Lightweight Software Process Improvement using Productivity and Sustainability Improvement Planning (PSIP)

Michael A. Heroux[1], Elsa Gonsiorowski[*2], Rinku Gupta[*3], Reed Milewicz[*1], J. David Moulton[*4], Gregory R. Watson[*5], Jim Willenbring[*1], Richard J Zamora[*∘6], and Elaine M. Raybourn[*†1]

[1] Sandia National Laboratories, Albuquerque, NM USA
[2] Lawrence Livermore National Laboratory, Livermore, CA
[3] Argonne National Laboratory, Lemont, IL USA
[4] Los Alamos National Laboratory, Los Alamos, NM USA
[5] Oak Ridge National Laboratory, Oak Ridge, TN USA
[6] NVIDIA, Santa Clara, CA USA

Abstract. Productivity and Sustainability Improvement Planning (PSIP) is a lightweight, iterative workflow that allows software development teams to identify development bottlenecks and track progress to overcome them. In this paper, we present an overview of PSIP and how it compares to other software process improvement (SPI) methodologies, and provide two case studies that describe how the use of PSIP lead to successful improvements in team effectiveness and efficiency.

Keywords: software development, software engineering, software process improvement

1 Introduction

The Department of Energy (DOE) Exascale Computing Project (ECP) provides an opportunity to advance computational science and engineering (CSE) through high performance computing (HPC). Central to the project is the development of next-generation applications that can fully exploit emerging architectures for optimal performance and provide high-fidelity, multiphysics, and multiscale capabilities. At the heart of the effort is the need to improve developer productivity, positively impacting product quality, development time, staffing resources, and software sustainability, reducing the cost of maintaining, sustaining, and evolving software capabilities in the future.

∗: These authors contributed equally to this work.
∘: Author's work was conducted while employed by Argonne National Laboratory.
†: Corresponding author.

This paper presents the Productivity and Sustainability Improvement Planning (PSIP) process: a lightweight, iterative workflow where teams identify their most urgent software development and sustainability bottlenecks and track progress on work to overcome them. Through the PSIP process, teams are able to realize process improvements, without a huge disruption to any current development processes.

PSIP is best understood as an instantiation of the Plan-Do-Check-Act management cycle (PDCA, also known as Plan-Do-Study-Adjust), first described by Deming in 1950 [5], which provides the intellectual foundation for much of the modern process literature (including SPI methods and Agile as a philosophy) and is itself a translation of the scientific method to the study of work process. It is both a method to be used by an individual or with a team, and a method to be taught to a team, with the team driving the improvement process. One PSIP process output is the development of Progress Tracking Cards (PTC), brief, shareable documents containing the target, or goal of the planning activity, title of the topic of improvement, and a step-by-step list of activities or outcomes that incrementally lead to improvements in team effectiveness and efficiency. Through the PSIP process, teams are able to realize process improvements, without a huge disruption to any current development processes.

The contributions of this paper are:

- We compare PSIP with other software process improvement (SPI) methodologies
- We present the PSIP process, a lightweight software process improvement framework.
- We discuss two case studies where a PSIP cycle was implemented by an existing software development team with the help of a facilitator.

2 Background

Techniques for software process improvement (SPI) aim to assess, design, and realize effective development processes. SPI has been the subject of an extensive body of literature, with "classic" SPI frameworks such as CMM(I) or SPICE dating back three decades [15][4][7]; this trend can be traced back even further to the "software crisis" starting in the late 1960s [12] and the realization that an intentional process of creating software was as important to software development as software itself or, as Osterweil put it, "software processes are software too" [14].

Within the context of scientific software research and development, SPI is an undeveloped territory. The most recent notable mention came in 2015; Mesh et al. reported plans to develop a Scientific SPI Framework (SciSPIF) by cataloging common decision factors and project characteristics and their relationships to software engineering practices, ideally realized as an online self-assessment tool [13]. Further work in this area is needed. In 2006, Baxter et al. observed that a lack of basic knowledge about development processes was a significant challenge for the scientific software community [2], but recent studies

suggest the situation is now more nuanced. A 2019 survey of scientific software developers by Eisty et al. found that respondents not only saw value in software process, but even "[preferred] using a defined software development process" over ad hoc approaches [6]. This may be a reaction to the increasing importance of software in science: a 2018 survey by Pinto et al. found that 8 out of 10 researchers reported spending "more time" or "much more time" developing software than they did 10 years ago[17]. We argue that the challenge moving forward is to cultivate ways of doing software work that align with the needs of the scientific software community.

Scientific software teams are typically focused on obtaining scientific results from the software they write. Funding is for generating results, not software. This is a competitive process and teams cannot usually expend much time or effort outside of writing the software features that support generating new results. Therefore, any productivity or sustainability improvements must be incremental and integrated into the primary feature development process. Bug fix or refactoring releases are rare.

In our experience, scientific software teams typically have little or modest formal software engineering training. They may be aware of formal terminology such as software lifecycle, requirements elicitation and technical debt, but often have incomplete or incorrect understanding. Furthermore, these teams have an inherent skepticism about formal, heavyweight approaches that might significantly delay their current scientific activities or require large investments before seeing benefits.

In this context, a lightweight, adaptable, iterative and informal approach to improving developer productivity and software sustainability is necessary. A comprehensive review by Kuhrmann et al. found that from 1989 to 2016 an average of 11 new or customized SPI methods were introduced per year [11]. One hypothesis put forward by the authors is that process improvement is a context-specific activity and that methods must be adapted to their context in order to be successful. This trend towards adaptation echoes recent studies that suggest most software companies today use individualized development processes that are hybrids of different methodologies and philosophies [10][20]. Along these lines, PSIP is neither wholly new nor unsupported but is instead a tailored approach that builds upon previous SPI approaches; likewise, it is not all-encompassing but is instead a lightweight toolkit that can be incorporated into existing CSE software development workflows.

3 Methodology

The Interoperable Design of Extreme-scale Application Software (IDEAS-ECP) [8] conducts research on topics of developer productivity and software sustainability in CSE domain. Its PSIP efforts focus on methodologies for improving productivity and sustainability by working with CSE software teams to identify opportunities to iteratively and incrementally improve software team practices and processes.

The objectives of the PSIP process are to capture and convey the practices, processes, policies, and tools of a given software project. The PSIP workflow is intended to be lightweight and fit within a project's planning and development process. It is not meant to be an assessment or evaluation tool. Instead PSIP captures the tacit, more subjective aspects of team collaboration, workflow planning, and progress tracking. Additionally, in the potential absence of planning and development processes, and as scientific software teams scale to larger, more diverse, aggregate teams, unforeseen disruptions or inefficiencies can often impede productivity and innovation [8]. PSIPs are designed to bootstrap aggregate team capabilities into best practices, introduce the application of appropriate resources, and encourage teams to adopt a culture of process improvement.

Fig. 1: The Productivity and Sustainability Planning (PSIP) cycle.

At its core, the PSIP framework is an iterative, incremental, repeatable, cyclic process for improvement planning. The cyclic nature of the PSIP process enables software development teams to improve overall project quality and achieve science goals by encouraging frequent iteration and reflection. The multi-step, iterative PSIP Workflow is described below (see Figure 1). Beginning at the start located at the left of Figure 1, software teams may work through these steps on their own, or with the assistance of a PSIP facilitator:

1. *Summarize Current Project Practices*: The first phase involves briefly documenting current project practices. It is important to record the original state of the project to both provide a baseline for measuring progress and to help identify areas that are ready for improvement. We find it important to use plain language when defining current project practices to reduce

misunderstanding of software engineering terms that might be vaguely or incorrectly understood by the software team, whose member may not be formally trained in software engineering concepts.

2. *Set Goals*: Completing this step typically brings to light project practices that can benefit from a focused improvement effort. Although any number of goals may be identified in this step, a limited set is selected at any given time to best impact the project and are achievable within a predictable span of time (a few weeks to a few months). Goals not chosen at this time may be tabled for future iterations.
3. *Construct a Progress Tracking Card (PTC)*: Recall that a PTC is a brief document containing the target, or goal of the planning activity, title of the topic of improvement, and a step-by-step list of activities or outcomes that incrementally lead to improvements in team effectiveness and efficiency. Each practice will have its own PTC. Teams may select PTCs from the PTC catalog[7]; or define their own PTC or modify PTCs found in the catalog. The purpose of the PTC is to help teams set and achieve improvement goals. The PTC is not a tool for external assessment or comparison with other projects. In fact, since PTCs are custom-designed for each project, comparisons are typically not possible.
4. *Record Current PTC Values*: In order to establish baseline capabilities and track progress, teams record the initial values (0-5 are suggested) for each PTC.
5. *Create a Practice Improvement Plan*: In order to increase the values in a PTC (corresponding to improvements in software productivity and sustainability), teams develop a plan to reach a higher value for each PTC.
6. *Execute the Plan*: Team efforts are focused on improving the selected practices described in the PTC. At first, teams may see a slowdown, as they work to start or improve a given practice. It is possible for teams to use complementary SPI methods in executing their plan. The slowdown in most cases is proportional to the amount of change, but ideally teams should see steady progress on a weekly basis after the initial phase and be able to complete execution of a particular practice improvement within a few months.
7. *Assess Progress*: During execution, teams assess, and determine the rate of progress each week. They adjust their strategy for success if needed. If progress is delayed too long, teams usually start the next PSIP iteration.
8. *Repeat*: The PSIP process is iterative. Continual process improvement is a valuable attribute for any software project. The PSIP process may be used to guide improvement planning within software projects and across aggregate projects.

During a PSIP process or at its conclusion, teams may elect to share their PSIP PTCs, best practices, and/or lessons learned with other teams. Teams may share their results with the community in a variety of ways including contributing blog posts on PSIP progress to the Better Scientific Software (BSSw)

[7] https://github.com/bssw-psip/ptc-catalog/

website[8], presenting lessons learned in the HPC Best Practices for HPC Software Developers webinar series[9], and by modifying, curating, or creating tools such as new PSIP templates, PTCs, or resources for inclusion in the PSIP catalog[10].

4 Related Work

In this section, we situate PSIP in the context of SPI methods. PSIP breaks from classic first wave SPI (such as CMM(I) or SPICE, or non-software methods translated to software like ISO 9000 or Six Sigma), in that it trades comprehensive standards and certification-driven assessment for self-defined, internally-driven goals. It does, however, carry forward first wave ideas such as having staged models of improvement like CMM(I), in the form of progress tracking cards. Additionally, PSIP is more aligned with numerous second wave SPI approaches which incorporate lean and agile thinking into their methods; it adopts their emphasis on iterative improvement and continuous learning.

PSIP is best understood as an instantiation of the Plan-Do-Check-Act management cycle (PDCA, also known as Plan-Do-Study-Adjust), first described by Deming in 1950 [5], which provides the intellectual foundation for much of the modern process literature (including SPI methods and Agile as a philosophy) and is itself a translation of the scientific method to the study of work process. It is both a method to be used by an individual or with a team, and a method to be taught to a team, with the team driving the improvement process. This situates PSIP within a constellation of bottom-up, inductive SPI methods as characterized by Stojanov 2016 [19]. These include QIP [1], AINSI [3], LMPAF^2 [18], and iFLAP [16]. Each of these methods emphasizes collaborative discovery of goals, tailored improvement solutions, and is designed to be lightweight to support small, resource-constrained organizations. However, they differ in their approaches.

QIP and AINSI (an implementation of QIP) differ from PSIP in that both use a Goal-Questions-Metrics approach to create measures for goals whereas PSIP utilizes progress tracking cards (PTC) that express either quantitatively- or qualitatively-defined goal states. PSIP, like LMPAF^2, may utilize a moderator to conduct interviews in order to construct documentation that captures team's practices, and draw out targets for process improvement. However, LMPAF^2 focuses specifically on maintenance activities and relies on frequent feedback sessions with the moderator to track progress. Finally, iFLAP conducts interviews with individuals drawn from across an organization and triangulates improvement activities by comparing different interview results (as opposed to a group interview), and the assessor(s) are responsible for creating a prioritization scheme for improvement targets (PSIP has no analogous concept).

[8] https://bssw.io

[9] https://bssw.io/events/best-practices-for-hpc-software-developers-webinar-series

[10] See https://github.com/bssw-psip/ptc-catalog and https://github.com/bssw-psip/practice-guides for further elaboration

5 Case Studies

In this section, we present two case studies where an existing software development team collaborated with a facilitator to work through the PSIP process.

5.1 EXAALT

The Exascale Atomistics for Accuracy, Length and Time (EXAALT) project[11] is a part of the Chemistry and Materials Applications area of the Exascale Computing Project. It is a materials modeling framework designed to leverage extreme-scale parallelism to produce accelerated molecular dynamics simulations for fusion and fission energy materials challenges. The official team comprises approximately 10 researchers at Los Alamos National Laboratory (LANL) and Sandia National Laboratories (SNL) working on four sub-projects. While two of these sub-projects are driven by a handful of people, the others are larger open-source efforts with many external contributors.

The EXAALT team was keen to utilize PSIP to improve their software engineering practices, particularly in the area of continuous integration (CI). Since the project integrates four distinct software packages, each with its own list of dependencies, the team frequently struggled with build regressions in the early days of development. After a few informal discussions with the authors, the team agreed that it would be necessary to (1) improve their end-to-end build system, (2) implement a CI pipeline to automatically detect build regressions, and (3) add unit/regression testing to the CI pipeline.

Although the team had not committed to an explicit project-management process at the early stages of the collaboration, the steps taken during these discussions correspond to the first two steps of the PSIP cycle shown in Figure 1. In order to prioritize their efforts, it was critical to clarify the current project practices and specify both near and long-term goals.

For the initial stage of the implementation of an automated end-to-end build system, the PSIP process was only used implicitly for project planning and execution. For the two remaining goals, however, PSIP was followed explicitly using the PTCs shown in Figure 2 (in summarized form). During steps 3-4 of the PSIP cycle, these PTCs were both fully annotated, but reflected a "score" of zero. For step 5 of the PSIP cycle, each PTC step was resolved in both Jira and GitLab as distinct stories and issues, respectively. The actual implementation of these Jira/GitLab issues corresponded to step 6 of the PSIP cycle, and the following assessment of the completed work was the final step.

[11] https://www.exascaleproject.org/project/exaalt-molecular-dynamics-at-the-exascale-materials-science

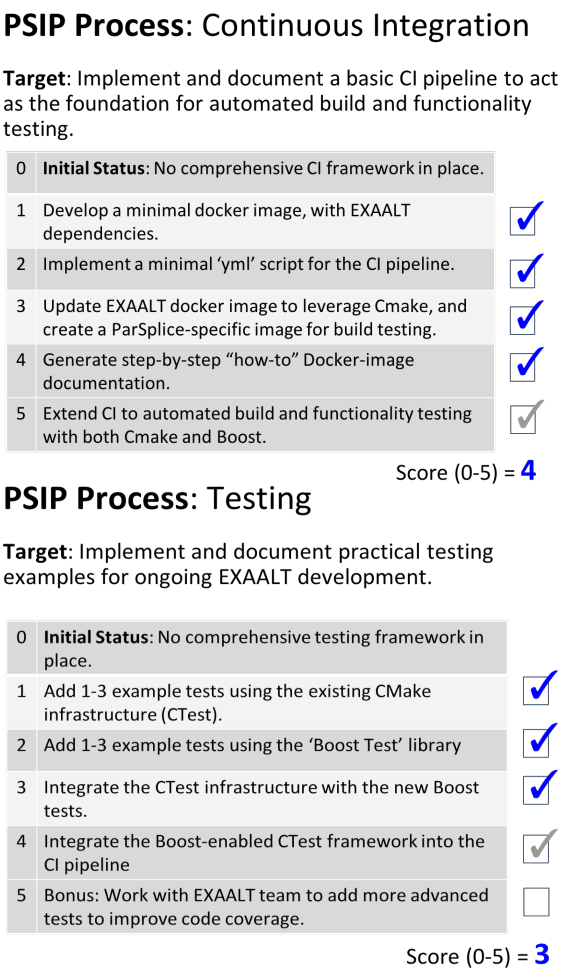

Fig. 2: Summarized versions of PSIP PTCs used for the EXAALT-IDEAS collaboration. The specific scores in the figure correspond to the state of the project. Note that some details about dependencies and timeline are excluded from the PSIP cards for clarity.

The completion of these cards does not mean that the EXAALT team is finished improving their CI and/or testing infrastructure. Like most aspects of software engineering, PSIP is an iterative process, and the initial plan may need to change if unexpected roadblocks emerge. Whether or not a progress tracking card can be followed to completion, documenting, revising, and repeating the process, makes sense when a natural finishing point is reached. The PTC used in this effort (see Figure 2) is available in the PSIP PTC catalog.

At this stage, the EXAALT team members have successfully adopted a minimal CI framework and are ready to apply the PSIP process to improve their CI pipeline further. The current plan is to modify the existing infrastructure to interface with ECP-supported facilities (e.g., Argonne Leadership Computing Facility and Oak Ridge Leadership Computing Facility). In addition, they are applying the PSIP process to further improve test coverage, specifically in the area of statistical tests for non-deterministic components and task management.

5.2 Exascale MPI

The Exascale MPI (Message Passing Interface) project consists of team members from Argonne National Laboratory. The project focuses on developing a production-ready, high-performance MPI implementation that scales to the largest supercomputers in the world. One particular challenge for the Exascale MPI project is a continuous influx of new contributors within the project. These new contributors are expected to already have technical expertise with MPI or learn these skills on-the-fly as needed by the job. While the team does provide some mentoring to new members, limited resources require that newcomers be fairly independent and proactive when it comes to learning the basic technical aspects of MPI.

The Exascale MPI team worked with a facilitator to implement the PSIP process. Through the documentation of current practices, the need to improve the project's onboarding processes was identified. Both the Exascale MPI team and the facilitator agreed to work on a PSIP cycle focused around improving the training for team contributors. This would take the form of a single destination resource containing all the required training material that could be provided to new team members during the onboarding process.

Once the overall goal was identified, the Exascale MPI team, together with the facilitator, identified key aspects of a satisfactory solution. These aspects included the need for:

- A central "repository" for all training material, relevant to the Exascale MPI team.
- Visually interesting, and easy navigation across all topics.
- Easy administration and ability to update the "repository" sustainably.
- Open collaboration to allow external contributors to contribute new technical topics and resources.

With the overall goal and desired outcome defined, the Exascale MPI team worked with the facilitator to create a timeline based on the resources available, followed by the creation of a PTC. Figure 3 shows a snapshot of the PTC created for this PSIP activity. Each step in the PTC, in this case, serves as an important checklist step to move towards the desired goal. As mentioned, these PTC cards are live entities and they may change depending on unexpected progress or bottlenecks.

Once the PTC was created, the team focused on the execution aspect of the PTC. The PSIP process aims to engage a full team through the execution of PTC

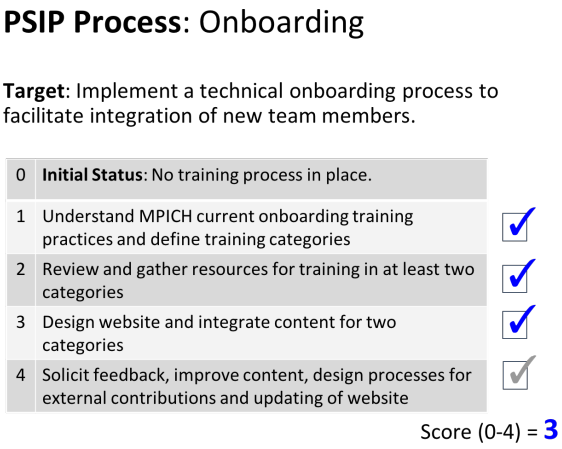

Fig. 3: Progress Tracking Card snapshot for Exascale MPI team, including date of completion. Row 0 indicates the team's status at the beginning of implementing the PSIP process.

steps. Each step is approachable, yet builds towards a larger goal. Throughout the PSIP process, the Exascale MPI team, with help from the facilitator, continually evaluated their progress on the path to building a resource for improving onboarding and training.

For improving the training process, the teams identified what categories of topics needed to be covered in the onboarding training. For each category, the team worked and solicited resources (based on the potential expertise level of new onboarding members), reviewed the material for accuracy and applicability, and worked on the design to integrate them into the training website/portal[12]. The Exascale MPI team, with help from the facilitator, did explore the viability of using existing cloud repository services (e.g., Google Drive). In the end, the team decided to design a custom stand-alone website to serve as a training portal, based on their needs and input from the team members. The training-base portal is a continual work-in-progress by the Exascale MPI team and can be a resource for the entire HPC community. At this stage, the Exascale MPI team is testing the training portal with their new hires and soliciting feedback. The next step for this PSIP is to create a plan to improve the training portal, which will focus on adding new content categories and establishing processes to sustain the content and its validity.

6 Discussion and Future Work

With the Exascale MPI PSIP process, we learned that PSIP topics developed for a particular team may sometimes be generalizable enough to be relevant and

[12] Exascale Onboarding training portal: https://sites.google.com/view/hpc-training-base/home

important to several teams in an organization or across multiple organizations. The topic of technical onboarding training for new hires may sometimes be team-specific, however most teams working in the HPC field need to train people in the common practices of the community. Thus, a PTC created for one team (as in the case of the Exascale MPI team) can be used by many other teams as a starting point. We also realized that not only the PTC, but also the resulting output (as in the case of the training portal) could end up being immensely useful across many other teams as well. The PTC used in this effort (see Figure 3) is available in the PSIP PTC catalog.

For EXAALT, we learned that one significant advantage of the PSIP management approach is that it forces the team to specify the 4-6 steps needed to reach a given goal. In this case, the process helped formulate the actionable items needed to lay the foundation for CI within the existing EXAALT software repository. Although PSIP can be used to manage the goals of any software project, the specific details of each step are highly dependent on the project. For example, different projects will most likely need to work with slightly different technologies to build a practical CI pipeline. Specific details will depend on where and how the source code repository is organized, as well as the limitations/capabilities of the existing library dependencies. For EXAALT, this process required careful discussion between teams in order to determine the key technologies to use.

In summary, PSIP is a lightweight adaptable framework for iterative and incremental improvement, applicable to any CSE software project, regardless of how software is developed and used. PSIP is easy to learn, especially for scientists who cannot dedicate time and resources to more formal or heavyweight approaches.

PSIP is meant to provide a mechanism to set goals collaboratively, get team buy-in, and enable periodic status checks to ensure the goals and execution are aligned. In some cases, teams may want to utilize a facilitator. This person may augment the PSIP by bringing process experience and objectivity to the effort, coaching the team on improving effectiveness and efficiency.

PSIP does have limitations. It is not as quantitative as other tools (not designed that way) and will possibly seem trivial to professional software engineering teams. It is presently best applied in research software settings and untested in large enterprises with product deliverables. Finally, no empirical research on PSIP available currently, only a few case studies.

Presently, PSIP improvements will likely come from further experience using it. We already know that one of the challenges of PSIP is making sure that progress on the topic of a PTC is not blocked by some prerequisite impediment that must be addressed first. Other needs include more documentation to support conducting PSIP without a facilitator to improve scalable application of PSIP. We also need to grow our PTC catalog and conduct more research on the use of PSIP with open source software teams, and large enterprises. Finally, PSIP does not address the issue of teams not being rewarded for efforts to improve developer productivity and software sustainability. In order for PSIP

to be broadly effective, the CSE community must prioritize the value of these improvements, something that we observe is happening slowly.

7 Conclusion

In this work we introduced PSIP, a lightweight, iterative SPI framework and method best understood as a Plan-Do-Check-Act management cycle. Drawing upon a well-supported foundation of software improvement theory and practice, PSIP was developed to help CSE software teams, and specifically HPC teams achieve software process maturity, an answer to the call by the National Strategic Computing Initiative for "a portfolio of new approaches to dramatically increase productivity in the development and use of parallel HPC applications" [9]. We provided two case studies where the PSIP cycle was implemented by an existing scientific software development teams with the help of a facilitator. Beyond scientific computing, we hope that our study of SPI methods and their use outside of conventional software development environments will inform and drive further innovation in the domain of software processes and methodologies.

8 Acknowledgements

Special thanks to Lois McInnes (ANL) and the members of IDEAS-ECP. Thanks to PSIP partners Danny Perez (LANL), Art Voter (LANL), Christoph Junhans (LANL), and Pavan Balaji (ANL). Images used by permission.

This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), Office of Biological and Environmental Research (BER), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2019-9693 C.

References

1. Basili, V.R., Caldiera, G.: Improve soft-ware quality by reusing knowledge and experience. Sloan management review **37**, 55–55 (1995)
2. Baxter, S.M., Day, S.W., Fetrow, J.S., Reisinger, S.J.: Scientific software development is not an oxymoron. PLoS Computational Biology **2**(9), e87 (2006)
3. Briand, L., El Emam, K., Melo, W.L.: Ansi–an inductive method for software process improvement: Concrete steps and guidelines (1995)

4. Chrissis, M.B., Konrad, M., Shrum, S.: CMMI guidlines for process integration and product improvement. Addison-Wesley Longman Publishing Co., Inc. (2003)
5. Deming, W.E.: Elementary principles of the statistical control of quality: a series of lectures. Nippon Kegaku Gijutsu Remmei (1950)
6. Eisty, N.U., Thiruvathukal, G.K., Carver, J.C.: Use of software process in research software development: A survey. In: Proceedings of the Evaluation and Assessment on Software Engineering. pp. 276–282. ACM (2019)
7. Emam, K.E., Melo, W., Drouin, J.N.: SPICE: The theory and practice of software process improvement and capability determination. IEEE Computer Society Press (1997)
8. Heroux, M., McInnes, L., Bernholdt, D., Gamblin, T., Marques, O., Moulton, D., Norris, B., Raybourn, E.M., et al.: Developer productivity and software sustainability report: Advancing scientific productivity through better scientific software (September 2018)
9. Holdren, J.P., Donovan, S.: National strategic computing initiative strategic plan. Tech. rep., National Strategic Computing Initiative Executive Council (2016)
10. Klünder, J., Hebig, R., Tell, P., Kuhrmann, M., Nakatumba-Nabende, J., Heldal, R., Krusche, S., Fazal-Baqaie, M., Felderer, M., Bocco, M.F.G., et al.: Catching up with method and process practice: An industry-informed baseline for researchers. In: Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice. pp. 255–264. IEEE Press (2019)
11. Kuhrmann, M., Diebold, P., Münch, J.: Software process improvement: a systematic mapping study on the state of the art. PeerJ Computer Science **2**, e62 (2016)
12. McIlroy, M.: Software engineering: Report on a conference sponsored by the nato science committee. In: NATO Software Engineering Conference, NATO Scientific Affairs Division. pp. 138–155 (1968)
13. Mesh, E.S.: Supporting scientific se process improvement. In: Proceedings of the 37th International Conference on Software Engineering-Volume 2. pp. 923–926. IEEE Press (2015)
14. Osterweil, L.: Software processes are software too. icse'87: Proceedings of the 9th international conference on software engineering. monterey (1987)
15. Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V.: Capability maturity model, version 1.1. IEEE software **10**(4), 18–27 (1993)
16. Pettersson, F., Ivarsson, M., Gorschek, T., Öhman, P.: A practitioner's guide to light weight software process assessment and improvement planning. Journal of Systems and Software **81**(6), 972–995 (2008)
17. Pinto, G., Wiese, I., Dias, L.F.: How do scientists develop scientific software? an external replication. In: 25th International Conference on Software Analysis, Evolution and Reengineering, SANER 2018, Campobasso, Italy, March 20-23, 2018. pp. 582–591 (2018)
18. Stojanov, Z., Dobrilovic, D.: Learning in software process assessment based on feedback sessions outputs. Information Technology and Development of Education (ITRO) 2015 p. 259 (2015)
19. Stojanov, Z.: Inductive approaches in software process assessment. In: International Conference on Applied Internet and Information Technologies (2016)
20. Tell, P., Klünder, J., Küpper, S., Raffo, D., MacDonell, S.G., Münch, J., Pfahl, D., Linssen, O., Kuhrmann, M.: What are hybrid development methods made of?: an evidence-based characterization. In: Proceedings of the International Conference on Software and System Processes. pp. 105–114. IEEE Press (2019)