Computing&AI Connect
ISSN: 3104-4719
2024, Article ID. 2024.0005, Cite as: https://doi.org/10.69709/CAIC.2024.103102

SCIFINITI
Research Article

Refining the Scrum Paradigm: A Comprehensive Research of Software Development Practices (2020–2023)

Denis S. Pashchenko✉,1

[1] SlavaSoft, 03500 Alc, Spain

Article History
Submitted: August 25, 2024 Accepted: October 24, 2024 Published: October 27, 2024

Abstract
This article presents a complex vision on software production practices aimed at enhancing the Scrum methodology within software project management. The recommended best practices are closely aligned with contemporary trends in the IT sector, including the total digitalization and virtualization of production processes, the transition to fully remote software development models, the incorporation of artificial intelligence technologies, and the implementation of cost-effective models in team organization. The modifications to software production processes discussed here are based on research conducted between 2020 and 2023, incorporating insights from prominent Russian, European, and international IT companies. This research underscores a significant transformation in IT business practices, reflecting a shift towards new benchmarks for Scrum team efficiency. These benchmarks emphasize the data-driven formalization of production models, automation facilitated by AI tools, and noticeable cost optimization within engineering teams, which collectively align with the evolving demands of the modern IT landscape. The article details a series of management strategies designed to adapt to these shifts, including the formalization of sprint goals and processes within Scrum, management of technical debt, and the cost-efficient organization of development teams.

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

1. Introduction

The software development landscape is in constant flux, driven by rapid technological advancements and shifting market demands. For IT businesses to stay ahead, they must embrace innovation and continually refine their processes. Among the various methodologies, Agile frameworks such as Scrum have gained significant popularity for their focus on adaptability and iterative development. Scrum, with its focus on cross-functional teamwork, collaboration, and incremental development, has emerged as a leading approach for managing complex projects. Recent research highlights that since 2017, Scrum has solidified its position as the predominant iterative methodology across Europe and among major global IT corporations such as Amazon, Google, and Yandex [1,2]. This rise is attributed to Scrum's ability to enhance responsiveness to changes and facilitate better alignment with evolving project requirements. The framework's iterative cycles, or sprints, enable teams to deliver incremental updates and improvements, which in turn fosters a culture of continuous enhancement.

To maintain a competitive edge, IT businesses must emphasize the ongoing refinement of their software development practices, including the Scrum framework. This involves systematically evaluating and enhancing development processes and tools, investing in professional development for teams, and adapting practices to address the needs of clients and software users. The integration of new technologies can significantly improve Scrum's efficacy. Moreover, fostering a culture that supports regular feedback and adaptive change ensures that the Scrum method-

* **Corresponding Author:**
Denis S. Pashchenko, SlavaSoft, 03500 Alc, Spain, denpas@rambler.ru

ology remains effective in navigating the complexities of modern software development.

By continuously refining their Scrum practices, IT organizations can streamline development efforts while enhancing their ability to deliver high-quality software products that meet or exceed customer expectations. This proactive approach to process improvement and innovation is crucial for sustaining success in an ever-changing 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% [3]. 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 software development 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. In the healthcare industry, Scrum methodology has been adapted to address the unique challenges of developing software for electronic health records (EHRs). This includes developing custom frameworks that incorporate healthcare-specific regulatory requirements, such as HIPAA compliance [4]. 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 [5] found that companies that focus on employee engagement and development see a doubled increase in employee retention, which is highly important in SD branch, despite of the global lay-offs in 2023–2024 in tech corporations.

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 methodology, IT businesses can reduce costs, improve productivity and project visibility, and deliver high-quality software products, ultimately leading to greater economic success.

So, Scrum's adaptability and focus on iterative progress have made it the dominant software development framework, enabling IT companies to improve productivity, and enhance flexibility software production. Continuous refinement of Scrum practices drives economic success by optimizing processes and fostering employee engagement, ensuring long-term competitiveness in the evolving tech industry.

2. Materials and Methods

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 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 production, corresponding with the modern trends in digital world economy.

The following methods should be emphasized as approaches for addressing the scientific task and achieving the stated study objectives:

Systems Analysis and Deduction: Provides a structured framework for analyzing complex processes.

Pareto Principle and Occam's Razor: These principles help identify the most widely used and practically effective elements for enhancing the production paradigm.

Generalization and Classification: Facilitates the identification of the most promising production improvement elements by balancing costs, risks, and benefits. Trends with the significant impact on SD are listed below:

(1) COVID pandemic and world-wide visible shift to fully remote model of working (FRM)\"hybrid" model with few days\weeks in offices;
(2) Digitalization in productive processes in SD;
(3) Cost-saving model in IT businesses since 2022;
(4) Avalanche-like growth in demand for software development using AI since 2023.

The basis for improving Scrum practices in software development in an IT company is the main results of scientific research conducted in 2020–2023 in Russia and Europe on the following topics:

- formalization and consolidation of the "hybrid" work schedule model and completely remote software development in the IT industry [2,6–8];
- digitalization of production processes in IT companies [9,10];
- the use of AI tools in the production processes of high-tech companies [11–13].

Author's scientific research [2,6,8,9,11,12] in 2020–2023 covered about a hundred IT companies (mainly in Russia and Europe, less share – USA and international). The main scientific approach and methodology should be described in this section in common highlights:

(1) Every research was conducted during two months with expert panels (engineers and managers from the corresponding industry);
(2) In every panel the corresponding company and their teams were presented by 1–2 experts, every panel consists of 18–35 experts or engineer's teams;
(3) Every research had two rounds – first round via Google.Forms questionnaire and second round via live meetings, phone calls or video conferences;
(4) Every Google.Forms questionnaire had few sections from current status of research's topic to expert's forecast about corresponding trends in short- and midterm periods;
(5) In every research after the first round the summarized and structured preliminary results were sent to experts and in second round were discussed in live meetings, phone call or video conferences with experts (in every research from 30% to 60% of experts took part in second round);
(6) Final results after round 2 never had a significant difference with structured results of round 1 in all describing 6 studies in 2020–2023.

Research showed a steady increase in the share of teams in the industry for which completely remote software development has become the production standard: from 31% in 2020 [2] to 58% by 2023 [8]. The share of companies using a "hybrid format" of work is also growing, and in Europe the first organizations have appeared that are returning their employees to offices from 2023 (about 13% of teams in the study [8]). At the same time, for 63% of teams the level of productivity did not change when switching to a completely remote (and even "hybrid") model, and in every fifth team it increased significantly [8].

Another study [7], covering the experience of several dozen IT companies from the Scandinavian countries of Europe, also confirmed that the "forced" shift to fully remote work in the IT industry due to the 2020 lockdowns led to the development of a new production culture along with a comprehensive set of production approaches and methods. These changes (formalization and automation of tasks, virtualization of processes, complication of communications, etc.) became the basis of new production process models that became entrenched in companies even after the risks of the pandemic subsided. Production tools and the organization of engineering teams turned out to be easily adaptable to new working conditions, and the technological level of supporting corporate processes turned out to be quite high, although it required additional investments. Moreover, the study [7] also confirms: since the early adaptation to lockdowns in 2020, the subjectively perceived productivity of engineers working "out of the office" has increased by mid-2022.

According to [8], the complex impact of fully remote work on software development parameters and the delivery process is not as pronounced as it was in 2020 or even in 2021, as the main challenges have been addressed. More than 70% teams in the study are spending time, money and efforts for additional motivation of teams in remote software development. In half of the cases, all investments in this model of working are done and work well. The same study also defined the interesting aspect of pandemic impact on Scrum methodology. The very spirit of Agile and project practice states that the team should work together and as close as possible to each other [14]. More than 70% of teams in study [8] were able to overcome the traditional problem of having engineers work together in the Scrum teams, combined with completely remote software development "without physical offices". Engineers continue to work closely together, but in a virtual space used modern communication and project management tools.

The IT domain is also undergoing a continuous process of digitalization [9].Also, one of the most powerful results of this process is a correct and constant flow of digital data that helps in autonomous decisions and corrective actions in company or project management. Agile methodologies are under digital transformation, and signs of this process could be overviewed in various directions, including software development environments (SDE) and relevant tools (like continues integration and delivery automation - CI\CD), modern methods of quality assurances, including automatic testing, and continuous integration and delivery, including system monitoring processes.

For SDE we could see changes in all interactions with software developers and quality assurance (QA) engineers:

1. every action from spelling a letter in code to usage of elements of software development language semantics is counting and makes SDE to demonstrate a rapid reaction;
2. every process like release building, run application and it's tests or activating of emulators is accompanied by detailed logging, this logging is giving a clear picture about all steps in process and final results;
3. all mechanisms of debug give on-line flow of a data about every aspect of building app: from value of variables to the behavior of the hardware;
4. Modern QA and CI\CD provide the constant flow of a data about all stages in the software automated testing, integrating and releasing.

Such data could lead to serve as a foundation for decision-making regarding the design and construction of software, along with project management activities conducted online across various levels. This inculcates analyzing individual efficiency and facilitating group decisions about code refactoring and the utilization of additional types of hardware and emulators. Modern practices of refactoring became "digital" as well: engineers from abstract estimation of code re-factoring of function or process went over to estimated tasks with forecast of benefits in numbers: level of code legacy, software performance parameters, speed of release building or testing, etc. Modern methods of quality assurances (include unit-tests, automatic tests and automatic methods of software verification) also received more and more estimated digital parameters. The modern release building process includes launching of auto-tests after every master-branch building that provides the engineers with:

- results of the software auto-testing;
- digital parameters of process: execution time of every stage, availability of elements of process, accessibility of integrated modules and external information systems and emulators (in case of integration testing).

According to [9] the average test's coverage in modern software production is about 40–50% for products. Typically, this entails extensive hours of automated testing during each release cycle. Current flow of digital data in quality assurance process could reduce such kind of loss of time and makes standard process more specific and selective. Meanwhile, processes of software quality assurance are a part of release management and they are integrated with production processes of continuous integration and delivery. Modern CI/CD automated by smart tools like GitHub and Docker and the tools are providing engineers with a data about every step of processes: from parameters of release building till packing into containers and transfer to user's areas. It means that every release building, integration and delivery update information about optimal digital parameters of release management and could be a base for decision making in this area. For example, simplification of GUI (as well as user-friendly interfaces) is the well-known trend and it has a huge impact from the software industry digitalization as well. There are strong changes in manner of presentation of valuable data for users in GUI and it has an influence even of the stages of design and construction of software including user interfaces. Modern software not only produces more data, it divides the flows of data to different roles in information systems and different types of visualization: charts, graphics, logs, report (including special types of software like Business Intelligence). Moreover, common business applications often feature complex reporting and data view configuration systems that enable users to effectively utilize all available data. For example, financial systems use the colors of labels and icons, for communication systems use emoji and gifs, the video-games use the vibrating of remote controller, sounds and visible marks of user actions in the game's past. Obvious things like statuses or events in software are gaining emotional nuances that also have its own digital track with own influence on user experience in software.

Software engineering teams should embrace the continuous digitalization of the IT industry enhance their value in 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 software development. 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 savings can be achieved through various means, ranging from economic measures such as salary reductions and cycles of layoffs and hiring to technological and organizational strategies. 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 software development teams. In the following section, solid practices of Scrum paradigm excellence and transformation are presented as perspective ways to improve software development in the IT industry.

Another significant trend shaping the IT industry is the rapid growth in demand for software development using AI. This trend is closely connected with the significant increase in the availability of AI tools since the end of 2022. According to [11], the rising demand for AI-augmented software development was anticipated, and key factors such as the COVID-19 pandemic and self-isolation accelerated this development. By mid-2023, the European software development sector had started formalizing the use of AI tools in software engineering [12]. 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 software development projects since 2023. Additionally, 20% of experts rated the impact of AI tools on their professional work as average but valuable for specific tasks.

Experts from [12] anticipate a further acceleration in the implementation of AI tools across all areas of software engineering by 2027:

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

In 2024, it is expected that AI tools will become integral components of various software engineering technologies, including platforms such as Apple's Xcode IDE and JetBrains IDEs. The practical implementation of AI tools in software projects is changing common production development patterns [13] and should be considered in Scrum paradigm excellence.

Thus, the trends described above significantly change the effective models of software production management in software companies. The next section of the article shows how the given theoretical framework defines best practices for improving the Scrum paradigm.

This section outlines the methodological framework for improving Scrum practices in response to major trends shaping the IT industry, such as remote/hybrid work models, digitalization, cost-efficiency measures, and the rise of AI-driven development. Using systems analysis, the Pareto and Occam razor principles, and generalization techniques, the study analyzes data from industry research conducted between 2020–2023 in Russia and Europe to identify the most impactful, cost-effective strategies for Scrum optimization in alignment with evolving technological and economic conditions.

3. Results and Discussion

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 software development on customer and user satisfaction, and formalizing and considering sprint parameters. Moreover, it became more important in fully remote model of working in software teams, where common management needs more formal criteria of software project success.

Defining specific goals for each sprint is not sufficient; it is essential to accurately combine well-measured and articulated elements, including the business objectives of the products being developed, prioritized functional and non-functional requirements in sequential backlogs, and a logical sequence of stages in the software development process. 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 software development process and its 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 also actively influences the development process.

In global IT branch we may see a lot of practical examples of constant focus of software development company on gathering the feedback from software users:

(1) Atlassian corporation (USA) is gathering and managing the feedback about Jira for 10+ years in full cycle: from Feedback Collection (user surveys, in-app feedback tools, and support tickets) and anal-

ysis of feature requests and user behavior through data analytics to making changes in Jira's development roadmap. These enhancements have led to increased user satisfaction and retention. Regular updates and feature additions, driven by user feedback, have kept Jira relevant and valuable for project management professionals.

(2) Sportify (Sweden) utilizes feedback loops to continuously enhance its user experience. Company is using the Feedback Collection through user surveys, in-app feedback mechanisms, and analysis of user behavior data. The company also conducts A/B testing to evaluate the impact of new features. As a visible result, Spotify has implemented features such as improved playlist curation algorithms, social sharing enhancements, and personalized recommendations based on user feedback.

(3) Skyscanner (United Kingdom) uses feedback loops to refine its flight, hotel, and car rental search functionalities. Skyscanner collects feedback through user surveys, customer support channels, and in-app feedback forms. Based on gathered information Skyscanner has improved its search algorithms, added more granular search filters, and enhanced its price prediction features. They regularly update the platform based on user feedback to ensure a better travel search experience.

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 software development 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 the 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 entire teams, aligning them with the business needs of the software company and the rationale for implementing Scrum (see Table 1). The flexibility to adapt these recommendations to each team's unique situation is in line with Agile values.

An example to be considered is from the author's practice in 2018 in software Russian-Chinese company, illustrating the use of formalized sprint parameters in software development 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, it is important to acknowledge that complete adherence to an "ideal" trajectory is a detrimental utopia; however, the refusal to pursue it reflects a low level of managerial maturity within 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 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 software development and achieving goals. Certainly, 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.

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 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.

Next element of the excellence of the Scrum software production paradigm, based on IT branch digitalization, is a technical debt management.[1] 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 software development production processes is beyond doubt. The solution to this problem encompasses multiple aspects. Several levels can be examined:

a. 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;

b. tasks of technological development and decommissioning of tech debts form independent epics in product backlogs and have own assessments (labor

[1] It is the cost of maintaining and supporting software systems. Technical debt management has two key aspects – preventing technical debt from accumulating – efforts to identify it, be aware of it and implement certain procedures, and repaying debt – prioritizing, incentivizing quality work, refactoring, etc.

intensity, priority, risks and connections with other tasks);

c. 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;

d. 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;

e. 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 software development projects. Managing technical debt is an ongoing activity that starts with the planning phase of each sprint. 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. Several general recommendations exist for structuring 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.

The shift to fully remote work or a "hybrid" model has brought significant changes to certain traditional aspects of production processes. The key area – team communications – also have changed due to decreasing or even lack of personal face-to-face interactions. Different studies showed unlike results in estimation of success in adapting this trend. According to [16] 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 [8,17] state that fully remote work in software domain didn't change the efficiency of communications and it's transferring into digital channels have a positive impact on team's productivities. In the base 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.

It's senseless to describe the case studies of software companies in modification Scrum by the impact of FRM in production: during pandemic times all software companies worked without offices for periods of 3–9 months and most of them continued this practice after 2020. Much more interesting to observe the common Scrum changes according to [2,6,8,16,18]:

(1) Stricter and direct management: from scheduling the regular online meetings and detailing the tasks and defects in software development management systems to obligation of additional artifacts in project management and organizing additional ceremonies to provide the feedback about development's results (more demo, more focus groups, more involved managers);

(2) Fast implementation of new software tools for every production process with special focus on video communication tools and learning in e-channels;

(3) Additional efforts and spends (especially in 2020 and 2021) in engineer's motivation and involvement in projects, product's environment, corporate life (in virtual type of).

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 software development 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 [18]. Based on successful industry experience, the author recommends the following optimizations:

f. 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;
g. 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;
h. internal outsourcing of services of DevOps-engineers, instead of including an engineer in the team.

To begin, it is important to clarify that the Scrum Master role 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 [20] (and similar roles such as system administrators, environment engineers, software development automation engineers, etc.). 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. Further, many project managers and team leaders would like to have a dedicated engineer for product environments, CI\CD support, etc., 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 [21] and informal measures such as creating good working relationships and assigning engineers to areas of responsibility.

Another significant trend is the increasing use of AI tools in software development projects since 2023 [12,22], that is aligning with global shift to FRM and fully corresponding to demand of more effective cost-saving models in IT businesses. As it was presented above [12] 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 software development 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 few things are clear now:

(1) with AI tools many types of ordinary tasks (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.

According to [22,23] there are main aspects in analyzing the impact of AI tools implementation in software production on Scrum processes (from major to average importance) in corresponding case studies:

(1) Increasing cross-functionality among developers by enabling them to handle simpler tasks using AI tools;
(2) Involvement of AI in all routine works: from code writing to product's documentation;
(3) Total automation of code review with search and fix errors and vulnerabilities for simple modules without complex business logic.

In [23] there is a case study when almost all software product documents had been created via AI tools.

Thus, the focus of this article in Scrum methodology excellence is on following aspects:

(1) Data-driven formalization of Scrum production process: from sprint goals and sets of parameters to tech debt management;
(2) Ongoing automation of production processes through AI tools, enhancing the efficiency of Scrum teams Cost saving optimization of developer's teams in a changing paradigm of labor relations (FRM\ "hybrid");

These ideas can be applied to planning Scrum excellence within a software company, as they are well-supported by industry practice and align with current digital trends. As a result of this study there is a Table 2 with main findings: those elements are building the solid approach in scrum paradigm excellence and it's a resolution of corresponding scientific task of the article:

This section explores the evolving Scrum software development paradigm, focusing on measurable sprint goals, customer feedback integration, and managing technical debt to ensure long-term quality. It emphasizes the importance of formalizing sprint success criteria and highlights best practices like using the SMART framework for sprint objectives, ensuring measurable outcomes, and regularly gathering user feedback. The shift to remote/hybrid work models has introduced challenges in team communication, but structured digital tools and formalized processes help maintain productivity. Additionally, the rise of AI tools in software development is transforming production, automating routine tasks, enhancing cross-functionality, and improving overall project efficiency.

4. Conclusions

Competition within the software production and IT industries is characterized by distinct dynamics, most notably the perpetual demand for adaptive modifications in core production processes driven by intense competitive pressures. Scrum, with its iterative and flexible framework, offers a structured yet adaptable approach that aligns well with the complexities of modern software production. However, the true potential of Scrum lies not only in its methodology but also in its capacity to harness the benefits of digital transformation. The shift towards total automation and the emphasis on measurable processes have resulted in an abundant flow of digital data, which can be leveraged across various levels of IT business development, production processes, and even in anticipating and exceeding client expectations.

The integration of this digital data into the Scrum process marks a significant evolution in how core activities are managed, making them increasingly sophisticated, data-driven, and flexible. By utilizing relevant and accurately estimated indicators, management can make more informed decisions that enhance the agility and responsiveness of the development process. In this context, the continuous improvement of Scrum methodologies becomes paramount. This article delves into the most experienced, widely adopted, and forward-looking approaches to enhancing Scrum practices, focusing on the data-driven formalization of the Scrum production process and the cost-efficient optimization of engineering teams. As emphasized earlier, sustained investment in the production process is a critical determinant of competitive advantage in the software development industry. The recommendations outlined in this article are grounded in industry practices and offer practical guidance for implementing these process changes while mitigating associated risks.

One of the key strategies discussed involves the implementation of measurable and specific sprint goals, using the SMART (Specific, Measurable, Achievable, Relevant, Time-bound) format. This approach ensures that sprint goals are not only clear and focused but also quantifiable, which is essential for tracking progress and achieving desired outcomes. Furthermore, the continuous assessment of the short-term impact of software development on customer and user satisfaction is highlighted as a crucial practice. Methods such as product manager and customer reviews, user focus groups, and A/B testing are recommended as effective means of gathering immediate and actionable insights that can inform the ongoing development process. These practices contribute to a more responsive and customer-centric development approach, which is essential for maintaining a competitive edge in the industry.

The formalization of Scrum processes, particularly the parameters of development sprints, is identified as a critical factor in the success of software projects. Formalized sprint goals and parameters, including delivery speed, the relevance of created functionality, employee cross-functionality, and the management of risks associated with timely changes, are necessary to improve the effectiveness and efficiency of the Scrum process. Additionally, the management of technical debt is emphasized as a crucial aspect of maintaining long-term software quality. Addressing technical debt requires a multifaceted approach, including its identification and reduction in each sprint and the creation of specific tasks aimed at technological development and the systematic elimination of accumulated technical debt.

Optimizing the roles within Scrum teams is another significant strategy for improving software project outcomes, particularly in scenarios where budgetary constraints are a concern. The article discusses various widely-used actions, such as defining the role of the Scrum

Table 2: Main findings for scientific task resolution.

#	Element of Scrum Paradigm Excellence, in Correspondence with the Impact of Digital Economy Trends	Comments
1	SMART-goal for every sprint	Key principal here is focus on not only goal setting, but on the whole cycle of sprint management including the analysis of results and retrospective meeting.
2	Feedback loop in product development	All members of Scrum team have the impact on the product, but quick loop of feedback from users and customers might change the efficiency of all traditional ceremonies: from sprint planning to retrospective meeting.
3	Formalizing the parameters of Scrum development sprints	Depends on the main goals in sprints and software project at all, digital form with measurable values
4	Technical debt management	Based on formal QA indicators of software product, not on “the vision” of tech lead or architect
5	Formalization of digital communication processes	Usage of new digital tools, including AI agents
6	Optimizations in Scrum teams	Roles optimization (Scrum master, tech and team leads vs. architect), budget optimizations (including DevOps outsourcing)
7	Constant automation of production processes (via AI tools)	All production process from business needs analysis to maintain the production environment and final product should be automated (including corresponding AI tools)

master more precisely and considering the outsourcing of DevOps services for specific software projects within an IT company. These approaches contribute to a more streamlined and cost-effective development process while maintaining the high standards required for successful software production.

The approaches presented in this article form a robust foundation for achieving Scrum excellence within a software company. The shift towards data-driven formalization, automation through advanced tools, and cost optimization in Scrum teams is in alignment with the evolving demands of modern IT businesses. These practices not only enhance the efficiency and quality of software production but also ensure that development teams remain responsive to rapidly changing business requirements. By integrating feedback loops and formalizing sprint parameters, Scrum teams can achieve higher levels of satisfaction and loyalty from both users and business customers, eventually leading to the successful completion of software projects.

5. List of Abbreviations

AI	Artificial Intelligence
CI\CD	Continuous Integration\Continuous deployment
FRM	Fully remote model
FTE	Full-time employer
HIPAA	Health Insurance Portability and Accountability Act
IDE	Integrated development environment
MSF	Microsoft Solutions Framework
ROI	Return of investments
RnD	Research and Design
RUP	Rational Unified Process
SAF	Scaled Agile Framework
SD	Software development
SDE	Software Development Environment
QA	Quality Assurance

Availability of Data and Materials

All details (methodology, participants, restrictions) about applied research might be find in corresponding published articles.

Consent for Publication

The authors give the publisher the permission of the author to publish the work.

Conflict of Interest

The authors declare that they have no conflicts of interest to this work.

Funding

This research received no specific grant from any funding agency in the public or commercial domains.

Acknowledgments

The authors declare that they have no acknowledgements except warm welcome their parents and families.

References

[1] R. Uppal. (2022). *Agile and Scrum have become dominant software project development methods. IDST. AI & IT* [Online]. Available: https://idstch.com/technology/ict/agile-and-scrum-have-become-dominant-software-project-development-methods.

[2] D. Pashchenko, "Fully Remote Software Development Due to COVID Factor: Results of Industry Research (2020)," *International Journal of Software Science and Computational Intelligence (IJSSCI)* , vol. 13, no. 3, pp. 64–70, 2021. [Online]. Available: https://www.igi-global.com/article/fully-remote-software-development-due-to-covid-factor/280517. [CrossRef]

[3] W. Aghina, C. Handscomb, O. Salo, S. Thaker, "The impact of agility: How to shape your organization to compete," *Mckinsey Study*, 2021. [Online]. Available: https://www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/the-impact-of-agility-how-to-shape-your-organization-to-compete.

[4] N. Juhasz. (2021). *How Custom Healthcare Software Is Developed for Secure HIPAA Compliance* [Online]. Available: https://tateeda.com/blog/how-healthcare-software-is-developed-for-secure-hipaa-compliance.

[5] Volini E., Schwarz J., Indranil R., etc. (2019). *Delloite Grobal Human Capital Trends. Report by Deloitte Insights* [Online]. Available: https://www2.deloitte.com/content/dam/insights/us/articles/5136_HC-Trends-2019/DI_HC-Trends-2019.pdf.

[6] D. S. Pashchenko, "Russian experience in organizing fully remote software development: an industry study of 2021," *Software Engineering*, vol. 6, pp. 311–318, 2021. [Online]. Available: http://novtex.ru/prin/rus/10.17587/prin.12.311-318.html. [CrossRef]

[7] D. Smite, A. Tkalich, N. Brede Moe, E. Papatheocharous, E. Klotins, M. Pettersen Buvik, "Changes in perceived productivity of software engineers during COVID-19 pandemic: The voice of evidence," *Journal of Systems and Software*, vol. 186, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221002715. [CrossRef] [PubMed]

[8] D. S. Pashchenko, "The Consolidation of the Fully Remote Software Development Practice in Europe: Study of 2023," *Computer Science Journal Open Access*, vol. 1, no. 1, p. 102, 2023. [Online]. Available: https://www.yumedtext.com/files/publish/published-pdf--6-CSJ-102.pdf.

[9] D. S. Pashchenko, "Digitalization in Software Engineering and IT Business," *International Journal of Software Science and Computational Intelligence (IJSSCI)*, vol. 12, no. 2, 2020. [Online]. Available: https://www.igi-global.com/article/digitalization-in-software-engineering-and-it-business/252212. [CrossRef]

[10] K. Warner, M. Wäger, "Building dynamic capabilities for digital transformation: An ongoing process of strategic renewal," *Long Range Planning*, vol. 52, no. 3, pp. 326–349, 2019. [CrossRef]

[11] M. Barenkamp, J. Rebstadt, O. Thomas, "Applications of AI in classical software engineering," *AI Perspectives*, vol. 2, 2020. [Online]. Available: https://aiperspectives.springeropen.com/articles/10.1186/s42467-020-00005. [CrossRef]

[12] D. S. Pashchenko, "Early Formalization of AI-tools Usage in Software Engineering in Europe: Study of 2023," *International Journal of Information Technology and Computer Science*, vol. 15, no. 6, p. 8, 2023. [CrossRef]

[13] N. M. Komarov, D. S. Pashchenko, "Application of artificial intelligence technologies in the innovative activities of industrial enterprises," *The Eurasian Scientific Journal*, vol. 15, no. 6, p. 101ECVN623, 2023. [Online]. Available: https://esj.today/PDF/101ECVN623.pdf.

[14] M. Moreira, *The Agile Enterprise: Building and Running Agile Organizations*, 1st ed. Berkeley: Apress, 2017.

[15] S. C. Misra, V. Kumar, U. Kumar, K. A. Fantazy, M. Akhter, "Agile software development practices: Evolution, principles, and criticisms," *International Journal of Quality & Reliability Management*, vol. 29, pp. 972–980, 2012

[16] D. Ford, M. A. Storey, T. Zimmermann, C. Bird, S. Jaffe, C. Maddila, et al., "A Tale of Two Cities: Software Developers Working from Home during the COVID-19 Pandemic," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 2, p. 27, 2021. [CrossRef]

[17] O. A. Baakeel, "Impacts of Remote Working on Employees During the COVID-19 Pandemic," *International Transaction Journal of Engineering, Manage-*

ment, & Applied Sciences & Technologies, vol. 12, no. 10, pp. 1–14, 2021. [CrossRef]

[18] A. Cucolaş, D. Russo, "The impact of working from home on the success of Scrum projects: A multi-method study," *Journal of Systems and Software*, vol. 197, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121222002382. [CrossRef] [PubMed]

[19] A. Lewis. *Tech Lead Survival Guide*. Medium Inc. 2018. [Online]. Available: https://ann-lewis.medium.com/tech-lead-survival-guide-aeee065fe0f5.

[20] C. Ikerionwu, D. Edgar, "DevOps: Introducing agility and flexibility to BPO-IT organizations – service providers perspective," *Software Engineering*, vol. 4, no. 5, pp. 65–74, 2016. [CrossRef]

[21] R. M. Amaro, "Capabilities and metrics in DevOps: A design science study," *The Information Manager*, vol. 60, no. 5, 2023. [CrossRef]

[22] S. Peng, E. Kalliamvakou, P. Cihon, M. Demirer. (2023). *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot* [Online]. Available: https://arxiv.org/pdf/2302.06590.pdf.

[23] D. S. Pashchenko, "Implementation of software development projects using artificial intelligence tools," *Program and Project Management*, vol. 4, 2024. [Online]. Available: https://grebennikon.ru/journal-20.html#volume2024-4.