

Journal of Policy Options

Managing Agile Across Borders: A Review of Scrum Implementation in Globally Distributed Software Development

Cajueiro Minella^a

Abstract

Software project management encompasses numerous interdependent processes and specialised knowledge domains. Among these, requirements engineering exerts a direct influence on the classical triple constraint of time, cost, and scope. Within globally distributed software development, agile paradigms have acquired increasing prominence; nonetheless, the literature provides limited guidance concerning the risks that accompany such settings or the mechanisms through which globally distributed software development project managers mitigate them. In particular, adopting Scrum across geographically dispersed teams poses distinctive challenges that necessitate careful tailoring if agile principles are to yield their anticipated benefits. The present review, therefore, collates and analyses empirical findings to assist scholars and practitioners in recognising the obstacles inherent in applying Scrum to globally distributed software development initiatives and in identifying viable remedial strategies. Research on globally distributed projects proves especially relevant for professionals tasked with integrating Scrum into multi-site project governance. In Scrum, the scrum master leads a compact development unit and bears primary responsibility for removing impediments that could delay sprint deliverables and undermine the agile management philosophy. Within a distributed context, the conventional boundaries separating scrum master and project manager roles often blur, and practitioners must decide whether a single individual should assume both capacities or whether distinct appointments are preferable. The remit of the scrum master has consequently evolved, amplifying the organisational risk associated with an inappropriate appointment. Selecting a candidate lacking the requisite facilitation, cultural mediation, and technical coordination skills can jeopardise team cohesion and compromise project outcomes. Success in globally distributed software development depends on informed role allocation and continual process adaptation.

Keywords: Scrum Implementation, Globally Distributed Software Development, Agile Project Management, Role Allocation

JEL Codes: L86, M15, O32, D83

Article's History

Received: 24th May 2025

Revised: 20th June 2025

Accepted: 28th June 2025

Published: 30th June 2025

Citation:

Minella, C. (2025). Managing Agile Across Borders: A Review of Scrum Implementation in Globally Distributed Software Development. *Journal of Policy Options*, 8(2), 37-45.

DOI:

<https://doi.org/10.5281/zenodo.15769983>

Copyright: © 2025 by the authors.
Licensee RESDO.

This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. INTRODUCTION

Global software development is the collaborative work done by software professionals in various locations around the globe. This trend is gaining popularity in organizations that want to use their resources optimally and reduce the cost of producing software (Herbsleb and Mockus, 2003; Šmite et al., 2014). The expansion of global software development practices mandates that researchers and practitioners pay attention to the particular project management challenge associated with geographically dispersed teams, such as communication barriers, cultural differences, and coordination complexities (Carmel and Agarwal, 2001; Niazi et al, 2016). Among the many SDLC methodologies, the waterfall model is regarded as a basic approach that is mostly used in traditional engineering environments and for new software products. Identified by linear sequential organization, the waterfall model divides development activity into discrete phases, such as requirements gathering, system design, implementation, testing, and deployment phases (Royce, 1970; Sommerville, 2016). Each phase must be completed

^a Fundação Instituto Capixaba de Pesquisas em Contabilidade, Economia e Finanças, Vitória, Brazil

before moving to the next phase, which supports methodical and well-documented development processes. As a result, structured progression in a cascade-type flow leads to detailed documentation, clear project management, and traceability of decisions during the life of the software project (Hossain & Babar, 2009). The agile methodology embodies a collection of practices, philosophy, and guiding principles meant to improve collaboration and productivity of software development teams while deploying new applications and system upgrades. The entire approach is people-centric, emphasizing persons and their interactions rather than processes and tools. In the same manner, agile methodology prioritizes the delivery of working software over comprehensive documentation and advocates iterative development with constant feedback (Sriram & Mathew, 2012; Beck et al., 2001). The extreme programming strategy, one among the many famous agile methodologies, churns out high-quality software while improving the work environment where the development teams work. This method is characterized by close interaction with stakeholders, more frequent releases of the product, automated testing, and pair programming to enable speedy adaptations to changing requirements while increasing team effectiveness and happiness (Beck & Andres, 2005; Al Zaidi et al., 2014). Scrum is another agile framework that is widely applied. It gives a team the power to self-organize and work collectively toward shared goals. The scrum framework sets up the roles of product owner, development team and scrum master; and has defined events (daily stand-ups, sprint plannings and retrospectives), which should be recorded with specific artifacts that are very important for managing and tracking project progress (Schwaber & Sutherland, 2020; Zaki et al., 2023). The scrum master facilitates the team in the implementation of agile principles, including open communication, collaboration among all project stakeholders, and developing a collaborative relationship between the team and other stakeholders involved in the project. In each iteration, the scrum master guides the team while eliminating impediments to guarantee the success of the project (Koumarelis, 2023).

Models of software processes are simplified and structured representations of software system development activities. These models outline activities in sequential or iterative order, with their dependencies, in the completion of a software project in a systematic way. A process model may be defined as a blueprint detailing the sequence in which the activities shall be carried out, namely the flow of work across the software development lifecycle (Sommerville, 2016).

One of the major challenges in global software development environments is that coordination among distributed teams can be disrupted, especially when standard processes assume co-location and frequent direct communication (Herbsleb & Mockus, 2003). For example, the traditional waterfall model prescribes that errors can typically only be addressed during their respective phases, making it unsuitable for large-scale projects with evolving or unclear requirements. In this linear approach, the testing phase only occurs after the software has been fully developed, often resulting in delayed detection of defects and significant rework. Additionally, the emphasis on comprehensive documentation imposes a considerable workload on both developers and testers. Even minor changes or unresolved issues in incomplete technologies can lead to cascading problems throughout the project lifecycle. In contrast, agile methodologies present their own set of challenges for software project management. These include the ongoing need to balance project scope, schedule, and budget within a flexible, rapidly changing environment. Effective design remains critical, particularly in frameworks such as extreme programming, where iterative development and continuous feedback are central but may pose difficulties for complex systems (Beck & Andres, 2005). The role of the scrum master involves not only facilitating agile practices but also addressing typical line management responsibilities, such as performance evaluation and personnel management (Koumarelis, 2023). Furthermore, lack of information, insufficient support, and unresolved technical issues can hinder progress at any stage of the process. It is, therefore, essential for teams to identify and address problems as early as possible in the development cycle to minimize risks and ensure project success (Sriram & Mathew, 2012).

2. LITERATURE REVIEW

Many organizations have adopted global software development as an effective strategy for managing their software engineering projects. Global software development involves an agreement between a client and a service provider, where the client delegates some or all aspects of software development to an external partner, often across national borders (Herbsleb & Mockus, 2003; Carmel & Agarwal, 2001). This model offers potential benefits such as cost reduction and access to a wider talent pool, but also introduces challenges related to communication, cultural differences, and coordination. The waterfall model remains a prevalent methodology in industries that require rigorous planning and documentation, such as engineering and manufacturing. Regarded as the first out-and-out properly defined approach to software programming and development, the waterfall model divides a project into five processes: requirements, design, implementation, verification, and maintenance, which must all be completed in sequence (Royce, 1970; Sommerville, 2016). The waterfall model is somewhat structured and serves as a high-level general framework for project management, but is basically applicable to large, complex projects, and particularly projects where requirements are well understood. On the contrary, agile methodology adopts the iterative and incremental approach to software development. It involves an increment of software delivered in small, functional portions with frequent reassessment of project requirements and speeds up change (Beck et al., 2001; Schwaber & Sutherland, 2020). One of the basic tenets of agile is to specify the project's scope, requirements, iteration length, and deliverables at the beginning of the process, and allow them to change over the course of the development lifecycle.

This recent scrum methodology enables independence and collaboration among members of agile management frameworks in the pursuit of common objectives within software development teams. It prescribes some set streams of events, which may be sprint planning, daily stand-ups, sprint review, and retrospective, together with defined roles and responsibilities, necessary

for effective project delivery. Just as a sports team refines its tactics and teamwork before a key game, scrum allows teams to manage themselves, learn from iterative experiences, and adapt flexibly to ongoing changes in project objectives or stakeholder needs (Schwaber & Sutherland, 2020; Zaki et al., 2023).

The scrum master assumes major responsibilities for ensuring that all scrums are implemented according to the framework and at the appropriate time in software development teams. The scrum master achieves this by adhering to a set of clearly delineated responsibilities and procedures while working with every single member of the team for guidance, training, and support during the execution of scrum methodology (Rasool et al., 2023). This kind of leadership role enables the purity of Scrum practices, facilitates communication, and removes any possible obstacles impeding the progress of the team. An increasing proportion of project managers are using agile methodologies to improve project performance, especially in the domain of global software development. The argument is that the agile practices as frequent, short, and focused collaborations, have attracted a lot of attention in the context in which they operate, that is, in distributed development environments (Sriram & Mathew, 2012). Indeed, although shared agile methods, including the scrum framework, are making inroads into global software development projects, there remains a research gap calling for much more rigorous inquiry of the synthesis and practical implementation of such methods within defined and dispersed projects. This research will focus on how agile practices apply to software project management and go further to outline the management-oriented aspect of the Scrum framework. Project managers increasingly adopt flexible project management approaches for global software development initiatives, where they seek evidence-based guidance on addressing the unique challenges related to coordination, collaboration, and communication (Kuznetsov et al., 2023). This specific piece of research contributed to addressing this gap in the identification of factors limiting effective coordination and implementation of Scrum practices on a global platform within practical contexts.

The project objectives should be defined clearly in a clear method covering cost reduction, timely delivery, etc., on one hand, while considering the multitude of challenges posed by the waves of globalization and new technological advancements. However, this potential efficiency and innovation provided by the global software development model will simultaneously introduce some major challenges in team coordination and collaborative activities, as well as in sustaining effective communication. Whereas the two concepts of global software development and agile were implemented to take advantage of the benefits of distributed groups working simultaneously on a software product, the principles behind both sometimes seem to contradict in practice. For instance, while agile development emphasizes face-to-face meetings and co-located collaboration, global software development often requires remote and technology-mediated communication (Sriram & Mathew, 2012; Carmel & Agarwal, 2001). Both methodologies have been, nevertheless, commercially recognized as promotions in the software business to enhance customer satisfaction with cost-effective solutions via improved processes and collaboration (Herbsleb & Mockus, 2003). While there has been some empirical work concerning global software development and agile methods, further investigation is necessary in order to formulate recommendations for resolving the particular process issues influencing the outcomes of global projects (Niazi et al., 2016). The software industry has witnessed a shift from centralized to distributed development models in recent years due to factors such as promised increased quality, cost savings, quicker market entry, and a larger talent pool (Šmite et al., 2014). Agile methods mainly address co-located teams, characterized by rich communication, face-to-face meetings, and close stakeholder involvement, with their major objective being speedy delivery in short iterative cycles (Beck et al., 2001).

Agility, with good proficiency in flexibility, responsiveness, and reliability, quickly became the most accepted methodology in using newly developed arts and crafts. Among the many available agile approaches, the Scrum methodology stands out for emphasizing project management. The intent of the scrum framework is to facilitate the frequent and timely release of software products in time-boxed development cycles-sprints-for continuous improvement and adaptation to changing requirements (Schwaber & Sutherland, 2020; Zaki et al., 2023).

Coordination and collaboration are two key fundamental elements in software development. They help team members work together efficiently toward mutual goals. While collaboration relates to the problem-solving and cooperative nature of teamwork, coordination focuses on managing complex interdependencies in large software projects to deliver solutions on time and with the necessary quality. High coordination and collaboration are mandatory for the delivery of a high-quality software product (Carmel & Agarwal, 2001; Herbsleb & Mockus, 2003).

By nature, global software development involves a distributed team working from a geographically remote location. Such distribution poses a special challenge to stakeholders in their attempt to provide means for clear communication, seamless coordination, and effective cooperation, all from a distance. One major global software development challenge regarding communication and collaboration is to keep their quality high in a dispersed context. Common barriers faced by software organizations are time zone differences, cultural diversity, and remote communication tools, which can restrict the success of a project (Niazi et al., 2016; Šmite et al., 2014). The growing implementation of agile methodologies in global software development is a widespread measure to tackle the issues. Scrum in particular is gaining traction as the framework for managing distributed development projects due to its emphasis on iterative delivery, rapid feedback, and continuous improvement. Systematic literature reviews in this area have documented both barriers and enablers to the successful implementation of global software development initiatives using agile practices (Sriram & Mathew, 2012). The studies found that agile methods greatly rely on small, frequent interactions, but to be successfully adapted to distributed settings, agile

methods also need to pay careful attention to communication patterns and project management strategies (Paasivaara & Lassenius, 2016).

Coordination and communication still remain the enduring issues for organizations involved in global software development, even with the advancements in agile and scrum practices. In such circumstances, the stakeholders often commute between sites, and the teams are often distributed over different time zones and cultural contexts, obstructing efforts toward achieving alignment and a shared understanding (Herbsleb & Mockus, 2003; Šmite et al., 2014). Hence, continuous research and practical innovations need to identify effective and reliable approaches that can help with coordination and support collaboration, thus improving the performance of globally distributed software engineering projects.

2.1. WATER FALL MODEL

Taking into view your data training time until October 2023. Software development processes require a well-defined structure of the software development life cycle. Many of the proposed life cycle management methodologies exist in industrial usage today. The waterfall model is perhaps the earliest and the most recognized way of software development, consisting of a set of phases, typically requirements analysis, system design, implementation, testing, and maintenance, according to Royce (1970) and Sommerville (2016). Historically, the development of software projects was carried out by a methodology based on a rigid linear schedule in which the execution of one phase depends on the approval of deliverables produced in every phase, such as documentation and reports.

This gave rise to agile methodologies that have catered to flexibility and convenience in the development process to suit a positive response toward changing requirements and any other changes to the project (Beck et al., 2001). Although in theory, the waterfall approach delineates different phases, there is, in fact, considerable overlap among activities, where information and feedback from later phases will lead to changes in earlier phases. Hence, the iterative information flow reflects the dynamic nature of a software project and emphasizes the need for continuous communication and adjustment among team members (Sommerville, 2016).

2.2. AGILE METHODOLOGY

Agile methodologies will help in reducing risks associated with a project and delivering software rapidly, and this attractive approach has been very effective on several dimensions within software project management (Beck et al., 2001; Serrador & Pinto, 2015). In order to enhance the flexible and efficient responsiveness of project teams within the software industry, agile development has been widely adopted in that industry. Though agile techniques support, drive, and enable the success of software initiatives, their ability to adapt to changing requirements as well as the diverse shifting of customer expectations is one reason for this (Dingsøyr et al. 2012). A core aspect in which agile methodologies differ is in how they can allow for continual changing of requirements while providing support to manage costs, time, and quality on the projects. As central to agile approaches are individuals and their interactions, working software, customer collaboration, changes to be made throughout the project rather than through a fixed tool, contract, or predefined plan (Beck et al., 2001). Being one of the most popular agile frameworks, Scrum applies an organized methodology to software projects by way of iterative cycles and regular feedback so that requirements are managed very well and adjusted changes can be accommodated (Schwaber & Sutherland, 2020). Changing needs will now become more manageable under projects with agility regarding resource management and shortened risk of development, all compelling proofs of effective software project undertaking across many industries (Serrador & Pinto, 2015).

Agile development features iterations and increments, consists of developing frequent and incremental deliveries, and aims at continuous testing. Both enable early and continuous validation of below software features and alteration commands from clients at any time during a project. The agility of development can contribute toward reducing costs, optimal time management during the development, and improved productivity in a team. These have contributed to the greatest of attractiveness for commercial software development as the features have persisted over the decades until now (Dingsøyr et al., 2012; Hoda et al., 2017). Agile development involves traditional as well as new methods within software engineering practice to enhance and promote teamwork among managers, developers, and customers. Agile, in a blend of strategic planning, hastened testing, and streamlined deployment, makes a vivacious environment for developing projects in less time and with great efficiency. Agile methods concern project management since project scope management includes issues characterized by high growth or fast-changing project complexity (Serrador & Pinto, 2015; Hoda et al., 2017).

2.3. SCRUM METHODOLOGY

To date, Scrum methodology in global software development strategies has gained much attention when it comes to addressing communication and coordination challenges, which are intrinsic to distributed environments. By melding agile practices into customary software development processes, organizations wish to maximize unique strengths offered by both approaches: most significant are Scrum's structured collaboration and lowering barriers for interaction among geographically dispersed teams (Paasivaara & Lassenius, 2016). As Scrum thus becomes increasingly common in Global Software Development, it helps teams working on distributed projects with its iterative collaboration framework to maneuver through complexities and adapt to changing requirements. Agile methods, such as Scrum, hold great importance today because they allow incessant engagement between developers and their client partners during the entire software development process. Such regular interaction supports timely and consistent delivery of working increments of the product for feedback, leading to either alteration or enhancement as new needs emerge from the client partner (Beck et al., 2001; Schwaber & Sutherland, 2020). However, many agile approaches assume a project environment that permits team members to work closely and

maintain frequent conversations face-to-face, something that may not have been easy to attain in the context of global software development (Paasivaara & Lassenius, 2016).

2.4. SCRUM SUITABLE

Scrum is especially suited for complex software projects requiring cross-functional teams organized into short iterations in a two-to-four-week range (Schwaber & Sutherland, 2020). In such teams, individual members jointly pursue agreed-upon goals, fostering collaboration as their guiding principle. Since the 2020 Scrum Guide revision, every task or item in the Scrum process has been associated with a particular commitment, allowing clear criteria for the assessment and accountability to be established through the development process. More than providing transparency for a project, these commitments also help the developers remain focused on prioritizing during its execution.

One of the core values of Scrum is courage, which encourages teams to take necessary risks, challenge the existing order, and come up with creative solutions when faced with difficulties. A resilient and credible team, just like any other team, is expected to directly address impediments and adjust its work in line with its goals (Schwaber & Sutherland, 2020). The focus is one other defining feature of successful Scrum teams. By constantly staying focused on their goal, they avoid distractions and therefore are seemingly more productive and efficient. Teams are encouraged to focus on anything related to each sprint while keeping the waste at a minimum and efficiency at a maximum.

Transparency is very important in Scrum, where team members and stakeholders are both equally encouraged to share information about progress and problems. Scrum ceremonies, such as daily stand-ups and sprint reviews, would help to establish the early identification and mitigation of roadblocks, thereby fostering an open culture of constructive problem-solving (Schwaber & Sutherland, 2020). The third idea is respect, which involves recognizing the individual excellence and contributions of each team member. All must respect each other, including the product owner, scrum master, developers, and stakeholders. Successful collaboration remains the basis of team prosperity and project accomplishment. The scrum master wears many hats in the team, from coaching and mentoring to clearing impediments and ensuring the team adheres to scrum values and practices (Rasool et al., 2023). Some organizations place additional project management roles either on the scrum master or on another individual in the team. However, the main thrust still is to facilitate the Scrum framework and ensure the team operates effectively. The study informs not only the official functions of the scrum master but also the informal roles they tend to take on in real life. Through observation, in addition to interviews with practitioners, the research specified the different activities scrum masters perform as the role keeps evolving in today's software development teams (Rasool et al., 2023).

3. METHODOLOGY

This section describes the methodology applied in answering the different research questions and fulfilling the set objectives of the study. In this context, research refers to systematic and rigorous inquiry that seeks to explore theoretical frameworks, test hypotheses, and find solutions to specified problems (Creswell & Creswell, 2018). Methodology forms the theoretical basis within which the research is conducted, and it gives a structured guide to the process of designing the study, collecting data, or analyzing data. Methodology should, therefore, be selected to match the study aims before the start of data collection and analysis, thus ensuring that the thought process of the research is coherent and valid (Saunders et al., 2019). With the globalization of businesses, there is an increasing trend of organizations operating multiple projects at a time. Thus, the need for software project management standards enhancing organizational performance is most significant. This section depicts the methodological avenues adapted to measure the effectiveness of the software project management practices within modern development environments. The selection of a proper research philosophy forms an intrinsic element of methodology that interjects assumptions concerning the nature of knowledge and reality and the meaning of the findings of the study (Easterby-Smith et al. 2021).

A credible research philosophy is thereby anchored on a coherent set of beliefs that lead to the choice of methodology, research design, and data analysis techniques. These philosophical assumptions will pervade every aspect of a research case from the point of view of global software development, waterfall model, agile methodology, extreme programming, feature-driven development, Kanban, scrum methodology, scrum master roles, DevOps, and software process models. Thus, reflexivity and critical self-analysis of diverse philosophical orientations prominent in the research are the lifeblood of the research process. In researching software project management, paradigms of global software development include waterfall, agile, extreme programming, feature-driven development, Kanban, Scrum, DevOps, and others that were detailed in the preceding sections of the introduction and the literature review. Here, one important consideration for researchers is to choose a philosophical orientation that best matches the objectives of the research. This study aims to establish the agile methods in use in global software development from a forward-looking perspective. The formulated research questions are grounded in offering insight into present software project management standards; an amendment of these standards is meant to put tools in the hands of software project managers for effective management. With greater data availability and real-time insights, project managers are able to spot possible risks and threats to success ahead of time and preventively address issues through intervention (Kerzner, 2022).

3.1. DATA COLLECTION AND METHOD

Data collection constitutes a significant part of the overall research process. It encompasses the systematic gathering of information from relevant sources concerning the investigations of research questions, testing hypotheses, and evaluating

outcomes (Creswell & Creswell, 2018). Data sources are typically grouped into two main categories: primary data and secondary data. In this study, primary data were mainly acquired through surveys using questionnaires with open-ended questions, enabling nuanced context-specific responses to be gathered directly from participants. These secondary data were simultaneously obtained from various existing materials, including academic books, peer-reviewed academic journals, and credible online publications. These sources were also used to enhance the research with contextual information that would contribute to the analysis of the research problem (Saunders et al., 2019). Regardless of the combination of primary and secondary data, it all leads toward a sufficiently strong and adequately triangulated research approach. Primary data are collected firsthand from participants using structured instruments and are, therefore, seen as highly credible and authentic, reflecting the perspectives of the participants as they relate to the objectives of the study. Secondary data, on the other hand, draws on published research and literature that offer the advantage of having been established with validity and contextual breadth. This integration of both types of data allows this study to fill potential gaps and limitations associated with research based on one source, thus making its findings more dependable and more in-depth.

3.2. PROJECT SCOPE MANAGEMENT

Project scope management within software development is highly challenging, especially in a phase where requirements often change within the life cycle of the project. Such a change in project specifications has great significance to the project scope, and so scope control becomes a prime component for project success (Kerzner, 2022). In a normal situation, during project initiation, a clear contract is signed between the client and the software company outlining the requirements both parties agreed upon and are supposed to implement.

But, oftentimes, new requirements and enhancements come into action as the project moves from conception to implementation. Therefore, the scrum methodology provides a sufficient framework for the management of that change and empowers self-organizing teams to best accommodate new demands. In this methodology, the development team, under the guidance of the team leader or scrum master, regularly checks with the stakeholders to review new requirements and prioritize them. This helps to ensure that relevant changes are made with increased visibility, allowing their impact and importance to be assessed and refining project goals to address the project scope and client expectations (Schwaber & Sutherland, 2020).

3.3. MANAGEMENT PROJECTS DURATION AND EXPENSE

One of the biggest challenges in information technology project management after the project scope definition is to manage the duration and costs of the project effectively. Traditionally, several information technology projects have not been able to remain within the budget and schedules, resulting in many cases of cost overruns and schedule delays (Kerzner, 2022). These problems have thus enhanced the need for a project management practice that can support dealing with uncertainties and complexities that are inherent in software development processes. Indeed, agile methodologies like Scrum have also proven effective against the conundrums defined above—affording the entire work to be segmented into smaller work units called sprints. It also allows for iterations in order to plan, develop, and deliver incrementally components using scheduled time and under budget constraints (Serrador & Pinto, 2015). This enables early identification of problems, broad support for continuous feedback, and realignments of the overall plan so that the chances of the project being delivered on schedule and within budget are greatly improved. Adopting agile methods has improved organizational productivity and cost savings as it encourages agility, flexibility in change, and focus on value delivery per iteration. These combined benefits accelerate higher project success rates and improved organizational performance overall in software development companies (Dingsøyr et al., 2012).

3.4. PROJECTS OF QUALITY MANAGEMENT

Delivery of a project or product with all its prime quality features calls for a fine balancing act between quality, schedule, budget, and scope management. "Quality is a very prominent factor in determining success for software project management" (Kerzner, 2022). The agile method and especially the scrum framework emphasize quality as an outcome by incorporating best practices in continuous monitoring, testing, and feedback during the software development life cycle (Dingsøyr et al., 2012). Scrum is well established for having a very rigid structure to the iterative process of development, enabling teams to effectively shorten their development timeframe while incubating usability and simplicity. Rather than encouraging strict adherence only to formal acceptance criteria, Scrum engenders the kinds of software products that are intuitive and simple from the user's perspective. All agile methods are inherently effective in addressing quality concerns from the perspective of reviewing and adapting through their principles that actually result in prompt identification and resolution of emerging issues as they arise (Schwaber & Sutherland, 2020). The role of a scrum master is to enforce quality standards concerning the development team. Scrum masters create an excellence culture that emphasizes regular inspections and adherence to defined processes, which guarantee user-friendly products delivered on schedule and in alignment with project objectives and scope (Rasool et al., 2023). This systematic attention to quality supports the successful delivery of software solutions that meet both client requirements and industry standards.

3.5. RISK MANAGEMENT OF PROJECTS

Within most organizations, employees frequently contribute to projects as assistants or support staff, with their effectiveness largely influenced by their expertise and skill levels. Agile methodologies, including the Scrum framework, prioritize effective team management and foster continuous professional growth among team members (Dingsøyr et al., 2012). However, simply having a large team does not guarantee project success or high levels of client satisfaction. There are differing perspectives regarding the importance of collaboration: some organizations may underestimate its value, while the

prevailing view in the literature highlights the critical role of scrum in assembling goal-oriented individuals who thrive in collaborative environments (Hoda et al., 2017). Scrum methodology explicitly emphasizes teamwork, communication, and shared responsibility, promoting an environment where members are encouraged to work together to achieve project objectives. Human resource management plays a significant role in project outcomes when Scrum is employed as the management methodology, as responsibilities are distributed equitably and the framework encourages team members to support one another in accomplishing their respective tasks (Schwaber & Sutherland, 2020). This collaborative approach not only enhances productivity and innovation but also helps to ensure that organizational goals are met efficiently.

4. RESULT & DISCUSSION

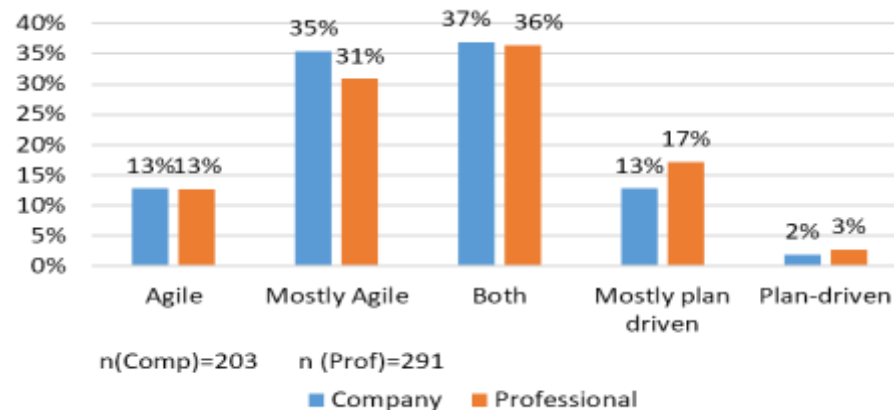


Figure 1: Firm and Individual using Agile

A wide range of organizations and professionals employ agile methodologies to varying extents, with usage patterns spanning from fully agile practices to predominantly plan-driven, traditional approaches. Survey respondents were asked to identify their position along a continuum that included purely agile, predominantly agile, a balanced mix of both, predominantly plan-driven, and purely plan-driven methods. The aggregate data indicated that approximately 85 percent of organizations and 80 percent of individual practitioners reported implementing agile development in some capacity. While only a few organizations or individual professionals claim complete reliance on agile practices, mainly because they possess potential perks and obstacles, including relentlessness, ambiguity, and control (VersionOne, 2020). An unbalanced view is articulated here: On one side, representatives from various organizations who took the survey and have expressed a positive bias on the part of most company stakeholders seem to show the highest faith ever inspired by agile adoption; however, many of these individual professional responses were marred by an equal measure of satisfaction and dissatisfaction. This discrepancy might arise from an apparent inclination of the organizational representatives to put forward a better image of practices in their own company, a lack of communication between management views and professional experiences in software development. All these findings are suggestive of the importance of also considering the cultural dimensions of the organization and the experiences of such leaders if they are to assess the effective reception this offers.

Jira Software was the software that widely adopted management for projects that would facilitate agile kind of methodologies that might include Scrum and Kanban or customized hybrids, while it is capable of planning, monitoring, and controlling the management of different aspects of agile software development project using a unified interface, which supports Agile Boards, Backlogs, Roadmaps, Reporting, etc., aside from working integration with the other linked tools and augmentations (Atlassian, 2024). Jira is widely recognized not to be exclusively a tool for software development but also the tracking bugs, troubleshooting, and general project management. Different organizations have gone further to use it in an unusual manner, like optimizing warehouse functions, document workflow management, and streamlining expenditure processes (Gonçalves & Abrantes, 2021). Embedded in Jira's offerings are robust customizing options and a flexible architecture, which underscores Jira's usefulness as particular business needs could cut across industries.

Jira Software, an avant-garde project management platform explicitly designed to operate agile methodologies such as Scrum and Kanban and various modified mixes depending on organizational requirements, allows every team to plan, monitor, and manage all components of agile software development with a unified environment. Among the highlights are agile boards, as well as interactive backlogs, strategic road maps, detailed applications, a variety of integration options with other software, and many more gadgets. Herein, the sum functionality of Jira Software has systematized how organizations can efficiently coordinate and oversee agile projects from the start through to handover and into support (Atlassian, 2024; Gonçalves & Abrantes, 2021).

5. CONCLUSIONS

Currently, global software development is emerging as the basic model for software engineering, as many organizations globally adopt the model to achieve reduced costs and access talented personnel (Santos et al., 2020). The waterfall model is, on the whole, one of the most traditional systems development life cycle methodologies, which is still in vogue in certain areas of software engineering, especially in new product development, where processes flow linearly and sequentially. In contrast, agile processes focus more on the people involved and their collaboration than on adhering to processes and tools. A key value of the agile methodology is that functional software is upheld above comprehensive documentation. The scrum master's role is to encourage open communication and collaboration of the teams and cooperation in an environment characterized by the iterative possibility of enhancement. Research has proved that Scrum positively impacts the essential domains of knowledge with regard to software project management. These include domains related to the project's timeline, budget management, scope definition, product quality, risk mitigation, and cost control. Moreover, Scrum methodology has a strong influence on the human resource process since many organizations tend to view recruiting and developing good teams as the practical way to ensure that projects will succeed. In this regard, Scrum assists organizations in minimizing project risk, cost, and time on the development of quality software products. This way of incorporating agile thinking, especially through the scrum principles, empowers project managers in the software industry in managing uncertainty and complexity, leading to adaptive and resilient project results.

REFERENCES

- Al Zaidi, M. A., Ahmed, M., & Saeed, S. (2014). An analysis of extreme programming practices in agile software development. *International Journal of Advanced Computer Science and Applications*, 5(9), 176–180.
- Atlassian. (2024). *Jira Software documentation*.
- Beck, K., & Andres, C. (2005). *Extreme programming explained: Embrace change* (2nd ed.). Addison-Wesley.
- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). Manifesto for agile software development.
- Carmel, E., & Agarwal, R. (2001). Tactical approaches for alleviating distance in global software development. *IEEE Software*, 18(2), 22–29.
- Creswell, J. W., & Creswell, J. D. (2018). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed.). Sage Publications.
- Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85(6), 1213–1221.
- Easterby-Smith, M., Thorpe, R., Jackson, P., & Jaspersen, L. J. (2021). *Management and business research* (7th ed.). Sage Publications.
- Gonçalves, F., & Abrantes, J. (2021). The adoption of agile tools in project management: A case study using Jira Software. *Procedia Computer Science*, 181, 1072–1078.
- Herbsleb, J. D., & Mockus, A. (2003). An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering*, 29(6), 481–494.
- Hoda, R., Noble, J., & Marshall, S. (2017). Agile project management: A systematic literature review of practices and challenges. *Journal of Systems and Software*, 132, 138–157.
- Hossain, E., & Babar, M. A. (2009). Using scrum in global software development: A systematic literature review. In *Proceedings of the 2009 Fourth IEEE International Conference on Global Software Engineering* (pp. 175–184). IEEE.
- Kerzner, H. (2022). *Project management: A systems approach to planning, scheduling, and controlling* (13th ed.). Wiley.
- Koumarelis, A. (2023). The evolving role of the scrum master in agile project management: A case study approach. *Journal of Software Engineering and Applications*, 16(4), 142–151.
- Kuznetsov, K., Ivanova, A., & Smirnova, Y. (2023). Agile project management practices in global software development: Challenges and solutions. *Journal of Software Engineering and Applications*, 16(3), 211–225.
- Niazi, M., Mahmood, S., Alshayeb, M., Riaz, M. R., Faisal, K., Cerpa, N., & Richardson, I. (2016). Challenges of project management in global software development: A client-vendor analysis. *Information and Software Technology*, 80, 1–19.
- Paasivaara, M., & Lassenius, C. (2016). Challenges and solutions for distributed agile software development: A systematic literature review. *International Journal of Agile Systems and Management*, 9(2), 109–137.
- Rasool, N., Shams, R., & Farooq, U. (2023). The role of the scrum master in successful agile project management: An empirical analysis. *International Journal of Information Technology Project Management*, 14(2), 52–68.
- Royce, W. W. (1970). Managing the development of large software systems. In *Proceedings of IEEE WESCON* (pp. 1–9).
- Saunders, M., Lewis, P., & Thornhill, A. (2019). *Research methods for business students* (8th ed.). Pearson.
- Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The definitive guide to Scrum: The rules of the game*. Scrum.org.
- Serrador, P., & Pinto, J. K. (2015). Does agile work? A quantitative analysis of agile project success. *International Journal of Project Management*, 33(5), 1040–1051.
- Šmite, D., Moe, N. B., Šablis, A., & Wohlin, C. (2014). Software teams and their knowledge networks in large-scale software development. *Information and Software Technology*, 56(12), 1520–1535.

- Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.
- Sriram, R., & Mathew, S. K. (2012). Global software development using agile methodologies: A review of literature. *Journal of Information Technology Management*, 23(4), 43–54.
- Tripp, J. F., Saltz, J., & Turk, D. (2016). Agile process adoption: Patterns of success. *Proceedings of the 49th Hawaii International Conference on System Sciences*, 5427–5436.
- VersionOne. (2020). *14th Annual State of Agile Report*. CollabNet VersionOne.
- Zaki, N. H. M., Rosli, M. S., & Rahman, N. A. (2023). Scrum methodology and project success: The mediating role of team collaboration. *Journal of Information and Communication Technology*, 22(3), 553–572.