76th International Astronautical Congress (IAC 2025), Sydney, Australia, 29 Sep-3 Oct 2025.

IAC-25,D3,3,7,x101831

Best Practices Tailored for Research Teams Handling Diverse Projects in Parallel

Suditi Chand*, Stefanie Bremer, Jaspar Meister

Department of Relativistic Modelling, Institute for Satellite Geodesy and Inertial Sensing, German Aerospace Centre (DLR), Germany

* Corresponding Author

Abstract

Established technology management (TM) methodologies need to be tailored for small research teams with multiple projects in parallel. This paper aims to demonstrate some customized agile practices that our team adapted for our diverse research goals and software development in the last four years. Henceforth, the practices, tools and methods are categorized into automation, collaboration, visualization and management.

For instance, Scrum is an agile method intended for a single project. But despite the challenges or obstructions in the earlier stages of adapting Scrum, it has shown success in our team of six people working on four to five different projects. Additionally, for one research software project a separate Scrum board was needed with traceability from the primary board. This helped create an agile work environment with improved efficiency and tackling of challenges that earlier went unnoticed until the later stages of the work.

For project version control and database management Gitlab is used. In Gitlab - milestones, issue boards, iterations and performance charts - are useful for TM and performance measurement and management (PMM). Moreover, the Gitlab CI/CD pipelines help automate and improve the speed, efficiency and accuracy of the research software toolchain being developed. Moreover, Unified Modeling language diagrams, Architectural Decision Records (ADRs) and design decision meetings have helped across the different phases of the projects. It is also important to show which challenges were faced in the implementation and which trade-offs had to be made. Concluding, all key aspects, conventional or tailored, that helped us in our diverse team with limited time and human resources, will be presented in this paper.

As a result, in all our projects some key facets of TM have improved. These include planning, milestones tracking, cross-functional collaboration and feedback, development and operations (DevOps), re-prioritization and PMM. It also helped us in knowledge management for the faster onboarding of new developers as well as recording the tacit knowledge of team members. This was achieved with sprint retrospective logs, mind maps, established workflows. In summary, despite hindrances we were able to customize conventional TM methods for the improved working of a diverse research team.

Keywords: Technology Management, Scrum, Small Multi-project Team Management, Research Project Management, Team Collaboration, Multi-tasking using Agile

Acronyms/Abbreviations

Architectural Design Records (ADRs), Continuous Integration and Continuous Delivery (CI/CD), Knowledge Management (KM), Performance Measurement and Management (PMM), Product Owner (PO), Project Management (PM), Research and Development (R&D), Technology Management (TM), Unified Modeling language (UML).

1. Introduction

This paper aims to explore how and which best practices from industry can be used for small to medium research teams with multiple-projects and changing requirements or resources. We will present the benefits and challenges that our team faced on adopting some of the agile methodologies and tools. We will begin with the definitions of the terms of management relevant to our study, namely, Technology Management (TM), Performance Measurement and Management (PMM), Scrum, DevOps and Continuous Integration and Continuous Delivery (CI/CD).

Our team of six researchers and two students adopted Scrum in June 2023 and have conducted a total of 25 sprints until August, 2025. We adopted DevOps, with a focus on CI/CD, for two projects since 2022. When we started with Scrum, we were five core employees working on two internal and four external projects. Currently, we are working on six external and three internal projects and continue to use the agile-waterfall hybrid of management practices to adapt them to the projects.

One of the standard definitions of TM was defined as early as 1987 by the National Research Council of the USA [1]. It states that “Management of technology links engineering, science, and management disciplines to plan, develop, and implement technological capabilities to shape and accomplish the strategic and operational objectives of an organization.” This includes the feasibility study and future relevance of the technology

being developed, which is also much required for the space industry.

An agile practice used for TM is Scrum. Scrum is a time-boxing, iterative and incremental practice for small teams working on a single project where work is split into tasks achievable in a given sprint of periods between 2 to 6 weeks [2] [3]. Five events define the Scrum cycle: *sprint planning* sets the sprint goal and selects backlog items; the *sprint* where developers are allowed to work uninterrupted on the planned backlog; the *daily scrum* synchronizes the team and surfaces impediments; the *sprint review* lets stakeholders and the team inspect the increment and give feedback; and the *sprint retrospective* enables the team to reflect on its process and define improvements. Scrum's artifacts provide structure and visibility. The *product backlog* is an ordered list of all desired work, continually refined; the *sprint backlog* contains the subset of backlog items selected for the current sprint. Furthermore, Scrum defines three roles. The Product Owner is responsible for the product backlog, continuously prioritizing items and ensuring the team works on the most important tasks. The Scrum Master acts as a servant-leader, removing impediments, coaching the team in Scrum practices, and protecting the team from external disruptions. The Development Team, a self-organizing, cross-functional group plans, builds, tests, and delivers the increment.

DevOps was originally formed for the management of software products, where development and operations workflows were integrated and streamlined for the fast deployment of software releases or products [4]. It constitutes of methods for agile development, version-management, security and quality control as well as CI/CD. CI/CD is the automation of pipelines for the build, test, documentation and deployment of released packages.

PMM has been defined in several ways as per the application field. Lebas [5] argued in his paper that performance measurement and management cannot be separated. Performance management is essential to define what key parameters, both qualitative and quantitative, can be defined for the given project or firm/institute to assess the progress. It paves the way for performance measurement in the later phases. For example, a project can be assessed based on its monetary profits, efficiency, accuracy, adaptability, applicability or productivity.

The above-mentioned methodologies and tools cannot be homogenized and used by everyone. In the field of research and development (R&D) with limited time, human resources, certainty or changing requirements, one needs to be open to move from traditional methods to new ones [6] [7] or practice the best of both based on the relevance to the team and their feedback [8]. In the following sections, we will briefly explain our motivation and difference in TM and PPM in R&D (sec. 2), the best practices and tools adopted to address our needs (sec. 3) followed by the results (sec. 4) and concluded with a brief summary (sec 5).

2. Motivation

Research for applied sciences also requires iterative development and testing akin to the software industry. Many projects start with the final goal or research question but with minimal or no definitive requirements or solutions that can be reviewed at the end of each milestone. Alternatively, even if definite deliverables are defined for each milestone, there are changing requirements or design solutions due to a new discovery along the experimental or development cycle. There are tools present in the waterfall project management (PM) methodology, such as, Risk Analysis and Quality Assurance that are used in R&D to ascertain any critical obstacles. Ideally, if caught early, there are methods in PM to implement changes in the project or its timeline to adapt to the challenges faced. But when this does not happen, it is too late or costly to re-design or terminate the project [6].

Moreover, with no incremental steps, it is common in research teams to prematurely invest a lot of time and planning into methods or features that are never implemented. This happens more frequently for small to medium teams with multiple projects and feasibility studies occurring simultaneously and long gaps between formal reviews in the project. Other consequences of multi-project teams are constant interruptions and meetings that affects the efficiency of an individual in multi-tasking [9].

Collaboration and knowledge exchange also profit from agile TM and PM tools. As the team becomes fragmented into different projects and groups, it is difficult without a structured workflow to pool team expertise on one platform for problem-solving or task delegation. In a monthly Scrum, the team meets frequently for a small meeting to discuss if the planned tasks are progressing or if someone is stuck. Such meetings are enough to convey what general expertise is needed. Teammates volunteer for one-on-one meetings if they can offer help to find a solution or share their experience. Research institutes can also benefit from such meetings in terms of project ideas for inter-disciplinary research and novel solutions as they are more aware of the team's knowledge pool. Moreover, collaboration tools like Git or SVN are useful for version control and parallel development.

Automation tools are also powerful elements of novel TM and software management that are most vital to save time and human resources for research teams. This allows for more robust development and more time for feature advancement. Moreover, automated documentation and archiving is required for better knowledge management (KM) with a varying

configuration of students, researchers and employees working on multi-year projects.

Lastly, certain visualisation tools such as Unified Modeling language (UML) or Scrum/Kanban boards are useful for visualizing complex design problems or research questions. They can help researchers break complex and complicated goals into small sections that provide steps for near-term, incremental and/or modular development [3]. It also helps scientists stuck in negotiation/planning cycles with project partners who cannot reach a consensus on a common solution to proceed with, due to misinterpretation of data or information conveyed over meetings or emails. It can be used to effectively showcase the work being done to stakeholders.

Why is TM, PM or PMM not the same for small to medium research teams?

Industries like automation, software or pharmaceuticals have dedicated roles for project managers in the different fields of product performance, customer relations or resource management. They are hired specially with the relevant managerial degrees or experience. But in typical science or technical research institutions or universities, projects are managed by experienced scientists or researchers with some on-boarding training on PM and prior experience in managing research teams. This is not a disadvantage as they have tacit knowledge and experience needed for the highly-specialized field of work. Nonetheless, many do not adapt inter-disciplinary management techniques for their teams without a lot of initial resistance or apprehension as they have limited resources and no obligation to do so. Hence, a primary reason why standard TM and PMM practices differ in R&D is the scepticism and the lack of willingness to invest time in them if not explicitly required by the organisation management.

Another reason is that while in industry the results of implementation of new practices is visible in terms of monetary or time profits, in research projects it takes time to perceive the benefits. Thus, many fail to see the reason for investing time and personnel upfront on changing to new managerial infrastructure or tools.

In the case of motivated teams who want to make changes, the specific processes and goals of the targeted tools of agile or TM are not directly executable or achievable. For instance, Scrum is meant for teams of up to ten members who want to deliver a single product in incremental fashion in a short time cycle [2]. Whereas, research teams often are working on multiple projects at the same time. If they decide to use Scrum for each project, they would spend a considerate amount of time on meetings rather than actual work. Additionally, Scrum has defined roles of Product Owner, Scrum Master, Stakeholders and Developers that might not always fit a research project. Marchenko and Abrahamsson [10] studied the implementation of and changes in Scrum in a 20-persons team working simultaneously on multiple R&D projects. Their results are similar to our experience detailed in Section 4.

Hence, research managers need to also conduct an additional initial planning to cherry-pick and customise the definitions of the different elements or processes of a methodology. Traditional organisations face several challenges in adapting agile practices due to conflicts in development, management, personnel or business factors or processes. Boehm and Turner [3] explain these in details and provide a list of issues that are relevant for R&D entities in any field. While the founders of Scrum argue that it needs to be adopted in its entirety or else it is not Scrum, this is not possible in research. Literature also proves that Scrum has been adapted for different team structures and multiple-projects in the past and yet its benefits have been reaped [3] [6] [9].

3. Best Practices and Tools Adopted by Us

A retrospect of the different areas in which our team benefited from adapting new tools and methods, as summarised in sec. 2, allowed us to categorize the best practices in this section into four fields where we saw most progress namely, automation, management, visualisation and collaboration.

3.1 Automation

All research projects in applied sciences require software development for proof-of-concept or data analysis. The software environment and infrastructure need to be configured to different settings or for different versions of the codebase. For instance, automation of virtual environment set-up using ‘uv’ or ‘poetry’ libraries saved us a lot of time for programming in Python. While all the code is set-up remotely in an automatic version control system, GitLab, it also helps to automate the build, test, documentation and release of research software using GitLab CI/CD as shown in Figure *1* for the release of the VENQS desktop application for the packaging and distribution of the orbit simulation and satellite test mass dynamics software library developed by us.

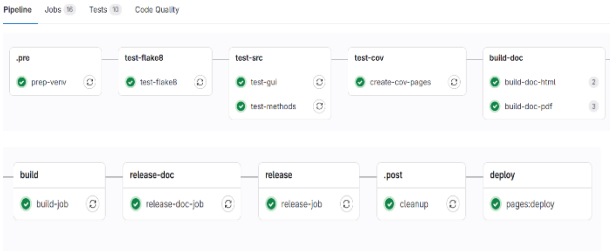

Figure 1 Gitlab Pipelines dashboard for the VENQS application release

Infrastructure-as-code is found to be useful for the configuration of the environment settings using scripts. Using a '. pre' stage in the CI/CD pipelines, each time a pipeline is triggered, the required infrastructure is also re-created virtually in the active GitLab Runner (virtual machine). Automation helps to streamline processes and detect software bugs in a timely manner, saving both time and personnel costs.

Another advantage is the automatic generation of documents, job artifacts and bug reports in each pipeline. This ensures good TM as well as PMM as the progress of the project and its performance can be seen in these reports, pipeline statuses and test reports. Automatic documentation is not fully achievable as an expert is required to write the research-based part of the manuals and design records, but once the content is pushed to GitLab it can be formatted and compiled automatically in a PDF or HTML file [11].

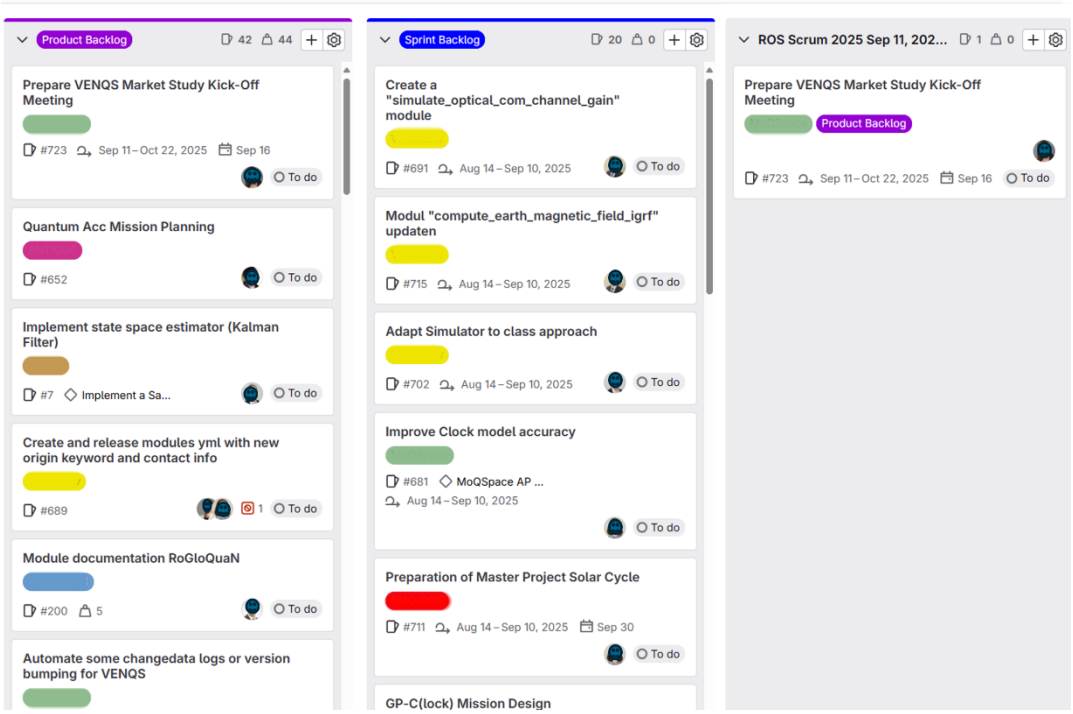

Figure 2 Sprint Planning board of the team where each issue has a unique-coloured label for a project and another for Scrum

3.2 Management

Our team adopted Scrum in June 2023 and have conducted a total of 25 sprints until August, 2025 for six third-party projects and three internal projects. Most sprints were of four weeks with a few exceptions of five/six weeks to accommodate holiday seasons or absences due to conferences. Since we already used GitLab for the projects, we used the following elements of it for implementing Scrum:

- Labels – Each issue is tagged with the label 'Product Backlog'. Once it is assigned to a sprint its label changes to 'Sprint Backlog' or 'Sprint Task' based on how big or small its list of Definition of Done (DoDs) is. Moreover, each project has its own label.
- Assignee: It was important to only add one main assignee in each issue as the primary responsible person for that task.
- Iteration: Each Sprint time period is saved as one iteration and all child items of that sprint are assigned with it. GitLab creates automatically a 'Burn-down Chart' for each iteration based on open issues versus closed ones.
- Epics and Issues: Epics are used for a complex task and its incremental steps are created into smaller child issues.
- Milestones: Milestones are used for tracking planned work packages or milestones of a project based on traditional PM plans which are part of the standard kick-off documents for most R&D projects.
- GitLab Boards: Two boards are used – one for sprint planning and one for the current sprint as shown in Figure 2 respectively.

Lastly, it is very helpful to have only general Product backlogs with some DoDs in the group sprint board. Team members create and link issues with more detailed work steps from their project repository to the backlog issues in the central group Scrum boards. This ensures that there is as much transparency in individual work as the confidentiality level of the project allows.

The first sprint was only for a week as it was used to plan how Scrum might work best for our team. It was crucial to discuss which rules of Scrum defined our practice and what were let go. Firstly, it was decided to not assign the Scrum roles dogmatically. In some external projects where only one group member was involved, it was difficult to define a separate Scrum master, Product Owner (PO) and Developer. Hence, the group leader offered to be the Scrum Master for all the projects in the context of sprints while the responsible members were in effect both the PO and developer of their projects. If there were multiple persons involved then the primary responsible acted more as the PO and created Product backlogs and the other members were the developers. In internal projects, the group leader was also the PO representing all the Stakeholders involved. At the end of the first sprint, a list of all projects was created with their PO, Developers, Customers and Stakeholders assigned as logic dictated.

It was decided to follow the traditional waterfall PM documents of the research projects for longer goals (of two to four years) but try to break them down to incremental and achievable backlogs to be tackled sequentially or iteratively in monthly sprints. Hence, GitLab Milestones are used for the former and Issues, Iterations and Boards for the latter. Lastly, another process decision made in the first sprint was to have the daily sprints only twice a week and increase the frequency only if deemed necessary.

As the sprints progressed, the sprint retrospective has been useful to recognize good practices that we should continue and to tackle challenges faced in using Scrum. A perusal of the documentation of the last 25 sprints shows that everyone faced some challenges in working agile in traditional research-intensive projects or breaking down complex work to smaller monthly tasks. But this improved significantly after four to five sprints thereby improving multi-tasking and enabling faster task-switching. However, it is very easy to lose focus over time and commit less to adapting work to the Scrum processes. This may cause the aforementioned benefits to reduce for individuals or the whole team.

Other management tools used by us include DevOps for research software that helped automate pipelines (discussed in sec. 3.1) and adapt Scrum in iterative software development. But this is not implemented for all projects. The implementation of CI/CD for a large software infrastructure was also assessed in a qualitative performance metrics. This metrics showed large improvement of criticality and frequency of certain errors in the build and release process of scientific software modules. Hence, this helped us to document some PMM that is done in the team and affirm the efficacy of re-using it in similar projects in the future.

3.3 Visualisation

Visualisation in the context of R&D is the use of graphical formats to represent data, entities, processes, tasks, concepts or comparative/objective performance measurement. While it is common knowledge for researchers to use visualisation tools, we aim to describe here how the new and old practices combined helped us to improve in this area. For the TM tasks like planning, tracking, development and review of projects it is very useful to view Gitlab Sprint Boards and milestones with linked items. Moreover, knowledge management and sharing are facilitated with visual ease of Gitlab dashboards where one can filter backlogs according to assignees and/or labels to view who is working on or worked on what types of tasks.

Another best practice taken from the software industry is the documentation of design decisions in Architectural Design Records (ADRs). These documents reflect the problem context and what was the decision made in a minimalistic way with simple screenshots or graphs if needed. Whereas, software design document and review reports were used for more elaborate information with formal UML diagrams or other graphs/plots. Automating documentation workflows via CI/CD motivated people to document more. Hence, there is more active documentation of research code, which we hope to sustain over project lifecycles. A desirable by-product of consistent documentation of knowledge/data with visualisation techniques is the faster integration of new employees in research projects and workflows.

3.4 Collaboration

Schwaber and Sutherland [2] defined five Scrum values of commitment, focus, openness, respect and courage. They are also essential in bolstering the team spirit and collaboration. Although theoretically sound, their presence in actual groups depends on the personality of its leader as well as members. A prime example of this is that we were only able to implement and customise Scrum in our group due to these values of our leader to switch to new management techniques. It was not easy to deal with the overhead work of Scrum and incorporating the feedback of the group with diverse personalities and work culture preferences. But it got easier with time and saved her a lot more time on planning and management than anticipated. Moreover, it would not have worked for over two years without her commitment along with the commitment, focus, openness and respect of all the members to her and to each other.

The bi-weekly daily sprints, sprint review and sprint retrospectives provide frequent opportunities for the team to come together and listen to or help the individuals, as well as, assess the sprint goals or future effectiveness of Scrum. Collaboration was made clear and concise by using certain visualization tools as mentioned previously. For instance, desktop/online

applications that serve as digital whiteboards or software add-ons to make UMLs via scripts make drawing of ideas/solutions quicker over online meetings. Even in offline meetings these apps are useful to record the results of the meeting online so that they are accessible and editable in the future from anywhere. These graphics are sometimes easier to understand than formal charts for an introduction to new members.

Another best practice that helps in effective team work and delegation is to allot one or two entire days for a large task that involves the whole team. We organised a documentation day a few times to standardize documentation of research software and implement parallelly in many affected science library modules. While not everything planned on such days carries forward to the follow-on work it definitely helps to give a kick-start to complex and/or dormant tasks. Domain-based skills are tricky to share in Scrum framework but each colleague working on a software for a considerable time gives an introductory tutorial on how to use it to the whole team. This also helps colleagues in the future if they have to work with the same software as they can revisit the tutorial files and/or demo projects stored in a central place in the department's remote file system.

Design decision meetings help for software development where teammates, who do not work in the same project, participate as mock users to give feedback. It is natural for such events to get more difficult to organise with everyone with an increasing number of projects or team size. But they are useful for the timely handling of collective tasks that are important and get postponed unknowingly until they become critical.

4. Results

The overall agile practices and supporting software tools helped us in structuring the near-term work for long-term project goals. Most of the projects last from 18 months to 3 years. Collaboration and meetings improved as the agile structuring and time-boxing dictated their frequency, duration and agenda. A feedback survey of the benefits of the overall practices was done which resulted in a qualitative performance metrics graph as shown in Figure 3 with an aggregate for responses for each entry for the different aspects of work that were influenced by the new tools and practices. The response was recorded from each teammate for how they benefited as an individual employee and how they perceived improvements in the team dynamics. The group leader responded to values on how she felt the group progressed with the new tools and how she as a group leader benefitted from the changes. The results show that:

1. All three – the group, its individuals and the group leader, saw a very good to excellent improvement in (a) visualisation and tracking of work across projects and people and (b) automation of pipelines, work quality and delivery time.
2. Project and technology management also improved significantly for all three entities.
3. The group leader benefitted the most in collaboration. Everyone also valued highly the importance of Scrum and meetings for knowledge sharing and problem solving.
4. Resource management - handling personnel, costs and time, improved also very well for the group and group leader but for individuals it is understandably not a big change from earlier.
5. Adaptation to changes received an average rating of good, making it the field with the least improvement in the polls.
6. The group leader of multi-project single teams definitely came out as the clear winner of adapting the best practices. While she had to implement those practices and had the most overhead work, over the couple of years the profits were also most visible for her.

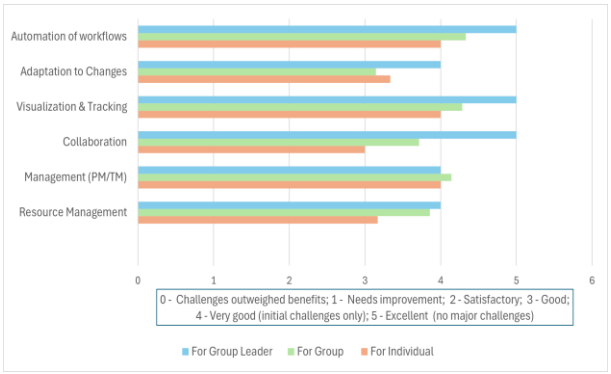

Figure 3 Qualitative performance metrics of using agile best practices for the last three years in a small team of 7 members

4.1 DevOps

DevOps and its practice of CI/CD helped in automation, visualisation, management and collaboration in project focussed on research software development and release. They help reduce risks/errors and make the software safer and more robust. The duration and efficiency of the package release operations also improved for a small team. Earlier the release of a software package took two weeks or more if the release operations and testing was done by one to two employees. After using CI/CD it reduced to a range of 2 to 3 days. The qualitative performance matrix of using DevOps in a large software codebase affirmed our intuition to re-use it in future projects.

Other elements of DevOps such as version control and agile management are also implemented successfully with GitLab and Scrum. The various GitLab dashboards,

job artifacts and test coverage reports help to visualise the project performance of research software.

4.2 Scrum

Scrum helped to increase efficiency, transparency and accountability in the team. Additionally, it helped us to incorporate the group goals in the monthly sprint retrospectives. This resulted in producing results that are more accountable in the institute's key performance indexes (for instance, more published research work). Work planning and delegation for reviews or milestone meetings also improved with Scrum boards and advanced creation of product backlogs. Hence, Scrum contributes to the PMM of the group and projects indirectly rather than directly, like the direct assessment we did for DevOps. But in the future, we would like to use performance measurement techniques for assessing the use of Scrum in our group.

It helped individuals to break down complex problems and tackle it incrementally, especially for those who struggled with daily work breakdown and time allocation for big projects. When an employee returns from a long absence, sprint boards and backlogs are very helpful to re-orient them to near-term tasks and resume work. Communication also improved amongst team members on sharing current work and challenges. For some projects the benefits of using Scrum were visible from the beginning and for others it seemed counter-intuitive to spend time in planning and sharing updates or problems with other team mates who did not work on the project. Eventually, the responsible colleagues for such projects recognised the use of Scrum for better management or evaluation of their work and next steps. As the team became more fluent in Scrum it was easier to plan realistic backlogs for a single sprint and carefully estimate time budgets. It was realized that this planning required separate sprint planning of each project with only its developers and the Scrum Master. The next day, the sprint retrospective and final group sprint planning were done together with all the group members. This helped to avoid having long project-specific discussions in the group sprint planning.

For larger teams, Marchenko and Abrahamsson [10] highlight the diverse challenges in cross-functional involvement of a team in projects where each member has his/her domain knowledge expertise and is conditioned to traditional PM practices.

Occasionally, team members tend to fall lax in finishing backlogs in the sprint timeframe. This is resolved by active help from the Scrum Master by monitoring delays in the sprint backlogs in the daily sprint meetings and by adapting those DoDs or backlogs if needed. Other recurring challenges in implementing Scrum for individuals are – time-budgeting, prioritization, tunnel-vision, writing clear backlogs, over-planning, losing morale if a backlog unravels into many obstacles, and the right use of deadlines and weights for backlogs and tasks.

In the case of research institutions, it is common to have the sprint interrupted with project proposals, publishing research work, review cycles or some work for the institute. Hence, some buffer time should be kept in each sprint for such unprecedented tasks. Also, it should be avoided to have backlogs without an assigned person as they tend to be overlooked. Hence, one key lesson learned is that implementing and using Scrum over the long haul requires commitment, focus, active retrospectives and adaptation from all members of the sprint team. These are the primary issues and benefits that were observed in our small group.

5. Conclusions

Our team of six research scientists/engineers and two students has benefited from adopting Scrum, DevOps and Gitlab PM tools to plan, prioritize and track progress in five or more projects at the same time. Some colleagues even use the new practices in personal GitLab projects to organise and track their personal research or administration tasks that did not fall into any projects. While estimating time and planning work in incremental steps improved for most team mates, it is also easy to fall back into old ways and not follow the Scrum processes efficiently.

Automation using GitLab CI/CD for customised pipelines saved a significant time in doing the work manually or in debugging the manual errors that are now completely circumvented with scripts. Visualization with GitLab boards as well as UML diagrams, simple screenshots and documentation helped in team work, online meetings, knowledge management and PMM. Overall, we would recommend other small to medium sized research groups working on diverse research topics and projects to try these best practices as it fits the group and individuals of the team.

References

[1] N. R. C. NRC, Management of Technology: The Hidden Competitive Advantage, Washington, DC: National Academies Press, 1987.

[2] K. Schwaber and J. Sutherland, The Scrum Guide, 2020.

[3] B. Boehm and R. Turner, "Management Challenges to Implementing Agile Processes in Traditional Development Organizations," *IEEE Software*, vol. 22, p. 30–39, September 2005.

[4] L. Leite, C. Rocha, F. Kon, D. Milojicic and P. Meirelles, "A Survey of DevOps Concepts and Challenges," *ACM Computing Surveys*, vol. 52, p. 1–35, November 2019.

[5] M. J. Lebas, "Performance measurement and performance management," *International Journal of Production Economics*, vol. 41, p. 23–35, October 1995.

[6] L. Rising and N. S. Janoff, "The Scrum software development process for small teams," *IEEE Software*, vol. 17, p. 26–32, 2000.

[7] M. de Bayser, L. G. Azevedo and R. Cerqueira, "ResearchOps: The case for DevOps in scientific applications," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015.

[8] M. Ccallo and A. Quispe-Quispe, *Adoption and Adaptation of CI/CD Practices in Very Small Software Development Entities: A Systematic Literature Review*, arXiv, 2024.

[9] C. J. Stettina and M. N. W. Smit, "Team Portfolio Scrum: An Action Research on Multitasking in Multi-project Scrum Teams," in *Agile Processes, in Software Engineering, and Extreme Programming*, Springer International Publishing, 2016, p. 79–91.

[10] A. Marchenko and P. Abrahamsson, "Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges," in *Agile 2008 Conference*, 2008.

[11] R. K. V K, J. Ukko, T. Rantala and M. Saunila, "The value of novel technologies in context to performance measurement and management: A systematic review and future research directions," *Data and Information Management*, vol. 8, p. 100054, March 2024.

[12] S. S. Sebrek, A. G. L. Romme and Z. T. Kosztyán, "How to create dynamic capabilities: A design science study," *Technovation*, vol. 143, p. 103204, May 2025.