PRODUCTIVITY IN A WATERFALL VERSUS AGILE DEVELOPMENT TEAM

Albert-Cristian CRĂCIUN[1]

Ioana-Alexandra MATEI[2]

Silviu-Gabriel FLORIAN[3]

Cătălin TUDOSE[4]

Costin-Anton BOIANGIU[5]

Abstract

This research delves into a comparative analysis of productivity in Waterfall and Agile development teams. It challenges the traditional metrics of productivity, like lines of code and hours worked, revealing their limitations in the complex realm of software development. The study emphasizes the need to consider both qualitative aspects - such as code quality and team collaboration - and quantitative measures. It explores how Waterfall and Agile methodologies influence team dynamics, efficiency, and project success, providing insights through industry examples and modern tools. The comparison presents a comprehensive view of evolving productivity metrics in software development.

This article comprehensively explores the various reasons why Agile methodology is widely considered to significantly enhance productivity compared to the Waterfall approach.

Though both methodologies contribute to increased productivity, Agile distinguishes itself through its inherent characteristics that prove advantageous not only for achieving project success but also for the well-being of developers..

Keywords: Productivity, software development, Waterfall, Agile, Code quality, Project impact, Quantitative and qualitative measurements, dynamics

JEL Classification: C61

[1]Student, POLITEHNICA National University for Science and Technology of Bucharest, Romania, albert.craciun@stud.acs.upb.ro
[2]Student, POLITEHNICA National University for Science and Technology of Bucharest, Romania, ioana.matei2108@stud.acs.upb.ro
[3]Student, POLITEHNICA National University for Science and Technology of Bucharest, Romania, silviu.florian@stud.acs.upb.ro
[4]PhD, Lecturer, POLITEHNICA National University for Science and Technology of Bucharest and Luxoft Romania, Romania, catalin.tudose@gmail.com
[5]PhD, Professor, POLITEHNICA National University for Science and Technology of Bucharest, Romania, costin.boiangiu@cs.pub.ro

1. Introduction

1.1. Overview of Waterfall and Agile Methodologies

The Waterfall methodology, conceived in the 1970s, is based on a sequential structure, where each phase of a project must be completed before moving on to the next one. The Waterfall is effective for projects with well-defined and stable requirements, where the end goal is clear and no significant changes are expected along the way.

Agile, a term that took hold in the software industry in the early 2000s, is founded on the Agile Manifesto, a set of principles designed to improve software development. This methodology evolved in response to the rigidity and limitations of traditional development processes. Agile emphasizes adaptability, cross-disciplinary collaboration, and continuous feedback, allowing teams to respond quickly to change. By breaking down projects into small, manageable segments (sprints), Agile facilitates continuous and flexible software delivery, focused on customer needs.

The main difference between Agile and Waterfall is flexibility. Agile allows for quick adaptation to changes, while Waterfall follows a strict and well-defined plan. Despite the differences, both approaches aim to deliver high-quality software and meet project goals. Waterfall is preferred for projects with clear and unchanging requirements, while Agile is ideal for projects where requirements evolve.

Agile and Waterfall have had a significant impact on the way software is developed and delivered. Agile has ushered in a new era of flexibility, collaboration, and adaptability, while Waterfall continues to be relevant for projects with well-defined parameters. The choice between these methodologies depends on the nature of the project, the culture of the organization, and the needs of the customers [1].

1.2. Overview of Productivity Measurements

In the world of software development, productivity measurement is a complex area that is essential for optimizing processes and ensuring efficient product delivery. Traditional metrics, such as the number of lines of code written or bugs fixed, are no longer sufficient to capture the essence of true productivity in such a dynamic and innovative environment.

Lines of code are not necessarily correlated with a better product. On the contrary, more lines of code can show a bad organization of the overall project, either due to the existence of duplicate code or to unnecessary complications of a task [2]. This can lead to a higher level of complexity and an increased chance of errors and future issues.

Furthermore, the number of bugs fixed is not a reliable indicator of efficiency, given the fact that a large number can result from poorly written code. Therefore, greater emphasis

should be placed on minimizing the time taken to fix bugs or to have as few bugs as possible.

Measuring the amount of code produced or bugs fixed and focusing on quantity over quality isn't an effective way to understand productivity. In contrast, modern approaches focus on a more diverse set of metrics that are more relevant to today's reality.

Measuring developer productivity typically boils down to tracking the work completed and the quality or importance of the task accomplished. By combining these two criteria, the monitoring of productivity ensures project success.

One notable example is Cycle Time, which tracks a task's duration from start to completion. It provides valuable insight into the efficiency of development processes. For example, in an Agile team, a short cycle time can indicate rapid responsiveness to requirements and excellent adaptability. On the other hand, Deployment Frequency measures how often the team releases software or updates. In a Waterfall environment, where deployments are less frequent but more comprehensive, this metric may reflect a more meticulous and structured development process.

The rework rate, which indicates the percentage of tasks requiring revision or correction after completion, is another critical indicator. A high rate may suggest problems in requirements definition or internal communication. In addition to these quantitative measures, qualitative assessment of productivity, such as feedback from colleagues and the quality of collaboration, plays a key role.

In recent years, innovative metrics have emerged that have enriched the productivity measurement landscape. DevOps Research and Assessment (DORA) metrics, such as change response time and system stability, are used extensively in the DevOps industry to evaluate and improve development and operational processes [3]. Covering a wide range of issues, another set of metrics addresses aspects such as employee satisfaction, well-being, performance, activity, communication, collaboration, efficiency, and workflow. These metrics are collectively known as SPACE.

A concrete industry example is Google's approach, which combines log analysis with socio-technical assessments to understand developer productivity better [4]. This blended methodology not only assesses what developers do but also how they feel about their work, a key aspect of maintaining sustainable productivity. Companies like Microsoft are integrating DORA metrics into their DevOps processes, aiming for continuous improvements in team performance. Tech startups also often adopt SPACE metrics to assess employee satisfaction and team effectiveness, reflecting a shift in how the industry values and measures productivity.

The multidimensional approach to these metrics reflects the continuing evolution of the industry and the need to assess productivity in a more holistic way adapted to contemporary realities.

2. Theoretical Background

The theoretical underpinnings of modern software development methodologies and metrics represent a paradigm shift in how productivity and efficiency are understood and measured in the software industry. This shift reflects an acknowledgment of the multifaceted nature of software development, extending beyond technical output to include human factors and the quality of the work environment. These contemporary approaches, grounded in diverse disciplines, highlight the necessity of adaptable, iterative methodologies and holistic metrics. They emphasize the significance of balancing technical efficiency with the well-being of development teams and the satisfaction of end-users, crucial for sustainable and successful software development in today's dynamic technological landscape.

2.1. Agile Development Evolution

The Agile development approach [5] moves from rigidity to flexibility. The evolution from traditional to modern methodologies in software development signifies a shift from structured, plan-driven processes to more adaptable, iterative approaches. This transition has been driven by the increasing complexity of software projects, necessitating methods that can accommodate rapid changes and deliver solutions more efficiently.

Incorporating feedback loops and team collaboration means that modern methodologies, especially Agile, emphasize the importance of regular feedback loops and team collaboration. This perspective is rooted in the idea that continuous improvement and adaptation, coupled with empowered, self-organizing teams, lead to higher productivity and innovative solutions.

2.2. Modern Metrics

Modern metrics to be considered include:

- DORA Metrics in the DevOps Context: Emerging from the need to enhance software development and operations, DORA metrics embody principles of continuous integration and deployment. These metrics, which include deployment frequency and mean time to recover, underscore the importance of swift and resilient software delivery in DevOps.
- Cycle Time and Efficiency: Drawing from lean manufacturing, Cycle Time in software development measures the efficiency of the development process from start to finish. It reflects a focus on minimizing waste and optimizing process efficiency, crucial in the Agile framework for rapid and responsive software delivery.

- SPACE Framework: A Holistic Approach: SPACE, encompassing Satisfaction, Performance, Activity, Communication, and Efficiency, offers a comprehensive view of productivity. This framework challenges the notion that productivity is solely quantifiable, integrating qualitative aspects such as team satisfaction and communication effectiveness.
- Qualitative Measurements: A Human-Centric View: Recognizing software development as a human-centric process, modern metrics also focus on team dynamics, creativity, and user satisfaction. These qualitative measures assess aspects often overlooked by traditional metrics, emphasizing the importance of team morale and customer satisfaction.

3. Factors Influencing Productivity

A study performed by Babu Veeresh Thummadi and Kalle Lyytinen [6] shows that methodologies like Waterfall and Agile have an impact of 40% on a project. This means that development strategies play a significant role in shaping the trajectory of the overall process. However, it's crucial to recognize that the remaining 60% of the equation is intertwined with the intricate tapestry of individual contributions, project-specific conditions, and the unpredictable nuances of the project's environment. This means that the people involved, the unique circumstances of the project, and the external influences on the development landscape contribute to the overall productivity and success of a software project.

By analogy, it can be stated that developers' productivity level is closely linked to factors such as motivation, customer satisfaction, mental health, and team dynamics.

3.1. Motivation in Workspace

Motivation plays a role in driving developers to consistently produce high-quality work.

Intrinsic motivation, which stems from personal interest or enjoyment in the task itself, is a key driver in workplace productivity. Ryan and Deci's Self-Determination Theory (SDT) [7] highlights the importance of intrinsic motivation in fostering employee engagement and satisfaction. In the context of software development, methodologies that offer autonomy, mastery, and purpose, like Agile, tend to enhance intrinsic motivation among developers.

Locke and Latham's Goal Setting Theory [8] suggests that clear, challenging goals and appropriate feedback contribute to higher levels of employee motivation. Agile methodology, with its iterative cycles and continuous feedback, aligns well with this theory. In the Agile methodology, which promotes collaborative development, motivation thrives thanks to its focus on quick feedback and adaptability. On the other hand, the Waterfall

model follows a progression where sustained motivation might be challenged due to longer feedback cycles and difficulty incorporating changes.

The impact of recognition and rewards on motivation is well-documented. Herzberg's Two-Factor Theory [9] considers that while certain factors (like salary and job security) prevent dissatisfaction, it's factors like recognition and achievement that truly motivate employees. Agile methodologies often incorporate regular reviews and acknowledgments of achievements, which can positively impact motivation. This regular recognition provides immediate gratification and acknowledgment of developers' efforts, contributing to a more motivated workforce.

Transformational leadership, which involves inspiring and motivating team members towards a collective goal, is particularly effective in Agile environments. Leaders who demonstrate vision, provide inspiration, and encourage intellectual stimulation can significantly enhance the motivation and productivity of their teams [10].

Both approaches prioritize customer satisfaction, but the iterative nature of the Agile model allows for frequent customer involvement and adjustments, resulting in a more satisfying outcome for the customer and higher levels of satisfaction for developers [11].

The study performed by A. Westendorp reveals that Agile teams exhibited a more robust knowledge network in contrast to their Waterfall counterparts. Additionally, there was an adverse relationship between knowledge boundaries and the transactive memory system within waterfall teams. Furthermore, the transactive memory system demonstrated a positive correlation with both team satisfaction and perceived team productivity, further highlighting potential limitations within the waterfall methodology [12].

3.2. Mental Health

The environment in which software developers work can significantly influence their mental health. Studies have shown that high-stress environments, characterized by tight deadlines and heavy workloads, can lead to increased anxiety and burnout among developers. The Agile methodology, with its emphasis on regular breaks and a sustainable work pace (as outlined in the Agile Manifesto), can mitigate these stressors, promoting a healthier work environment [13].

Social support within the workplace plays a crucial role in the mental health of employees. Agile methodologies often foster a strong sense of community and collaboration, which can provide emotional and professional support to developers. This support system can be particularly beneficial in managing work-related stress and preventing feelings of isolation, common in more rigid and compartmentalized working environments like those found in Waterfall methodologies [14].

Mental health is crucial in software development, and Agile's emphasis on well-being can lead to a more pleasant and healthier work environment, considering the more rigid structure of Waterfall development.

3.3. Team Dynamics

Team dynamics are important for work quality in both methodologies. However, the fact that Agile emphasizes teams managing themselves encourages the formation of cross-functional teams and continuous collaboration promotes communication and an overall better synergy among team members, positively influencing productivity [15].

4. Advanced Metrics

4.1. Cycle Time

Ever since the 90s, a study of NPD best practices sponsored by the Product Development and Management Association [16] showed that nearly 41% of those surveyed indicated that their organizations were developing new products much faster than they did in the past 5 years. This is due to competition, but mostly to a shorter product life cycle and, implicitly, the growing need for new products. Even in the present time, the goal is to minimize cycle time so that new products are launched more quickly.

Cycle time can be influenced by a lot of factors, such as project complexity. Projects that involve advanced features, difficult algorithms, or integrations are often more challenging, so the development process tends to last longer.

Additionally, incorporating new technologies into software development can represent an advantage, as they are faster and support more functionalities needed by developers. However, the time required to learn how to use them can vary and become quite expensive, perhaps even to the point where using older technologies would have obtained the same results in a much shorter time. Therefore, team members must be carefully selected, ensuring they possess a high level of adaptability and quick learning skills.

Team members play an important part in the duration of task completion. For example, cross-functional teams including members with diverse skill sets can enhance productivity, reduce cycle time, and even create an environment conducive to innovation [17].

Product management is also a crucial element in reducing production time. That is why the well-defined and structured development processes successfully meet the expected deadlines. Consequently, cycle time can be directly related to project management methodologies, and reducing it is more likely to be achieved using Agile rather than Waterfall. Agile breaks down the project into manageable and small iterations, delivering a

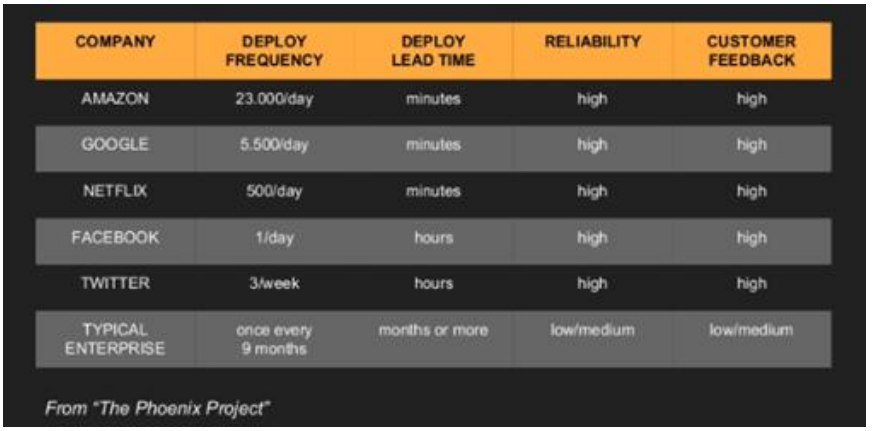
prototype at the end of every sprint. Hence, usable versions of the product are released faster, reducing the overall cycle time [18].

In addition, Agile methodology implies flexibility to change, so adapting to the customer needs and feedback doesn't cost as much time as if it were for a Waterfall project.

4.2. DORA Metrics

The DevOps Research and Assessment (DORA) metrics are a set of key performance indicators designed to measure the effectiveness of software development and delivery processes. Introduced by Dr. Nicole Forsgren, Jez Humble, and Gene Kim in the State of DevOps Reports, these metrics include four main areas: Deployment Frequency, Lead Time for Changes, Change Failure Rate, and Mean Time to Recovery (MTTR) [3].

Major tech companies like Google, Amazon, and Netflix have adopted DORA metrics to optimize their DevOps practices. For instance, Amazon reportedly deploys new software to production every 11.7 seconds [19], demonstrating a high Deployment Frequency. Netflix, known for its robust DevOps culture, emphasizes rapid recovery, aligning with the MTTR metric, to ensure the high availability and reliability of its streaming service.



COMPANY	DEPLOY FREQUENCY	DEPLOY LEAD TIME	RELIABILITY	CUSTOMER FEEDBACK
AMAZON	23.000/day	minutes	high	high
GOOGLE	5.500/day	minutes	high	high
NETFLIX	500/day	minutes	high	high
FACEBOOK	1/day	hours	high	high
TWITTER	3/week	hours	high	high
TYPICAL ENTERPRISE	once every 9 months	months or more	low/medium	low/medium

From "The Phoenix Project"

Figure 1. Comparison of Deployment Metrics Across Leading Tech Companies and a Typical Enterprise [20]

The table compares several major tech companies and a typical enterprise based on their deployment metrics. These metrics include Deployment Frequency, Deploy Lead Time, Reliability, and Customer Feedback. Amazon leads with 23,000 deployments per day and has a deploy lead time in minutes, along with high reliability and customer feedback. Google follows with 5,500 daily deployments. Netflix has 500 deployments per day, and

Facebook has one per day, both with high reliability and customer feedback. Twitter deploys three times a week, again with high marks in the remaining metrics. In contrast, a typical enterprise deploys once every 9 months, with longer lead times, and lower reliability and customer feedback. These metrics illustrate the efficiency and effectiveness of DevOps practices in these organizations.

In practice, these metrics are collected and analyzed through a combination of automated tools and processes. Deployment Frequency and Lead Time for Changes are often tracked using version control and continuous integration tools, while Change Failure Rate and MTTR are monitored through incident management systems. For instance, tools like Jenkins for continuous integration can provide data on deployment frequency, while JIRA or PagerDuty might be used to track incident response times and change failure rates.

The application of DORA metrics has led to significant improvements in software development and operational performance. Companies that excel in these metrics are often categorized as 'elite performers' in DevOps, showcasing higher deployment frequencies, faster lead times, lower change failure rates, and quicker recovery from incidents. This translates into faster time-to-market, improved customer satisfaction, and enhanced competitiveness in the industry.

Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

Figure 2. DORA Metrics Categorization for Software Delivery Performance [21]

The table outlines the categories of performance metrics for DevOps practices, classifying them into 'Elite', 'High', 'Medium', and 'Low' based on Deployment Frequency, Lead Time for Changes, Time to Restore Service, and Change Failure Rate.

While DORA metrics provide valuable insights, their implementation can be challenging, especially in organizations with legacy systems and traditional development practices. It requires a cultural shift towards embracing DevOps principles, investing in the right tools and technologies, and training teams to effectively leverage these metrics for continuous improvement.

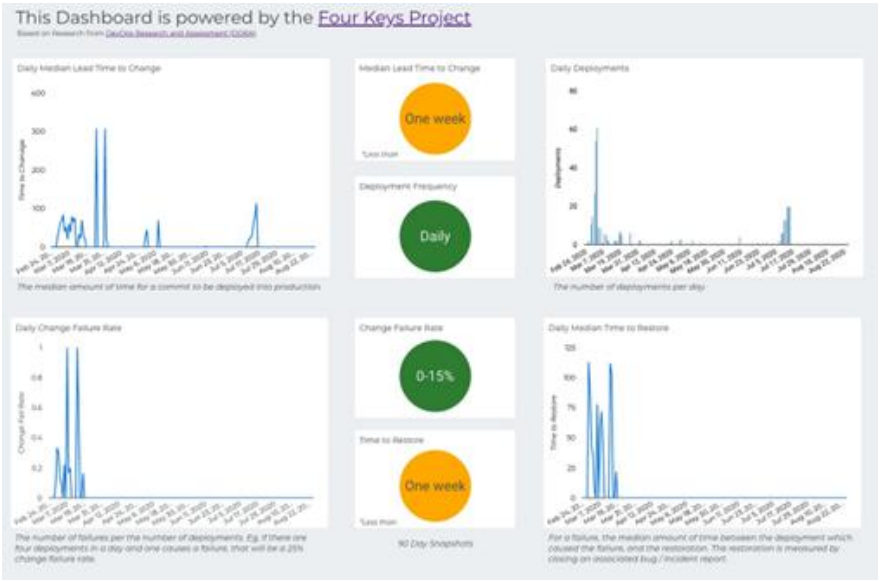


Figure 3. DevOps Performance Metrics Dashboard Based on DORA Research [21]

This dashboard presents a visual representation of key DevOps performance metrics over time, based on the Four Keys Project and DORA research. Daily Median Lead Time to Change: This chart shows spikes in the time it takes for changes to go from commit to production, suggesting variability in the process. The median lead time seems to fluctuate significantly over the months displayed. Daily Deployments: This histogram indicates the number of deployments per day, with noticeable peaks and troughs, implying inconsistency in the daily deployment frequency. Daily Change Failure Rate: The chart illustrates the ratio of deployments that fail, represented as a proportion. The data points show that most of the time, the change failure rate is low, with occasional spikes. Daily Median Time to Restore: This graph shows the time taken to restore service after a failure, with the median time also varying but with some high peaks, indicating instances of longer restoration times. In the center of the dashboard, summary metrics are provided: Median Lead Time to Change is summarized as "less than one week.", Deployment Frequency is noted as "daily.", Change Failure Rate is between "0-15%." and Time to Restore is summarized as "less than one week." This summarizes the overall performance, suggesting a generally high operational

standard, with daily deployments and change lead times and restoration times within a week, alongside a low change failure rate.

4.3. SPACE Metrics

Tech giants like Microsoft and Google have started to adopt frameworks like SPACE [22] to evaluate developer productivity. These companies recognize that productivity is not just about output but also about employee satisfaction, effective communication, and efficient workflows. For example, Microsoft uses various internal tools to gauge developer satisfaction and performance, integrating these insights into their software development process.

Implementing SPACE involves collecting data across different dimensions. Satisfaction can be measured through regular surveys and feedback mechanisms. Performance might be tracked via project milestones and individual contributions. Activity data can be gathered from version control systems. Communication effectiveness is often assessed through team meetings and collaborative tools analytics. Lastly, efficiency is evaluated by analyzing the time and resources utilized for completing tasks [23].

Adopting SPACE in an organization is not without challenges. It requires a cultural shift towards valuing qualitative aspects of work, as well as implementing systems for regularly collecting and analyzing diverse data types. Additionally, there is a need to balance the collection of these metrics with concerns about privacy and the potential for micromanagement. Ensuring that data collection is transparent and used constructively is crucial for the successful implementation of the SPACE framework.

When implemented effectively, SPACE can provide a comprehensive view of productivity, encompassing both the well-being of software developers and the efficiency of the development process. This holistic approach can lead to improved team morale, higher quality of work, and ultimately, more successful software projects. Companies that have embraced similar multifaceted productivity frameworks report better employee retention, increased innovation, and enhanced project outcomes [24].

5. Survey and Data Analysis

5.1. Survey Methodology

The survey methodology is based on the following principles:

- Data Collection Approach: A survey was conducted on a sample of 51 working students from the Polytechnic University of Bucharest, all within the age bracket of 21 to 23 years and with less than two years of experience in the IT industry. This demographic was selected to yield insights into the productivity and work habits of

nascent professionals in the IT field. The survey was disseminated using the public communication channels frequented by fourth-year students from the Faculty of Automatic Control and Computers.

- Survey Design: The survey was crafted to consist of three segments: participant sorting questions, main questions, and questions for understanding demographics.

The main questions concerned:

- Self-assessment of workplace productivity
- Perceived efficiency of the development model in use
- Average weekly time spent in team or managerial meetings
- Focus on discussions during meetings.
- The proportion of tasks completed within the initial time frame allocated
- Ease of finding help for tasks within their team.

The survey's design aimed to encapsulate a comprehensive understanding of the participants' experiences, measuring not only quantitative aspects such as task completion rates and meeting times but also qualitative dimensions like satisfaction with the development model and teamwork dynamics.

5.2. Survey Results and Interpretation

Data highlighted that a mere 3.8% of respondents were using the Waterfall methodology, while a significant majority of 84.6% employed Agile methodologies. This distribution underscores the prevalence and popularity of Agile practices among the participating companies and teams.

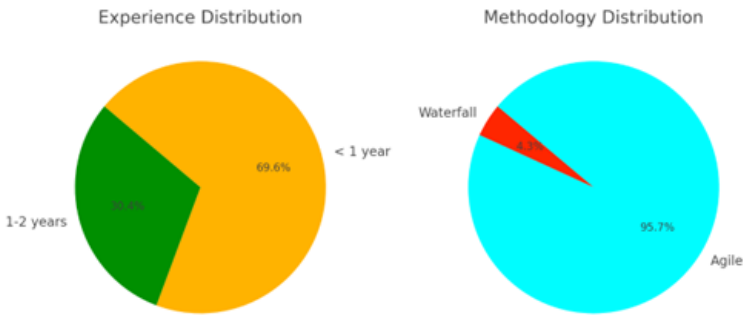


Figure 4. Analysis of Methodology Preference and Experience among Young Software Developers

The analysis highlights:

- Experience Distribution: The pie chart illustrates the distribution of experience among the students. 30.4% have 1-2 years of experience, while 69.6% have less than a year of experience in software development.
- Methodology Distribution: This pie chart displays the methodologies used by the students. Only 3.8% use the Waterfall methodology, whereas a significant majority of 84.6% use the Agile methodology.

The collected data revealed that Agile users reported an average productivity level of 7.5 out of 10 and rated the methodology's effectiveness at 7.8. Conversely, Waterfall users reported lower productivity and satisfaction levels, with scores of 6 (productivity level) and 5 (methodology rating). These findings illustrate a distinct preference for Agile in terms of both perceived productivity and satisfaction.

Regarding task completion, Agile practitioners reported an 83.2% success rate in completing tasks within the allocated time, slightly higher than the 80% reported by Waterfall users. Meeting durations varied, with Agile meetings averaging 3 hours weekly, typically in the form of daily stand-ups, compared to 2 hours for the more traditional, weekly Waterfall status meetings.

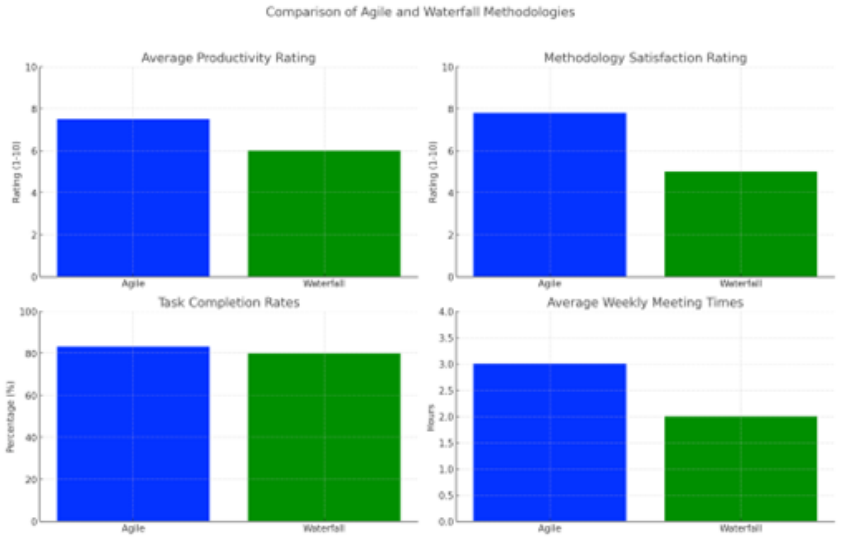


Figure 5. Comparative Analysis of Agile and Waterfall Methodologies Based on a Survey of Early-Career IT Professionals

The graphs present a comparison of Agile and Waterfall methodologies based on a survey conducted with early-career IT professionals. Average Productivity Rating: Agile

methodology users report higher productivity ratings, with an average score of around 7.5 out of 10, while Waterfall users report an average of 6. Methodology Satisfaction Rating: Satisfaction with the Agile methodology is higher, scoring around 7.8 out of 10, as opposed to a 5 for Waterfall. Task Completion Rates: Agile users report a higher task completion rate at approximately 83.2%, compared to 80% for Waterfall users. Average Weekly Meeting Times: Agile methodology entails longer weekly meetings, averaging about 3 hours, whereas Waterfall averages about 2 hours per week. These findings suggest that the Agile methodology is perceived to be more productive and satisfactory among the participants, with a slightly higher task completion rate and longer meeting times compared to the Waterfall methodology. The study highlights the prevailing preference for Agile practices among the surveyed early-career professionals.

As statistical relevance, it is imperative to note the limited representation of Waterfall methodology among the survey participants. This underrepresentation may skew the comprehensiveness of the data related to Waterfall, necessitating a cautious interpretation of these results.

The overarching aim of this survey was to acquire empirical evidence on the efficacy and perceived productivity of Agile versus Waterfall methodologies among budding IT professionals. The study meticulously evaluated factors such as self-assessed productivity, developmental model efficiency, meeting durations, the focus of discussions, task completion rates, and the ease of obtaining assistance within teams. The subsequent analysis endeavors to provide an exhaustive overview of how these methodologies tangibly influence the daily working experiences of those just commencing their careers in the IT sector.

6. Conclusions

In conclusion, this article has delved into the multifaceted reasons why Agile methodology is a substantial enhancer of productivity compared to the traditional Waterfall approach. The core principles of collaboration, flexibility, and customer-centricity that underpin Agile, not only foster a more responsive and efficient work environment but also contribute to the superior productivity achieved through Agile methodology.

However, thoughtful consideration when selecting a development methodology is crucial. Project managers and teams should weigh the benefits and constraints of each approach against the unique characteristics of the project at hand. Software development has a dynamic nature, where there is a synergy between the chosen methodology and the collective efforts of the individuals involved. The adaptability to project conditions and environmental dynamics contributes significantly to the ultimate success and productivity of the endeavor.

References

[1] D. RADIGAN *Agile vs. Waterfall Project Management,* https://www.atlassian.com/agile/project-management/project-management-intro Date of access: 17.01.2024

[2] J. COLINA *Software Development Metrics: Top 5 Commonly Misused Metrics* https://www.usehaystack.io/blog/software-development-metrics-top-5-commonly-misused-metrics Date of access: 17.01.2024

[3] N. FORSGREN, J. HUMBLE, G. KIM *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*, IT Revolution Press, 2018.

[4] C. JASPAN, C. GREEN Podcast *Engineering Enablement by Abi Noda*, episode 44 - *How Google Measures Developer Productivity* https://share.transistor.fm/s/b36e515c Date of access: 17.01.2024

[5] K. BECK, M. BEEDLE, A. VAN BENNEKUM, A. COCKBURN, W. CUNNINGHAM, M. FOWLER, D. THOMAS *Manifesto for Agile Software Development*. Agile Alliance, 2001.

[6] B.V. THUMMADI, K. LYYTINEN *How Much Method-in-Use Matters? A Case Study of Agile and Waterfall Software Projects and their Design Routine Variation*, Journal of the Association for Information Systems, 2020, https://aisel.aisnet.org/jais/vol21/iss4/7/ Date of access: 18.01.2024

[7] R. M. RYAN, E.L. DECI *Intrinsic and Extrinsic Motivations: Classic Definitions and New Direction*s, Contemporary Educational Psychology, Volume 25, January 2000, Pages 54-67

[8] E.A. LOCKE< G.P. LATHAM *Building a Practically Useful Theory of Goal Setting and Task Motivation (A 35-Year Odyssey),* 2002, DOI: 10.1037/0003-066X.57.9.705

[9] F. HERZBERG *One More Time: How Do You Motivate Employees?* Harvard Business Review Magazine (January 2003) https://hbr.org/2003/01/one-more-time-how-do-you-motivate-employees Date of access: 17.01.2024

[10] B.M. BASS, R.E. RIGGIO *Transformational Leadership*, 2nd ed., Lawrence Erlbaum Associates Publishers, 2006

[11] I. FATEMA, K. SAKIB *Factors Influencing Productivity of Agile Software Development Teamwork: A Qualitative System Dynamics Approach*, 24th Asia-Pacific Software Engineering Conference (APSEC), 2017

[12] A. WESTENDORP *Agile versus Waterfall Methods: Differences in Knowledge Networks and Performance in Software Engineering Teams*, 2016 https://essay.utwente.nl/69311/ Date of access: 17.01.2024

[13] C. ROOK *Managing to Make a Difference: An Investigation into the Influence of Managers' Activities on Employee Well-Being* Journal of Business Ethics, 124(3), 2016

[14] S. OZKAN, K. CENGIZ *The Relation between Employee Social Support, Stress, and Work-Family Conflict in IT Organizations*, 2017.

[15] R. V. O'CONNOR, S. BASRI *The Effect of Team Dynamics on Software Development Process Improvement*, 2012 https://www.researchgate.net/publication/262201629_The_Effect_of_Team_Dynamics_on_Software_Development_Process_Improvement Date of access: 17.01.2024

[16] A. GRIFFIN *The Effect of Project and Process Characteristics on Product Development Cycle Time*, 1997, DOI: 10.2307/3152062

[17] V. KASHYAP *What Is a Cross-Functional Team, and How to Build One?* 2022,

https://businessmap.io/blog/cross-functional-teams Date of access: 17.01.2024

[18] K.S. RUBIN *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley, 2012.

[19] C. NULL 10 Companies Killing It at DevOps, TechBeacon, https://techbeacon.com/app-dev-testing/10-companies-killing-it-devops Date of access: 17.01.2024

[20] https://www.altexsoft.com/blog/dataops-essentials/ Date of access: 17.01.2024

[21] https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance Date of access: 17.01.2024

[22] A. ZAGALSKY, F. FIGUEIRA FILHO, L. SINGER, D.M. GERMAN (2019). *Towards an Understanding of the Productivity of Software Developers*. In Proceedings of the 41st International Conference on Software Engineering. IEEE Press, 2019.

[23] C. BIRD, T. ZIMMERMANN *Assessing the Value of Productivity Tools for Software Engineers*. In Proceedings of the 40th International Conference on Software Engineering. ACM, 2018.

[24] N. FORSGREN, M. KERSTEN *The SPACE of Developer Productivity: There's More to It than You Think*. Communications of the ACM, 64(7), 30-35, 2021.

Bibliography

S. KOMAI, H. SAIDI, H. NAKANISHI *Man-Hour Comparison Between Two Methods of Agile and Waterfall in IT System Development* Proceedings of the 13th International Conference on Innovation and Management, VOLS I & II, Page 829-836, 2016

G. MELNIK, P. KRUCHTEN, M. POPPENDIECK *Moving from Waterfall to Agile* Agile 2008, Proceedings Page 97-101, DOI10.1109/Agile.2008.49

A. RAHMAN, L.M. CYSNEIROS, D.M. BERRY *An Empirical Study of the Impact of Waterfall and Agile Methods on Numbers of Requirements-Related Defects* 39th Annual ACM Symposium on Applied Computing, SAC 2024, Page 1143-1152, DOI: 10.1145/3605098.3635901

T. TURECEK, R. SMIRÁK, T. MALÍK, P. BOHÁCEK *Energy Project Story: From Waterfall to Distributed Agile* Agile Processes in Software Engineering and Extreme Programming, Volume 48, Page 362-371, 2010

O. ADELAKUN, T. IYAMU *Translation of Activities in a Global Virtual Teams Software Development: Agile vs. Waterfall* Journal of Cases on Information Technology Volume 23, Issue 4, 2021, DOI: 10.4018/JCIT.20211001.oa11