International Journal of
Information Systems and Project Management
ISSN (print):2182-7796, ISSN (online):2182-7788, ISSN (cd-rom):2182-780X
Available online at ijispm.sciencesphere.org

Agile software development approach for 'ad-hoc' IT projects

Michal Kuciapski
Department of Business Informatics, University of Gdansk
Piaskowa 9, 81-864 Sopot
Poland
michal.kucipski@ug.edu.pl

Bartosz Marcinkowski
Department of Business Informatics, University of Gdansk
Piaskowa 9, 81-864 Sopot
Poland
bartosz.marcinkowski@ug.edu.pl

Abstract:
Restrictive Scrum assumptions make the effectiveness of this approach debatable in projects deviating from typical execution conditions. This article delivers a comprehensive software development approach for both academic and commercial Information Technology (IT) projects effectuated by teams that are hampered by significantly unsystematic participation of project members and mercurial internal communication. The nature of 'ad-hoc' projects imposes another level of difficulty in terms of both managing the conduct of such a project and ensuring the quality of the end product. Multicyclic action research enabled a gradual adaptation of the Scrum approach to support such project conditions. This study introduces major alterations to Sprint implementation and minor enhancements within the documentation process to streamline knowledge sharing among Development Team members. Proposed key alterations include the evolution of Daily Scrum towards Weekly Scrum, the possibility of extending Sprints length, the eventuality to switch team members during Sprint due to substantial failure to meet deadlines, having at least two team members responsible for a single Product Backlog Item (PBI) at all times, as well as exclusion of Burndown Chart in favor for Development Team members updating their working time. Positive validation of enhancements in mixed settings confirms that the generic Scrum framework can be adapted to support highly volatile projects. The proposed approach is suitable not only for carrying out software development initiatives that rely heavily on the skills of external experts and/or volunteers. It also supports traditional Scrum teams that seek to reduce their exposure to risk arising from organizational changes.

Keywords:
project management; agile; software development; systems engineering; Scrum; adaptation.

DOI: 10.12821/ijispm110402

Manuscript received: 15 April 2023
Manuscript accepted: 5 August 2023

1. Introduction

Challenges that most modern business entities face when launching Information Technology (IT) projects, put pressure on both global development teams and management staff to adapt novel techniques, approaches, and tools [1] – and thus decrease failure rates. Leveraging the strengths of various existing processes more effectively owing to the adaptability of Agile Methods (AMs) contributed to establishing the dominance of the latter [2]. That said, the leading agile approach to project management, i.e. Scrum, necessitates a fairly restrictive and precise application of underlying principles throughout the development process [3] – which often cannot be facilitated by companies [4]. Daily meetings at a specific time and duration, minimum and maximum constraints on the duration of individual Sprints, presence at meetings required from all team members as well as significant experience in implementing projects using this framework required from local programming teams, in fact, constitute the underlying problem behind the non-negligible share of IT projects. Moreover, being obliged to deliver demonstrable results in a short timeframe, regardless of their scope, tricks teams into ignoring software quality and creates a pileup of quality-related challenges. These challenges make Scrum – should it be used by the book – unsuitable for running projects in unfavorable conditions, such as highly irregular cooperation between development team members. The Scrum guide itself neglects to discuss how this approach might be adapted [5].

Therefore, our motivation was to retain Scrum's ability to deliver the final product on time and budget under the required functionality while enabling projects to be carried out effectively considering the significantly unsystematic project members' participation as well as mercurial internal communication and deliverables provision. Efficiency of project realization is especially important for individuals with more years of experience, as they have a higher perception of the importance of metrics related to Scrum team performance [6]. We reckon that the Scrum framework can be adapted to the successful execution of a wide range of IT projects falling into the 'ad-hoc' category. The irregular way of cooperation basically comprises collaboration based on infrequent (or even rare) meetings of Development Team members, with the possibility of considerable gaps between them. Such a manner of collaboration is typical for projects that we henceforth address as 'ad-hoc' ones. Our early conceptual work explored and highlighted the determinants of such projects, including:

- no systematic daily implementation [7];
- the Development Team structure varying significantly not only between the initiation points of individual Sprints, but also during their execution [8];
- a bulk of the Development Team members devoting only a small share of their professional work to contribute to the project or participating in the project as a side contract, resulting in divergent availability of members [9];
- strongly diversified levels of time commitment to the project between different developers [10];
- high volatility in the availability of certain members of the team [11],[12];
- virtual form of the Development Team due to the significant geographical dispersion of its members [13],[10];
- collaboration between Development Team members likely to take an irregular form [14],[15].

'Ad-hoc' projects are, ipso facto, not in line with a number of aforementioned agile assumptions. We define 'Ad-hoc' projects as ones realized by distributed teams with highly fluctuating structures. We identified no approach that adapts the generic Scrum framework's Sprints to provide for such projects in the subject-related literature. The preliminary analysis of related research was re-checked based on the full contents of the AIS eLibrary, ACM Digital Library, IEEE Xplore, Springer, Web of Science, Scopus, Science Direct, as well as the EBSCOhost multi-source, full-text repository. In total, 92 journals and 221 conference proceedings articles that are directly or indirectly related to the keywords such as Scrum, agile project management, software development method, systems engineering method, software development methodology, systems engineering methodology; combined with adoption, adapt, adaptation, limitations, customize, elaboration were explored.

Building upon our early concepts for overcoming the limitations of the generic Scrum framework to support 'ad-hoc' IT projects, the goal of this manuscript is to deliver a comprehensive, practically applicable, and meticulously validated solution for projects with such specificity. In doing so, we strive to find an answer to the following research question –

Can Scrum as the key representative of agile software development approaches be adapted to the successful execution of 'ad-hoc' IT projects? The importance of both the research aim and the question is supported by the findings of the study by Almeida and Carneiro [6], who highlighted that there is a great need for extending the knowledge about Scrum project management processes and their teams – in addition to offering important insights into the implementation of metrics for software engineering companies that adopt it.

The target solution must not be confined to academia but take into account the specifics of commercial IT projects as well. Thus, the proposed streamlined approach was field-tested throughout the development of IT solutions during projects that featured significantly non-systematic Development Team collaboration in both environments.

After the introduction, related studies are discussed. The research design employed to elaborate an agile project management approach along with an in-depth discussion on individual cycles is presented next. Subsequently, the results of the study are introduced, followed by a discussion.

2. Related research

IT projects face rapidly changing environments. This applies to both business and technological perspectives, where the determinants of technology acceptance often depend on the area and organizational context of its application [16]. The evolution of cooperation among project members led to the establishment of virtual teams, in which geographically distributed, and often culturally diverse individuals use sophisticated technologies for interaction and collaboration; raising organizational maturity transformed those teams from an innovative source of competitive advantage into an important and necessary part of any organization strategy [14].

Such fluctuations particularly hinder project planning and further implementation of requirements [17]. AMs are better adapted to projects with high uncertainty and risk, where significant changes in systems requirements are necessary [18]. Agile frameworks and methods focus on close cooperation between team members, delivery of demonstrable products being components of a system under development in relatively tight cycles, and limiting documentation processes to a necessary minimum [19]. Moreover, prioritizing responding to change swiftly over following a plan allows for more flexible execution of software development initiatives and a reduction in the number of fatal shortcomings [20]. Successful attempts were made to ensure synergies between the IT-enabled agile approach and business process management, thus making business processes highly adaptable rather than rigid [21].

Scrum, leveraged by as many as 87% of enterprises that take advantage of Agile techniques (which constitutes a rise from 58% over three years) – remains a safe leader as far as AMs are concerned [22]. As with any approach, the Scrum framework has several assumptions and artifacts explained in detail in its guide [23]. Many of them were tagged as inexpedient to be modified. Plenty of studies indicate that this is impractical in business practice [24],[4],[25], and the adaptation of the agile approach to the specifics of a given project is a must [5],[18],[10]. These assumptions mean that the Scrum approach that strictly adheres to the indicated principles may be of little use in projects with participants who do not cooperate on a regular basis and communication is exercised downright 'ad-hoc' within a virtual team.

To meet the challenge of dissonance between the rigidity of certain Scrum features and IT practice, profiling the approach to different organizational contexts might be considered a viable option [26]. When considering such an enhancement, it is crucial to avoid the 'Scrum but' trap; numerous organizations declare adherence to Scrum, yet tend to skip Scrum features that are uncomfortable for them and are directly related to internal issues of those organizations at the same time [27]. Scrum adaptations tend to be introduced primarily across three organizational settings: (1) vertical scaling – individual projects of embedding Scrum in larger organizational aspects, such as strategic planning [11],[12]; (2) size scaling – implementation targeted at small, medium, and large software development initiatives [9],[11]; and (3) distributed structure – overall cooperation between distributed teams [15]. This study falls into the last of those areas. Simultaneously, it extends this organizational setting with 'ad-hoc' project specificity that necessitates surrounding organizational context, such as team structure, to be highly agile. An abrupt increase in such sets of constraints on the operations of companies was noted in Europe during the COVID-19 pandemic.

All tailor-fittings ought to be carefully measured and supported by empirical data. In this regard, Stålhane et al. [28] assessed how to adapt Scrum for safety-critical projects, so that it can be used without losing the benefits that one gets from both incorporating AMs and conformance with the IEC 61508 standard – which manifests itself e.g. in strict planning. Rolland et al. [29] attempt to overcome the challenges associated with using Scrum in large-scale projects. They come up with a customized approach that incorporates experiences from a 3.5-year project involving 120 participants. To improve the way of working – especially between teams and stakeholders – task forces were established across teams, champion roles were implemented, the practice of specifying in advance prior to Sprints was introduced, redistributing tasks within Sprints and improvising mini-demos in the middle of Sprints were pioneered. Caballero and Calvo-Manzano [30], on the other hand, discuss the lesson learned while tailoring Scrum to the needs of a very small company. To achieve more efficient resource management, a high-level functional specification was introduced through requirement elicitation. Sprint design usually lasting less than three days was treated as a systematic activity aimed at improving the knowledge of the project before starting coding, and a Sprint test was carried out by the quality team.

Despite failing to identify contributions discussing adapting the Scrum framework's Sprints to solve the problem of irregular cooperation between the members of the Development Team, the literature review enabled highlighting studies indirectly related to this contribution, pointing out:

- the existence of factors that interfere with Sprint execution, such as 'ad-hoc' requests during mid-Sprint and low inter-departmental communication, as they impede knowledge transfer between project members [31],[32];
- difficulties in maintaining traditional project meetings, along with a solution providing for using global virtual teams (GVTs) to enable a distributed team structure and project type [25];
- skipping or modifying some Scrum components for practical reasons, so that the overall approach is no longer in line with Scrum's guidelines, tends to be employed in organizational environments [18],[10];
- the necessity to elaborate hybrid methods to adjust to fluctuating requirements [33],[34].

Adapting the Scrum framework for 'ad-hoc' projects in line with the goal of this study is also justified by the belief that it is impossible to design one general theory of project management due to the vast differences between project types and contexts [35]. Additionally, cultural differences can affect how a project is run, especially from the perspective of team interaction [36],[37]. Moreover, an abrupt increase in high fluctuation of development team structure with the requirement to work remotely as distributed teams was noted across Europe during the 1st wave of the COVID-19 pandemic [38] and, to our best knowledge, similar settings during IT projects realization with Scrum framework were not explored in prior studies.

Importantly, the article aims to adapt Scrum only to the minimum extent required. The premise for it is the finding of Havstron and Karlson [39], pointing out that when adopting and using an SDM it is important to stay true to the philosophy of the method. Otherwise, software developers might execute activities that do not lead to the intended outcomes.

3. Research design

3.1 Method

Development of hybrid approaches from one side benefits from the strengths of each approach, and, at the same time allows to avoid their weaknesses [34]. On the other side, adopting and using a Software Development Method (SDM) without compliance with its philosophy might lead to unintended outcomes [39]. Therefore, to meet the goal of the article, the action research (AR) method was implemented. Software engineering is empirical knowledge, a synthesis of the experience of thousands of software development centers [40]. AR is a method of qualitative inquiry strongly oriented towards resolving practical problems of real-world organizations [41]. In contrast to waterfall-based methods, AR is capable of recognizing local perspectives [42] and is cyclic in nature. At least two complete AR cycles are required to fulfill the requirement of the process being iterative [43]. In our study, three strictly related research cycles were taken into account – with a 5-phase AR design employed within each cycle (Fig. 1). Decision to opt for five

phases was prompted by the team's previous involvement in research work based on such design, and thus knowledge of the subtleties of the method.

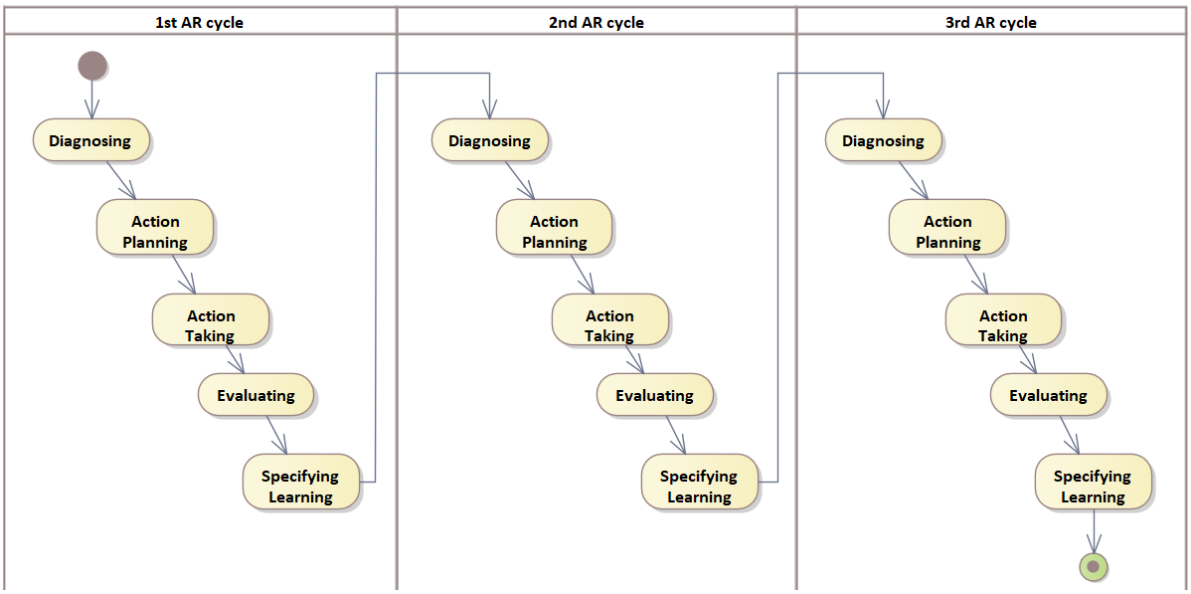

Fig. 1. High-level AR design underlying the study

Evaluation-oriented activities that ought to be considered while designing and running each cycle enable putting forward working hypotheses/propositions as well as assessing the legitimacy and direction of activities undertaken to drive organizational change and verify those propositions in a real organizational setting. At the same time, the unpredictability of real-world settings in which the change is implemented combined with the necessity to provide added value beyond an analysis of a given phenomenon calls for high flexibility of the method. The working nature of hypotheses and the ability to redirect research efforts between succeeding cycles deliver just that.

The study covered a couple of IT projects. One of the projects was executed within a student-exclusive environment, whereas the other was within a commercial one. Both scrutinized projects were effectuated in an agile way by virtual and geographically dispersed teams cooperating in a highly irregular manner, which contributed to a complex project implementation environment. The initial cycle was accomplished in a strictly academic setting, as persuading companies to allow conducting early stages of experimental research in the project management field during real-world software development projects often proves to be challenging. Deliverables of the study (Fig. 2) were subsequently drafted and tuned in within academic and commercial settings respectively (2nd and 3rd AR cycle) to ensure their universal nature. The decision to cover both non-commercial and commercial IT projects of different sizes was dictated by presumably dissimilar deviation levels from respective project plans due to Development Team structure changes resulting from no financial dependence of Development Team members. Moreover, we were vitally interested in determining whether these deviations are higher for volunteer-based projects, and whether individual adaptations may play a more important role than for commercial ones or not.

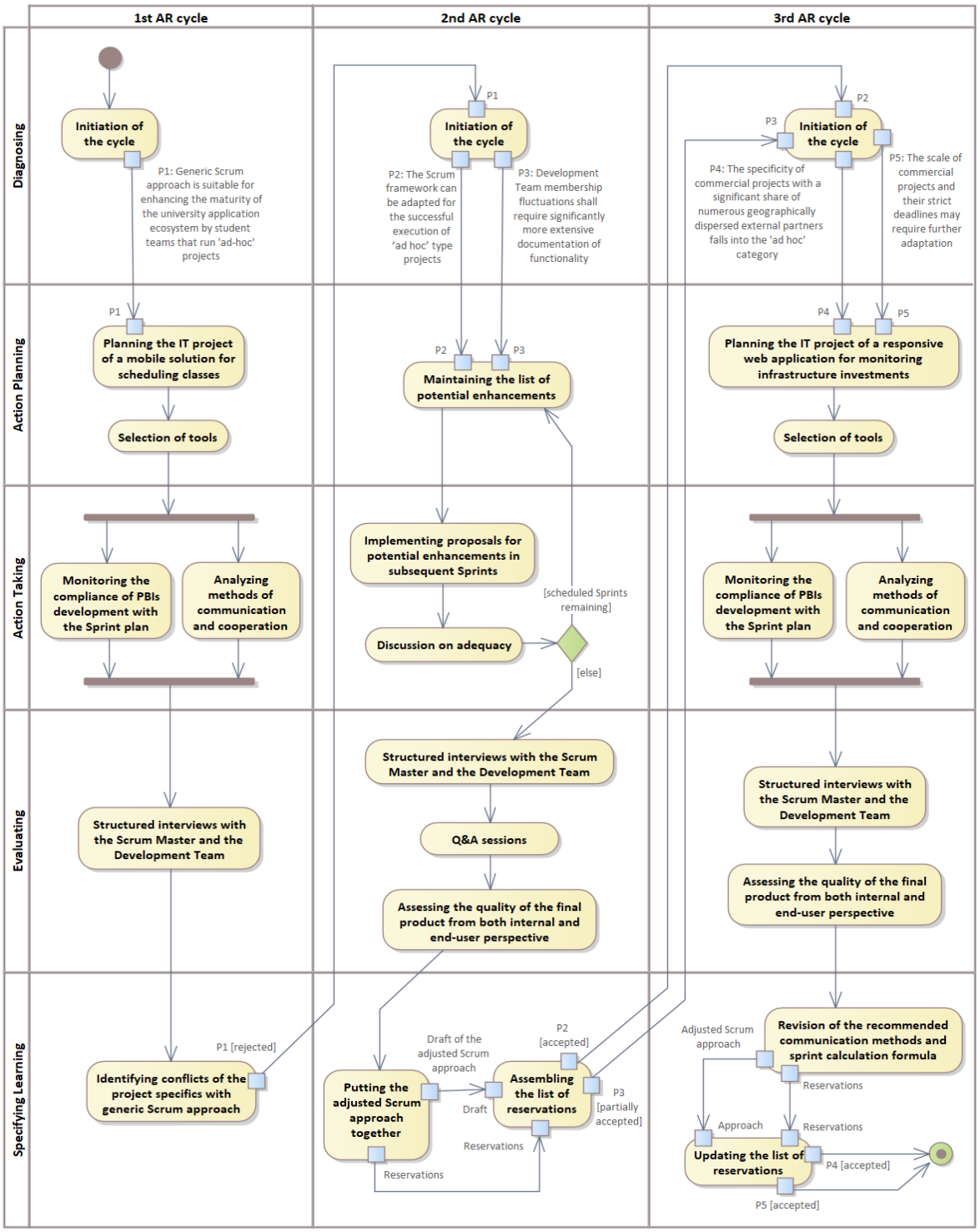

Fig. 2. Detailed AR design underlying the study

3.2 The first AR cycle

The groundwork for elaborating Scrum approach enhancements was laid in the early stages of a project executed by nine IT students: members of a student scientific society at the University of Gdansk, Poland. Five members of the Development Team combined university studies with the occupation of junior/regular developers. Their professional experience ranged from half a year to two years. In detail, the Development Team consisted of:

- mid developer – one student already working a part-time job for two years;
- junior plus – one student already working a part-time job for one year and a half;
- juniors – three students already working a part-time job for one year (two students) or half a year (one student);
- four outstanding students with no professional experience – two final year (third) students and one student in the second year.

The learning curve was of moderate length as the Development Team had complete freedom when choosing the technologies used. The frameworks adopted for system development were the ones already learned by students during their studies, and used for professional projects by mid, junior plus, and one junior developer.

The team planned and launched a project aimed at designing and building a mobile solution for both students and academic staff, which enables the management of class schedules. The specifics of the project pointed towards a good fit with generic AMs. First of all, only major requirements were known at the inception of the project, as the list of functionalities was closely related to the technical capabilities for importing relevant data from the main scheduling system that had evolved over time. Moreover, the mobile application was to be built following a flexible Web Oriented Architecture (WOA) to enable data exchange between different mobile platforms, as well as the web version of the solution. The risk of replacing some of the technologies used also had to be considered. Therefore, working research proposition P1 was put forward: *the generic Scrum approach is suitable for enhancing the maturity of the university application ecosystem by student teams that run 'ad-hoc' projects*.

Within the *Action Planning* phase, the team agreed upon the following primary functionality:

- collating data from relevant university websites: course plans, faculty staff's consultation hours, classroom availability, announcements, and information on important academic events;
- presentation of data obtained in a highly interactive way;
- multi-layer filtering of data related to class schedules, academic staff, and academic events;
- data processing through supplementing, removing, or modifying information on class schedules or consultation hours of academic staff;
- subscribing to events and receiving reminders via email;
- displaying alerts for upcoming events, such as exams.

Two Sprints have been covered by the first AR cycle. *Action Taking* phase soon revealed that project team members rarely had the opportunity to communicate with each other to transfer and share knowledge through face-to-face meetings. It was caused mainly by conflicts in students' class schedules where divergent days of the week and hours were available. Moreover, two out of nine students studied extramurally, combining working during weekdays with studying during weekends. As a rule, an online solution was used for (1) synchronous communication using TeamViewer video conferencing software; and (2) asynchronous communication based on Facebook groups and email. Project management was supported by Visual Studio Team Services (currently Azure DevOps), Trello, and Scrum templates prepared in Excel and shared via Google Drive. Since the *Evaluating* phase confirmed many issues, the list of conflicts with the generic Scrum approach was delivered within *Specifying Learning*, and P1 was eventually rejected.

3.3 The second AR cycle

The 2nd AR cycle corresponded with a bulk of Sprints (9) across the academic project and led to delivering a preliminary version of the Scrum approach adjusted for 'ad-hoc' IT projects. Given revealed limitations, the cycle was fueled by two working research propositions: (P2) *the Scrum framework can be adapted for the successful execution of*

'ad hoc' type projects; and (P3) *Development Team membership fluctuations shall require significantly more extensive documentation of functionality*.

From this moment on, monitoring the course of the project regarding the schedule and progress of work on Product Backlog Items (PBIs) ran in the background. It was the Scrum Master who was the primary person responsible for identifying sources of inefficiency; he had the support of an experienced Product Owner in coming up with proposals to address those. *Action Taking* involved both field-testing the proposals and ongoing ratification of the components of the approach based on three questions discussed with the entire Development Team after each Sprint review: (1) Which project management activities do you consider the most beneficial? (2) Which project management activities are the least beneficial for you? (3) What other activities would you integrate into the project execution process to make it more efficient?

To avoid embellishment and maintain the integrity of received responses for all of the conducted surveys we used a deductive analysis. This allowed us to make sense of interview respondents' individual stories via the narrative analysis method, mapped with structured or predetermined categories in advance. Therefore, we mapped connections in the data to distinguished categories strictly related to the aforementioned connections [44]. For instance, the first questionnaire came with such categories as: the most beneficial project management activities, the least beneficial project management activities, and missing project management activities.

Upon the project's completion, *Evaluating* the customizations featured a few interrelated steps. First, structured, in-depth interviews with the Scrum Master and all members of the Development Team were conducted to assess the project from two perspectives: the efficiency of the project management approach used and the quality of the final product. The questionnaire form to validate the former featured eight questions with the option of providing narrative feedback:

1. Are Sprint planning mechanisms appropriate?
2. Do the formulas for calculating the duration of a Sprint allow for precise measurement?
3. Does the lack of a Daily Scrum have a negative impact on project performance?
4. Do virtual synchronous and asynchronous communication solutions enable effective communication in the project?
5. Does the possibility of increasing the length of Sprints harm the efficiency of project execution?
6. Does discontinuing the practice obliging the members of a Development Team to participate in only one project have a negative impact on project execution?
7. Does the monitoring system detect exceptions?
8. Are project templates suitable for carrying out project tasks?

Secondly, a Q&A session was held to gather more detailed information on possible concerns. The quality of the final release of the mobile solution was assessed both internally and from the perspective of the end-user. The procedure for collecting feedback from Development Team members was again based on sessions including questionnaire forms and extensive discussion. Two open questions were addressed: (1) How would you rate the completeness and consistency of the application's functionality? (2) How would you rate the quality of the application in terms of readability of the layout, consistency of navigation mechanisms, intuitiveness of interaction with the user interface, as well as user-friendliness of solutions used for data visualization?

Hence, the former question was aimed at assessing the functionality of the final product, while the latter addressed the user experience associated with using it. As for the end-user perspective, the quality of the mobile solution was confirmed by both students and lecturers of the Faculty of Management. Altogether, 132 end-users (121 students and 11 lecturers) of the product expressed their opinions during face-to-face group interviews lasting from 15 to 30 minutes. They were approached with the same questions as the Development Team across a two-month-long timeframe. *Specifying Learning* of the 2nd AR cycle ended with accepting P2 and only partially accepting P3.

3.4 The third AR cycle

Further development and tuning in the proposed approach were based on a Diagnosis that (P4) *the specificity of commercial projects with a significant share of numerous geographically dispersed external partners falls into the 'ad hoc' category*; and (P5) *the scale of commercial projects and their strict deadlines may require further adaptation*. The layout of the 3rd AR cycle closely mirrored the 1st cycle up to the *Evaluating* phase across 17 Sprints. Yet, following the propositions the project setting shifted to commercial: a large electricity supply company was provided with a responsive web application for monitoring infrastructure investments. Depending on Sprint, the number of Development Team members fluctuated from 9 to 14. For its entire timespan, one backend senior software engineer (8 years of experience), and one frontend senior software developer (11 years of experience) were part of the project. Also, depending on Sprint there were 5 mid-developers (3-6 years of experience) and 9 junior developers (1-3 years of experience). The essential web system development component was carried out over fourteen months. Users of the system are employees representing infrastructure development departments, inspectors of investment progress, and the top management of the electricity supply company that employs over 8,700 professionals. The target solution was successfully integrated with the IT solutions used to date – namely, SAP (Systems Applications and Products in Data Processing) and SID (Distribution Information System). The project fully met the 'ad-hoc' IT project specificity:

- Scrum was used as a project management approach in which the development team consisted of twelve members on average.
- The Development Team was significantly distributed. System analysts and designers, front-end developers, back-end developers, testers, and product owner were employed by two universities and four companies. All members were located in different cities, a few hundred kilometers from each other.
- The team was virtual. Members of the Development Team conducted 94% of synchronous communication via video conferencing using Skype or Skype for Business. Project management was supported by JIRA and Google Drive.
- Participation in the project was 'ad-hoc'. Individuals' involvement in the project was an additional job, even though most of them devoted at least a dozen hours per week to it.

That said, both projects covered by the research differed in size and duration constraints. The academic project was relatively small, with no strong time pressure, whereas the commercial one featured a large scale and a strict deadline.

Action Taking confirmed the general adequacy of the agile software development approach for 'ad-hoc' IT projects elaborated during academic initiative for use in described conditions. Participation in the commercial project was not part of the regular work of team members. Therefore – such as the students – the professionals had very limited time available to carry out project tasks. Team meetings were also affected by numerous restrictions, and thus had to be very irregular. This setting forced the decision to run the project as a virtual team (which constitutes another similarity), with (a)synchronous communication limited to that provided by dedicated software without the benefit of direct meetings.

Evaluating incorporated procedures and questionnaires used previously in a student-exclusive environment (see section 3.3). The assessment was conducted by 43 employees from various departments and six branches during seven face-to-face meetings. As of the *Specifying Learning* phase, adaptation of communication methods, a revised Sprint calculation formula, and an updated list of reservations were introduced. Ultimately, both P4 and P5 were accepted.

4. Results

4.1 Academic project

As soon as feedback gathered during the 1st AR cycle contradicted that the generic Scrum constituted a good fit for enhancing the maturity of the university application ecosystem by student teams that run 'ad-hoc' projects, academic project contributors embarked on a systematic assessment of activities that exist under Scrum. In particular, conducting project meetings at fixed periodical dates (with no option to seamlessly swap team members) was considered detrimental (Table 1).

Table 1. Project realization conflicts with the generic Scrum approach

Scrum assumption	Conflict with the Scrum assumption
All Scrum members are obliged to participate in daily meetings and cannot participate in other projects during Sprints.	Development Team members are engaged in the project as part of their additional jobs. Only periodic availability was declared.
There has to be a Daily Scrum in the form of a 15-minute time-boxed event for the Development Team. It enables synchronizing activities and creating a plan for the next 24 hours.	Daily meetings are not possible, and the number of participants may change over time depending on the availability of participants. Moreover, communication must be remote.
Once a Sprint runs, its duration is fixed and cannot be shortened or extended.	Since the availability of Developer Team members during a Sprint may change dynamically, it seems useful to be able to extend the Sprint length to fix disturbances instead of canceling the Sprint.
Scrum Teams are self-organizing and cross-functional. No one, not even the Scrum Master, tells the Development Team how to turn a Product Backlog into releasable functionality.	Due to loose cooperation between team members during a Sprint, there is a noticeable need for the traditional role of Project Manager. He/she constantly controls the accomplishment of tasks members of the Development Team.

Developers and the Scrum Master also pointed out activities they would like to integrate during succeeding Sprints, as they believed those to be viable solutions to problems related to the least beneficial activities. Such activities included:

- determining Sprint duration based on the average availability of team members per week (introduced after completing the 1st Sprint and developed further after the 2nd and 3rd);
- scheduling meetings one-by-one, at different time intervals and with different lengths accounted for (put to work after the 1st Sprint and extended into virtual collaboration after the 2nd one);
- periodical evaluation of the availability of Development Team members with the option of changing its structure by swapping members or hiring new ones (introduced after the 2nd Sprint and escalated after the 3rd one);
- introducing the possibility of increasing the Sprint length or shifting some of the features being implemented to the next (with the former included in the adapted approach after the 3rd Sprint and the latter upon completion of the 4th one).

The intervention-oriented nature of the AR method enabled the integration of those activities gradually into the adapted Scrum framework (Fig. 3). It is the significant tailoring of the Scrum Sprint execution that constitutes the key modification to support volatile development team collaboration. As presented in the figure, both major and minor enhancements were included compared to traditional Sprint execution (marked with a dashed line). First of all, the list of developed features was based on their duration. This contradicts the relative system used in Scrum [45]. Features were chosen according to their priority, measured as a quotient of their business value in relation to their effort. The higher the quotient value, the higher the feature priority. Such a prioritization system is in line with the generic Scrum. As this occurred at very early project stages, it aggravated the potential risk of important features, with a high expected workload, ending up getting a relatively low priority. Thenceforward, features developed within succeeding Sprints were ranked by business value. If there were features assigned the same rank value, then priority was calculated. Subsequently, the list was limited to 100 hours of work (as velocity), which must have been provided by the team members during the Sprint. Such a value was ascertained as a result of measuring members' availability during the first two months of a project. Development Team members could only allocate an average time of close to 11 hours for a 4-week-long period. Moreover, depending on the period, values differed significantly among members where deviations reached hundreds of percent.

Also, the maximum Sprint time was extended to ten weeks (instead of the usual 30 days). This decision required the consent of stakeholders regarding the time required to prepare a decent number of PBIs and was introduced after the 3rd Sprint. The justification for this change was to mitigate the risk that several Product Backlog Items would not be implemented during a Sprint due to the limited availability of certain Development Team members. Therefore, the Development Team calculated the Sprint duration based on the *100 hours / average available hours per week* formula. The formula was determined based on the analysis of the 2nd Sprint, in which few features were developed given the

intermittent availability of project team members – a total of only 57 hours per month. This means that the estimation of the Development Team velocity is project-dependent.

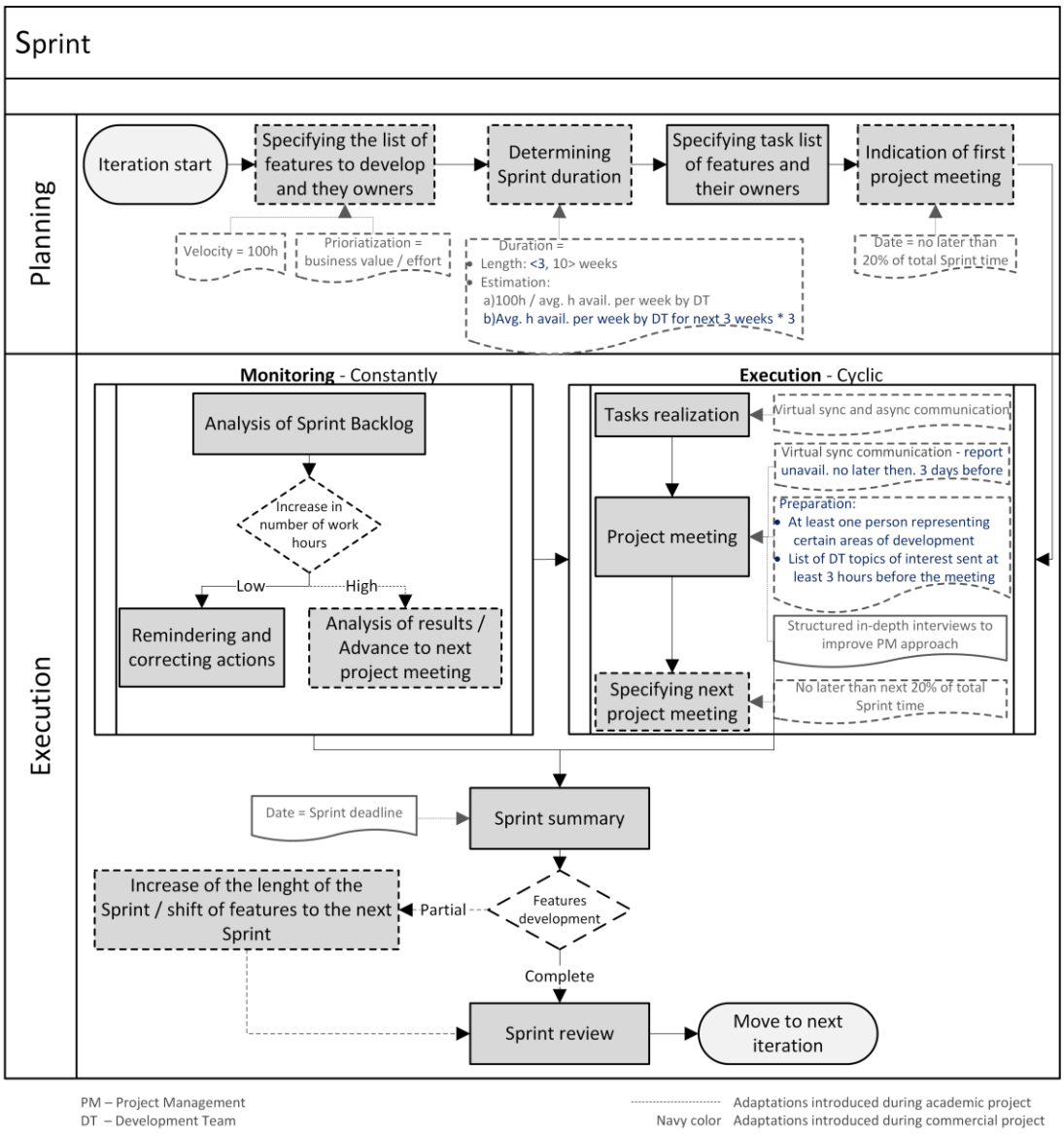

Fig. 3. Modified Sprint execution process

An important determinant of the project realization was the inability to maintain Daily Scrum (see Table 1). During the 1st Sprint, many members of the Development Team did not participate in most everyday meetings. A typical root source of the problem was that the daily schedule of meetings conflicted with other important duties associated with working or studying. All team members listed daily meetings as Scrum's activities they found least favorable. Therefore, the 1st Sprint contributed to the project management approach being adapted to periodically organize project

meetings with dates set during previous meetings (Fig. 3). To ensure the adequate number of project meetings during Sprints, the frequency of participation in daily meetings throughout the 1st Sprint was subjected to analysis. Based on the results and consultations between the Scrum Master and the Development Team, it was decided that meetings should be scheduled no later than the following 20% of the total Sprint duration. Due to the varying availability of the contributors, the exact date and time were jointly agreed upon a few days before the pre-determined date, and the action was coordinated by the Scrum Master. Beyond that, it was considered acceptable that not all members of the Development Team were required to attend a project meeting if it was not possible to find a suitable date for all of them. Such enhancement was integrated into the proposed approach after the 1st Sprint.

Many contributors were not available on-site and complained that they had to spend a lot of time traveling to participate in daily meetings instead of working on developing the mobile application functionality. To facilitate communication and allow more flexible participation in project meetings, digital communication channels were mainly used instead of traditional face-to-face meetings, becoming an integral part of the adapted Scrum after the 2nd Sprint.

The loss of the Daily Scrum caused significant changes within the project monitoring process. The Scrum approach assumes daily monitoring of the remaining working hours (which was not applied). This was dictated by the various work patterns of individual members of the Development Team, depending on their availability. The analysis of the Burndown Chart after the 1st Sprint revealed a great downtime in its update, traced back to highly varying work periods of certain contributors. Some of them had over week-long interludes in developing PBIs. Thus, the Burndown Chart proved to be of little use as a monitoring tool. Therefore, in cooperation with the Scrum Master, it was decided that the Development Team members should systematically update their working time for specific dates in the Sprint Backlog. This is contrary to generic Scrum, where working hours for given dates are set at the beginning of a Sprint, and, during its execution, the values are decreased according to the work done. The Scrum Master was to analyze the results of updates systematically. When he deemed it fitting, he could decide to proceed to the next project meeting. If the increase in the number of hours was small, the Scrum Master carried out appropriate corrective actions. Those included motivating project members, replacing contributors, or expanding the Development Team. Said tweak was integrated into the solution after the 2nd Sprint.

Per Scrum principles, each Sprint ended with a summary and review (Fig. 3). In this respect, one significant modification was introduced. When only partial implementation of specified features occurred, it was permissible to increase the length of the Sprint. The lack of daily collaboration between programmers and their intermittent availability posed a serious risk in ensuring that all functions were completed before the deadline. This was the case in the 3rd Sprint because one of the Development Team members was replaced by another developer due to a substantial failure to meet deadlines. For the third task, one contributor (MK, see Fig. 4) was substituted with another one (JL). Recruiting a new Development Team member and familiarizing one with the current state of the product under development (and the project management approach being used) prevented the development of two PBIs according to the schedule. Unfortunately, one of the PBIs was developed solely by the dismissed contributor. Hence, the basic solution that was included in the adapted Scrum approach beginning with the 3rd Sprint was to always have at least two team members responsible for a single PBI, except trivial ones.

Additionally, the Scrum Master was hereafter authorized to increase the length of the Sprint should deviations from the plan be minor, or even to transfer some of the developed features to the following Sprint without canceling the current one. He was obliged to consult first with all contributors involved in the development of PBIs that would not be prepared on time. The choice of a measure – increasing lengths or offsetting PBIs – depended on the scale of time discrepancy and was made individually for each PBI during consultations. Ipso facto, the role and responsibility of the Scrum Master was extended compared to the generic Scrum. His effectiveness in coordinating the project was recognized as a highly relevant factor for the project's accomplishment. Further empowerment of the Scrum Master resulted not only from the dispersion and virtuality of the team, the impermanent involvement of individual contributors, and the inability to conduct regular, cyclical meetings. The repeated absence of several members of the team at given meetings, even though the meetings were virtual, was another factor. The issues limiting team collaboration did not enable full self-organization of the team. Apt decisions regarding the implementation of individual

PBIs and Sprints lengths, however, were made each time after consulting selected team members who were directly affected by the problem.

As a result of the far-reaching modification of Sprinting, the 2nd element of the proposed Scrum adaptation involved customized documentation (beginning with the 3rd Sprint). This modification met the reported necessity to augment the project realization monitoring process. On top of that, a backup accountable developer for each PBI was nominated beginning with the same Sprint – and had to be precisely indicated in the documentation. Taking generic Scrum as a benchmark, the enhancements (in contrast to streamlining the Sprint execution process) simplified things. First, the Sprint Burndown Chart (usually being an integral part of Scrum-realized projects) was excluded. As already mentioned, the volatile availability of contributors meant that the information provided by the Burndown Chart was not very useful. Usually, on a given day the number of working hours devoted to given tasks was much lower or much higher than indicated by the Ideal Trend. Some customizations were included in the Sprint Backlog (Fig. 4).

"Mobile Student Schedule WZR" Application
Sprint 3 - ETL (Extract, Transform and Load) faculty data from the university website

Initial Developer Team members:	5
Duration in days:	14
Start of Sprint:	04 May

Id	Type	Task	Status	M1	M2	M3	M4	04/05	05/05	06/05	07/05	08/05	09/05	10/05	11/05	12/05	13/05	14/05	15/05	16/05	17/05
1	Design	Analysis of data structure on schedules website		AM	JL			4													
2	Coding	Programming connection and data downloading from website		AM	JL					4AM	2AM	3JL	3AM 2JL								
3	Coding	Programming data transformation and load to SOA Web Service standard		MK	JL										5	2		6			
4	Testing	Testing validity of ETL with the website		MK																1	3

.....

ID	Name	Function
1	JL, AM	Senior developer
2	MK	Developer
3	TK	Junior developer, tester
4	MK, TK	Tester
5	AM	Requirements
6	MK	Deployment
7	MK	Documentation

Status
Finished
In Progress
To do
Canceled
Bug
Other

Fig. 4. Customized Sprint Backlog example

Columns M1-M4 were added to the Sprint Backlog document to represent the initials of Development Team members accountable for handling each task. Value in the M1 column is the initials of the main person responsible for PBI's task. Values in M2-M4 columns are initials for other Development team members responsible for the task, whereas importantly muted values represent reserve person(s) for the task. The latter have basic information about a PBI, and will only contribute to task realization in case of disturbances – such as failure to meet deadlines by another member. Therefore, the proposed approach addressed the moderate risk of 'ad-hoc' projects that some project members may suddenly become unavailable. The former initials indicate the main person accountable for a PBI, whereas the latter – a deputy, should disruptions within a given Sprint implementation emerge. The initials of the contributors who were unable to continue working on the task are marked in grey. When the recruitment of a new PBI supervisor is conducted, the availability of the deputy is checked first (see Task 3 in Fig. 4). If the task's supervisor remains involved in its execution, yet support is required, an additional backup person is recruited. When more people perform the task, initials accompany the number of hours for particular dates (see Task 2 in Fig. 4). Providing such information is crucial to predict future problems and introduce corrective measures. The corresponding spreadsheet template was created and shared via Google Drive. IT solutions (e.g., JIRA) commonly used for agile project management enable only one person

responsible for each PBI to be assigned. These solutions also feature limited ability to mark Development Team members as temporarily unavailable or no longer available.

The Sprint Backlog template stores dates when the Development Team members update their working hours. This is also a result of the monitoring system adopted. As a Sprint begins, estimates for individual dates are provided in grey. Then, during the execution of the Sprint, actual values are recorded in black. If the Scrum Master detects a significant discrepancy between estimates and actual values, information on the root causes of this situation is obtained first from individual project members. Subsequently, corrective actions are conducted. Other elements of the Sprint Backlog template are consistent with the generic Scrum approach.

4.2 Commercial project

Fine-tuning the draft jointly with the professional IT community confirmed the added value of this approach in respect of:

- adaptation of the Sprint execution process (Fig. 3, marked with navy color), except for the formula for calculating a Sprint duration as well as meetings planning;
- customized Spring Backlog documentation (Fig. 4);
- crisis management methods that address the unavailability of members of the Development Team.

The *100 hours / average available hours per week* formula used to calculate the duration of a Sprint did not seem fully suitable. When planning the 1st Sprint, eleven contributors had an average availability of 13.7 hours per week every month. Thus, applying the formula would end up with a Sprint length taking just 66% of a single week. This constitutes approximately three business days, as *100 / (11 * 13.7) = 0.66*. In a setting that significantly limits the capability to organize meetings of the entire Development Team (which, in our experience, is typical for commercial projects with the close involvement of external experts), it was impossible to plan Sprints every three days. After consulting developers and the Scrum Master, it was agreed that Sprints should last at least three weeks so that Sprint planning would not take place too often. It was justified that, for shorter periods, time overhead for Sprint planning meetings would be too high regarding later development time, as commercial project development team members could allocate moderate time (several hours a week). This is contrary to the very limited availability of the Academic project developers (a few hours a week). Therefore formula "*100 hours / average available hours per week*" has been wrapped with "*at least 3 weeks condition"* (Fig. 3).

As a result, an additional formula was fashioned for calculating the minimum Sprint length based on the number of working hours of 'ad-hoc' teams with increased availability: *the average available hours per week by the Development Team for the next three weeks * 3*. In the 1st Sprint, it amounted to 452 hours – *(13.7 * 11) * 3*. The choice of one of two alternative basic formulas for planning Sprint duration depended on the average availability of the Development Team per week. Such enhancement was integrated upon completing the 1st Sprint of the commercial project.

After the initial two Sprints, the members of the Development Team highlighted the inefficiency of both the planning and implementation of the knowledge-sharing infrastructure. Communication via JIRA, emails, and video conferencing enabled cooperation between front-end and back-end developers. Unfortunately, there were situations in which interactive communication featuring screen sharing was required, involving members responsible not only for programming. Above all, it covered requirements analysis, development of business algorithms, system specification, interface prototyping, and deployment. Established communication patterns, where a determination of the next date of the entire Development Team meeting is based on the analysis of the advancement of PBIs, revealed that:

- some contributors deliberately delayed the fruition of their respective tasks, waiting for the next project meeting to be able to consult complex or ambiguous problems with the entire team;
- scheduling difficulties required additional, synchronous project meetings of an ad-hoc nature involving the entire Development Team;
- the use of asynchronous communication tools to obtain properly correlated information from various contributors turned out to be time-consuming and, in some cases, caused misunderstandings.

To address the aforementioned challenges of sharing knowledge among contributors, after the 2nd Sprint it was decided to hold regular meetings of the entire team via videoconferencing instead of physical ones. The modification was integrated into the proposed approach (see Fig. 3, remarks regarding *Tasks realization* and *Project meeting* activities as part of the *Execution*) beginning with the 3rd Sprint. Scheduling such a meeting ought to be pursued during Sprint planning. To limit the issues arising from individual contributors canceling participation in a given meeting, all members of the Development Team were required to report this to the Scrum Master no later than 3 days before the meeting. The 2nd vital ruleset was that at least one person had to represent certain areas of system development each time, namely: requirements analysis, specification, front-end development, back-end development, testing, and deployment. Finally, members of the Development Team were obliged to provide the Scrum Master with a list of topics of interest to them at least three hours before the meeting. The Scrum Master was responsible for preparing a joint list and passing it to the entire team no later than two hours before the meeting. This minor, yet still important prerequisite was added to the elaborated approach after the 3rd Sprint.

5. Discussion

5.1 Development Team structure

Periodic assessment of Development Team members' availability proved useful for the overall performance of 'ad-hoc' projects. The frequency of taking the option to change the structure of the team slightly exceeded rough estimates. Team members who turned out to be much less available during the development of PBIs than they declared could be replaced seamlessly. This happened twice during the academic project of mobile solution development and once during the commercial one. If minor issues related to declared availability were reported by some contributors, new team members could be added – which is convergent with a study by Tekin et al. [46]. This solution was used across four student Sprints, and only once during the execution of the commercial project. Pre-planned intervention prevented shifting certain PBIs to the following Sprint or extending the current one three times. The academic project once required an emergency modification of the Sprint schedule despite extending the team. Strict adherence to the generic Scrum framework would not allow such an event and would imply abandoning the Sprint and restarting the whole scheduling process. Such results build upon reports from Heikkilä et al. [11] and Rolland et al. [29], who considered the stability of a Development Team an important aspect of strategic planning during project vertical scaling. Therefore, retaining the Development Team structure might be considered project size-dependent.

5.2 Sprint execution

The admissibility of lengthening Sprints (or shifting some of the features under development to succeeding ones) also gained much traction in challenging real-world settings. Apart from holding additional flexibility in high esteem, project members highlighted a lower degree of frustration that the scheduled number of PBIs during the Sprint could not be developed compared to previous projects. Naturally, delaying the development of PBIs or extending Sprint timespans generally has a negative impact, especially if it occurs frequently. Such action cannot be automatic, and it should be preceded by a detailed Sprint execution analysis. The student project featured a single situation of transferring PBIs to the following Sprint. On two occasions, the Sprint was stretched. Including such a feature in the structure of Sprint prevented it from being canceled and avoided spending extra time planning a new one. The time needed to reorganize the Sprint schedule in the 'ad-hoc' setting simply proved considerably shorter. Moreover, to make up for the delay resulting from re-scheduling/shifting PBIs, further human resources were secured beforehand for the next Sprint.

Transferring PBI development to subsequent Sprints was not necessary in the commercial setting, and the need to extend the Sprint happened twice. The latter concerned the first two Sprints and resulted from communication patterns in force at that time – where the determination of the next date for the entire Development Team meeting was based on an analysis of PBI progress. The proposed modification of the framework approached the problem by planning regular meetings of all participants via videoconferencing once a week. Our results, highlighting the necessity to ease restrictions related to Daily Scrum, are consistent with a few other studies concerning Scrum adaptation for distributed

organizational settings [10],[15]. However, in contrast to these, the level of relaxation is much higher – project meetings can be held even only once per week. This is justified by the 'ad-hoc' features of IT projects.

Ensuring that (1) representatives of each area of system development must always be present during online meetings; and (2) a list of reported topics is to be sent to the Development Team members before the meeting were another two interventions that nipped the recurring issue of extending Sprints in the bud. Narrowing down the number of Development Team members obliged to take part to the representatives of given areas only is a new solution that is not brought up in other studies [2],[31],[15].

5.3 Benefits of the adapted Scrum approach

The Development Team members highly rated the modified Scrum approach with respect to its use in 'ad-hoc' projects. All twenty-one developers representing both projects provided convergent responses to the entire set of questions (Table 2). This clearly indicates that the enhancements introduced were perceived as alike. As projects met their original deadlines, the proposed Scrum adaptation alone did not lead to exceeding the premeditated durations of product delivery – what is beyond doubt a risk accompanying the introduction of organizational change. In our opinion, the reason for receiving highly consistent feedback on evaluation questions was related to the method chosen for tailor-fitting the project management approach. AR method enabled continuous improvement by introducing new components after each Sprint within successive cycles, owing to observations and extensive consultations. Any method that would assume a linear scenario of *approach adaptation → holistic implementation → final assessment* would be unlikely to be rated as favorably. The study implies the legitimacy of loosening some Scrum principles within 'ad-hoc' projects. These include keeping a strict daily meeting routine, retaining planned Sprint lengths, and enforcing Development Team stability while running Sprints.

5.4 Risks associated with the approach

Even though the approach was positively validated, we carefully considered the concerns that were raised in the final stages of the 2nd and 3rd AR cycles. Three students expressed the view that accepting reshuffles in the team structure during Sprints as a normal practice makes developers feel less connected to the project and reduces the call of duty (see Table 2). Similar feedback was not recorded during the final assessment of the approach after the commercial project. We believe that this inconsistency was caused by having just a single replacement of the developer in the latter case. Moreover, each professional was engaged in the venture for at least several hours per week. Sweeping fluctuations in the structure of the Development Team constitute a possible negative effect of our solution. Therefore, it seems reasonable to allow the structure of the team to change during a Sprint, yet treat it as an emergency solution only.

Replacing Development Team members with new ones requires additional time to become familiar with the project and its setting. Also, it might enforce (be beneficial) to recalculate the Development Team velocity before/during Sprint (Table 2). The negative impact of this situation is unavoidable. The primary solution to reduce such a drawback is to document the project to a slightly greater extent than is traditionally found in Scrum. Typically, lean documentation is created; both in terms of specifying functionalities to be implemented and code delivered already. The academic project featured an attempt to incorporate a light version of Unified Modeling Language for providing a system overview and sequencing crucial Use Cases to enhance cooperation among system developers, yet this proposal proved too far-reaching and was rolled back. Real-life conditions demonstrated that the documentation ought to be detailed just enough to enable the successors of PBI developers to pursue implementing PBIs being taken over on its basis. The class structure might be recreated after actual implementation using reverse-engineering functionality of contemporary CASE tools if needed. It is also vital to have communicative team members, and always ensure that at least two people are familiar with each topic. Such an approach is original to our Scrum adaptation, as it was not featured in other reported cases [11],[9],[15],[29].

Minor negative comments addressed the capability of stretching some Sprints by members of the student project. Four contributors highlighted that allowing PBIs to be moved to the next Sprint or extending lengths could harm overall project performance. Developers might be less motivated and even effective, as they are aware that the development of

PBIs can always be shifted – so that other duties can be treated as having higher priority. The same opinion was expressed by only a single contributor to the commercial project, who noted that projects running simultaneously and requiring one's input could be considered more important for the same reason. Across both ventures, the Scrum Masters' motivational skills were at a level high enough to avoid such drawbacks. If the Scrum Master were a mediocre motivator, this might become significant. Therefore, in our opinion, PBI relocation between Sprints should be limited to the features of secondary importance. These conclusions are partly divergent from previous studies [12],[9] that highlight the need to maintain consistency in the implementation of Sprints. Therefore, the proposed feature of our Scrum adaptation should be considered particularly valuable for small and medium software development projects. For large initiatives, a prior individual assessment of this adaptation's usefulness should be conducted.

Table 2. Validation of the approach – feedback summary

Question	Feedback	Comments
Are Sprint planning mechanisms appropriate?	Yes	
Do the formulas for calculating the duration of a Sprint allow for precise measurement?	Yes	The academic project only: ▪ Might be beneficial if the Development Team velocity is recalculated before each Sprint especially if the team structure significantly changes.
Does the lack of a Daily Scrum have a negative impact on project performance?	No	Both projects: ▪ Could have a negative impact on less motivated Development Teams.
Do virtual synchronous and asynchronous communication solutions enable effective communication in the project?	Yes	
Does the possibility of increasing the length of Sprints harm the efficiency of project execution?	No	Both projects: ▪ Possible negative impact on overall project performance – developers might be less motivated. ▪ Other projects featuring a less flexible approach to project management may be prioritized.
Does discontinuing the practice obliging the members of a Development Team to participate in only one project have a negative impact on project execution?	No	Academic project only: ▪ Developers feel less connected with a project. ▪ Additional time is required to familiarize developers with the project and its setting.
Does the monitoring system detect exceptions?	Yes	Academic project only: ▪ Possible deterioration of the quality of test cases – additional workflows are required to fix bugs and optimize source code.
Are project templates suitable for carrying out project tasks?	Yes	

As the monitoring system is concerned, the most significant reservations were raised by members of the academic project. Although it was rated positively overall, three of the contributors linked the changes in the Development Team to the quality of test cases. Potential frequent fluctuations in membership contributed to shifting the elaboration of test cases to the final, rather than to the early stages of development of individual functionalities. Should the developers tasked with implementing given PBIs be replaced, their successors would prepare test cases covering the functionality they did not develop themselves. The study revealed this led to deteriorating the overall quality of test cases. Therefore, assuring high-quality target software required additional workflows to fix bugs and optimize application code. The

natural reason for this state of affairs is that newly introduced team members have limited knowledge of previous work – including whether the PBI-related source code already written was tested and optimized or not.

Moreover, the likelihood that the added value provided by new developers ceases the existing code to work properly proved to be greater than in traditional teams. Test-driven Development (TDD) was the countermeasure introduced during the 4th Sprint. Despite that, two members of the team found this insufficient. Therefore, additional solutions should be explored. Again, this issue was not raised by the members of the commercial project.

6. Implications and limitations

Research results allow several implications to be pointed out. To begin with, the adapted Scrum approach supports the accomplishment of software development projects by enthusiasts – who run such projects during their spare time, which is difficult to predict. Whereas half of the proposed Scrum adaptations were equally suited to non-commercial and commercial 'ad-hoc' software development projects executed by distributed teams with highly fluctuating structures, some of them constituted a better fit for the former (Table 3). A bulk of variations associated with running Sprints result from fluctuations in team membership, and these are more common in volunteer-based projects. One reason is that there is no link between the success of a project and the remuneration of contributors.

That said, both the viability of the far-reaching rebuild of a team during a Sprint and having two members ascribed for each PBI (instead of a single one) meet some challenges of contemporary companies. Not only does it address the need for the operational management of the intensive use of external contractors, but also greatly reduces the risk of derailing IT projects should a major change in a setting occur. Under normal circumstances, project teams must cope with potential changes in the responsibility of company departments or activities related to mergers, international rollouts, and acquisitions. In 2020-2021, however, a large part of the globe experienced conditions far from normal, manifested in particular by local and national lockdowns. Being equipped with a minimal amount of information about PBI that would allow taking over its realization or supporting the main developer accountable is vital when employees are let go or shifted to other duties. Even though scrutinized projects were fully realized in an agile manner, traditionally managed projects are also prone to the identified challenges of 'ad-hoc' conditions. Proposed solutions might therefore also support human resources management within waterfall-based software development initiatives.

On top of that, the tailored framework provides the means for companies to increase the flexibility of multi-project employee allocation. In such conditions, challenges with planning Daily Scrums accumulate. Although the use of videoconferencing tools is not necessary under normal working conditions in non-dispersed organizations, developers involved in several parallel projects benefit from (1) the evolution of Daily Scrum towards 'Weekly Scrum'; (2) greater flexibility in planning Sprints while maintaining transparent scheduling rules; and (3) liberalized rules of attendance at meetings. The adapted approach also facilitates the involvement of external experts who, due to their primary contractual obligations, often have planning conflicts regarding meetings. Once again, such practices increase the organizational maturity of entities highly dependent on experts outside their payrolls and the adaptability of entire organizations to potential crises that thwart earlier planning arrangements.

Table 3. Comparative analysis of adaptations' fit

Scrum framework adaptation		**The setting of a project**	
No.	**Description**	**Non-commercial**	**Commercial**
1	Option to change the structure of the Development Team Structure during a Sprint	*Highly useful*: The lack of financial dependence causes frequent disruptions in the structure of the team, even during Sprints.	*Moderately useful*: Employee contracts in place reduce the risk of Development Team membership fluctuating during Sprints.
2	Ascribing backup accountable developer to each PBI	*Highly useful*: provides a viable tool to counteract frequent fluctuations in the structure of the Development Team.	*Moderately useful*: In this setting, developer substitutions are relatively rare.

Scrum framework adaptation		The setting of a project	
No.	**Description**	**Non-commercial**	**Commercial**
3	Option to extend the duration of a Sprint	*Useful*: The time needed to reorganize the Sprint schedule in 'ad-hoc' settings is usually shorter than to cancel it and initiate planning another one; a more useful solution than the PBI shift (#4), which requires securing additional human resources beforehand.	
4	Option to shift some PBIs under development to subsequent Sprints	*Rather useless*: extending Sprint durations (#3) turned out to be a much better solution; the occurrence of such a situation is also limited by other Scrum adaptations (#1 and #2).	
5	Passing over Daily Scrum restrictions with deadlines calculated from PBI progress and holding virtual meetings	*Highly useful*: It allows scheduling and successfully conducting cyclical meetings less frequently, yet often enough for dispersed teams, in which individual members of the Development Team have significantly different availability dates.	
6	Narrowing down the Daily Scrum participants to development areas representatives only	*Highly useful*: It supports conducting cyclical meetings while retaining the capability to solve a wide spectrum of programming challenges thanks to full coverage of the required IT competencies.	
7	A slight increase in PBI documentation's level of detail and prevention of cursory descriptions of technical tasks	*Highly useful*: a tool to reduce the level of chaos in case of an emergency PBI takeover (#1).	*Moderately useful*: In this setting, developer substitutions are relatively rare.
8	Extending Scrum Master's powers towards managerial	*Highly useful*: it allows reacting faster when instability in the Development Team structure occurs (#1).	*Useful*: Although the team's stability is at a reasonable level, it facilitates the use of external competencies.

Therefore, the study results discussed above enabled us to answer the research question – *Can Scrum as the key representative of agile software development approaches be adapted to the successful execution of 'ad-hoc' IT projects?* Indeed, Scrum can be adapted for such projects, even though tailor-fitting of Sprint process and project documentation is not straightforward.

Conducted empirical research addressed both a small and a large IT project, whereas medium projects might potentially require adjusting formulas to calculate the maximum Sprint time. Data collection took place during software development initiatives in a single European Union member state, while several studies confirmed that culture settings might be important [36],[37]. We consider this a clear limitation of this study. Therefore, replicating the study and comparing results between developed and emerging economies is a potential direction for future research and a measure to mitigate a threat to the external validity of the study. Finally, organization-specific contingency team reconstruction constraints should be developed and integrated into the proposed approach to prevent abuse of this feature.

7. Conclusions

The study revisited restrictive assumptions of the Scrum framework across organizations with the means to run agile IT projects in 'ad-hoc' settings. Findings allow several implications to be pointed out. To begin with, the adapted Scrum approach supports the accomplishment of software development projects by enthusiasts – who run such projects during their spare time, which is difficult to predict. A bulk of variations related to Sprinting result from fluctuations in team membership, and these are more common in volunteer-based projects like open-source framework development based on volunteer contributors. One reason is that there is no link between the success of a project and the remuneration of contributors. That said, both the viability of the far-reaching rebuild of a team during a Sprint and having two members ascribed for each PBI (instead of a single one) meet some challenges of contemporary companies. Not only does it address the need for the operational management of the intensive use of external contractors, but also greatly reduces the risk of derailing IT projects should a major change in a setting occur. Being equipped with a minimal amount of information about PBI that would allow taking over its realization (or supporting the main developer accountable) is vital when employees are let go or shifted to other duties. Even though scrutinized projects were fully realized in an agile manner, traditionally managed projects are also prone to the identified challenges of 'ad-hoc' conditions. Proposed

solutions might therefore also support human resources management within waterfall-based software development initiatives.

On top of that, the tailored framework provides the means for companies to increase the flexibility of multi-project employee allocation. In such conditions, challenges with planning Daily Scrums accumulate. Although the use of videoconferencing tools is optional under normal working conditions in non-dispersed organizations, developers involved in several parallel projects benefit from (1) the evolution of Daily Scrum towards 'Weekly Scrum'; (2) greater flexibility in planning Sprints while maintaining transparent scheduling rules; and (3) liberalized rules of attendance at meetings. Therefore, vital changes introduced to the generic Scrum's Sprint included:

- Providing custom formulas to calculate the maximum Sprint time depending on the average availability of Development Team members per week.
- Introducing asynchronous and synchronous virtual communication as the primary means of conducting project meetings with additional requirements regarding attendance and topic list distribution. Especially efficient asynchronous communication methods (dedicated chat/forum-based channels/groups/teams) are important due to potentially highly fluctuating teams with members available in disjoint time slots.
- Establishing a general rule for scheduling a future project meeting during the preceding meeting but no later than the following 20% of the total Sprint duration.
- Making the Scrum Master responsible for managing the Development Team tasks.
- Empowering the Scrum Master to make decisions regarding extending Sprints in case of minor deviations from the plan.
- Enabling PBI implementation to be moved to the next Sprints.
- Adjusting generic Scrum to project setting by allowing changes within the Development Team structure during a Sprint.
- Introducing daily Sprint Backlog analysis to assess the increase in the number of working hours for individual tasks and project team members and take adequate motivational and/or corrective actions.

The adapted approach also facilitates the involvement of external experts who, due to their primary contractual obligations, often have planning conflicts regarding meetings. Once again, such practices increase the organizational maturity of entities highly dependent on experts outside their payrolls and the adaptability of entire organizations to potential crises that thwart earlier planning arrangements.

Acknowledgments

The authors would like to thank Mr. Daniel Golicki for shedding light on the particularities of the class schedules management mobile app as well as the nuances of its delivery, which served as the background for the development of the pre-adjusted Scrum approach enhancements.

References

[1] U. Muhammad, T. Nazir, N. Muhammad et al., "Impact of agile management on project performance: Evidence from I.T sector of Pakistan," *PLOS One*, vol. *16*, no. *4*, *e0249311*, 2021.

[2] A. W. Cram, "Agile Development in Practice: Lessons from the Trenches," *Information Systems Management*, vol. *36*, no. *1*, pp. *2-14*, 2019.

[3] D. Martinez, X. Ferre, G. Guerrero and N. Juristo, "An Agile-Based Integrated Framework for Mobile Application Development Considering Ilities," *IEEE Access*, vol. *8*, pp. *72461-72470*, 2020.

[4] A. Srivastava, S. Bhardwaj and S. Saraswat, "Scrum Model for Agile Methodology," in *International Conference on Computing, Communication & Automation*, Greater Noida, India, 2017, pp. 864-869.

[5] A. Agrawal, M. A. Atiq and L. S. Maurya, "A Current Study on the Limitations of Agile Methods in Industry Using Secure Google Forms," *Procedia Computer Science*, vol. *78*, pp. *291-297*, 2016.

[6] F. Almeida and P. Carneiro, "Perceived Importance of Metrics for Agile Scrum Environments", *Information*, vol. *14*, no. *6*, 2023.

[7] F. M. Fowler, *Navigating Hybrid Scrum Environments: Understanding the Essentials, Avoiding the Pitfalls*. New York, NY, USA: Apress, 2019.

[8] A. Raza and H. Majeed, "Issues and Challenges in Scrum Implementation," *International Journal of Scientific & Engineering Research*," vol. *3*, no. *8*, pp. *1-4*, 2012.

[9] A. Martini and J. Bosch, "A Multiple Case Study of Continuous Architecting in Large Agile Companies: Current Gaps and the CAFFEA Framework," in *13th Working IEEE/IFIP Conference on Software Architecture*, Venice, Italy, 2016, pp. 1-10.

[10] A. Przybyłek and D. Kotecka, "Making Agile Retrospectives More Awesome," in *Federated Conference on Computer Science and Information Systems*, Prague, Czechia, 2017, pp. 1211-1216.

[11] V. T. Heikkilä, M. Paasivaara, C. Lasssenius, D. Damian and C. Engblom, "Managing the Requirements Flow from Strategy to Release in Large-Scale Agile Development: A Case Study at Ericsson," *Empirical Software Engineering*, vol. *22*, pp. *2892-2936*, 2017.

[12] M. Laanti, "Implementing Program Model with Agile Principles in a Large Software Development Organization," in *32nd Annual IEEE International Computer Software and Applications Conference*, Turku, Finland, 2008, pp. 1383-1391.

[13] D. Batra, W. Xia, D. VanderMeer and K. Dutta, "Balancing Agile and Structured Development Approaches to Successfully Manage Large Distributed Software Projects: A Case Study from the Cruise Line Industry," *Communications of the Association for Information Systems*, vol. *27*, pp. *379-394*, 2010.

[14] K. Siau and M. Ling, "Mobile Collaboration Support for Virtual Teams: The Case of Virtual Information Systems Development Teams," *Journal of Database Management*, vol. *28*, no. *3*, pp. *48-69*, 2017.

[15] S. Lee and H. S. Yong, "Distributed Agile: Project Management in a Global Environment," *Empirical Software Engineering*, vol. *15*, no. *2*, pp. *204-217*, 2010.

[16] M. Kuciapski, "How the Type of Job Position Influences Technology Acceptance: A Study of Employees' Intention to Use Mobile Technologies for Knowledge Transfer," *IEEE Access*, vol. *7*, pp. *177397-177413*, 2019.

[17] N. Cerpa and J. M. Verner, "Why Did Your Project Fail?," *Communications of the ACM*, vol. *52*, no. *12*, pp. *130-134*, 2009.

[18] M. Senapathi and M. L. Drury-Grogan, "Refining a Model for Sustained Usage of Agile Methodologies," *Journal of Systems and Software*, vol. *132*, pp. *298-316*, 2017.

[19] G. Borrego, A. L. Morán, R. R. Palacio, A. Vizcaíno and F. O. García, "Towards a Reduction in Architectural Knowledge Vaporization during Agile Global Software Development," *Information and Software Technology*, vol. *112*, pp. *68-82*, 2019.

[20] P. Rodríguez, M. Mäntylä, M. Oivo, L. E. Lwakatare, P. Seppänen and P. Kuvaja, "Advances in Using Agile and Lean Processes for Software Development," *Advances in Computers*, vol. *113*, pp. *135-224*, 2019.

[21] B. Marcinkowski and B. Gawin, "A Study on the Adaptive Approach to Technology-Driven Enhancement of Multi-Scenario Business Processes," *Information Technology & People*, vol. *32*, pp. *118-146*, 2019.

[22] Digital.ai. (2022, December). *The 16th Annual State of Agile Report* [Online]. Available: https://info.digital.ai/rs/981-LQX-968/images/AR-SA-2022-16th-Annual-State-Of-Agile-Report.pdf

[23] K. Schwaber and J. Sutherland. (2020, November). *The Definitive Guide to Scrum: The Rules of the Game* [Online]. Available: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

[24] V. P. Eloranta, K. Koskimies and T. Mikkonen, "Exploring ScrumBut – an Empirical Study of Scrum Anti-Patterns," *Information and Software Technology*, vol. *74*, pp. *194-203*, 2016.

[25] G. Kiely, J. Kiely and C. Nolan, "Scaling Agile Methods to Process Improvement Projects: A Global Virtual Team Case Study," in *23rd Americas Conference on Information Systems*, Boston, MA, USA, 2017, paper 13.

[26] M. Hron and N. Obwegeser, "Scrum in Practice: An Overview of Scrum Adaptations," in *Hawaii International Conference on System Sciences*, Waikoloa Village, HI, USA, 2018, pp. 5445-5454.

[27] J. F. Tripp and D. J. Armstrong, "Agile Methodologies: Organizational Adoption Motives, Tailoring, and Performance," *Journal of Computer Information Systems*, vol. *58*, no. *2*, pp. 170-179, 2018.

[28] T. Stålhane, T. Myklebust and G. K. Hanssen, "The Application of Safe Scrum to IEC 61508 Certifiable Software," in *11th International Probabilistic Safety Assessment and Management Conference and the Annual European Safety and Reliability Conference*, Helsinki, Finland, 2012, pp. 6052-6061.

[29] K. H. Rolland, V. Mikkelsen and A. Næss, "Tailoring Agile in the Large: Experience and Reflections from a Large-Scale Agile Software Development Project," *Lecture Notes in Business Information Processing*, vol. *251*, pp. *244-251*, 2016.

[30] E. Caballero and J. A. Calvo-Manzano, "A Practical Approach to Project Management in a Very Small Company," *Communications in Computer and Information Science*, vol. *301*, pp. *319-329*, 2012.

[31] M. L. Drury-Grogan, K. Conboy and T. Acton, "Examining Decision Characteristics & Challenges for Agile Software Development," *Journal of Systems and Software*, vol. *131*, pp. *248-265*, 2017.

[32] M. Tanner and A. Mackinnon, "Sources of Interruptions Experienced During a Scrum Sprint," *The Electronic Journal of Information Systems Evaluation*, vol. *18*, pp. *3-18*, 2015.

[33] O. Adelakun, R. Garcia, T. Tabaka and R. Ismail, "Hybrid Project Management: Agile with Discipline," in *International Conference on Information Resources Management*, Santiago de Chile, Chile, 2017, paper 14.

[34] J. Reiff, and D. Schlegel, "Hybrid project management – a systematic literature review," *International Journal of Information Systems and Project Management*, vol. *10*, no. *2*, article *4*, 2022.

[35] K. Remington and J. Pollack, *Tools for Complex Projects*. Abingdon, UK: Routledge, 2008.

[36] K. Conboy and N. Carroll, "Implementing Large-Scale Agile Frameworks: Challenges and Recommendations", *IEEE Software*, vol. *36*, no. *2*, pp. *44-50*, 2019.

[37] R. Vinaja, "The Scrum Culture: Introducing Agile Methods in Organizations," *Journal of Global Information Technology Management*, vol. *22*, no. *4*, pp. *309-311*, 2019.

[38] M. Logemann, J. Aritz, P. Cardon, S. Swartz, T. Elhaddaoui, K. Getchell … and J. Stapp, "Standing strong amid a pandemic: How a global online team project stands up to the public health crisis," *British Journal of Education Technology*, vol. *53*, no. *3*, pp. *577-592*, 2022.

[39] T. E. Havstorm, and F. Karlsson, "Software developers reasoning behind adoption and use of software development methods – a systematic literature review," *International Journal of Information Systems and Project Management*, vol. *11*, no. *2*, article *4*, 2023.

[40] V. Stray, R. Florea and L. Paruch, "Exploring human factors of the agile software tester," *Software Quality Journa*l, vol. *30* no. *2*, pp. *455–481*, 2021.

[41] A. Joskowski, A. Przybylek and B. Marcinkowski "Scaling Scrum with a customized nexus framework: a report from a joint industry-academia research project," *Software-Practice & Experience*, vol. *53*, no. *7*, pp. *1525-1542*, 2023.

[42] A. Alami, P. A. Nielsen and A. Wąsowski, "A Tailored Participatory Action Research for FOSS Communities," *Empirical Software Engineering*, vol. *25*, pp. *3639-3670*, 2020.

[43] M. Grabowski, J. Madej and J. Trąbka, "Using IT to Improve Efficiency of Polish Courts: An Action Research Study," in *20th Americas Conference on Information Systems*, Savannah, GA, USA, 2014, paper 5.

[44] K. Proudfoot, "Inductive/Deductive Hybrid Thematic Analysis in Mixed Methods Research," *Journal of Mixed Methods Research*, vol. *17*, no. *3*, pp. *308–326*, 2023.

[45] Y. A. Harb, C. Noteboom and S. Sarnikar, "Evaluating Project Characteristics for Selecting the Best-Fit Agile Software Development Methodology: A Teaching Case," *Journal of the Midwest Association for Information Systems*, vol. *1*, pp. *33-52*, 2015.

[46] N. Tekin, M. Yilmaz and P. Clarke, "A novel approach for visualization, monitoring, and control techniques for Scrum metric planning using the analytic hierarchy process," *Journal of Software: Evolution and Process*, e2420, 2021.

Appendix A. Acronyms

AM	Agile Methods
AR	Action Research
GVT	Global Virtual Teams
PBI	Product Backlog Items
SAP	Systems Applications and Products in Data Processing
SDM	Software Development Method
SID	Distribution Information System
TDD	Test-Driven Development
WOA	Web-Oriented Architecture

Appendix B. Action research (AR) cycles' surveys

Survey I. Ratification of the components of the Scrum approach

1. Which project management activities do you consider the most beneficial?
2. Which project management activities are the least beneficial for you?
3. What other activities would you integrate into the project execution process to make it more efficient?

Survey II. Validation of the adapted Scrum approach

1. Are Sprint planning mechanisms appropriate?
2. Do the formulas for calculating the duration of a Sprint allow for precise measurement?
3. Does the lack of a Daily Scrum have a negative impact on project performance?
4. Do virtual synchronous and asynchronous communication solutions enable effective communication in the project?
5. Does the possibility of increasing the length of Sprints harm the efficiency of project execution?
6. Does discontinuing the practice obliging the members of a Development Team to participate in only one project have a negative impact on project execution?
7. Does the monitoring system detect exceptions?
8. Are project templates suitable for carrying out project tasks?

Survey III. The quality of the final product developed via an adapted Scrum approach

1. How would you rate the completeness and consistency of the application's functionality?
2. How would you rate the quality of the application in terms of readability of the layout, consistency of navigation mechanisms, intuitiveness of interaction with the user interface, as well as user-friendliness of solutions used for data visualization?

Biographical notes

Michal Kuciapski
Michał Kuciapski, Ph.D., is an Associate Professor at the University of Gdansk. He was a member of its e-learning council from 2006 to 2016. Former secretary of PLAIS – the Polish Chapter of the Association for Information Systems. His main research areas involve agile project management, mobile and cloud technologies acceptance and adoption for knowledge transfer, as well as the digital transformation of Supply Chain Management processes. In this regard, he has authored over fifty publications as journal articles, conference papers, and book chapters. He combines scientific work with business practice as a senior software engineer. He has participated in over twenty scientific and implementation projects, both national and international.

Bartosz Marcinkowski
Bartosz Marcinkowski, Ph.D., is an employee of the University of Gdansk and the Polish-Japanese Academy of Information Technology, currently engaged in research on the development of IT solutions in globalizing enterprises as well as IT-supported energy efficiency management. Author of several best-selling books on Systems Analysis and Design (SAND) published on the Polish market. He is also an expert in the field of computer network administration and an instructor at numerous professional training ventures devoted to the development of IT competences. His professional skills are confirmed by international certificates, inter alia: OMG-Certified UML Professional, OMG-Certified Expert in BPM, Cisco Certified Network Associate Instructor Trainer, and ITIL V3.