often 30-day iteration in which the Scrum team adds new features to the product. The sprint begins with a "Sprint Planning Meeting" where the team picks from the product backlog the items to be handled in the sprint and plans the work to be performed. The team will estimate the selected items based on their velocity (e.g., the number of "story points" they can execute within a predetermined time limit). The result of planning is turned into an objective known as the "Sprint Goal" The Scrum Team then has an internal meeting and utilizes the Sprint Goal to generate a list of the necessary requirements to achieve the target. These requirements are decomposed into "tasks" that become entries in the "Sprint Backlog." The success of the sprint is based on the achievement of the sprint goal [2].

During the sprint, the team holds daily 15-minute stand-up meetings called “Daily Scrum” with the purpose of assessing progress and maximizing the chance that the development team will accomplish the sprint goal. Each team member responds to three questions [13]:

- What progress has been made since the last meeting?
- What will be accomplished by the next meeting?
- What impediments stand in the way?

The Sprint yields a deliverable product increment as its final output. During a 4-hour Sprint Review, the Scrum Team inspects the product increment, evaluates what they were able to accomplish during the sprint, and modifies the product backlog as needed prior to the next Sprint Planning meeting [17]. The “Product Owner” will approve the needs for the live system based on the requirements and their preset acceptance criteria.

The last meeting of a "sprint" is the "retrospective," at which time the team examines and comments on itself and the project in terms of people, relationships, processes, and tools. As a consequence, an improvement strategy may be developed.

Merzouk et al. [12] proposed, in their paper titled "Towards a New Metamodel Approach of Scrum, XP, and Ignite Methods," a metamodel of Scrum (see Fig. 2).

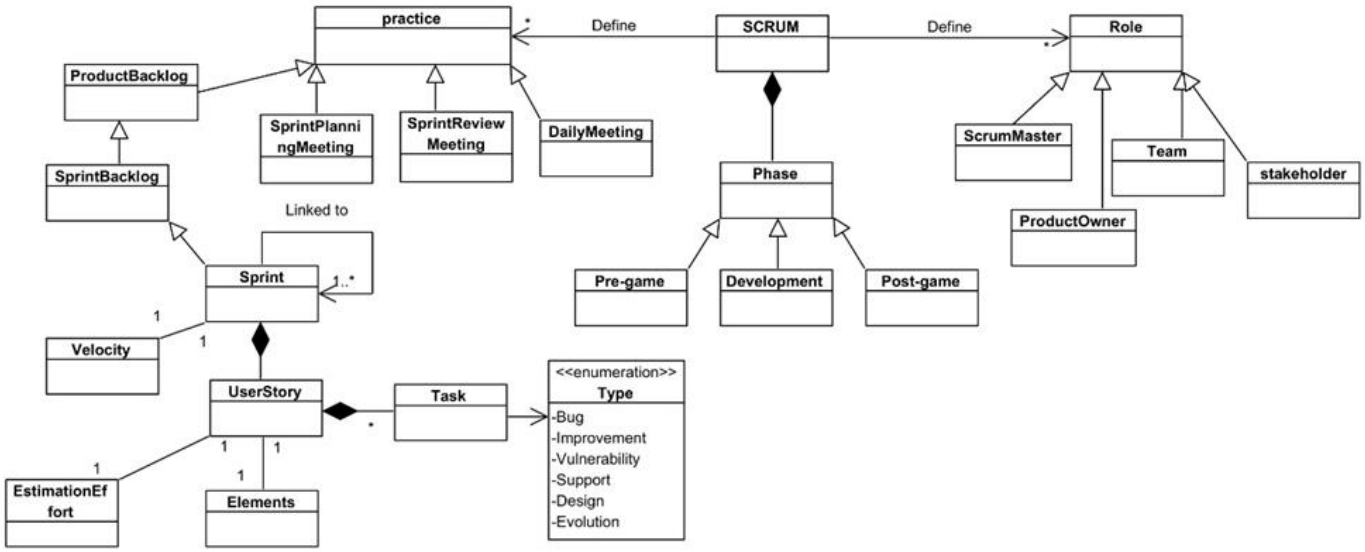

Fig. 2. Proposed Metamodel for Scrum Method [12]

IV. Critical Analysis of Scrum

Scrum is the most popular and effective Agile methodology due to its many advantages, including quick delivery and flexibility to change. It emphasizes customer satisfaction, continuous feedback, and process transparency. In spite of its many advantages, it has three disadvantages. First, Scrum backlogs focus only on functional requirements (FR) and tend to neglect non-functional requirements (aka Quality Attributes; see Fig. 3) [6] [14]. Second, the majority of quality assurance activities are skipped in scrum due to the sprint's short period and the lack of a dedicated quality management role [8]. Finally, in Scrum, the requirements are typically managed by a person with a business-oriented profile. Thus, the focus is on the development activities that produce business value, while quality assurance practices receive less attention and an overall picture is missing [7].

This study proposes the SQrum as the solution to the aforementioned problems. SQrum is built on the traditional scrum to present a more effective method.

Our new approach will improve classic Scrum in three ways.

- The quality attributes will be considered and included in the product backlog.
- Quality assurance activities will be effectively handled.
- The tester's responsibilities will be precisely outlined.

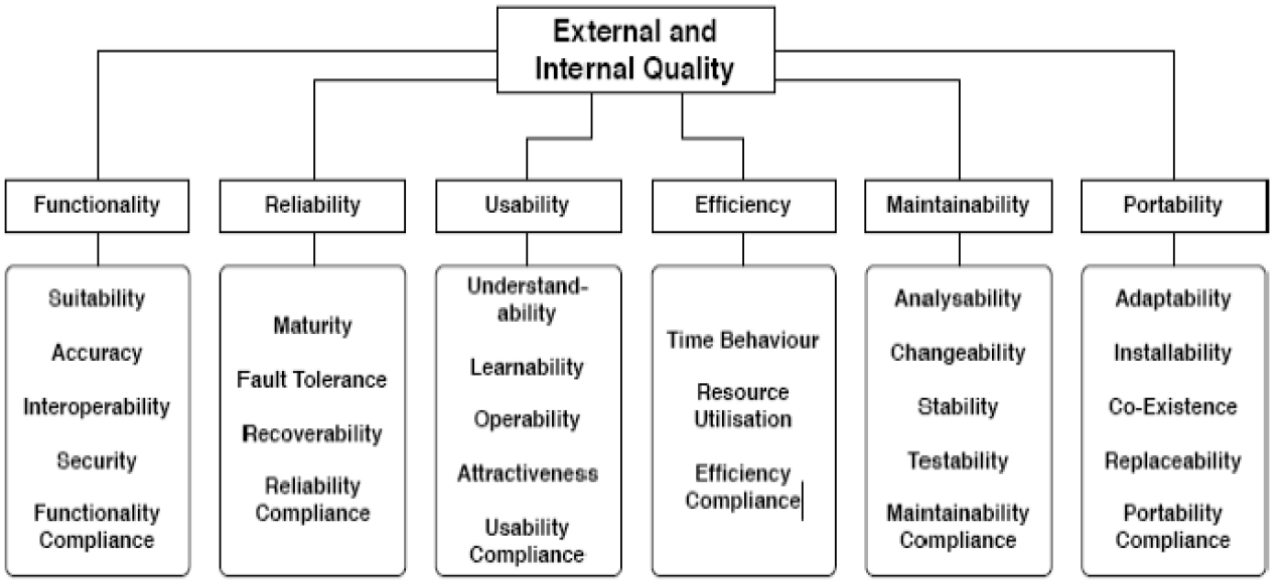

Fig. 3. ISO 9126 Model for Quality Attribute [15].

V. Proposed Method

A. SQrum Quality Artifacts

SQrum intends to incorporate the DoD as an artifact into its new model and proposes four new artifacts: QA strategy, test plan, defect backlog, and test library.

1) Definition of done: DoD is a significant part of the test plan. It is used by the QM as a check list of items, each one used to validate a story or a PBI for completeness. Unlike the acceptance criteria, the DoD is applicable to all items in the Product Backlog, not just a single user story. It is applied to the product increment as a whole [16]. The QM collaborates with the PO and the dev team to identify all the conditions that make an increment shippable or not at the end of the sprint. This proposal's DoD should emphasize quality attributes.

2) QA strategy: A QA strategy is a long-term plan of action, the key word being “long-term” about the overall test approach to projects. It defines the project’s testing guidelines on how to test the target system.

Using the ship analogy once more, the QA strategy symbolizes the "course" to follow in order to reach the final destination.

QA strategy is a static high-level document that can be used as a reference that doesn’t change much over time and needs to be updated only if processes change. It generally includes decisions made in terms of sprint timelines, test types required, infrastructure such as test management tools, defects management tool, test environments, test monitoring and reporting.

3) Test plan: A test plan is a concise and lightweight document used to organize the test activities. Each sprint has its own test plan, which is a living document that changes and evolves based on sprint requirements. A test plan outlines the scope, approach, resources, and schedule of planned testing activities. It identifies, among other things, the items and features to be tested, the assignment of tasks, the DoD and the test types to be performed, the test environment, the test design techniques, test data requirements and test measurement techniques to be used, the risk and dependencies assessment carried out, automation tests programmed, and time budget allocated. Continuing with the ship metaphor, the sprint is the trip, and the test plan is all that is required to ensure its success.

An Agile Test Plan is a crucial document since it collects all the answers to test-related questions in one place.

4) Defect backlog: A defect backlog is an ordered list of all the known defects in the project that haven't been fixed yet.

Defects may be functional, describing misbehavior or technical related to quality attributes such as performance, security, or other. The remaining bugs can be calculated by subtracting the fixed bugs from the total bugs:

Remaining bugs = number of total bugs - fixed bugs

There are three types of defects that we can find in the defect backlog:

- Defects within the current iteration: These are defects that can’t be fixed immediately and do not impact the increment date of release. Ideally, defects should be corrected as soon as they are discovered, before they become massive, tangled defects.
- From the Legacy System: These are inherited defects of the old system that have remained hidden until now. When found, they are logged to the defect backlog and the QM with the team can choose to fix them or not. If so, they will be prioritized as part of the product backlog.
- Found in Production: These are bugs found by the customer in production. Depending on their severity, these bugs may be fixed immediately, at the time of the next release, or they’ll be estimated, prioritized, and put in your product backlog.

5) Test library: Scrum uses an iterative approach for product development; the same approach can also be applied to testing. As a product is produced, the test library expands progressively. It includes test cases, test scenarios, and test campaigns. User stories are the basis for the creation of test cases. The relationship between test cases and user stories in the sprint backlog is one-to-many, as a single user story may have numerous test cases. The test cases are similar to puzzle pieces that compose the test scenario (see Fig. 4), and test scenarios are the main components of test campaigns. Each campaign is designed to evaluate an increment. Utilizing the library enables testers to save time since for each new script, there are reusable components that can be used to develop a new test campaign. Also, all they have to do is make test cases and scenarios for the new features. For example, a first sprint is conducted to develop a command-launching feature. The test campaign is then utilized to check that this increment functions properly. Sprint 2: This sprint's objective is to develop functionality that allows for the management of billing for placed orders. Testers won't have to start from scratch to test this new feature. Instead, they can use the first tests they ran to launch the order and finish this test campaign with tests that make sure billing is handled correctly.

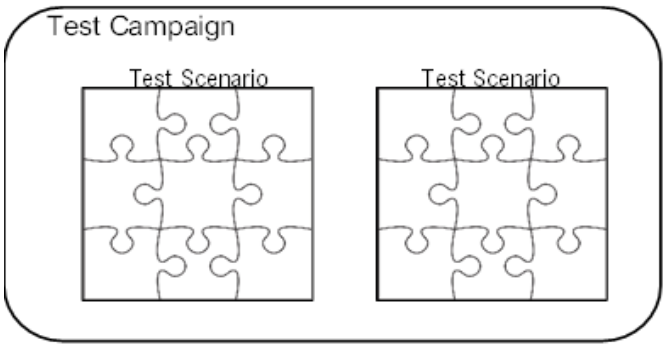

Fig. 4. The Components of a Test Campaign.

B. SQrum Events

1) Refactoring iteration: it is a didicated sprint for continuously enhancing the design of an existing solution without modifying its core behavior. Agile teams incrementally maintain and enhance their code from Sprint to Sprint. If code is not refactored, the resulting product will be of poor quality, with unhealthy dependencies between building blocks and improper allocation of component responsibilities [17].

2) Product backlog grooming: is a meeting that helps the Product Owner maintain his Product Backlog . He can ask for the aid of the team in creating and refining Product Backlog Items, estimating the amount of work required to complete a PBI, and prioritizing PBIs in the Product Backlog to ensure that the Product Backlog is ready according to the Definition of Ready [18].

C. Quality Manager Responsibilities

1) Manage quality attributes: Quality Manager is responsible for NFR Also known as quality attributes (see Fig. 3), QM works with the scrum team to analyze the quality attributes based on the functional requirements. With the agreement of the product owner, he complete the product backlog by mapping out the functional requirements and the quality attributes. QM can use three ways to elicit NFR 1) Acceptance criteria 2) User story 3) Definition of done.

QM plays an important role in every stage of the sprint:

- During the Product Grooming meeting, the QM begins gathering NFR by asking the PO and the development team several questions, such as: the system's scalability when more resources are added; the likelihood of the system performing without failure; and how long data should be retained in the system for reference (this might be a government/national regulation); What are the consequences if the user cannot access the system?
- Prior to Sprint planning: QM works with the PO to complete the product backlog.

QM incorporates NFR testing into the test plan.

- Sprint planning: QM presents and explains to the development team the identified NFRs for each user story.
- Daily meeting: QM tracks NFR work progress and assist the team in overcoming difficulties.

QM assists the team in remembering the importance of the quality attributes.

- Review meeting: While the PO focuses on the FR, the QM examines the NFR.
- Retrospective: QM recommends enhancements to increase product quality. For instance, devote more time to quality attributes such as security.

2) Manage testing activities: The SQrum method enhances the classic scrum "whole team" approach, in which every team member is responsible for quality and every team member is a tester [19]. SQrum proposes to give the responsibility for managing testing tasks to a single member who is the Quality Manager. QM's mission is not to pilot and supervise the SQrum team, but to accompany and coach them to ensure that they have all they need to execute QA activities. His role is to:

- Break down testing activities into several tasks.
- Ensure that the appropriate testing tasks are scheduled during release and iteration planning sessions.
- Assign testing responsibilities to team members so that everyone is aware of what to accomplish.
- Ensure that all testers meet their deadlines for work completion.
- Take notice of test-related challenges and attempt to address them.
- Ensure that each tester has the skills and knowledge necessary to develop and perform the tests required for each user story.
- Employ pair testing to address the skills gaps of the tester.
- Help to estimate the overall test effort and the technical resources need it.

3) Define QA strategy: In Sprint Zero (also known as the pre-planning phase of a sprint project), the QM collaborates with the scrum team to develop the QA strategy. However, his responsibilities do not end there because he is also responsible for keeping the QA strategy updated. The QA strategy is used by the quality manager to provide a new tester with an overview of the test process.

4) Establish the test plan: Before the sprint planning, the QM asks the PO about what stories will be in the next sprint, so he takes time to understand functionally and technically the requirements, and he starts working on his sprint plan. When the sprint planning comes, he already has an idea about quality attributes, possible issues and dependencies, data creation estimation, and test effort. This raises awareness of potential resource, time, and scope of work constraints confronting testers, as well as risks that must be discussed and addressed. This also allows the PO to reevaluate the level of quality he requires, and how much work should fit within the actual, achievable velocity of the sprint.

The Quality Manager may delegate the preparation of the test plan to a member of the team, but he remains responsible for the veracity of the information included within.

5) Help the team in expressing its DoD: The Quality Manager assists the team in creating a common understanding of quality to ensure that each user story makes sense within the context of the product's bigger story. The QM helps the team in formulating its DoD by asking the appropriate questions, such as:

Are functional tests passed?

Is acceptance testing finished?

Are quality attributes considered?

6) Tracking test tasks and status: At any point in the sprint, the Quality Manager must be able to quickly determine how much testing work remains on each story and which stories are "done." He must also ensure that no story is "done" until it has been tested at all appropriate levels. This helps him to check if the team is on schedule and to anticipate if there is a story that cannot be completed and he must remove it or ask programmers to help with the testing tasks.

Tracking the number of tests produced, executed, and passed at the story level helps indicate the status of a story. The number of tests written shows the progress of tests to drive development. Knowing how many tests aren't passing yet gives you an idea of how much code still needs to be written. The burndown chart is an example of a method used for measuring team progress.

Story or task boards are a helpful visual way to determine the state of an iteration, particularly if color coding is utilized; there are different colored index cards for the various types of tasks, such as green for testing, white for coding, and yellow and red for defects . Progress tracking can be achieved by any method and with both virtual and physical storyboards, as long as it enables the QM to see at a glance how many stories are "done," with all coding, database, and testing completed, and whatever the team's DoD is.

7) Identify risks and threats to sprint: Every user story in the product backlog is a potential risk. A story risk is the level that a user story will fail (the impact of the failure multiplied by the probability of failure). The key purpose of the risk prediction is to accurately anticipate the testing work so that all user stories can be tested in accordance with the risk level defined by the entire team. A simple test is sufficient for stories with a low likelihood of failure. Unlike stories with a high failure risk, which require a very careful test plan containing a variety of test techniques. Stories with a high failure risk require a very careful test plan containing a variety of test techniques. The QM assists the team in achieving the ideal balance between sufficient quality and acceptable risk, on the one hand, and time and resource limits, on the other.

The QM can initiate the identification and assessment of risk for both functional and non-functional requirements prior to sprint planning, and the team can finish the risk analysis during sprint planning. If there is not enough time to address the relevant risks at this meeting, the QM can ask the Scrum Master to organize a risk poker session. Once the risks have been identified, the team classifies and evaluates them based on likelihood and impact. This assessment is recorded in the test plan and taken into account during the design, implementation, and execution of tests for this iteration.

8) Track defect: Since quality is the concern of the entire team, everyone works collaboratively throughout. The Quality Manager's responsibility is to assist the SQrum team in setting targets relating to defects and using the right metrics to assess progress toward these goals. Gathering metrics on defects helps reflect the trend, which means the growing attitude in the number of defects in the defect backlog over a period of time. Another QM's responsibility is to communicate the trend in the defect backlog to the SQrum team. If the defect backlog is decreasing, there are no concerns. If it is increasing, the SQrum team must invest time in analyzing the underlying cause. In order to address the root cause, the QM must tell the SQrum team about the nature of the defects. If the defects could not have been detected using unit tests, then perhaps the programmers need additional training in unit test writing. If defects are missed or functional requirements are misinterpreted, then perhaps not enough effort is spent on sprint planning or acceptance tests are insufficiently thorough.

The QM can use a visual technique such as "Defect Trend chart." As shown in Fig. 5, the "Defect Trend chart" is a graphical representation of reported defects over time. The x-axis represents a period of time, while the y-axis represents the number of defects.

9) Manage defects backlog: The Quality Manager is in charge of the Defect Backlog, which includes what's in it, when it's available, and how it's organized. Before sprint planning, the quality manager makes an initial selection of defects based on their severity. Product backlog grooming is an excellent opportunity to discuss this selection with the product owner in order to determine which defects to fix first. The selected defects are presented to the team during the sprint planning and are scheduled for the next sprint.

The QM must ensure that the team does not accumulate technical debt, particularly when working with legacy code. The longer a defect remains in the system, the greater its impact. Defects in a code base have negative effects on code quality, system security and flexibility, team effectiveness, and velocity.

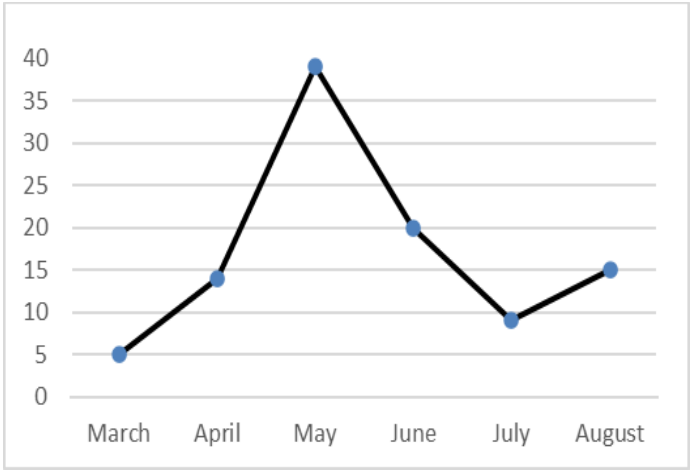

Fig. 5. Defect Trend Chart.

Early detection and correction of defects is more cost-effective. In iSixSigma Magazine, Mukesh Soni [20] cites an IBM report stating that a defect discovered after product release costs four to five times as much as one discovered during product design and up to 100 times as much as one discovered during product maintenance.

The QM must constantly evaluate the amount of technical debt dragging it down and work on reducing and preventing it. The QM needs to persuade the product owner of the benefits of addressing technical debt by demonstrating that technical debt may be costing the business money due to decreasing velocity. The team's velocity is sometimes consumed by bug fixes and trying to make sense of the code.

The QM can request that the PO reduce the scope of his desired features to allow sufficient time for good practices such as continuous small refactoring, which results in improved test coverage, a solid foundation for future development, decreased technical debt, and higher overall team velocity.

If it is insufficient and the PO cannot budget time in each iteration, the QM may suggest to the PO that a "refactoring iteration" be planned as a last resort to upgrade or add necessary tools, reduce technical debt, automate more tests, and perform major refactoring efforts. Planning refactoring iterations at regular intervals improves quality, maintains the system and its infrastructure, and preserves the team's velocity, allowing the team to move faster.

10)Help the team stay focused on the big picture: he Quality Manager tries to put each story in the context of the whole system by looking at possible risks, dependencies, and unplanned effects on other parts. The QM assumed the perspectives of the user, product owner, programmer, and tester, as well as everyone engaged in building and using the features. He can consider the effects of FR and NFR on the larger system and bring this to the attention of the team. Everyone on the team may easily focus their attention on the work or story at hand. This is a disadvantage of working on small feature portions at a time. The goal of the QM is to help the team take a step back and evaluate how their current stories fit into the big picture. QM keeps challenging the team to do a better job of delivering real value.

11)Keep testing environment updated: Testers cannot test effectively in the absence of a test-controlled environment. The QM must continuously inspect test environments and collect information regarding the deployed build, database schema, whether or not somebody is altering the schema, and other processes operating on the system. This information enables him to sustain the test environment with the most recent or updated version and eliminate the obsolete test environment, its tools,and techniques. This is also true for databases. Sometimes other teams can modify fields, add columns, or remove obsolete ones. The QM must be aware of all these changes in order to keep his database updated.

D. SQrum Process

We have identified eleven quality manager responsibilities. This list of eleven responsibilities indicates the tasks for which the QM is accountable; the remaining QA activities can be performed by the rest of the team, since quality is still owned by the entire team (everyone is a tester), but managed by just one person.

SQrum method follows eight phases, which are: Project Initiation, Sprint 0, Product Grooming, Sprint Planning, Sprint Execution, Sprint Demo, Sprint Retrospective, and Release. These phases are described in Table I with the artifacts of each phase.

E. SQrum Metamodel

This section presents the proposal of the new SQrum method as a metamodel. As shown in Fig. 6, the SQrum Metamodel is based on the transformation of the method's concepts into metaclasses linked by meta-associations, which define the kinds of relationships between these concepts. The green metaclasses represent the new concepts added to the Scrum method related to QA viz, Test Library, Defect Backlog, Definition of Done, QA Strategy, Refactoring Iteration, Test Plan, Product Grooming, and Quality Manager.

TABLE I. SQRUM PROCESS

	Project Initiation	Sprint 0	Grooming	Sprint Planning	Sprint Execution	Sprint Demo	Sprint Restro	Release
Activities	• *Create project Idea* • *Define project start/end dates* • *Team composition* • *Get a Quality Manager* • *Define sprint length*	• *Train the team in the SQrum method* • *Communicate the role of QM* • *Define the QA strategy* • *Setup the test environment* • *Build the test infrastructure*	• *Identify product's quality attributes* • *Identify risk and dependencies* • *Review the future scope*	• *Plan tests* • *Keep testing environment updated* • *Define DoD* • *Identify acceptance criteria* • *Estimate test effort* • *Plan tests automation* • *Identify test data* • *Participate in sizing stories* • *Complete product backlog with NFR*	• *Track tests activities* • *Track defect* • *Report defect* • *Write and execute tests campaigns* • *Perform nonfunctional testing* • *Communicate tests results* • *Report test impediments* • *Create test data* • *Run automated testing scripts* • *Automate new functional tests* • *Review of resolved defects* • *Pair-test with other testers*	• *Check functional and non-functional requirements* • *Report defect*	• *Inspect the process and people* • *Identify improvement s*	• *Participate in release to production* • *Train end users*
Artifacts	• *Project idea*	• *QA strategy*	• *Product backlog* • *Defect backlog*	• *Product backlog* • *Sprint backlog* • *DoD* • *Defect backlog* • *Test plan*	• *Sprint backlog* • *Increment* • *Defect backlog* • *Test library*	• *Increment*	• *QA strategy* • *Test plan*	• *Increment*

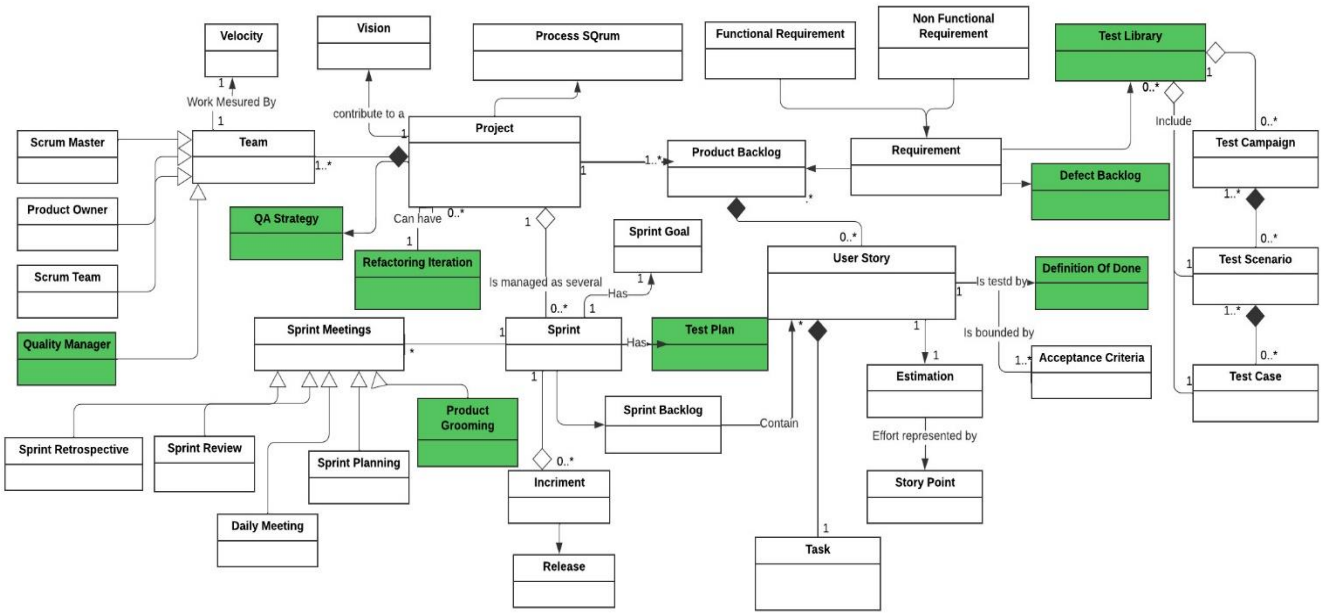

Fig. 6. Proposed Metamodel for SQrum Method.

VI. CASE STUDY AND EVALUATON

A. Project Description

To validate our model, we conducted a four-month case study on a software development project in a multinational telecommunications and IT services company, which is a large mobile network operator that serves European and worldwide companies. When we began the case study, the project had been underway for more than a year, and the team was using the agile scrum methodology. The main goal of this project is to rebuild a legacy system used by the project managers to plan and manage the deployment of internet solutions for enterprise

customers (btob business model). By rebuilding the system, the company wants to establish a new technological system, standardize further development and maintenance procedures, improve the system's quality and facilitate project managers' work. The team has agreed to test our method to determine if it will help them achieve this goal. Before starting the use trial, the team received SQrum training. The case study was distributed into four increments. Three are development sprints, and one is a refactoring sprint. Sprints were 4 weeks long. One quality manager, one product owner, seven developers, and one scrum master made up the team that worked on the project.

The company required a 10-month trial period, and if the deployment is successful, it will adopt SQrum as a standard methodology for two of its projects.

B. First Results

This section presents results after only four months of using SQrum. It is a classification of the team's feedback about which aspects they think have been improved using SQrum.

1) Awareness about quality: All the team members reported that SQcrum made them more aware of software quality during the sprint. Developers paid more attention to quality and were more focused.

2) Better organization and more communication: Communication and organization were also cited by the team as SQrum improved aspects. Since the QM took over the management of everything related to quality, the Scrum Master was able to focus on monitoring development activities and assisting the team in overcoming obstacles.

The team was also able to communicate more effectively as a result of the new SQrum events, since each event provided an opportunity for sharing and interacting.

3) Efficiency improvement: The team members reported improved efficiency. By mapping out the functional requirements and quality attributes, the QM helps the PO define clearer user stories from a technical perspective. Due to the clarity of the user stories, the team was more efficient and had a better knowledge of what to do and how to do it, as well as a better understanding of potential risks. which has improved not only the efficiency of the team but also the quality.

4) Better use of team velocity: Analysis and verification of quality attributes on a periodic basis, as well as the implementation of a refactoring sprint, led to a reduction in defect counts and defect-fixing time, resulting in a 21% increase in development productivity. The team spent more time in providing value rather that fixing issues.

5) Testing properly: The team found that the tests had been enhanced; they were better documented and structured as a result of the use of the test library. That helped them keep the balance between communication and documentation. Developers also reported that with the help of the test plan, they were able to test more thoroughly and were aware of the aspects to be taken into account when testing in order to improve quality.

6) Improving quality: Initiating the refactoring iteration, checking the quality attribute, and using the newly proposed artifacts decreased the defect density and time required to fix defects. The prevention and control of bugs contributed to a 36% decrease in the defect density of the project, and the time necessary to remedy defects was decreased by 41% by easily locating the spot of change and estimating side-effects.

C. Aspects to be Improved

Despite the positive results of SQrum, months is not enough time to test all its aspects. Also, for a better assessment of quality, the method needs to be used on large-scale projects with SAFe.

The members suggested detailed guidance on analyzing the quality attributes and enhancing the traceability, they also suggest a burn-down chart to assess the current state of the QC activities in SQrum.

VII. Discussion and Conclusion

Scrum is the most used method because it is adaptable to all project types. It is an iterative and incremental method that helps teams to deliver a high-quality project. Scrum's primary issues continue to be process and product quality.

This study introduces SQrum which is an adaptation of Scrum that includes and promotes the existence of a quality owner role.

SQrum provides a quality enhancement by adapting the traditional Scrum to emphasize non-functional requirements, establishing a new role and new artifacts to focus on control assurance activities, and ensuring that the quality assurance process has been adhered to.

In an ideal world, the goal would be to have zero defects, but due to the sprint's short lifecycle, this goal is almost impossible to achieve. Quality should not be seen as a constraint, but rather as a tool for maximizing business value. The QM's responsibility is to assist the PO in finding the ideal.

References

[1] T. Khalane and M. Tanner, "Software quality assurance in Scrum: The need for concrete guidance on SQA strategies in meeting user expectations," in 2013 International Conference on Adaptive Science and Technology, Pretoria, South Africa, Nov. 2013, pp. 1–6. doi: 10.1109/ICASTech.2013.6707499.

[2] A. Srivastava, S. Bhardwaj, and S. Saraswat, "SCRUM model for agile methodology," in 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, May 2017, pp. 864–869. doi: 10.1109/CCAA.2017.8229928.

[3] L. Crispin and J. Gregory, Agile Testing: A Practical Guide for Testers and Agile Teams. Pearson Education, 2009.

[4] "MetaObject Facility | Object Management Group." https://www.omg.org/mof/ (accessed Sep. 01, 2022).

[5] G. K. Hanssen, B. Haugset, T. Stålhane, T. Myklebust, and I. Kulbrandstad, "Quality Assurance in Scrum Applied to Safety Critical Software," in Agile Processes, in Software Engineering, and Extreme Programming, vol. 251, H. Sharp and T. Hall, Eds. Cham: Springer International Publishing, 2016, pp. 92–103. doi: 10.1007/978-3-319-33515-5_8.

[6] S. Jeon, M. Han, E. Lee, and K. Lee, "Quality Attribute Driven Agile Development," in 2011 Ninth International Conference on Software Engineering Research, Management and Applications, Baltimore, MD, USA, Aug. 2011, pp. 203–210. doi: 10.1109/SERA.2011.24.

[7] O. P. Timperi, "An Overview of Quality Assurance Practices in Agile Methodologies," p. 10, 2004.

[8] M. Aamir and M. N. A. Khan, "Incorporating quality control activities in scrum in relation to the concept of test backlog," Sādhanā, vol. 42, no. 7, pp. 1051–1061, Jul. 2017, doi: 10.1007/s12046-017-0688-7.

[9] N. Bajnaid, R. Benlamri, and B. Cogan, "An SQA e-Learning System for Agile Software Development," in Networked Digital Technologies, vol. 294, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 69–83. doi: 10.1007/978-3-642-30567-2_7.

[10] K. Schwaber and J. Sutherland, "The Scrum Guide."

[11] K. Schwaber, "SCRUM Development Process," p. 18.

[12] M. Soukaina, E. Badr, M. Abdelaziz, and S. Nawal, "Towards a New Metamodel Approach of Scrum, XP and Ignite Methods," IJACSA, vol. 12, no. 12, 2021, doi: 10.14569/IJACSA.2021.0121225.

[13] R. Pichler, Agile Product Management with Scrum: Creating Products that Customers Love. Addison-Wesley Professional, 2010.

[14] F. Ramos, A. A. M. Costa, M. Perkusich, H. Almeida, and A. Perkusich, "A Non-Functional Requirements Recommendation System for Scrum-based Projects," Jul. 2018, pp. 149–187. doi: 10.18293/SEKE2018-107.

[15] ISO/IEC 9126-1:2001, "Software engineering – Product quality – Part 1: Quality model."

[16] A. Silva et al., "A systematic review on the use of Definition of Done on agile software development projects," in Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona Sweden, Jun. 2017, pp. 364–373. doi: 10.1145/3084226.3084262.

[17] R. Moser, P. Abrahamsson, W. Pedrycz, A. Sillitti, and G. Succi, "A Case Study on the Impact of Refactoring on Quality and Productivity in an Agile Team," in Balancing Agility and Formalism in Software Engineering, vol. 5082, B. Meyer, J. R. Nawrocki, and B. Walter, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 252–266. doi: 10.1007/978-3-540-85279-7_20.

[18] F. Ribeiro, A. L. Ferreira, A. Tereso, and D. Perrotta, "Development of a Grooming Process for an Agile Software Team in the Automotive Domain," in Trends and Advances in Information Systems and Technologies, vol. 745, Cham: Springer International Publishing, 2018, pp. 887–896. doi: 10.1007/978-3-319-77703-0_86.

[19] S. Najihi, S. Elhadi, R. A. Abdelouahid, and A. Marzak, "Software Testing from an Agile and Traditional view," Procedia Computer Science, vol. 203, pp. 775–782, 2022, doi: 10.1016/j.procs.2022.07.116.

[20] S. Mukesh, "Defect Prevention_ Reducing Costs and Enhancing Quality," p. 6.