Received: 18 June 2024 | Revised: 12 August 2024 | Accepted: 5 September 2024 | Published online: 12 September 2024

REVIEW

Journal of Data Science and Intelligent Systems
2025, Vol. 3(2) 79–86
DOI: 10.47852/bonviewJDSIS42023645

BON VIEW PUBLISHING

Excellence Practices in Scrum Paradigm in Software Development

Denis Pashchenko[1,*]

[1]*SlavaSoft, Spain*

Abstract: This article presents a comprehensive approach to improving Scrum methodology in practical software development. This article is intended for software engineering scientists and expert practitioners who develop their software products in the Scrum paradigm. The proposed best practices align with contemporary trends in the IT domain, including the complete digitalization and virtualization of production processes, the shift to fully remote software development, the integration of artificial intelligence technologies, and cost-saving models in team organization. The changes in software production processes discussed in this article are based on research conducted between 2020 and 2023. This research encompasses the experiences of leading Russian, European, and international IT companies and highlights the significant shift in IT business organization towards new standards of Scrum team efficiency. These standards include the data-driven formalization of production models, automation through AI tools, and visible cost optimization in engineer's teams, aligning closely with the demands of modern IT businesses. The article outlines a set of management actions designed to adapt to these evolving trends in the IT domain. These actions range from the formalization of sprint goals and processes in Scrum to sophisticated management of technical debt and the cost-effective organization of developer teams. As a main contribution to the article, there are presented most valuable Scrum paradigm excellence elements: implementing measurable and specific sprint goals in SMART format, cost-saving practices, management of sprint parameters, and usage of AI tools in software engineering. Presented in article recommendations might help in practical implementation of those process changes, based on industry practice and reduce any corresponding risks on software development projects.

Keywords: Scrum, software development, technical debt, project management

1. Introduction

The software development (SD) industry is constantly evolving, and IT businesses must continuously innovate and improve their processes to stay competitive. Agile methodologies, such as Scrum, have become a popular approach to SD due to their iterative and flexible nature. This methodology emphasizes cross-functional teams, collaboration, and continuous improvement to deliver software products with high-impacted changes in scope. The studies demonstrated that from 2017, Scrum is the dominating iterative methodology in Europe and among leading IT global corporations like Amazon, Google, Yandex, etc. [1].

To remain competitive, IT businesses must prioritize continuous improvement of their software production processes, including Scrum methodology. This means regularly evaluating and improving SD processes and tools, investing in training and support for developer's teams, and adapting production processes to meet the changing needs of their customers and IT industry.

The constant improvement and excellence of software production methodology can lead to significant economic benefits for businesses. For example, a study by McKinsey & Company found that implementing Agile methodologies, including Scrum, can reduce time-to-market by up to 40% and improve team's productivity by up to 25% [2]. Additionally, businesses that use Scrum methodology report a 50% reduction in defects, so it's leading to improved software quality.

Moreover, this methodology process framework is so "light and flexible" that improvement of SD processes takes less time and efforts in comparing with RUP/MSF models. It opens the specific ways to framework customizing for reaching the unique competitive advantages. First example of how businesses can adapt this methodology by customizing the framework to meet the specific needs of their industry. In the healthcare industry, Scrum methodology has been adapted to address the unique challenges of developing software for electronic health records. This includes developing custom frameworks that incorporate healthcare-specific regulatory requirements, such as HIPAA compliance [3]. Another example is customizing the Scrum in Russian Sberbank, who create their "Sbergile" (acronym – Sberbank + Agile) software project's delivery based on this methodology. Furthermore, investing in continuous improvement of Scrum methodology can lead to increased employee engagement and job satisfaction. The cohesive Scrum teams is the best example of collaboration in SD. A report by Deloitte Insights [4] found that companies that focus on employee engagement and development see a doubled increase in employee retention, that's very important in SD branch despite of the global lay-offs in 2023–2024 in tech corporations.

Scrum paradigm is a matured and well-documented paradigm in SD [5]. Searching the economic rational ways, tools, approaches to the constant improvement and excellence of Scrum methodology is an actual scientific task. By adopting and continuously improving the

*Corresponding author: Denis Pashchenko, SlavaSoft, Spain. Email: denpas@rambler.ru

methodology, IT businesses can reduce costs, improve productivity and project visibility, and deliver high-quality software products, ultimately leading to greater economic success. Also, this article is focused in several principal research questions: 1) how modern SD companies might improve the usage of Scrum in their production processes? 2) What's the cumulative effect of modern trends in IT domain (like usage of artificial intelligence or remote model of working) on changes in software production methodology Scrum?

Principal novelty of the article is in the clarification of assessments and recommended methods for managing various aspects of the production processes of an IT company in the Scrum paradigm. Such kind of clarifications are based on strong impact of several actual trends as fully remote model (FRM) of work of teams, AI tools usage, IT branch digitalization, and cost-saving models in IT business. Elements of novelty are including the following:

1) Formalization and Measurability: The article's approach to formalizing Scrum parameters using data-driven methods and SMART criteria is a novel contribution. This structured approach to defining and evaluating sprint goals offers a new perspective on improving Scrum efficiency and project success.
2) Role Optimization in Teams: The suggestions for optimizing team roles, including the combination of the Scrum Master role with other leadership roles and internal outsourcing of DevOps services, present innovative strategies for cost-saving and efficiency in Scrum teams.
3) Technical Debt Management: The article's structured approach to managing technical debt, through continuous activity in sprints and the creation of "technology epics", provides a new methodology for maintaining long-term software quality within Scrum projects.
4) Remote Work Adaptation: The insights on adapting Scrum processes to remote and hybrid work models, focusing on formalized communication and documentation, offer fresh perspectives on maintaining team effectiveness in a changing work environment.
5) AI Integration in Scrum: The exploration of AI tools' impact on SD and Scrum processes, including the potential for enhanced efficiency and data-driven decision-making.

Following structure of the article:

1) Introduction;
2) Theoretical framework, literature overview, and the goal of the study;
3) Research – Excellence of Scrum Paradigm in SD;
4) Conclusion and recommendations.

2. Theoretical Framework, Literature Overview, and the Goal of the Study

Listed above references are demonstrating the strong interest from software teams to agile methodologies and advantages of the constant investments in the excellence of SD processes and tools. The scientific task of the study is to create the focus on the most experienced in practice, wide-used, and perspective approaches in Scrum methodology improvement. The goal of the study is to present the solid practices of Scrum excellence in software companies, corresponding with the modern trends in digital world economy.

In the description of the research methodology, the following set of logical methods should be highlighted:

1) identifying the main and secondary within the framework of component analysis – in considering industry traditions in SD;
2) determining cause-and-effect relationships, Parretto's principle, and Ocam's razor – to determine the boundaries of the influence of industry trends on production processes in the IT industry and determining the limiting possibilities of modifying production processes in 2024;
3) synthesis of new ideas and recommendations – to create a practical set of approaches to improving the Scrum paradigm and the practical implementation of changes in an IT company.

Factors with a significant impact on SD according to recent economic trends include:

1) COVID-19 Pandemic and Transition to Remote/Hybrid Work Models: The shift to FRM or hybrid work models, where employees split their time between home and office, has become prevalent, especially in leading tech regions. Despite some companies' desire to bring engineers back to the office, remote work has become the industry standard and continues to evolve. This transition has significantly influenced communication formalization within and outside companies, necessitating the use of advanced digital communication tools.
2) Digitalization of Productive Processes in SD: Digital transformation has been a crucial factor, leading to the integration of digital data flows that enhance autonomous decision-making and corrective actions in project management. This transformation includes the adoption of Agile methodologies, continuous integration and delivery (CI/CD) automation, and advanced quality assurance (QA) techniques such as automated testing and system monitoring.
3) Cost-Saving Models in IT Businesses Since 2022: The economic landscape has driven IT companies to adopt cost-saving models, optimizing resources while maintaining productivity. This shift emphasizes efficient project management and the strategic allocation of financial resources.
4) Surge in Demand for AI-Driven SD Since 2023: The rapid increase in demand for AI-integrated software solutions has led to significant advancements in AI-augmented software engineering. This trend is expected to continue reshaping the industry, enhancing both the efficiency and quality of SD processes.

As part of a literature review should be considered, that Scrum is one of the most documented agile paradigm in SD. Unlike Agile manifesto[1] Scrum paradigm is well-learned [1], all practices are defined and documented by researchers and software engineers [5]. The critical issue here is not in establishing new "best" practices, but in practical implementation of Scrum culture, processes, and artifacts into real implementation in teams who declare Agile approach, but don't execute well the significant part of framework. Real implementation of Scrum paradigm (instead of its declaring) takes resources and needs to be economical reasoned [2]. In current article, there is a focus on the solid practices of Scrum excellence, based on the impact of mentioned-above economic industry trends.

The transition to FRM/hybrid work models has been widely adopted, with more than 70% of teams in research [6] investing in additional motivation and support for remote work. This shift has required teams to formalize communication processes, ensuring effective collaboration despite physical distances. Studies [6–8] indicate that many teams have successfully adapted to remote work, maintaining close cooperation through virtual spaces and modern project management tools.

[1]Manifesto for Agile Software Development. URL: https://agilemanifesto.org

Digitalization has further revolutionized SD environments (SDEs) [9], introducing changes such as:

1) Real-Time Feedback and Automation: Every action, from coding to using SD language semantics, is tracked, allowing for rapid responses and adjustments.
2) Detailed Logging and Analysis: Processes like release building and testing are accompanied by detailed logs, providing clear insights into every step and outcome.
3) Comprehensive Debugging: Modern debugging tools offer real-time data on variables and hardware behavior, enhancing problem-solving capabilities.
4) Continuous Integration and Delivery: CI/CD processes provide constant data flow about all stages of software testing, integration, and release, supporting informed decision-making.

This continuous flow of digital data supports decisions at various levels, from personal efficiency assessments to group decisions on code re-factoring and hardware utilization. Modern QA methods, including automated testing and verification, have become more data-driven, with auto-tests running after every master-branch build, offering digital parameters for process optimization.

For instance, current digitalization trends have significantly influenced graphical user interfaces, leading to more user-friendly designs that enhance user experience [10]. Business applications now feature complex reporting systems and data views, tailored to different user roles and visualization types.

The SD industry must adopt strategies that leverage these trends to maintain and enhance global competitiveness. This includes embracing AI tools, expanding digital and automated processes, and adapting to remote work models. By doing so, IT companies can remain at the forefront of a highly competitive and ever-evolving market.

Software engineering teams should embrace the continuous digitalization of the IT industry to add owns value on the market. This can be achieved by making operational and tactical decisions in Scrum team management more data-driven and by ensuring that the purpose, operational parameters, and technical debt of every sprint in a project are clear to all stakeholders. The project should be managed transparently, based on the flow of incoming data. As digital data usage in software engineering increases, there is a growing need for cross-functional roles in teams, which can lead to excellence in the Scrum process of software delivery. This is in line with the rising demand for software release automation and project operations.

Another valuable and visible trend is the strong market demand for cost-savings in SD. The lay-off in global IT corporations in 2023–2024 and the rising competition on the labor market might be dangerous for future stable development of IT branch, and in the mid-term, it should be taken into account. Cost-saving is affordable in many ways: from economical (like salary cut-offs and cycles of lay-offs and hiring) to technological and organizational. This demand can also be used to search for approaches to improve software production processes, including Scrum methodology excellence. Transparent management of engineer's teams can have a long-term impact on cost-savings in SD teams. In the following section, solid practices of Scrum paradigm excellence and transformation are presented as perspective ways to improve SD in the IT industry.

Another significant trend shaping the IT industry is the rapid growth in demand for SD using AI. This trend is closely connected with the significant increase in the availability of AI tools since the end of 2022. According to Barenkamp et al. [11] and Lv et al. [12], the rising demand for AI-augmented SD was anticipated, and key factors such as the COVID-19 pandemic and self-isolation accelerated this development. By mid-2023, the European SD sector had started formalizing the use of AI tools in software engineering [13]. A significant share (20%) of teams and organizations has already begun implementing AI tools in real software production, with around 43% planning to do so in the near future. Approximately 23% of experts use AI tools regularly in their work, which has had a strong impact on their tasks in SD projects since 2023. Additionally, 20% of experts rated the impact of AI tools on their professional work as average but valuable for specific tasks.

Pashchenko [13] anticipated a further acceleration in the implementation of AI tools across all areas of software engineering by 2027:

1) AI tools are expected to appear in all advanced IDEs (code editors, release building, documentation, etc.,) – 94% of experts;
2) AI tools are expected to appear in software project management systems (like Trello, MS Project, Jira, etc.,) – 73% of experts;
3) AI tools are expected to appear in product management tools (UX/UI, feature analysis, user tests, etc.,) – 60% of experts;
4) AI tools are expected to appear in DevOps tools (from CI/CD to user support software) – 57% of experts.

In 2024, we are likely to see AI tools becoming integral features of many software engineering technologies (such as Apple's Xcode IDE or JetBrains IDEs). The practical implementation of AI tools in software projects is changing common production development patterns [14] and should be considered in Scrum paradigm excellence.

3. Research: Excellence of Scrum Paradigm in SD

The complex software production paradigm aims to achieve project success and stakeholder satisfaction, including both customers and engineering teams. In Scrum methodology, the success of each sprint determines the overall success of the software project. However, evaluating the success of each Scrum sprint is a subjective parameter that requires formalization. This can be achieved by implementing standard actions such as defining criteria for achieving sprint goals, assessing the short-term impact of SD on customer and user satisfaction, and formalizing and considering sprint parameters. Moreover, it became more important in FRM of working in software teams, where common management needs more formal criteria of software project success.

Defining own goals for each sprint is significant and not enough: it is necessary to combine sufficiently accurately measured and formulated substances: business goals of the products being developed, prioritize validity and non-functional requirements in sequential level backlogs, logical sequence of stages in the development of software projects. A good practice for defining sprint goals is to use the SMART format, where "M" stands for measurability. Measurable goals are often expressed in numbers, such as "Achieve the maximum completion time for each transaction in the system for users in any role is less than 1 second". This statement is specific, relevant, and measurable, with a clear deadline for achievement, which is the duration of the sprint. Typically, a sprint has 1–2 main goals, unless it involves the development of complex ecosystems in Scrum of Scrum and Scaled Agile Framework formats.

Evaluating the short-term impact of SD process and corresponding product on business customer and user satisfaction can be quite straightforward. Methods such as Product Manager reviews, user focus groups, and beta tests offer immediate insights that, even if not perfectly precise, are highly valuable in gauging user sentiment and business satisfaction. The key principle is to continuously collect feedback and analyze it in relation to production issues identified during sprints and discussed in retrospectives. This ensures that feedback is not only heard but actively informs the development process.

Consider a scenario where a development team is working on an e-commerce platform. They receive feedback from a focus group indicating that the checkout process is too cumbersome. This feedback is discussed during a retrospective, and the team prioritizes improving the checkout process for the next sprint. After implementing a streamlined checkout, they release an update and monitor the impact through user reviews and customer satisfaction surveys. An increase in positive feedback and higher ratings on the app store would indicate a successful enhancement, demonstrating the value of integrating user feedback into development. Another example can be seen in a SaaS company developing a project management tool. Business customers might report through surveys that they need better integration with other tools they use. The product manager brings this up in a new sprint planning meeting, and the team decides to estimate new integrations in the next sprints. Post-release, they track customer satisfaction through the CRM system, noting positive feedback, which underscores the importance of responding to business customer needs. Additionally, a mobile game developer might notice through app store reviews that players are frustrated with the game's difficulty level. The team discusses this feedback in their sprint retrospective and decides to adjust the difficulty settings in the next update. They then monitor the impact through subsequent reviews and in-game analytics, seeing improved user retention and higher ratings, indicating that addressing this feedback has had a positive effect.

In all these examples, the feedback loop is crucial: collecting feedback, analyzing it within the context of ongoing development, and making informed changes. This continuous cycle helps ensure that the software not only meets but exceeds user and business customer expectations, leading to higher satisfaction and loyalty. The digitalization of processes in SD is including the mechanism of converting of quality estimations (like user reviews or business customer satisfaction) into numbers – from estimated ROI to visible increasing of customer satisfaction in official parameters.

Formalizing the parameters of Scrum development sprints is crucial to improve the success of software projects. The key reasons for implementing Scrum must be considered when formalizing these parameters. While comical reasons, such as "fashionable, stylish, youthful", should be disregarded, the most typical reasons in world-wide practice include the need to speed up and formalize development, adapt to rapidly changing requirements and business realities, and bring business customers. Formalizing sprint parameters in each case is specific and closely related to the parameters that determine these reasons, including task delivery speed, relevance of created functionality, cross-functionality of employees, and risk management of timely changes in products, including response to force majeure situations. This article proposes grouping formalized sprint parameters and recommends their use by whole groups based on the business needs of the software company and the reasons for implementing Scrum (Table 1). The flexibility to adapt these recommendations to each team's unique situation is in line with Agile values.

Let's consider an example from the author's practice in 2018 in software Russian-Chinese company, illustrating the use of formalized sprint parameters in SD and their impact on business parameters in the development of software products. One of the rarest, but potentially effective parameters of a sprint is the maximum deviations of the actual burned story points from the ideal "burn-down chart". Of course, we should immediately agree: full adherence to the "ideal" trajectory is a harmful utopia, but refusal to follow it is a low level of managerial maturity of the development team. It is important to take into account and analyze the causes of maximum deviations at the key moments of the sprint. Such deviations were counted for 7 sprints at two key moments: at the end of each sprint and in the middle of its calendar period. Deviations were calculated as a percentage (ranging from 2 to 43%) and allowed to make invaluable conclusions about both the speed of development and the potential of teams. In this example, a very clear pattern was

Table 1
Examples of groups of sprint's parameters in Scrum

№	Business need	Sprint parameters in group
1	Speed up the software development	1) Deviations of the actual burned of story points \ stories from the "ideal trajectory" on the "burn-down chart" at the key moments of the sprint; 2) The emerging vector of acceleration of the work of each team (with the stability of its composition) in burned points or released stories; 3) The emerging rising vector of the team's capacity at the planning stage (with the stability of its composition) in points or stories; 4) The amount of the stories or tasks that were included in the sprint, but not implemented due to weak formalization and unclear requirements;
2	Adopting to constant changes in requirements	1) The number of tasks with the highest priority implemented in the sprint; 2) The amount of the stories or tasks that were included in the sprint, but not implemented due to the loss of relevance; 3) A constant balance between the number of tasks and stories in three categories: (A) "urgent" with high priority, (B) technical and infrastructure (including QA and CI \ CD), (C) "long-term" actual core features.
3	Common goals in software development for product teams and core customers	1) Amount of product hypotheses that have passed all stages of development and became software feature; 2) Amount of stories and tasks that were removed from commercial operation due to negative user feedback during the reasonable period; 3) The growth of the team's capacity in story points or tasks; 4) A constant balance between the number of tasks and stories in three categories: (A) "urgent" with high priority, (B) technical and infrastructure (including QA and CI \ CD), (C) "long-term" actual core features.

observed – the more deviations from the ideal burn-down chart in the sprint, the less successful the sprint in terms of achieving its goals. One of the sprints had deviations of 27% and 43% – the maximum observed for the team – this sprint was the most unsuccessful and required team rework and additional resource costs. The analysis of mentioned parameters made it possible to form individual capacities of teams and team leads, adjust the task planning stage, and make effective changes in the process of stabilizing the quality of the functionality being put into operation. Thus, the potential of formalized sprint parameters is real, and the analysis and management of such parameters is a practical tool for improving the quality of SD and achieving goals. Of course, the implementation of such parameters is a process that requires professional skills, attention to detail, and adaptation to specific teams and their tasks.

Thus, formalizing the goals and parameters of the sprint and constantly working with digital indicators of customer satisfaction is the first and very important practice for improving software engineering production processes within the framework of the software production paradigm.

Next element of the excellence of the Scrum software production paradigm, based on IT branch digitalization, is a "smart" technical debt management. Rising set of digital data about the software product gives a new opportunity to evaluate the relevant needs of re-factoring or additional measures in software quality management. Criticism of the Scrum methodology relates to the project team's ability to neglect the parameters of long-term software quality [15]. However, such "savings" can rarely be justified in a large part of the software, where long-term quality is at the heart of the competitive opportunities of IT companies. Technological development of software products, and, for example, technical debt management positively affects a significant number of parameters of long-term software quality. The relevance of the problem of technical debt in Scrum and other "agile" paradigms (SAF, SofS) in the development of SD production processes is beyond doubt. The solution to this problem lies in several aspects at once. Consider several levels of it:

1) accounting for and eliminating technical debt is an activity in each sprint, specifically discussed in the planning, demo, and retrospective. Forming only "technology sprints" (in Scrum) or entire trains (in SAF) is not a best practice, but a necessary measure;
2) tasks of technological development and decommissioning of tech debts form independent epics in product backlogs and have own assessments (labor intensity, priority, risks, and connections with other tasks);
3) actual re-factoring is carried out according to the schedule, its connection with the functional development of the system is less important than the convenience of managing the team's capacity;
4) the best practice is to identify independent and long-term software quality indicators related to technological development and decommissioning of tech debt rather than trying to tie "tech tasks and stories" to functional epics and their quality/success indicators;
5) interaction between business customers and the development team in terms of technical debt should be clear in terms of goals and so long until "technological tasks and stories" find the right place in the understanding of customers in planning priorities.

The aforementioned list provides insight into the management of technical debt and technological development within Scrum projects. It is important to recognize that technological development is just as crucial as functional development in the majority of IT products. Therefore, team members must pay close attention to the time and resources dedicated to technological aspects of product development as it is an essential aspect of SD projects. Accounting for technical debt is a continuous activity that begins in the planning of each sprint [16]. A recommended approach involves the creation of "technology epics" which consist of stories that are allocated to development sprints according to the team's capacity. There are several general recommendations for forming these epics, including defining clear goals and linking them to product quality indicators, maintaining a balance between story components in each epic, and allocating independent process areas into separate epics if significant development is required in these areas, such as CI/CD or architectural redesign. These epics may contain stories that exclusively implement development in these areas.

Shift to fully remote work \ "hybrid" model made the significant changes in some traditional areas of production processes. The key area – team communications – also has changed due to decreasing or even lack of personal face-to-face interactions. Different studies showed unlike results in the estimation of success in adapting this trend. According to Ford et al. [17], the remote work necessitates a heavy reliance on digital tools like Slack, Zoom, and Jira. While these tools facilitate structured communication, they often fail to capture the nuances of in-person interactions. This can lead to misunderstandings, delays, and fragmented communication. For example, during Scrum ceremonies such as daily stand-ups, sprint planning, and retrospectives, the absence of non-verbal cues can hinder the effectiveness of communication and collaboration. Other studies [6, 18] define that fully remote work in software domain did not change the efficiency of communications, and its transferring into digital channels has a positive impact on team's productivities. On the basis of this, conclusion is total formalization of communications and production process documentation, that makes Scrum paradigm more effective. Detailed and accessible documentation ensures that all team members are aligned and have a shared understanding of project goals, tasks, and progress. This can mitigate the risk of misunderstandings and ensure continuity in communication.

The shift to full remote work has undeniably impacted the communication processes within Scrum teams in the software domain. While challenges such as lack of personal communications, potential misunderstandings, and feelings of isolation exist, these can be mitigated through strategic use of digital tools and practices [6]. Regular video meetings, clear documentation, and a culture of open communication are key strategies that can help remote Scrum teams thrive. As the landscape of FRM continues to evolve, these practices will be essential in ensuring that remote teams remain effective, cohesive, and productive.

Another promising direction in the development of the Scrum methodology, which is consistent with the market demand to reduce the cost of development teams and rising of cross-functionality of engineers because of the FRM labor factor, is the optimization of the role model in the team. The paradigm of Scrum SD is evolving at a high pace: new methods and approaches are emerging to optimize labor costs in product development, to increase productivity and engagement of engineers, and to improve other quality parameters of software products. Based on successful industry experience, the author recommends the following optimizations:

1) Scrum master is not an independent role; Scrum master is a current member of the project team: find a leader who will combine this role with his current own one;
2) team lead, tech lead, architect – it is necessary to find the right role for each team or determine a successful balance of their partial participation;

Table 2
Principal research findings

1 how modern software development companies might improve the usage of Scrum in their production processes?
Answer: There is a set of actions to improve the software production paradigm:
1) Formal and digital approach for every sprint: goals in SMART format, sprint parameters, estimated feedback from customers, "smart" technical debt management.
2) Respect to rising demand of fully remote work or "hybrid" model with redesign of the formal and digital communications in team via modern software tools;
3) Respecting the demand of cost-saving in software development: optimization of roles in the team (Scrum master, tech lead, DevOps engineer) and active usage of AI tools in all production processes.
2 What's the cumulative effect of modern trends in IT domain (like usage of artificial intelligence or remote model of working) on changes in software production methodology Scrum?
Answer: Despite of initial ideas of Scrum some production processes are changing:
1) All processes and estimations of success/failure in sprints are becoming more formal and digital;
2) All communications inside of team and with external agents (like customers) are becoming much more formal and automated;
3) Virtualization of the production processes is becoming much clear and supported by special software tools in every process area or internal outsourcing (like in the case with DevOps engineers);
According to the initial ideas of Scrum some production processes are developing very fast:
1) Total digital focus on the product and automation of feedback and review processes;
2) Cross-functionality in Scrum team is increasing;
3) Production process has new automation tools (like AI tools in software development).

3) internal outsourcing of services of DevOps engineers, instead of including an engineer in the team.

Let us begin by defining that the role of Scrum Master is not a standalone position within a team, but rather an important role that can be fulfilled by various leaders on the project, including team leaders, analysts, QA engineers, or project managers. Therefore, a Scrum master should be identified from among the current leaders of the development team in such a way that this role in the project team does not incur additional costs for the project.

In determining the balance of effort and need for team lead, technical lead, and architect roles, budgeting becomes a crucial factor. In smaller projects with up to 10 team members, it is advisable to combine these three roles into one or two people (1,5 FTE), with the team lead being the key role. For larger projects, the scale may require the involvement of solution and enterprise architects, with an estimated time commitment of no more than 0.5 FTE every six months. Technical lead is an essential role for scaling development (SAF, Scrum of Scrum), especially when using proprietary technologies and complex integrations between software projects [19], but can be sacrificed in projects experiencing budgetary constraints.

Finally, the approach to optimizing the composition of the project team in the face of budget constraints is to change the budgeting of DevOps engineers (and similar roles such as system administrators, environment engineers, SD automation engineers). The allocation of such engineers to internal units, which provide their services at the company's internal price, is a slow but effective option for optimizing the budgets of development projects. Of course, many project managers and team leaders would like to have a dedicated engineer for product environments, CI \ CD support, but from the point of view of optimizing project costs, it seems more promising to automate these regular needs through by special DevOps unit within a software company. This transformation carries certain risks associated with infrastructure problem-solving efficiency in the project and the reduction of DevOps engineers' involvement in each software project's problems. However, these risks can be managed through formal agreements such as SLA and KPI and informal measures such as creating good working relationships and assigning engineers to areas of responsibility.

One more significant trend – rising of active usage of AI tools in SD projects since 2023. This trend is aligning with global shift to FRM and fully corresponding to demand of more effective cost-saving models in IT businesses [20]. As presented above, AI tools are actively implementing in all production processes in software engineering: from analysis and coding to product documentation. It's the start of new format of SD team – where every team member might do simple cross-functional tasks via AI tools and be much more independent in software product realization. The impact of this trend on Scrum paradigm needs more time to final estimation, but a few things are clear now:

1) with AI tools, many types of ordinary tasks (e.g. software prototyping, RnD, product documentation, etc) need less time and lower level of collaboration within the teams;
2) communications in digital channels might be formalized and documented better with AI agents;
3) almost all technologies in software engineering are improving via AI elements, that generate much more data for managing decisions: from estimation of product's (and its technology's) quality to efficiency of the whole team in sprints.

As a main contribution, this article is providing the focus in Scrum methodology excellence on the following aspects:

1) Data-driven formalization of Scrum production process: from sprint goals and sets of parameters to "smart" tech debt management;
2) Constant automation of production processes (via AI tools) that leads to increased efficiency of Scrum teams;
3) Cost-saving optimization of developer's teams in a changing paradigm of labor relations (FRM \ "hybrid").

Those findings might be used in the planning of Scrum excellence in software company, have a solid industry practice of usage, and align with modern trends of digital world.

4. Conclusion and Recommendations

Competition in the software production and IT industries involves unique characteristics, including the continuous need for changes in major production processes under the pressure of competitive factors. The digital transformation in SD has a significant impact on competition within the industry, and the Scrum methodology is an effective process for managing these changes. From another point of view, the Scrum methodology excellence process should use the results of this transformation: total automation and focus on measurable processes give continuous flows of digital data that might be used on different levels of IT business development, production processes, and even software product client's expectation. Management of core activities is becoming sophisticated and more flexible, based on relevant and estimated indicators.

Principal research questions should be answered in the following manner (Table 2):

Main findings of the article in Scrum excellence:

1) Data-Driven Formalization of Scrum Processes: The article emphasizes the importance of formalizing Scrum sprint parameters through measurable and specific goals. It introduces the use of SMART criteria for sprint goals and advocates for constant evaluation of sprint success based on formalized parameters. This structured approach aims to enhance project management, improve development speed, and align product development closely with business needs.
2) Optimized Role Model in Scrum Teams: The article proposes optimizing team roles to reduce costs and improve efficiency. It suggests that the Scrum Master role can be combined with other leadership roles within the team and discusses the balance between team leads, technical leads, and architects, especially under budget constraints. Additionally, it advocates for the internal outsourcing of DevOps services to optimize project budgets.
3) Technical Debt Management: The article highlights the importance of managing technical debt within Scrum projects, emphasizing that it should be an ongoing activity integrated into each sprint. It provides practical recommendations for balancing technical and functional development instead of creating "technology epics".
4) Impact of Remote Work on Scrum: The article discusses the effects of fully remote and hybrid work models on team communication and productivity. It suggests that formalized communication and detailed documentation can mitigate the challenges of remote work, ensuring continuity and alignment in Scrum teams.
5) AI Integration in SD: The article acknowledges the growing use of AI tools in SD, which can enhance efficiency and reduce the need for extensive collaboration on routine tasks.

In this article, the presented focus on the most experienced in practice, wide-used, and perspective approaches in Scrum methodology improvement: data-driven formalization of Scrum production process and cost-saving optimization of engineer's teams. As it was mentioned in introduction, the constant investments in production process in SD are key factor in industry competition. Presented in article recommendations are the main contribution in this research area. Those recommendations might help in practical implementation of those process changes, based on industry practice and reduce any corresponding risks. Implementing measurable and specific sprint goals using the SMART format ensures clarity and focus. Additionally, continuously assessing the short-term impact of software product's development on customer and user satisfaction through methods like product manager reviews, user focus groups, and beta tests provides immediate and valuable insights.

Scrum paradigm is an excellent methodology for SD that requires formalization to achieve project success and stakeholder satisfaction. Formalizing the parameters of Scrum development sprints is crucial to improving the success of software projects. The use of formalized sprint goal and parameters, including delivery speed, relevance of created functionality, cross-functionality of employees, and risk management of timely changes, is needed. The management of technical debt is also crucial to long-term software quality, and its solution lies in several aspects, including accounting for and eliminating technical debt in each sprint and forming specific tasks for technological development and decommissioning of tech debts. Optimizing Scrum team roles is an effective change for software projects experiencing budgetary constraints. There were listed more widely used actions: from defining the Scrum master to outsource DevOps service for software projects within IT company.

Presented approaches are formed the solid base for Scrum excellence in software company that means the reaching of this study's goal. The shift towards data-driven formalization, automation via AI tools, and cost optimization in Scrum teams aligns with the demands of modern IT businesses. These practices enhance efficiency, ensure high-quality software production, and maintain alignment with rapidly changing business requirements. By integrating feedback loops and formalizing sprint parameters, developer's teams can achieve higher satisfaction and loyalty from both users and business customers, ultimately leading to software product success.

The ideas of further research should be considered the following:

1) tracking of the rising impact of AI tools in SD by Scrum teams;
2) future optimizations in Scrum teams and its role management with rising excellence of CI\CD technologies and IDE tools and strong demand of cost-saving in SD projects.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by the author.

Conflicts of Interest

The author declares that he has no conflicts of interest to this work.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Author Contribution Statement

Denis Pashchenko: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization, Supervision, Project administration.

References

[1] Pashchenko, D. (2021). Fully remote software development due to covid factor: Results of industry research (2020). *International Journal of Software Science and Computational Intelligence*, *13*(3), 64–70. https://doi.org/10.4018/IJSSCI.2021070105

[2] Aghina, W., Handscomb, C., Salo, O., & Thaker, S. (2021). The impact of agility: How to shape your organization to compete. *McKinsey & Company*, *25*.

[3] Khristich, S. (2021). *How custom healthcare software is developed for secure HIPAA compliance.* Retrieved from: https://tateeda.com/blog/how-healthcare-software-is-developed-for-secure-hipaa-compliance

[4] Deloitte Insights. (2019). *Leading the social enterprise: Reinvent with a human focus*. Retrieved from: https://www2.deloitte.com/content/dam/insights/us/articles/5136_HC-Trends-2019/DI_HC-Trends-2019.pdf

[5] Project Management Institute. (2000). *A guide to the project management body of knowledge (PMBOK guide)*. USA: Project Management Institute.

[6] Pashchenko, D. (2023). Fully remote software development as a new standard in the IT industry: European study 2022–2023. *Software Engineering*, *14*(5), 217–224. https://doi.org/10.17587/prin.14.217-224

[7] Cucolaş, A. A., & Russo, D. (2023). The impact of working from home on the success of Scrum projects: A multi-method study. *Journal of Systems and Software*, *197*, 111562. https://doi.org/10.1016/j.jss.2022.111562

[8] Emmanni, P. S. (2023). The impact of remote work on agile project management. *Journal of Scientific and Engineering Research*, *10*(2), 202–207. https://doi.org/10.5281/zenodo.11078900

[9] Laato, S., Mäntymäki, M., Birkstedt, T., Islam, A. N., & Hyrynsalmi, S. (2021). Digital transformation of software development: Implications for the future of work. In *Responsible AI and Analytics for an Ethical and Inclusive Digitized Society: 20th IFIP WG 6.11 Conference on e-Business, e-Services and e-Society*, 609–621. https://doi.org/10.1007/978-3-030-85447-8_50

[10] Rodríguez, P., Markkula, J., Oivo, M., & Turula, K. (2012). Survey on agile and lean usage in finnish software industry. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 139–148. https://doi.org/10.1145/2372251.2372275

[11] Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). Applications of AI in classical software engineering. *AI Perspectives*, *2*(1), 1. https://doi.org/10.1186/s42467-020-00005-4

[12] Lv, C., Guo, W., Yin, X., Liu, L., Huang, X., Li, S., & Zhang, L. (2024). Innovative applications of artificial intelligence during the COVID-19 pandemic. *Infectious Medicine*, *3*(1), 100095. https://doi.org/10.1016/j.imj.2024.100095

[13] Pashchenko, D. (2023). Early formalization of AI-tools usage in software engineering in Europe: Study of 2023. *International Journal of Information Technology and Computer Science*, *15*(6), 29–36. https://doi.org/10.5815/ijitcs.2023.06.03

[14] Terragni, V., Roop, P., & Blincoe, K. (2024). The future of software engineering in an AI-driven world. *arXiv Preprint*: *2406.07737*.

[15] Misra, S., Kumar, V., Kumar, U., Fantazy, K., & Akhter, M. (2012). Agile software development practices: Evolution, principles, and criticisms. *International Journal of Quality & Reliability Management*, *29*(9), 972–980. https://doi.org/10.1108/02656711211272863

[16] Murillo, M. I., López, G., Spínola, R., Guzmán, J., Rios, N., & Pacheco, A. (2023). Identification and management of technical debt: A systematic mapping study update. *Journal of Software Engineering Research and Development*, *11*(1), 1–8. https://doi.org/10.5753/jserd.2023.2671

[17] Ford, D., Storey, M. A., Zimmermann, T., Bird, C., Jaffe, S., Maddila, C., . . . , & Nagappan, N. (2021). A tale of two cities: Software developers working from home during the COVID-19 pandemic. *ACM Transactions on Software Engineering and Methodology*, *31*(2), 1–37. https://doi.org/10.1145/3487567

[18] Baakeel, O. A. (2021). Impacts of remote working on employees during the COVID-19 pandemic. *International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies*, *12*(10), 1–14. https://doi.org/10.14456/ITJEMAST.2021.196

[19] Spiegler, S. V., Heinecke, C., & Wagner, S. (2021). An empirical study on changing leadership in agile teams. *Empirical Software Engineering*, *26*, 41. https://doi.org/10.1007/s10664-021-09949-5

[20] Bhandari, K., Kumar, K., & Sangal, A. L. (2023). Artificial intelligence in software engineering: Perspectives and challenges. In *2023 Third International Conference on Secure Cyber Computing and Communication*, 133–137. https://doi.org/10.1109/ICSCCC58608.2023.10176436

How to Cite: Pashchenko, D. (2025). Excellence Practices in Scrum Paradigm in Software Development. *Journal of Data Science and Intelligent Systems*, *3*(2), 79–86. https://doi.org/10.47852/bonviewJDSIS42023645