Exchanging Best Practices for Supporting Computational and Data-Intensive Research, The Xpert Network

Parinaz Barakhshan
Rudolf Eigenmann
parinazb@udel.edu
eigenman@udel.edu
University of Delaware
Newark, Delaware, USA

ABSTRACT

We present best practices for professionals who support computational and data-intensive (CDI) research projects. The practices resulted from the Xpert Network activities, an initiative that brings together major NSF-funded projects for advanced cyberinfrastructure, national projects, and university teams that include individuals or groups of such professionals. Additionally, our recommendations are based on years of experience building multidisciplinary applications and teaching computing to scientists. This paper focuses particularly on practices that differ from those in a general software engineering context. This paper also describes the Xpert Network initiative where participants exchange best practices, tools, successes, challenges, and general information about their activities, leading to increased productivity, efficiency, and coordination in the ever-growing community of scientists that use computational and data-intensive research methods.

CCS CONCEPTS

• **Software and its engineering → Programming teams**; **Software development methods**.

KEYWORDS

Computational and data-intensive (CDI) research, Xpert Network, CDI research support, Best practices, Collaboration in CDI research

ACM Reference Format:
Parinaz Barakhshan and Rudolf Eigenmann. 2022. Exchanging Best Practices for Supporting Computational and Data-Intensive Research, The Xpert Network. In *Practice and Experience in Advanced Research Computing (PEARC '22), July 10–14, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3491418.3530293

1 INTRODUCTION

1.1 Motivation and Key Contributions

This paper and its underlying project were motivated by the fact that there appears to be little exchange of information between the many support groups for computational and data-intensive (CDI) research, which are of critical importance, as outlined in Section 1.2. In particular, knowledge of best practices is usually acquired by "training on the job" and not communicated to other projects. Furthermore, many domain science projects, due to budgetary constraints, use a single computational expert who has little to no access to peers and needs to self-train on such practices. We will argue that the practices exhibit differences from those in general software engineering contexts, which make their compilation and dissemination especially relevant. Filling this knowledge gap is the key opportunity and contribution of the present paper. We describe the initial outcomes of an initiative that brings together CDI support professionals, which we refer to as CDI experts or, short, Xperts. There are many related terms, such as research software engineers (RSEs), research facilitators, or research programmers. Despite possible differences, we consider the terms synonymous in this paper, as they all support CDI research. Bringing these professionals together through the Xpert Network leads to exchanging best practices and tools, discussing issues that arise in CDI research support, and desirable self coordination among the involved organizations. The present paper describes the outcomes of such activities.

1.2 Importance of Supporting Computational and Data-intensive (CDI) Research

The importance of CDI research is well evidenced. A relentlessly growing amount of computing power is consumed to conduct research using computational experimentation. Nobel prizes have been awarded to CDI science breakthroughs [7]. *Laszewski et al.* have shown that CDI work correlates with higher scientific impact [36]. *Apon et al.* have found that universities investing in computational infrastructure supporting CDI research tend to increase in ranking [1].

Supporting such CDI research by Xperts is critical, allowing domain scientists to focus on their research tasks. Recognizing this need, several large projects funded by the National Science Foundation (NSF) [26], such as XSEDE [43], CyVerse [6], and the NSF Software Institutes, support the community through groups of Xperts. National organizations have emerged, such as the US Research Software Engineer Association (US-RSE) [31] and Research Data Alliance United States (RDA-US) [16], that include Xperts for supporting CDI researchers. There are also a growing number of university IT departments, or research computing support groups, that include Xperts for boosting the productivity of

PEARC '22, July 10–14, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9161-0/22/07…$15.00
https://doi.org/10.1145/3491418.3530293

researchers on their campuses. Furthermore, many large domain-science projects include computational experts for supporting their own applications. The effectiveness of such computational support is remarkable. A study by *XSEDE's ECSS group* [9] has revealed that time invested by computational experts can have a four-fold return in terms of time a domain scientist would spend on the same tasks.

1.3 Structure and Activities of the Xpert Network

The Xpert Network [32] is an NSF-supported, community-serving initiative that offers

- Monthly online meetings (webinars and panels) to exchange best practices applied, tools used, and open problems faced in computational and data-intensive (CDI) research,
- Face-To-Face Meetings at major conferences, such as PEARC (Practice and Experience in Advanced Research Computing), ICS (International Conference on Supercomputing), and SC (Supercomputing) for in-depth discussions, and
- A website (sites.udel.edu/xpert-cdi) with access to recordings and reports of past events, a calendar of upcoming events, and community announcements.

Under preparation are also

- A discussion platform for community communication around the clock, and
- An exchange program that supports participants visiting other participants.

The ultimate goal of the initiative is to advance science by increasing the productivity of researchers who use CDI methods for pushing research frontiers. To this end, several communities are invited to participate in the Xpert Network activities, including (i) researchers developing and using CDI applications, (ii) those who assist these researchers with expertise in CDI technology and methods (Xperts, Facilitators, RSEs), (iii) domain experts/scientists on university campuses and other research organizations, and (iv) developers of tools that support the creation and use of CDI applications. These groups are invited to:

- Share success stories about how Xperts have assisted domain researchers,
- Present open problems and challenges faced when supporting CDI research,
- Exchange best practices that are being applied in their organization or in their research,
- Present tools and report on the use of tools that have made a substantial difference in supporting the work of Xperts and domain researchers,
- Coordinate activities that are of mutual interest.

1.4 Best Practices and Tools Guide

Two specific outcomes of the discussions of the Xpert Network are the creation of:

- A best-practices guide for computational and data-intensive research, and
- A catalog of tools that support CDI research and engineering.

They will serve as training material for new Xperts joining CDI support teams, for instructional events, and for university curricula. The present paper can be seen as a starting point for such educational material. This material is related to the large volume of best practices for software engineering and scientific programming that exists, but differs in the following significant ways.

While training material for general software engineering and scientific programming cover important skills that *Xperts* must have, we are looking for best practices specific to professionals that *support domain science teams*. We focus on information that is not commonly found in software engineering and scientific programming literature, or on advice that is unexpected. We also discuss skills that our participants have called out as especially relevant and those that need to be applied differently in a *CDI context*.

Since the Xpert Network initiative began three years ago, over 250 people have contributed to the knowledge base through over thirty webinars and three in-person events. Some of the in-person sessions held at major conferences include the ICS19 workshop, the PEARC19, and the SC19 Birds-of-a-feather sessions.

The Xpert Network's participants include major NSF-funded projects for advanced cyberinfrastructure e.g., the Extreme Science and Engineering Discovery Environment (XSEDE) [43], Extended Collaborative Support Services (ECSS) [9], Novel and Innovative Projects (NaIP) [23], The Molecular Sciences Software Institute (MolSSI) [18], Science Gateways Community Institute (SGCI) [30], Campus Research Computing Consortium (CarCC) [28], Virtual Residency (VR) [27], and Cyverse [6], as well as national organizations e.g., the US Research Software Engineer Association (US-RSE) [31], Research Data Alliance United States (RDA-US) [16], and campus organizations supporting CDI researchers.

Through Xpert network events, these efforts have showcased their projects, the types of support they provide to their users, their best practices for developing, debugging, parallelizing, and executing CDI applications, and the tools they have used. The present paper compiles that information. Our past work on several interdisciplinary projects with domain scientists has also contributed to identifying the best practices described in this paper.

Another important element of our approach is the inclusion of tools. Best practices need to be supported by tools that follow the underlying methodologies. Good tools for scientific software development can boost the productivity of domain science research as much as the best practices themselves. For space reasons, these tools will be discussed in a forthcoming paper, however.

The remainder of the paper is organized as follows. Section 2 describes the best practices that have emerged in the Xpert Network activities so far. Section 3 describes related efforts on best practices and creating synergy among Xpert groups, followed by conclusions in Section 4.

2 BEST PRACTICES

This section describes a set of best practices for professionals who support CDI research involving applications that are *compute-intensive* (spending most execution time on computational algorithms) and/or *data-intensive* (manipulating large volumes of data). The practices have been compiled from Xpert Network activities as described in section 1.4.

Two broad categories of best practices for supporting such research have emerged in the Xpert Network activities: (i) best practices related to project collaborations and (ii) best practices for software development. As mentioned in the introduction, the second category is covered well in the software engineering literature and in organizations focused on training, such as *Software Carpentry* [2].

This paper considers only those software development practices that have been described by participants as different from general software engineering practices or particularly important for CDI research. Of particular interest are practices that are *actionable*; we will highlight the recommended actions in each practice. In some cases, the action lies in *being aware* of something. The following subsection will begin with this category.

2.1 Diversity of Xpert Backgrounds

Be aware of different backgrounds Xperts may bring into a team; configure training so that less-familiar best practices can be acquired as needed. Members of *Xpert groups* may come from domain sciences, from computer science backgrounds, or from environments where they learned to assist *CDI* science teams through training on the job. For example, working with Unix commands, parallel programming models, and version control may be second nature to members with computer science backgrounds, whereas these very skills may be critical best practices to acquire by *Xperts* with domain-science background. Vice versa, *Xperts* who have a domain degree are often well aware of the terminology gap discussed in Section 2.5, which facilitates the communication with new science collaborators [10]. Training for Xperts should accommodate these differences, allowing the trainees to focus on best practices they are not familiar with.

2.2 Understanding the Academic Environment

Be aware of the academic reward system and activities to increase academic standing. An issue for *Xperts* that is essentially absent in a general software engineering context is the importance of understanding the academic environment. This issue is especially relevant for software engineers with background in industry, where hierarchical organization, and the overriding goal of creating a reliable product as rapidly as possible, are the norm. Understanding the academic reward system and the many side activities that researchers may get engaged in to maintain the academic standing of the science team [24, 25] will influence project decisions. In fact, some of the best practices need to be understood from this view point, such as the issues of documentation and testing, mentioned in Sections 2.12 and 2.11.

2.3 Breadth of Xpert Skills Needed

Prepare for skills needed beyond your current expertise, by networking with other Xperts. Modern CDI applications draw from a broad range of technologies, such as computing paradigms, programming languages, architectures, and algorithms.

What's more, this range keeps evolving. For example, new applications may demand expertise of machine learning techniques. Individual Xperts and those in small teams, serving a large CDI research community, will be unlikely to cover the needed skills. Besides, they may need to perform tasks across the entire software life cycle and be involved in project management. It is important to be able to recognize such situations and seek external advice. Maintaining contacts with other Xpert teams, who may be consulted when needed, is highly advisable.

2.4 Collaborative Assistance Between Xperts and Domain scientists

Help propel new projects though short-term, close collaboration with domain scientists. A recommended form of interaction between Xperts and CDI domain scientists is through *collaborative assistance*. For a period of one to several months, Xperts work side-by-side (physically or virtually) with the domain scientists whose project they support. During the collaboration, the Xpert takes care of computer-engineering issues that arise, while the domain researcher resolves the problems requiring application science expertise. This form of joint work allows issues that fall into the competency of the collaborator to be addressed immediately, significantly reducing, or even avoiding, the need for *domain researchers and Xperts* to get trained on each others' knowledge and skills at project begin. Over the course of the joint work, the collaborators tend to pick up sufficient knowledge of each others' skills and terminology. In particular, the domain scientist becomes familiar with the software development processes and tools, allowing them to continue the work independently. Vice versa, the *Xpert* will have acquired sufficient knowledge of the domain problem, allowing them to effectively provide help remotely, even after moving on to other projects. This model has been pioneered by *XSEDE's ECSS group* and was applied successfully by other Xpert teams [10].

2.5 Overcoming the Terminology Gap Between Computer and Domain Sciences

Carefully identify and resolve terminology gaps. Keep vocabulary to the essentials. Explain using many examples. The gap between computer-science and domain-science lingo is an often mentioned issue in Xpert Network discussions. The challenge can be big if the same term is used by both collaborators, but with different meanings. Participants reported significant confusions and even incorrect project executions due to this challenge. Awareness of the issue and patience in trying to understand the collaborators' viewpoints are critical. It was pointed out that the gap is wider than in a general software engineering context, when trying to understand a customer, as science terminology tends to use rich vocabularies that may complicate understanding the problem domain and project requirements. Keeping the vocabularies to the essentials and investing time to explain new terms is key to successful collaboration. Using many examples and frequent feedback from both sides will help bridge this gap.

2.6 Understanding the Domain Problem and Developing a Project Plan

Devote substantial time to understanding the domain problem, turning possibly vague ideas into a feasible solution approach, and developing a project plan. Many Xpert Network participants highlighted the importance of the first contact with

the supported domain scientists and the approach taken to understand problems and develop solutions. In addition to awareness of the terminology gap, the relevance of investing time in understanding the goals and the functional as well as non-functional requirements was emphasized. Showing patience in the process is critical. The domain researcher needs to be helped to transform an initially often vague idea into a concrete plan. Developing specific requirements for the computational application and the underlying system is essential. Devote sufficient time. One challenge for the Xpert is to introduce the researcher to the possibilities of the to-be-created or improved software implementation and the underlying system while introducing only a small number of new terms. Recommendations include the use of many concrete examples and an *inverted pyramid* approach. The latter describes a possible solution in initially very few, high-level terms, which are then successively refined in discussions.

The situation is again related to that of software engineers with their customers, two important differences being:

(1) the potentially large terminology gap, mentioned above, and
(2) the collaborative assistance situation, which often enables the inverted pyramid approach: As *Xpert* and domain researcher work side by side, refinements of high-level ideas can more easily happen over time, permitting the collaborators to invest their initial effort in defining the tip of the pyramid.

2.7 Prioritize Functional Requirements

Carefully vet all requirements by the application's end users and prioritize aggressively. The dilemma of a large number of desirable features but only a short project duration can be big in scientific software. Xperts need to learn to tell the difference between essential and desirable requirements users may have. Essential features should be strictly prioritized. Scientific software is also special in that its requirements can change rapidly as new insights into a research project emerge. Therefore, frequent reassessment and reprioritizing of the requirements are needed [4].

2.8 Issue Tracking

Track origin as well as implementation status of requirements and bug reports. Issue tracking is important in most mid-size and large-size software development. Two points make it a particular concern in scientific applications. The first is the belief that research projects – often of a three-year duration – will remain small and easy to oversee, obviating the need for issue tracking. The second is that developers (typically graduate students and post-doctoral researchers) tend to change often, losing important project memory. The reality is that, even within three years, remembering what feature was requested by whom and with what rationale, can be difficult. After a "change of guard", the same will become close to impossible. What's more, successful science applications may turn out to have a long life. Hence, being able to trace a feature request, its implementation status, and the reasons for accepting/rejecting the request to its origin can be key. Furthermore, having a clear record of who made what request can be critical for re-assessing and re-prioritizing functional requirements, mentioned previously.

Issue tracking tools, which maintain a list of tasks and sub-tasks to be performed, prevent duplicated efforts, and enable collaborative work. The tools support requirement traceability, that is, the association of an application update with the requirement that motivated it [38].

2.9 Source Code Management and Version Control

Make use of source code management and version control systems to track your software's evolution. An *Xpert* with software engineering background is well aware of the importance of software version control. At the same time, this importance needs to be advertised to domain experts, as pointed out strongly by the Xpert network participants.

Many successful science software applications had their origin in a *"toy program"* written by a graduate student. It got gradually expanded by several authors, caught the attention of a wide audience, and ended up becoming an important research tool. Without version tracking, the history (origin, authors, relationship of features, and specific extensions) may no longer be known, making it difficult to extend further and obtain needed documentation. Learning version control methods and tools will not only help overcome these difficulties; it will also increase the productivity of the software developer as soon as the application exceeds the boundaries of a small program. What's more, source code management tools greatly facilitate collaborative software development [3], enable software roll-back to a previous, well-defined state, and help developers start a new branch of the software. The latter can be important if, say, two graduate students want to add their own, separate feature sets.

2.10 Code Review

Xpert and domain scientist should review each other's software written. Code review – a second person reading newly written software – is one of the best ways to catch errors and also to improve the code. Despite its effectiveness, it is often ignored, unless strictly mandated, and hence warrants mentioning here. A number of additional reasons make this practice particularly useful for an Xpert-domain scientist relationship:

- The two collaborators have different backgrounds, increasing the chance of detecting an issue that eluded the other.
- The review is effective in verifying that requirement and implementation match.
- It is a good way of transferring the knowledge of what the Xpert did to the domain scientist, who will continue the work after the collaboration completes.
- The discussions happening during code review provide excellent material for documenting the code and project.

Code review helps improve the science as well as the code itself. Research software can benefit from code reviews, as much as industrial software.

2.11 Software Testing

Define test cases that the application and its components must pass before you begin their implementation. Testing is another issue that is covered well in the software engineering

literature. The topic has been mentioned often in our Xpert discussions, deserving a place in this list of best practices. Next to raising awareness of the need and the benefit of testing, a key point is that testing must not be an afterthought; one must avoid thinking of test cases only when the software is close to completion. Instead, testing should be considered in the design phase. In the initial communication with the domain researcher, the Xpert needs to ask the question "what test cases should be passed by the application once it is completed?" Coming up with a thorough set of such cases will not only facilitate the later testing phase; it will also help clarify the specific capabilities that need to be implemented. The implementation of features that do not satisfy any test case can be postponed or even avoided altogether. Designing test cases can go hand in hand with functional requirement prioritization (2.7).

2.12 Documentation

Document your project to ensure long-term success, reproducibility, and obtain proper credit for your work. Software engineering teachings cover extensively the fact that well documented programs are essential for software maintenance and extensibility. Yet, academic software is notorious for the lack of documentation, as pointed out forcefully by *Xpert Network* participants. While raising awareness of the benefits is important, one also has to understand that academic processes and reward systems often do not support spending time on documentation (2.2). Most academic research funding pays for showing principles and developing prototypes, not production-ready software. What's more, students are under pressure to graduate, whereby the functionality of the created software is more important than its documentation. If the software becomes successful, the original author often is no longer involved and thus has little incentive to retrofit a proper description. It is often the difficult task of the next generation of students in a team to accomplish these tasks. It helps, however, to understand these relationships. If one can identify the original creator, they may be willing to help, especially when offering them proper credit or co-authorship on a forthcoming publication. Vice versa, properly documenting the original authorship will insure that such credit can be given.

Reproducibility (2.14) of scientific research is a concern that is currently gaining attention. Proper software documentation can be key to reproducibility [3].

2.13 Continuous Integration

Integrate new software updates frequently into the application version seen by the end users in their end environments. Continuous Integration (CI) is generally good software engineering advice. It catches miscommunicated requirements early. CI also allows users to experience new features and provide feedback to the developer early and often. The process may be combined with automated builds and testing, ensuring that new functionality satisfies and continues to satisfy defined test cases.

CI was mentioned as particularly relevant for the work of Xperts, as scientific software tends to be a moving target with frequent changes of requirements. Ensuring that new features are what the end user had in mind, conform to defined test cases, and do not break previous requirements is critical. Many teams find that this approach leads to significantly reduced integration problems and enables the development of cohesive software more rapidly.

2.14 Reproducibility

Enable reproducibility and transparency by capturing data and software underlying scientific processes, using available software platforms. While the reproducibility of research results is a key requirement in any domain of science, there is recent, increased focus on this issue in computation-based research. Reproducibility was highlighted as a significant concern in our Xpert Network discussions as well, pointing out that adding supporting software and data to a publication can increase the value significantly. This is of particular importance in large computational studies, where data analysis may play a central role in reaching the conclusions. Transparency regarding the input data, the software underlying the research methods, and the analysis conditions will contribute to achieving consistent results in science projects.

2.15 Parallelization

Optimize the code only after it works correctly in serial. The question if parallel code should be written directly, versus creating serial code first, is an open one [20]. The Xpert Network participants were clear, however, in recommending the latter for creating Computational and Data-Intensive (CDI) applications. Among the arguments were that the benefits of lesser complexity of getting the serial code correct first, combined with better tool support for serial code, outweigh the negatives. The primary negative is that certain serial algorithms are intrinsically hard to parallelize. When keeping this issue in mind and selecting algorithms that are known to parallelize and scale well, this negative can be overcome, however.

3 RELATED EFFORTS AND SYNERGY

3.1 Related Work on Best Practices

A number of papers have recommended practices for scientific programming.

Wilson et al. [39, 40] have provided a set of "best practices" and also "good enough practices" for scientific computing community from the experiences of the thousands of people who have taken part in Software Carpentry [12] and Data Carpentry [11] workshops, and from a variety of other guides.

Dubois [8] has described some of his experiences to help writing scientific programs.

Heroux et al. [14] discuss practices used in the Trilinos [13] open-source software library project, some of which are close to practices advocated by the Agile software development community [21].

Naguib et al. [22] present an overview of two projects and describes the software engineering methods they applied on these computational science and engineering applications.

Wilson [37] talks about the "common core" of modern software development that is taught through software carpentry courses [12] to help computational scientists meet the standards that experimental scientists have taken for granted.

Riley [19] advocates bringing knowledge of useful software engineering practices to HPC scientific code developers. While not prescribing specific practices, she emphasizes adopting practices that *help productivity*.

The Better Scientific Software (BSSw) [5] community is focused on software development, and sustainability that leads to improved software for computational science and engineering (CSE) and related technical computing areas.

The Working Towards Sustainable Software Science: Practice and Experiences (WSSSPE) [41] community meet frequently to discuss the challenges of software for science.

The Software Engineering for Computational Science (SE4Science) [33] community is working to understand the differences, necessities, impacts, and barriers of applying general software engineering practices to research software So that scientific communities can adopt good software engineering practices.

The UK Software Sustainability Institute [17] has worked on facilitating the advancement of software in research by cultivating more sustainable research software.

IDEAS [15] community addresses issues of software productivity and sustainability in the Exascale Computing Project (ECP) [29].

The UCAR [35] Software Engineering Assembly (SEA) [34] is a community for software engineering professionals within UCAR, that addresses the issue of effective software engineering throughout UCAR.

While the contributions above provide important, general software engineering advice for science and engineering applications, in this paper, we focused on practices specific to computational Xperts (a.k.a. Research Facilitators or Research Software Engineers) that support domain scientists. We discussed the best practices related to project collaborations between Xperts and domain scientists, which is at the core of such projects. Furthermore, we discussed best practices for software development that are of particular relevance to CDI research. Also discussed are those software development practices that should be applied differently to CDI research than what is advised by general software development guidelines.

We reported best practices discussed by participants of the Xpert Network activities, which represent both large projects and individuals involved in such support roles. In addition, our experience working on several interdisciplinary projects with domain scientists has also contributed to the emergence of the best practices presented in this paper.

It is worth mentioning that even for those practices that are in common between CDI research and general software engineering, the priority and importance of the practices differ considering different scopes defined for the projects based on different environments (Academic Environment, or Industry) these projects are designed for. While in academic environment a working prototype would be enough to prove a point, in industry we are trying to make a reliable production quality application.

3.2 Related Efforts in Networking Xpert Groups

The Xpert Network is related to and collaborates with a number of activities that involve or facilitate idea exchanges among professionals supporting computational and data-intensive (CDI) research.

The *Research Software Engineer Association (US-RSE)* [31] is an initiative to create synergy among research software engineer individuals and groups at US universities and other institutions. This is a recent initiative with aims that are similar to those of the Xpert Network, and the two efforts seek to combine forces.

The *Campus Research Computing Consortium (CaRCC) effort* [28] also aims to engage a similar community as US-RSE, with the professionalization (professional development and advocacy) of those involved in campus research computing being a particular focus. CaRCC uses the term Research Facilitator, which is closely related to the terms Xpert and RSE.

The *Virtual Residency* [27] program focuses on training Xperts/ RSEs/ Research Facilitators.

The *The Extreme Science and Engineering Discovery Environment (XSEDE)* [43] project includes two thrusts that support CDI application researchers. The *Extended Collaborative Support Service (ECSS) group* [9] has a large number of computational and data experts who work collaboratively with domain scientists in improving their applications. The *Campus Champions* [42] program engages participants at many university campuses to raise awareness of XSEDE services and help local users accelerate CDI research.

The *CyVerse project* [6], supporting data-driven discovery, includes community support services similar to that of XSEDE's ECSS group.

Related support groups are also part of the NSF Software Institutes, including the *Science Gateway Community Institute* [30] and *the Molecular Sciences Software Institute* [18].

The *Xpert Network* exchanges ideas with all the above efforts. Best practices identified in this paper may provide training material for the mentioned projects. In addition to presenting best practices, the agenda of monthly Xpert Network webinars include discussions of coordination among the many involved activities, with the overall goal of increasing efficiency in the science community.

4 CONCLUSION

This paper presented best practices for those who support domain researchers in creating, improving, and running computational and data-intensive (CDI) applications. We refer to these professionals as computational experts, research software engineers, research facilitators or, short, Xperts. The information in this paper emerged from discussions and reports of the Xpert Network project, over a period of more than three years. The Xpert Network's participants include major NSF-funded projects for advanced cyberinfrastructures, national organizations, and campus organizations that support CDI researchers. This paper also describes some of the best practices identified through our work with domain scientists on several interdisciplinary projects. While some of the identified practices have overlap with those presented in general software engineering literature, we have emphasized aspects in which the work of Xpert professionals differs. Although none of these practices will guarantee error-free software development, when used together, they will reduce the number of errors in scientific software, facilitate its reuse, and save the authors of the software time and effort that can be devoted to addressing the underlying scientific questions. In addition, the paper discussed related projects that support CDI research and how to create synergy within this community. Together, these contributions aim to improve computational and data-intensive science in a significant way.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Award No. OAC-1833846.

REFERENCES

[1] Amy Apon, Stanley Ahalt, Vijay Dantuluri, Constantin Gurdgiev, Moez Limayem, Linh Ngo, and Michael Stealey. 2010. High Performance Computing Instrumentation and Research Productivity in U.S. Universities. *Journal of Information Technology Impact* 10, 2 (Sept. 2010).

[2] Dhavide A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Katy Huff, Ian Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Greg Wilson, and Paul Wilson. 2012. Best Practices for Scientific Computing. *CoRR* abs/1210.0530 (2012). arXiv:1210.0530 http://arxiv.org/abs/1210.0530

[3] Klaus Bartschat and Barry I. Schneider. 2019. (Position Paper) Workshop on Best Practices and Tools for Computational and Data-Intensive Research in atomic and molecular physics. https://sites.udel.edu/xpert-cdi/2019/08/06/ics-workshop/

[4] Adam Brazier. 2015. *Best Practices for Software Development in the Research Environment.* https://www.cac.cornell.edu/education/training/StampedeJan2015/BestPracticesForSoftwareDevelopment.pdf

[5] BSSW. 2020. *The Better Scientific Software (BSSw).* https://bssw.io/

[6] CyVerse. 2020. *CyVerse: Transforming Science Through Data-Driven Discovery.* https://cyverse.org/

[7] C. Day. 2012. Nobel prizes for computational science [The Last Word]. *Computing in Science & Engineering* 14, 06 (nov 2012), 88. https://doi.org/10.1109/MCSE.2012.123

[8] Paul F Dubois. 1999. Ten good practices in scientific programming. *Computing in Science & Engineering* 1, 1 (1999), 7–11.

[9] ECSS. 2020. *XSEDE Extended Collaborative Support Service.* https://www.xsede.org/for-users/ecss

[10] Rudolf Eigenmann and Parinaz Barakhshan. 2019. *THE XPERT NETWORK,Workshop on Best Practices and Tools for Computational and Data-Intensive Research.* Report P-29. University Of Delaware. https://cpb-us-w2.wpmucdn.com/sites.udel.edu/dist/6/8980/files/2019/08/ICS-workshop-report-RE4.pdf

[11] Data Carpentry Foundation. 2020. *Data Carpentry-Building communities teaching universal data literacy.* https://datacarpentry.org/

[12] Software Carpentry Foundation. 2020. *softwarecarpentry-Teaching basic lab skills for research computing.* https://software-carpentry.org/

[13] M. A. Heroux. 2009. *Trilinos project Home Page.* http://trilinos.sandia.gov

[14] Michael A Heroux and James M Willenbring. 2009. Barely sufficient software engineering: 10 practices to improve your CSE software. In *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering.* IEEE, 15–21.

[15] IDEAS. 2020. *IDEAS.* https://ideas-productivity.org/

[16] RDA in the United States. 2022. *Research Data Alliance.* https://www.rd-alliance.org/groups/rda-united-states

[17] Software Sustainability Institute. 2020. *UK Software Sustainability Institute.* https://www.software.ac.uk/

[18] The Molecular Sciences Software Institute. 2020. The Molecular Sciences Software Institute. https://molssi.org/

[19] Argonne National Laboratory Katherine Riley. 2020. *Good Scientific Process Requires Software Engineering Practices.* https://extremecomputingtraining.anl.gov/files/2016/08/Riley_935aug8_GoodScientific.pdf

[20] David B. Kirk and Wen mei W. Hwu. 2013. Programming Massively Parallel Processors. In *Programming Massively Parallel Processors, A Hands-on Approach, Second Edition.*

[21] Agile Software Development LLC. 2009. *Agile Software Development Home Page.* http://www.agile-software-development.com

[22] Hoda Naguib and Yang Li. 2010. (Position Paper) Applying software engineering methods and tools to CSE research projects. *Procedia Computer Science* 1, 1 (2010), 1505–1509.

[23] NaIP. 2022. *XSEDE, Novel and Innovative Projects.* https://confluence.xsede.org/pages/viewpage.action?pageId=1671332

[24] Henry Neeman, Hussein M. Al-Azzawi, Dana Brunson, William Burke, Dirk Colbry, Jeff T. Falgout, James W. Ferguson, Sandra Gesing, Joshua Gyllinsky, Christopher S. Simmons, and et al. 2019. Cultivating the Cyberinfrastructure Workforce via an Intermediate/Advanced Virtual Residency Workshop. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning)* (Chicago, IL, USA) *(PEARC '19).* Association for Computing Machinery, New York, NY, USA, Article 79, 8 pages. https://doi.org/10.1145/3332186.3332204

[25] Henry Neeman, Marisa Brazil, and Dana Brunson. 2019. (Position Paper)The Virtual Residency: A Training Program for Research Computing Facilitators. https://sites.udel.edu/xpert-cdi/2019/08/06/ics-workshop/

[26] NSF. 2020. *National Science Foundation.* https://www.nsf.gov/

[27] University of Oklahoma. 2019. *OU Supercomputing Center for Education & Research, Virtual Residency.* http://www.oscer.ou.edu/virtualresidency.php

[28] CaRCC Project. 2019. Campus Research Computing Consortium, https://carcc.org/.

[29] ECP project. 2020. *Exascale Computing Project (ECP).* https://www.exascaleproject.org/

[30] SGCI Project. 2015. Science Gateways Community Institute, https://sciencegateways.org/.

[31] US-RSE Project. 2020. U.S. Research Software Engineer Association, https://us-rse.org.

[32] Xpert Network Project. 2019. The Xpert Network, https://sites.udel.edu/xpert-cdi.

[33] SE4Science. 2020. *Software Engineering for Computational Science (SE4Science).* https://se4science.org/workshops/

[34] SEA. 2020. *SOFTWARE ENGINEERING ASSEMBLY (SEA).* http://sea.ucar.edu

[35] UCAR. 2020. *University Corporation for Atmospheric Research (UCAR).* https://www.ucar.edu/

[36] G. von Laszewski, F. Wang, G. C. Fox, D. L. Hart, T. R. Furlani, R. L. DeLeon, and S. M. Gallo. 2015. Peer Comparison of XSEDE and NCAR Publication Data. In *2015 IEEE International Conference on Cluster Computing.* 531–532. https://doi.org/10.1109/CLUSTER.2015.98

[37] Greg Wilson. 2006. Software carpentry: getting scientists to write better code by making them more productive. *Computing in Science & Engineering* 8, 6 (2006), 66–69.

[38] Greg Wilson. 2013. Software Carpentry: Lessons Learned. arXiv:1307.5448 [cs.GL]

[39] Greg Wilson, Dhavide A Aruliah, C Titus Brown, Neil P Chue Hong, Matt Davis, Richard T Guy, Steven HD Haddock, Kathryn D Huff, Ian M Mitchell, Mark D Plumbley, et al. 2014. Best practices for scientific computing. *PLoS biology* 12, 1 (2014).

[40] Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy K Teal. 2017. Good enough practices in scientific computing. *PLoS computational biology* 13, 6 (2017).

[41] WSSSPE. 2020. *Working Towards Sustainable Software Science: Practiceand Experiences (WSSSPE).* http://wssspe.researchcomputing.org.uk/

[42] XSEDE. 2020. *Campus Champions.* https://www.xsede.org/community-engagement/campus-champions

[43] XSEDE. 2020. *XSEDE – Extreme Science and Engineering Disvocery Environment.* www.xsede.org