# TIME AND COST OF QUALITY IN SOFTWARE VERIFICATION: A COMPARATIVE SURVEY OF RUP, EVOLUTIONARY PROTOTYPING, AND SCRUM

**MSI. Esteban Sanabria-Mora**
*Master's Degree in Computer Science and Information Systems*
*Instituto Tecnológico de Costa Rica. San José, Costa Rica*
*esteban.sanabria@tec.ac.cr*

## ABSTRACT

Quality assurance has become a fundamental component of modern software development, as project success is increasingly linked to the ability to deliver reliable and high-quality software within defined time and budget constraints. As a result, understanding how time and cost are affected by quality-related activities has become a major concern for organizations. This paper analyzes the role of the Time of Quality (ToQ) and the Cost of Quality (CoQ) within the software verification process of projects developed using three different methodologies: the Rational Unified Process (RUP), evolutionary prototyping, and Scrum. Through a comparative analysis of how each methodology integrates testing and verification activities throughout the software development lifecycle, the study highlights differences in the estimation, management, and impact of quality-related time and cost. The results provide a conceptual reference to support the selection of a software development methodology based on its ability to manage quality assurance efforts while balancing time and budget constraints.

*Keywords: Software development methodology, time, cost, quality assurance, verification, project management, prototyping, Scrum, RUP.*

## I. INTRODUCTION

As with all products and services in industry, software applications require systematic quality assurance (QA) processes to ensure correctness, completeness, and compliance with customer and business requirements [1]. Beyond validating the final product, QA also supports the control and improvement of the software engineering process throughout the project lifecycle. Consequently, quality assurance constitutes a broad and complex discipline that directly influences both project outcomes and resource consumption.

Quality assurance activities are commonly grouped into verification, prevention, and correction of nonconformities [2]. Among these dimensions, software verification plays a particularly critical role, as it involves evaluating software artifacts at different stages of development through reviews, inspections, and testing activities. Verification tasks are directly associated with measurable project variables such as time and cost, since they require dedicated effort, tooling, and human resources. Given the wide scope of QA, this paper focuses specifically on the verification phase, which allows a clearer analysis of how quality-related time and cost are managed within different development methodologies.

The aim of this study is to compare how three software development methodologies (the Rational Unified Process, evolutionary prototyping, and Scrum) address software verification activities and how these approaches affect the Time of Quality (ToQ) and the Cost of Quality (CoQ). The comparison is based on methodological characteristics defined in each model, together with commonly reported practices in the software industry regarding testing effort, iterative verification, and quality-related cost distribution. The findings provide a conceptual reference for organizations seeking to select a software development methodology that aligns with their quality assurance objectives and their constraints in terms of time and cost.

### A. Objective

To examine and compare the time and cost required to ensure software quality at the verification level in projects developed using RUP, evolutionary prototyping, and Scrum.


ISSN: 2015-5392

## B. Problem and Justification

Software development methodologies continue to evolve in response to technological advances and changing organizational needs. As a result, a methodology that is effective in one context may be less suitable in another. This highlights the need not only to analyze software development models, but also to examine how they incorporate and manage verification activities related to quality assurance.

In addition, despite the central role of quality assurance in software projects, many organizations often lack clear guidance on how different development methodologies affect the time and cost associated specifically with verification activities. For these reasons, this study justifies a comparative analysis of RUP, evolutionary prototyping, and Scrum, focusing specifically on how each methodology addresses verification activities and how these practices impact the Time of Quality and the Cost of Quality. The results aim to provide a conceptual reference to support informed decision-making when selecting a software development methodology aligned with quality, time, and budget objectives.

# II- BACKGROUND AND RELATED WORK

Nowadays, software development models have become widespread across most companies. In this context, selecting an appropriate software model that enables the early detection and reduction of errors during the quality assurance (QA) phase has become increasingly important. Some organizations choose to create their own models to meet specific needs, while others implement existing software models with specific adaptations. Regardless of the approach, framework, or software model adopted, ensuring the quality of software applications has become a critical concern, as have become more demanding than ever in terms of software reliability and performance. Some traditional software models, such as the Waterfall model, which are still used despite the rise of agile methodologies, treat software verification as a phase separate from other processes, such as analysis, design, and development.

In other words, functionality verification of functionalities, or testing, occurs during a single phase and only after development artifacts have been completed. By contrast, companies such as IBM, which have adopted the Rational Unified Process (RUP) as their primary software development model, state that RUP includes a dedicated testing phase. However, in RUP, this phase occurs at the end of each iteration that introduces new functionality to the increment—that is, the set of implemented features—rather than solely at the end of the project. To illustrate the testing phase within RUP, Fig. 1 presents this phase in comparison with the other phases that comprise the development lifecycle of a software project.

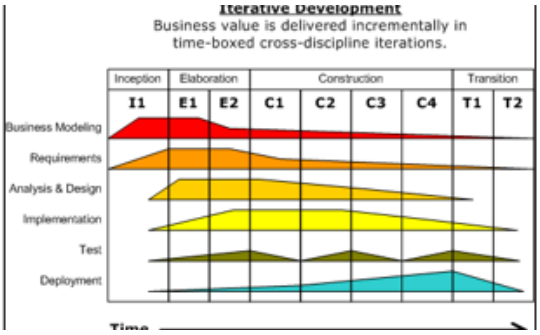


*Fig. 1. Development phases in RUP [3].*

Unlike traditional models, agile methodologies view quality assurance not as a standalone activity but as an integral part of development. For instance, in evolutionary prototyping, development and testing activities are tightly interwoven. Developers release functional prototypes as soon as they are testable, allowing users to provide feedback while development continues. Scrum manages quality assurance differently from both RUP and evolutionary prototyping. Each user story includes acceptance criteria that define the conditions under which the story is considered complete. Additionally, all sprints are governed by the Definition of Done, a set of quality criteria used to verify the correctness, completeness, and overall quality of the increment produced during a sprint.

Overall, quality verification is handled differently depending on the development model and organizational practices. Agile methodologies incorporate verification iteratively, while traditional models tend to separate verification from development activities, either iteratively, as in RUP, or sequentially, as in the Waterfall model.



ISSN: 2015-5392

# III- THEORETICAL FRAMEWORK

## A. Concepts

### *1) Quality Assurance*

According to Wiley and Wiley, the concept of quality assurance encompasses all handled of the software development lifecycle, ensuring that its implementation complies with both organizational and client quality standards, and that potential inconsistencies or errors are identified and corrected [4]. This concept differs from software testing, which focuses specifically on evaluating the software product by assessing the correctness of the results produced when executing operations that users are expected to perform once the product is released. In more colloquial terms, as stated by Joseph Man, "testing should intentionally attempt to make things go wrong to determine if things happen when they shouldn't, or things don't happen when they should" [4].

### *2) Verification testing*

The verification phase within quality assurance refers to the practice of tracking and evaluating all software development activities to certify their compliance with expected behavior. This phase is not limited solely to programming tasks only but also encompasses all other project-related activities that contribute to the software product as the primary outcome. Historically, the verification phase of QA was often conceived as a process applied only at the end of the software development lifecycle. However, it is now widely recognized that verification should take place throughout all phases of software creation, including requirements gathering, analysis, design, programming, implementation, and formal user training.

### *3) The time of quality (ToQ)*

The term "time of quality" refers to the number of working hours devoted to detecting and correcting defects identified through the use of testing strategies and techniques during the verification process. The time required to implement quality assurance practices in the verification phase depends largely on the complexity and scope of the application under test. For example, when automated testing is employed instead of manual testing, the time needed to assure software quality can be significantly reduced. Experts suggest that automated testing is particularly suitable for large-scale applications with frequent releases. Conversely, when an application is small or not a suitable candidate for automated testing, manual verification becomes the preferred approach, which typically requires a greater investment of time.

### *4) The cost of quality (CoQ)*

The term "cost of quality" refers to the costs incurred to correct defects that are discovered and addressed throughout the development and delivery of a software product. It may result from any of the following scenarios, according to the American Society for Quality [5]:

Performance of unnecessary work because of errors, poor organization, or communication.

Correction of defective results or errors.

Analysis of the causes of the failures to prevent them from happening again.

Wiley & Wiley [4] state that the cost of applying quality assurance in software products often exceeds half the total cost of software development and maintenance. That is because "there are no fixed recipes for attacking the problem of verifying a software product as even the most experienced specialists do not have pre-cooked solutions but need to design a solution that suits the problem, the requirements, and the development environment" [4].

## B. Software development methodologies for this survey

In order to understand the role of time and cost as key variables in quality assurance processes, it is essential to examine the software development methodologies through which software is produced. Accordingly, Scrum, the Rational Unified Process (RUP), and evolutionary prototyping are briefly introduced in the following sections.

### *1) Scrum*

Scrum is a framework used to develop complex projects, which began in the early 1990s. It is mainly characterized by using an iterative and incremental method in the development and delivery of results, which is known as "sprint" and has an approximate duration of two weeks to one month. This framework was developed by Ken Schwaber, Jeff Sutherland and Mike Beedle, and is particularly suitable for projects with rapidly changing requirements [8]. The result of each sprint, which is composed of "user


ISSN: 2015-5392

stories (requirements in a format similar to Kanban visual cards), is an increment that is presented to customers on a regular basis for approval or feedback. In Scrum, there are four artifacts that promote a collaborative and transparent strategy of the work to be done, the work in progress, and the work done.

### *2) RUP*

IBM defines "The Rational Unified Process" or RUP as "a generic software development methodology that works for varied types of software applications, organizations, levels of competitiveness, and size of projects. Its goal is to ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule and budget" [3]. RUP is an attempt to draw on the best features and characteristics of traditional software process models, such as Waterfall, and at the same time implement some of the best principles of agile software development [7]. RUP utilizes the Unified Modeling Language (UML) as its main tool to analyze and design software. This language allows developers, software architects, and other professionals involved in the creation of software products to clearly model requirements and bring them as close as possible to their implementation in a programming language.

### *3) Evolutionary prototyping*

Evolutionary prototyping is a lifecycle model that enables software applications to be developed incrementally. This approach relies heavily on customer feedback, allowing modifications and the addition of functionalities at a pace determined by customer approval. The model is particularly useful when software requirements are complex, incomplete, or not clearly defined, as it helps users better understand and refine their needs throughout the development process.

This model operates through evolutionary releases of the application, in which the most critical requirements are prioritized, designed, and implemented to deliver functional components that users or customers can evaluate at any stage of the project. The model can be applied to a wide range of software development projects, as it provides timely and effective feedback from future users, contributing to continuous improvements in product correctness and quality.

## IV- TIME AND COST OF QUALITY ASSURANCE IN THE THREE MODELS

Capgemini's World Quality Report, published in 2015-16, found that today's software development companies spend 35% of their budgets on QA and testing [9]. But is it the same case for all software models? This section addresses each model from the time and cost of QA's perspective.

### *1) Time and cost of QA in RUP*

In the Rational Unified Process (RUP), quality assurance activities are integrated across all project phases and involve all stakeholders involved in the assessment of the software product. Functional testing is not treated as a standalone activity performed by a separate team; rather, it is embedded within the development process itself [7].

The amount of time RUP devotes to verification testing depends on the number of features developed during the construction phase. For each iteration intended to deliver functional components to the customer, "all features are developed and integrated into the product, and all features are thoroughly tested" [3]. Although the construction phase is where testing and verification are most intensive, quality assessment is not limited to this stage. Verification activities may also occur during the inception, elaboration, and transition phases, even though new functionality is primarily introduced during the construction and transition stages. In this context, according to Pressman and Maxim [7], verification and testing activities typically consume between 30% and 40% of the total project effort in structured and iterative development models such as RUP.

From a cost perspective, RUP's iterative verification strategy contributes to a more predictable and controlled Cost of Quality. By identifying defects early in the lifecycle, the process reduces the cost of rework, defect propagation, and late-stage corrections, which are typically more expensive. The structured nature of RUP allows organizations to allocate resources explicitly to quality-related activities within each iteration, making quality costs more traceable and manageable throughout the project. Consequently, RUP projects often exhibit a predictable and controlled Cost of Quality, typically representing close to one-third of the total project budget, which aligns with its focus on risk mitigation and high-quality deliverables.


ISSN: 2015-5392

### 2) Time and cost of QA in Scrum

The Scrum Body of Knowledge (SBOK) defines quality as the ability of a completed product or its deliverables to meet the acceptance criteria of all user stories, which is ultimately reflected in the achievement of the business value expected by the customer [8]. To ensure that a project meets its quality requirements, Scrum advocates maintaining an updated and prioritized product backlog and defining clear acceptance criteria for every user story. These criteria function as quality parameters against which each unit of functionality is evaluated. According to the SBOK, "clearly defined acceptance criteria are crucial for timely and effective delivery of the functionality defined in the user stories, which ultimately determines the success of the project in terms of time and cost" [8].

From a time perspective, Scrum manages the Time of Quality by integrating verification activities into every sprint. Because development proceeds incrementally, defects can be identified early through repeated testing after each iteration, rather than being discovered only at the end of the project. This iterative verification reduces the time required for extensive rework and minimizes delays associated with late-stage defect correction. Testing effort is commonly estimated as part of the sprint workload using story points [10].

From a cost perspective, Scrum distributes quality-related costs across sprints instead of concentrating them in a single testing phase. Quality assurance activities such as testing, documentation, and review are performed as part of regular sprint tasks, allowing teams to control the Cost of Quality incrementally. Although Scrum does not prescribe fixed percentages for quality-related effort, the use of story points, sprint planning, and acceptance criteria enables teams to estimate and adjust quality-related costs with reasonable accuracy throughout the project lifecycle. This continuous integration of quality practices increases the likelihood of achieving the desired level of quality while maintaining control over both time and budget.

### 3) Time and cost of QA in Evolutionary prototyping

Evolutionary prototyping is a software development model that enables faster delivery when compared to traditional models such as the Waterfall model. It is designed to leverage modern development tools that support rapid construction and modification of software artifacts. Rather than emphasizing extensive upfront planning, this model prioritizes ongoing development, allowing requirements to evolve throughout the project based on continuous user feedback.

Development in evolutionary prototyping is conducted through short, time-boxed cycles that include implementation, testing, and verification activities. Each prototype represents an incremental version of the final product and is evaluated by users or customers to validate functionality, usability, and overall quality. Feedback gathered during these evaluations is incorporated into subsequent iterations, facilitating early defect detection and reducing the likelihood of costly rework at later stages of the project [11].

From a time perspective, the Time of Quality in evolutionary prototyping is closely intertwined with development activities, as verification occurs continuously alongside implementation. This integration can shorten feedback loops and accelerate defect identification. However, because verification tasks are not formally defined or standardized, the amount of time devoted to quality activities may vary significantly between iterations and projects, making precise estimation of ToQ challenging.

From a cost perspective, the Cost of Quality in evolutionary prototyping is similarly difficult to quantify. While early feedback and rapid validation may reduce the cost of late-stage defect correction, the absence of structured quality checkpoints, formal documentation, and predefined testing criteria can lead to inefficiencies, rework, and maintainability issues. These factors often result in unpredictable quality-related costs, particularly in projects with frequent requirement changes or poorly controlled iteration cycles.

Overall, evolutionary prototyping supports early verification and user-driven validation, which can positively impact both time and cost. However, compared to methodologies such as RUP and Scrum, the lack of explicit mechanisms for planning, measuring, and controlling quality activities limits its effectiveness in estimating the Time of Quality and Cost of Quality.

### 4) Comparison of the Time and cost of Quality in the three different software models

Table 1 summarizes the comparison of the Time of Quality (ToQ) and Cost of Quality (CoQ) specifically associated with the software verification phase in projects developed using RUP, Scrum, and evolutionary prototyping.


ISSN: 2015-5392

*TABLE 1. Comparison of the Time and Cost of Quality in the three different software models.*

| Software Model/ Measure | ToQ | CoQ |
|---|---|---|
| RUP | Approximately one-third of the total project development time is devoted to verification and testing activities, distributed across all iterations, with a strong emphasis during the construction phase. | Approximately one-third of the total project budget is allocated to verification and testing activities, including test planning, execution, automation, and defect correction. |
| Scrum | Quality activities are continuously integrated into each sprint through acceptance criteria, Definition of Done, testing, refactoring, and continuous integration rather than being treated as a separate phase. | Quality-related costs are distributed across sprints and managed incrementally through sprint planning, acceptance criteria, continuous integration, and defect resolution activities. |
| Evolutionary proto-typing | Time devoted to quality varies significantly across iterations and is often difficult to quantify due to evolving requirements, informal verification practices, and frequent user feedback cycles. | The cost of quality is highly variable and may increase over time due to frequent requirement changes, rework, limited architectural planning, and maintenance challenges in evolving prototypes. |

As shown in Table 1, RUP and Scrum provide clearer mechanisms for estimating and managing the time and cost associated with the verification phase. At the same time, evolutionary prototyping lacks explicit structures for verification planning and measurement.

## V- DISCUSSION

The comparison of the Time of Quality (ToQ) and Cost of Quality (CoQ) across the three software development models reveals substantial differences in how quality assurance activities are planned, executed, and controlled. The Rational Unified Process (RUP) exhibits the most structured approach, characterized by clearly defined phases and explicit verification activities. This structure enables relatively precise estimation of both the time and cost associated with quality assurance, making RUP particularly suitable for projects in which strict quality control, extensive documentation, and reliable budget forecasting are critical.

Scrum also integrates quality assurance throughout the development lifecycle but does so more flexibly and iteratively. Although its ToQ and CoQ are not expressed as fixed proportions, they can be reasonably estimated through sprint planning mechanisms, story point allocation, and clearly defined acceptance criteria and Definition of Done. This adaptability allows Scrum teams to continuously balance quality, time, and cost, making the methodology well-suited for projects with evolving requirements and frequent incremental deliveries.

In contrast, evolutionary prototyping lacks clearly defined metrics for ToQ and CoQ. While the model facilitates rapid development and early user feedback, the absence of formal quality checkpoints and structured planning can lead to inefficiencies that complicate time and cost estimation. Consequently, although evolutionary prototyping may be effective for exploratory or highly uncertain projects, it presents increased risks when rigorous control over quality-related time and cost is required.

Overall, methodologies with explicitly defined quality assurance practices, such as RUP and Scrum, offer stronger support for managing and estimating the time and cost associated with quality. Therefore, the selection of a software development model should consider not only development speed and flexibility, but also the organization's ability to plan, measure, and control quality-related activities within established time and budget constraints.



ISSN: 2015-5392

## VI- CONCLUSIONS

Before initiating a software development project, it is essential to understand its characteristics and to clearly define how it will be executed. This preliminary analysis enables organizations to optimize resources, manage time and cost effectively, and ensure a balanced distribution of responsibilities and benefits among project participants.

An important finding of this study is that the selected development methodology directly influences team participation and workload distribution. Depending on the model adopted, certain roles may require greater involvement than others, which in turn affects both the effort invested and the final cost of the software product. Consequently, the choice of methodology has a direct impact on quality-related expenditures and timelines.

The findings of this paper confirm that time and cost are key variables in quality assurance processes and that their management is highly dependent on the software development methodology employed. The testing phase is shown to be a critical component of the software development lifecycle, with its execution varying according to the model and the methodological adaptations applied by each organization. Agile methodologies apply quality assurance iteratively at the verification level, while traditional models tend to separate verification from programming activities, either iteratively, as in RUP, or sequentially, as in the Waterfall model.

RUP exhibits the most structured and predictable approach, with explicitly defined verification activities integrated throughout the lifecycle, resulting in relatively stable time and cost proportions. Scrum demonstrates a flexible but measurable investment in quality, where verification efforts are embedded within each sprint and represented as an incremental increase in effort and budget. In contrast, evolutionary prototyping lacks clearly defined boundaries for quality-related activities, leading to greater variability and uncertainty in both time and cost estimations. This comparison reinforces the importance of selecting a development methodology that aligns not only with technical requirements but also with an organization's capacity to plan, control, and sustain quality assurance efforts within defined time and budget constraints.

Overall, this study demonstrates that methodologies with well-defined and continuous quality assurance practices provide better support for controlling the Time of Quality and the Cost of Quality. Therefore, selecting an appropriate software development methodology is a strategic decision that should consider not only development speed or flexibility, but also the organization's capacity to plan, measure, and manage quality-related activities within defined time and budget constraints.

## REFERENCES

[1] P. Lledó and G. Rivarola, Gestión de proyectos: cómo dirigir proyectos exitosos, coordinar los recursos humanos y gestionar el riesgo. Buenos Aires, Argentina: Prentice Hall, 2007..

[2] Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOK Guide). Campus Boulevard, Pennsylvania, USA: Project Management Institute, Inc., 2013.

[3] "IBM - Costa Rica," IBM, 2017. [Online]. Available: https://www.ibm.com/cr-es/[4] J. Wiley and S. Wiley, Software Testing and Analysis, 1st ed., 2008.

[5] P. Mahmood and M. Beg, "Measuring Cost of Quality (CoQ) on SDLC projects is indispensable for effective Software Quality Assurance," International Journal of Soft Computing and Software Engineering, vol. 2, no. 9, pp. 1–15, 2012.

[6] "ASQ - A Global Leader in Quality Improvement & Standards," ASQ, 2017. [Online]. Available: https://asq.org/

[7] R. Pressman and B. Maxim, Software Engineering, 1st ed. New York, NY, USA: McGraw-Hill, 2015.

[8] K. Schwaber and J. Sutherland, Software in 30 Days, 1st ed. New York, NY, USA: John Wiley & Sons, 2012.

[9] "World Quality Report 2016-17," Capgemini Worldwide, 2017. [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/135804/Williams_Paivi.pdf?sequence=1&isAllowed=y[10] K. Schwaber and J. Sutherland, The Scrum Guide, 2020.

[11] I. Sommerville, Software Engineering, 10th ed. Pearson, 2016.



ISSN: 2015-5392