OPTIMIZING SOFTWARE QUALITY: LESSONS FROM AGILE AND DEVOPS PRACTICES

RALUCA NICOLAESCU [*]

ABSTRACT: *Software products have become integral to our everyday lives, making the quality of these products a crucial concern. Recent efforts to deliver software more rapidly have focused on iterative and fast-paced deliveries, a hallmark of Agile methodologies. However, despite these advancements, there remains a significant issue: both Agile and DevOps practices often fall short in meeting customer quality expectations. This paper analyzes how quality is managed within Agile and DevOps frameworks and explores potential integrations of quality management practices to ensure higher customer satisfaction. Through a detailed review of current literature and practical case studies, this study aims to identify gaps and propose actionable solutions for enhancing software quality in fast-paced development environments.*

KEY WORDS: *Quality, Software quality, Agile, DevOps, SDLC, quality metrics*.

JEL CLASSIFICATIONS: *C88, L15, L86.*

1. INTRODUCTION

The growing need for faster deliveries of software products that meet customer and stakeholder requirements led to the birth of new software development approaches and frameworks. Agile methodologies emerged as a response to the inefficiencies of traditional development models, prioritizing shorter development cycles, frequent releases, and enhanced communication with stakeholders. Frameworks such as Scrum, Extreme Programming, Lean, and Kanban have become widely adopted, offering flexibility and iterative releases of products. Despite their acknowledged benefits, Agile methodologies have faced challenges and critiques in the last couple of years, particularly concerning software quality and the associated high costs of managing poor quality. In order to address these issues, DevOps was introduced as an extension of Agile principles, aiming to integrate and streamline development and operations. By

[*] *Lecturer, Ph.D., University of Petroşani, Romania, raluca.dovleac@yahoo.com*

using key principles such as continuous integration, continuous delivery, and enhanced automation, DevOps aims to improve the speed, frequency, and reliability of software deployments. This approach not only extends Agile's core principles but also emphasizes proactive quality management through automation and continuous feedback.

The current paper analyses the most important aspects of quality within these two main software development approaches. For this, a comparative analysis of quality management practices and key performance metrics used within Agile methodologies and DevOps is conducted, with the role of exploring how these methodologies address software quality issues, the particular challenges each methodology poses, and the key performance metrics used to measure their effectiveness. Through this analysis, the paper aims to provide insights into the strengths and weaknesses of both approaches, offering a comprehensive understanding of how these approaches can help the development team achieve qualitative software products and meet customer and stakeholder requirements.

2. RESEARCH BACKGROUND

Agile methodologies emerged as an alternative to deliver software products in shorter time cycles and ensuring a better communication with the customer, and throughout time have been widely implemented and used. Agile methodologies encourage a transition from formal communications to frequent and often informal communications (Hermawan & Manik, 2021). Some of the most commonly adopted Agile methodologies include: Scrum, Extreme Programming, Lean and Kanban (Samarawickrama & Perera, 2017). Throughout time however, it has been noticed that the quality of software is decreasing, and there are high costs associated with these aspects. The main categories of costs related to poor software quality are related to costs of unsuccessful IT/software projects, cost of poor quality in legacy systems, cost of operational software failures and costs related to cybersecurity and technical debt (Krasner, 2021).

DevOps evolved as an alternative to Agile methodologies to fill a need for an agile infrastructure and interaction between the development and operations teams (Debois, 2008). DevOps is considered to be a mixture of different developments and operations, with the primary focus of increasing the deployment speed, frequency and quality of software products (Mishra & Otaiwi, 2020; Erich, 2017). The focus on the DevOps movement has been on using lean manufacturing processes in IT systems, with the goal of improving the collaboration between developers and IT operations (Colavita, 2016). Furthermore, we can witness an increase in organizations adopting DevOps with the purpose of accelerating the delivery speed and improving the quality of their products (StateOfAgile, 2020). DevOps extends the continuous development goals present in the Agile methodologies and core principles to continuous integration and release, and furthermore, encourages automations of the change, configuration and release processes (Perera, et al., 2017).

In the current paper, the main aspects of quality management in software development have been analyzed through a comparative analysis of the quality

management practices and key performance metrics used within two of the most widely implemented software development approaches – agile methodologies and DevOps.

3. QUALITY IN AGILE VS DEVOPS

The quality of software products is an essential aspect in today's society where we rely on software tools more than ever. Software quality is typically measured through the lens of the quality characteristics of the software product. These characteristics can be defined as a set of attributes that the software product has and which are evaluated in order to ensure that they meet the requirements established by the development team (Yarlagadda, 2019).

In order to obtain qualitative software products, it is important to first understand the key aspects of quality of software products. One definition of software quality established by the US Department of Defense states that software quality is "the degree to which the attributes of the software enable it to perform its intended end use" (Gillies, 2011).

The previously in-use ISO 9126 standard for software quality evaluation defined a software quality model based on six major quality characteristics, such as: reliability, efficiency, functionality, portability, maintainability and efficiency (Gong, et al., 2016). The new, updated standard "ISO/IEC 25019:2023 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality-in-use model" focuses instead on three characteristics (further subdivided into sub-characteristics) and provides a set of quality characteristics for specifying, measuring, evaluating and improving quality-in-use. The standard defines quality-in-use such as "the extent to which the system or product, when it is used in a specified context of use, satisfies or exceeds stakeholders' needs to achieve specified beneficial goals or outcomes." (ISO, 2023).

A way to measure the amount of customer expectations and development team requirements that are met is by establishing key performance metrics. Quality metrics are an essential aspect of ensuring that both customer expectations and operational performance regarding quality are met, by transforming these expectations into metrics that can be measured, evaluated and compared (Chakravarty & Singh, 2020).

Popular quality metrics that are used in traditional software development focus on three key aspects of the development process: product metrics (measure aspects such as: mean time to failure, time between failures, defect density, defects per unit software size and customer satisfaction often measured on scales from one to five), process metrics (focus on improving the development and maintenance process and measure aspects such as: defect arrival pattern, defect removal efficiency), and project metrics (focus on analyzing key aspects of the project execution parameters such as: cost, scheduling and staffing) (Chakravarty & Singh, 2020).

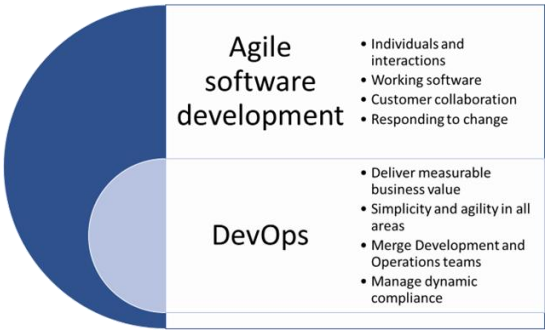
3.1. Key differences between Agile methodologies and DevOps

It is a well-known fact that in traditional software development models such as the Waterfall mode, the communication between the development team and the customers and stakeholders is often inadequate and results in longer development cycles (Zhu, et al, 2016).

Agile software development evolved as a solution to the problem that big companies were facing in the '90s with achieving the established targets while following heavily controlled "phase-gate development" methodologies also known as "waterfall development". As a result, several people created simpler and less prescriptive development methods which were called "lightweight methods", such as "Adaptive Software Development", "Crystal", "Feature-Driven Development", "Extreme programming" and "Scrum" (Shore, etal., 2022).

Agile development is characterized by four core values (1. Individuals and interactions over processes and tools, 2. Working software over comprehensive documentation, 3. Customer collaboration over contract negotiation, 4. Responding to change over following a plan) and twelve principles that emphasize the importance of good team cohesion and communication, flexibility, simplicity, integration of stakeholders in the decision-making process and rapid, frequent and iterative delivery of working products (Beck, et al., 2001).

Similarly, to the four key values of the Agile Manifest for software development, four main goals have been identified as being the backbone of the DevOps methodology, ranging from delivering measurable business value through continuous and high-quality service delivery, emphasizing simplicity and agility in all areas (from technology to human factors), eliminating barriers between the development and operations teams and managing dynamic compliance (Farroha, D and Farroha, B., 2014).



Ssource: Beck, K.; Beedle, M.; van Bennekum, A.; Cockburn, A.; et al. (2001) Manifesto for Agile Software Development, agilemanifesto, [Online], Available at: https://agilemanifesto.org/, [Accessed 05 07 2024]; Farroha, D.; Farroha, B. (2014) A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment, Military Communications Conference

Figure 1. Core values of Agile and DevOps

As it can be observed from Figure 1, DevOps is not necessarily a separate and completely distinct software development approach but rather an improved approach that is based on the core principles and values of the Agile software development. It can be noted that while Agile establishes a rather abstract view on how software development should take place, DevOps provides a couple of specific targets such as: Delivery of measurable business value and Management of dynamic compliance.

This is noted in the existing literature, where the fact that DevOps originated in the context of Agile software development has been established, and it has been observed that DevOps appeared as an appropriate approach to enable the continuous delivery and deployment of working software in small iterative releases (Ghantous & Gill, 2017).

One key difference between the two development approaches is related to the development team. Where Agile prioritizes individuals and interactions, DevOps tends to rather put an emphasis on tools and processes instead (Erich, et al., 2014).

3.2. Quality aspects in Agile methodologies vs DevOps

While both Agile and DevOps promise faster and better product releases and better customer and stakeholder compliance, when it comes to quality, both approaches are facing major issues, and one of the key factors that contribute to this is related to the fact that the work load of both approaches is hard to predict (Erich, et al., 2014).

Agile methodologies didn't exactly place a focus on quality management and used a more reactive approach to fixing quality issues rather than a proactive one. Efforts of integrating quality management tools and practices within the development cycle were noted though, and there was also a proposition of integrating software quality parameters such as: correctness, robustness, extendibility, reusability, compatibility, efficiency, portability, timeliness, integrity, verifiability and ease of use in the development process along with the proposition of integrating Total Quality Management (TQM) success factors in the Agile development methodologies (Dovleac & Suciu, 2018).

Because one of the key aspects of DevOps is streamlining everyday exercises and the focus is to add customer value, eliminate waste, reduce cycle times and remove bottlenecks, DevOps seems to be getting closer to meeting the quality expectations of customers and stakeholders (Katal, et al., 2019). Research shows that the implementation of DevOps has a positive effect on teamwork quality in software development (Hermawan & Manik, 2021; Mishra & Otaiwi, 2020; Perera, et al., 2017). Others however argue that organizations have difficulty in determining the right quality coverage in the DevOps process (Mishra & Otaiwi, 2020; Casale, et al., 2016).

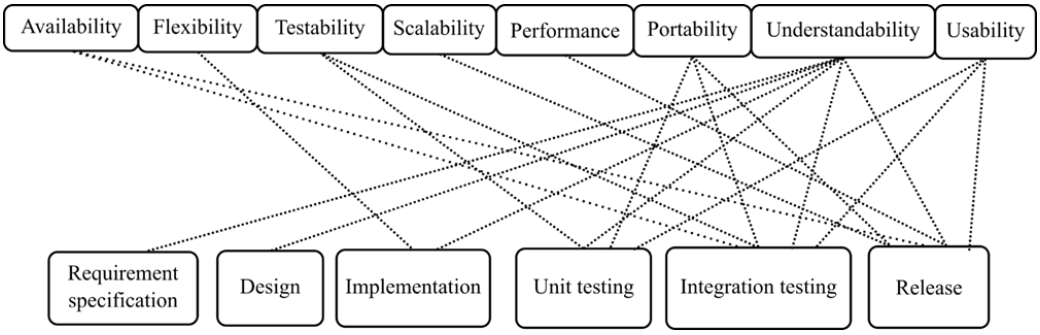
Efforts to improve the quality of the produced software in companies and teams using the DevOps approach have been noted, with a focus on coupling DevOps stages with tools and methods to help improve the quality. Solutions include the integration of practices such as Continuous Integrations/ Continuous Delivery (CI/CD), using pipelines and highly automated configuration solutions for the runtime environment (Alnafessah, et al., 2021; Virmani, 2015; Zhao, et al., 2017).

The fact that the CI process is integrated and that the pipeline system is used allows for a faster feedback cycle and given how quality gates are used later in the pipeline and are non-blocking allows for work to continue while the quality checks are underway (Ibrahim, et al., 2019; Chakraborty, et al., 2014).

Other approaches to ensuring quality in DevOps include using frameworks such as the SQUID framework for quality specifications in DevOps (Di Nitto, et al., 2016), the DevQualOps development model which can also be used when transitioning from Agile development environments to DevOps and puts an emphasis on continuous evolution, integration and delivery and the main concepts of Continuous Quality Engineering (Dovleac, 2023).

In order to look at a more objective approach of comparing how quality is achieved within Agile methodologies and DevOps, the key performance metrics used by these development approaches will be analyzed. This should highlight existing differences and could furthermore become the starting point for research regarding the adoption or elimination of key metrics for each type of development approach.

The quality attributes that have been identified as being critical for agile applications, and their most significant impact on each stage of the development lifecycle can be observed in Figure 2.



Source: Malik, U. M.; Nasir, H. M.; Javed, A. (2014) An Efficient Objective Quality Model for Agile Application Development, International Journal of Computer Applications, vol. 85, no. 8, pp. 19-24

Figure 2. Key metrics in Agile SDLC

As it can be observed most key metrics are essential to more than one stage of the development lifecycle. Of course, each metric brings its contribution to each one of the stages of the development lifecycle to some extent but for this visual representation, only the most significant contributions have been taken into account.

Others suggest taking into account the following quality attributes when analyzing quality within Agile development projects and even propose a set of possible metrics for these attributes (Ikerionwu & Nwandu, 2021):

- Maintainability – measured as total downtime over number of outages
- Availability – measured as mean time to failure over mean time to failure plus mean time to repair multiplied by 100%
- Reliability – measured as 1 over mean time to failure

- Portability – measured as number of successful ports over total number of ports multiplied by 100
- Testability – measured as k multiplied by visibility
- Reusability – measured as the added products of maintainability/ adaptability and their weighs.

As for DevOps, some of the most important key metrics identified by literature, and which are closely related to the core principles of the DevOps philosophy are:

- Mean Time to Recover (MTTR)
- Mean Lead time for Changes (MLT)
- Deployment Frequency (DF)
- Change Failure Rate (CFR)

Additionally, metrics related to the quality culture of the company overall have also been proposed for integration, such as (Amaro, et al., 2024):

- Organizational Culture
- Operational Performance
- Business Focus
- Incremental change

Others suggest integrating some of the Scrum metrics into the DevOps development process for improvement such as: velocity, work capacity, focus factor, percentage of adopted work, percentage of found work, accuracy of estimation, accuracy of forecast, targeted value increase, success at scale and win/loss record (Kruis, 2014).

Of course, these metrics are mere starting points and as more and more companies are going to be implementing DevOps and concerns will focus on quality of delivered product, we will most likely see an increase of key performance metrics being used in the future, as well as a diversification of the metrics being used.

4. CONCLUSIONS

The current paper aimed to identify the best lessons learned from the quality management practices of one of the most commonly used software development approach – Agile, and the more newly adopted and integrated approach DevOps. The purpose of the research was derived from the need identified to obtain better and more qualitative software products on the market. For this, a comparative analysis of Agile methodologies and DevOps has been used in order to highlight their strengths and areas of improvement in software quality management.

Agile methodologies, having emerged in an attempt to address the inefficiencies of traditional software development models, emphasize flexibility, rapid and iterative delivery, and close collaboration with customers and stakeholders. However, since their inception, these methodologies have often faced challenges in maintaining high software quality, in part due to their reactive approach to quality management.

On the other hand, the DevOps approach to software development evolved as an extension and enhancement of Agile principles, placing an emphasis on continuous integration, continuous delivery, and the automation of development and operational processes. This approach aims to improve collaboration between development and operations teams, thereby enhancing the speed, frequency, and quality of software releases. Despite these advancements, and certain steps being taken in the direction of more qualitative products, ensuring consistent quality in DevOps remains challenging due to some key aspects such as the unpredictable workload and the complexity of determining appropriate quality coverage.

The transition from Agile to DevOps is marked by a shift from prioritizing individuals and interactions to emphasizing tools and processes. DevOps incorporates advanced practices such as Continuous Integration/Continuous Delivery (CI/CD) and automated configuration management, which facilitate faster feedback cycles and continuous quality checks. These practices have been noted to contribute to a more proactive quality management approach compared to the reactive methods observed in Agile.

Key performance metrics play a crucial role in both approaches, serving as benchmarks for assessing software quality and operational performance. It has been noted that Agile methodologies focus on metrics related to maintainability, availability, reliability, portability, testability, and reusability. In contrast, DevOps emphasizes metrics such as Mean Time to Recover (MTTR), Mean Lead Time for Changes (MLT), Deployment Frequency (DF), and Change Failure Rate (CFR), along with organizational and operational metrics.

As organizations increasingly adopt DevOps, there is a growing need for comprehensive quality metrics that align with the core principles of both Agile and DevOps. The continuous evolution of these methodologies suggests that future research and development will likely introduce more refined metrics and practices, increasing therefore the ability to deliver high-quality software products in a consistent matter.

In conclusion, while Agile methodologies and DevOps have significantly improved software development processes, achieving and maintaining high software quality remains a complex challenge. The integration of proactive quality management practices and the continuous refinement of key performance metrics will be essential in addressing these challenges and ensuring that both development approaches can meet the evolving demands of the software industry.

REFERENCES:

[1]. Alnafessah, A.; Ul Gias, A.; Wang, R.; Zhu, L.; Casale, G.; Filieri, A. (2021) *Quality-Aware DevOps Research: Where Do We Stand?*, IEEE Access, vol. 9, pp. 44476-44488

[2]. Amaro, R.; Pereira, R.; Da Silva, M. M. (2024) *DevOps Metrics and KPIs: A Multivocal Literature Review*, ACM Computing Surveys, vol. 56, no. 9, pp. 231:1-231:40

[3]. Beck, K.; Beedle, M.; van Bennekum, A.; Cockburn, A.; et al. (2001) *Manifesto for Agile Software Development*, agilemanifesto, [Online], Available at: https://agilemanifesto.org/, [Accessed 05 07 2024]

[4]. Casale, G.; Chesta, C.; Deussen, P.; Di Nitto, E.; et al. (2016) *Current and Future Challenges of Software Engineering for Services and Applications*, Procedia Computer Science, vol. 97, pp. 39-42

[5]. Chakraborty, A.; Ramachandran, K. K.; Yamijala, S. S.; Pati, S. K.; Maji, T. K. (2014) *A hexanuclear Cu(i) cluster supported by cuprophilic interaction: Effects of aromatics on luminescence properties*," RSC Adv, vol. 4, no. 66

[6]. Chakravarty, K.; Singh, J. (2020) *A Study of Quality Metrics in Agile Software Development*, Advances in Intelligent Systems and Computing 1311

[7]. Colavita, F. (2016) *DevOps Movement of Enterprise Agile Breakdown Silos, Create Collaboration, Increase Quality, and Application Speed*, Proceedings of 4th International Conference in Software Engineering for Defence Applications

[8]. Debois, P. (2008) *Agile infrastructure and operations: how infragile are you?*, Agile 2008 Conference, Toronto

[9]. Di Nitto, E.; Jamshidi, P.; Guerriero, M.; Spais, I.; Tamburri, D. A. (2016) *A Software Architecture Framework for Quality-Aware DevOps*, QUDOS 2016: Proceedings of the 2nd International Workshop on Quality-Aware DevOps, New York

[10]. Dovleac, R. (2023) *The Rise of DevQualOps and Implications on Software Quality*, International Journal of Computers, vol. 8, pp. 5-10

[11]. Dovleac, R.; Suciu, C. (2018) *Quality assurance within the agile system development life-cycle*, International Journal of Economics and Management Systems , vol. 3, pp. 181-187

[12]. Erich, F. M. A.; Amrit, C.; Daneva, M. (2017) *A qualitative study of DevOps usage in practice*, Journal of Software: Evolution and Process, vol. 29, no. 6

[13]. Erich, F.; Amrit, C.; Daneva, M. (2014) *Report: DevOps Literature Review*, University of Twente, Enschede

[14]. Farroha, D.; Farroha, B. (2014) *A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment*, Military Communications Conference

[15]. Ghantous, G. B.; Gill, A. Q. (2017) *DevOps: Concepts, Practices, Tools, Benefits and Challenges*, PACIS 2017 Proceedings, LANGKAWI ISLAND

[16]. Gillies, A. (2011) *Software Quality: Theory and Management*, Lulu.com

[17]. Gong, J.; Lu, J.; Cai, L. (2016) *An induction to the development of software quality model standards*, 2016 Third International Conference on Trustworthy Systems and their Applications

[18]. Hermawan, A.; Manik, L. P. (2021) *The Effect of DevOps Implementation on Teamwork Quality in Software Development*, Journal of Information Systems Engineering and Business Intelligence, vol. 7, no. 1, pp. 84-90

[19]. Ibrahim, M. M. A.; Syed-Mohamad, S. M.; Husin, M. H. (2019) *Managing Quality Assurance Challenges of DevOps through Analytics*, ICSCA '19, Penang

[20]. Ikerionwu, C.; Nwandu, I. C. (2021) *Quantifying Software Quality in Agile Development Environment*, Software Engineering, vol. 9, no. 2, pp. 36-44

[21]. ISO (2023) *ISO/IEC 25019:2023 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Quality-in-use model*, ISO

[22]. Katal, A.; Bjoria, V.; Dahiya, S. (2019) *DevOps: Bridging the gap between Development and Operations*, 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), Erode

[23]. Krasner, H. (2021) *The cost of Poor Software Quality in the US: A 2020 Report*, CISQ

[24]. Kruis, S. (2014) *Designing a metrics model for DevOps at Philips IT*, Eindhoven University of Technology

[25]. Malik, U. M.; Nasir, H. M.; Javed, A. (2014) *An Efficient Objective Quality Model for Agile Application Development*, International Journal of Computer Applications, vol. 85, no. 8, pp. 19-24

[26]. Mishra, A.; Otaiwi, Z. (2020) *DevOps and software quality: A systematic mapping*, Computer Science Review, pp. 1-14

[27]. Perera, P.; Silva, R.; Perera, I. (2017) *Improve Software Quality through Practicing DevOps*, International Conference on Advances in ICT for Emerging Regions

[28]. Samarawickrama, S. S.; Perera, I. (2017) *Continuous scrum: A framework to enhance scrum with DevOps*, Seventeenth international conference on advances in ICT for emerging regions, Colombo, Sri Lanka

[29]. Shore, J.; Larsen, D.; Klitgaard, G.; Warden, S.; Fowler, M. (2022) *The art of Agile Development*, Sebastopol: O'Reilly

[30]. StateOfAgile (2020) *14th annual STATE OF AGILE REPORT*

[31]. Virmani, M. (2015) *Understanding DevOps & bridging the gap from continuous integration to continuous delivery*, Proc. INTECH, pp. 78-82

[32]. Yarlagadda, R. T. (2019) *How DevOps Enhances the Software Dévelopment Quality*, International Journal of Creative Research Thoughts, vol. 7, no. 3, pp. 358-364

[33]. Zhao, Y.; Serebrenik, A.; Zhou, Y.; Filkov, V.; Vasilescu, B. (2017) *The impact of continuous integration on other software development practices: A large-scale empirical study*, Proc. ASE, Urbana

[34]. Zhu, L.; Bass, L.; Champlin-Scharff, G. (2016) *DevOps and Its Practices*, IEEE Softw, vol. 33, pp. 32-34

This article was reviewed and accepted for presentation and publication within the 11th edition of the International Multidisciplinary Symposium "UNIVERSITARIA SIMPRO 2024".