Received: 14 September 2021 | Revised: 6 May 2022 | Accepted: 23 June 2022

DOI: 10.1002/spe.3120

EXPERIENCE REPORT

WILEY

Collaborative experience between scientific software projects using Agile Scrum development

Amanda L. Baxter[1] | **Segev Y. BenZvi**[2] | **Walter Bonivento**[3] | **Adam Brazier**[4] | **Michael Clark**[1] | **Alexis Coleiro**[5] | **David Collom**[6] | **Marta Colomer-Molla**[5,7] | **Bryce Cousins**[8,9] | **Aliwen Delgado Orellana**[10] | **Damien Dornic**[11] | **Vladislav Ekimtcov**[12] | **Shereen ElSayed**[13] | **Andrea Gallo Rosso**[14] | **Patrick Godwin**[9,15] | **Spencer Griswold**[2] | **Alec Habig**[16] | **Remington Hill**[14] | **Shunsaku Horiuchi**[17] | **D. Andrew Howell**[6,13] | **Margaret W. G. Johnson**[12] | **Mario Jurić**[18] | **James P. Kneller**[19] | **Abigail Kopec**[1,20] | **Claudio Kopper**[21] | **Vladimir Kulikovskiy**[22] | **Mathieu Lamoureux**[23] | **Rafael F. Lang**[1] | **Shengchao Li**[1] | **Massimiliano Lincetto**[11] | **Lindy Lindstrom**[6] | **Mark W. Linvill**[1] | **Curtis McCully**[6] | **Jost Migenda**[24] | **Danny Milisavljevic**[1] | **Spencer Nelson**[18] | **Rita Novoseltseva**[25] | **Erin O'Sullivan**[26] | **Donald Petravick**[12] | **Barry W. Pointon**[27,28] | **Nirmal Raj**[28] | **Andrew Renshaw**[29] | **Janet Rumleskie**[14] | **Tom Sonley**[30] | **Ron Tapia**[8,9] | **Jeffrey C. L. Tseng**[31] | **Christopher D. Tunnell**[32,33] | **Godefroy Vannoye**[11] | **Carlo F. Vigorito**[34] | **Clarence J. Virtue**[14] | **Christopher Weaver**[35] | **Kathryn E. Weil**[1] | **Lindley Winslow**[36] | **Rich Wolski**[13] | **Xun- Jie Xu**[37] | **Yiyang Xu**[32] | **The SCiMMA and SNEWS Collaborations**

Correspondence
Amanda Baxter, Department of Physics and Astronomy, Purdue University, West Lafayette, IN 47907, USA.
Email: adepoian@purdue.edu
Bryce Cousins, Institute for Computational and Data Sciences, The Pennsylvania State University, University Park, PA 16802, USA.
Email: bfc5288@psu.edu

Funding information
Division of Physics, Grant/Award Numbers: 1914409, 1914410, 1914416, 1914418, 1914426, 1914447, 1914448; Office of Cyberinfrastructure, Grant/Award Numbers: 1934752, 1940209

Abstract

Developing sustainable software for the scientific community requires expertise in software engineering and domain science. This can be challenging due to the unique needs of scientific software, the insufficient resources for software engineering practices in the scientific community, and the complexity of developing for evolving scientific contexts. While open-source software can partially address these concerns, it can introduce complicating dependencies and delay development. These issues can be reduced if scientists and software developers collaborate. We present a case study wherein scientists from the SuperNova Early Warning System collaborated with software developers from the Scalable Cyberinfrastructure for Multi-Messenger Astrophysics project.

Abbreviations: MMA, multi-messenger astrophysics; SCiMMA, Scalable Cyberinfrastructure for Multi-Messenger Astrophysics; SNEWS, SuperNova Early Warning System.

For affiliation refer to page 18

The collaboration addressed the difficulties of open-source software development, but presented additional risks to each team. For the scientists, there was a concern of relying on external systems and lacking control in the development process. For the developers, there was a risk in supporting a user-group while maintaining core development. These issues were mitigated by creating a second Agile Scrum framework in parallel with the developers' ongoing Agile Scrum process. This Agile collaboration promoted communication, ensured that the scientists had an active role in development, and allowed the developers to evaluate and implement the scientists' software requirements. The collaboration provided benefits for each group: the scientists actuated their development by using an existing platform, and the developers utilized the scientists' use-case to improve their systems. This case study suggests that scientists and software developers can avoid scientific computing issues by collaborating and that Agile Scrum methods can address emergent concerns.

KEYWORDS

Agile, cyberinfrastructure, multimessenger astrophysics, scientific computing, software development

1 | INTRODUCTION

The creation of scientific computing applications must address several software development challenges unique to the scientific software context. Unlike many commercial applications, a scientific application must often provide a precise solution to a problem before the software is modified or discarded.[1] This need for precise scientific solutions requires domain knowledge, leading to the inclusion of scientists on development teams. Indeed, scientists spend about 30% of their time developing scientific software, though they may not have prior training in software engineering and often must learn these methods during the development process itself.[2,3] Moreover, budgetary constraints or time limitations often make it impractical to employ professional developers to share responsibility for software integrity with the scientists. The timescales of scientific projects can be decades long with high personnel turnover, producing challenges for software sustainability. In particular, scientists who fulfill a development role can create legacy technical debt when they change roles or projects as they exploit new career opportunities. Efforts to introduce a new career trajectory of "research software engineer" attempt to address this problem of software sustainability.[4] Another approach to this issue involves the use of community-supported software, namely open-source codebases, that can "outlive" the tenure of any one developer.[5,6] Open-source is freely available and community maintained, but the release and maintenance life-cycles are not coordinated between technologies and services. Thus, the maintenance procedures for an application that contains multiple open-source technologies must be able to account for and manage the different software life-cycles of its dependencies, introducing a risk that a required feature may be unavailable by a software dependency. Moreover, the community-based maintenance model of open-source codebases can introduce issues with the rapid time frame of scientific software development. While users can submit feature requests, issues, and bug fixes in the form of pull-requests to the external codebase, the integration of these requests can take days or weeks, if they are integrated at all.[7]

These complexities are often unaddressed in modern scientific software development contexts since resolving them introduces additional risks to a positive and timely scientific outcome. Simply adding the tooling necessary to develop and maintain long-lived services can slow the time to scientific solution due to the time required to both learn and apply this tooling. As a result, the use of software auditing, life-cycle, and management methods like version control and issue tracking for scientific applications varies over a large range from absent to proficient, with many implementations either budding or unused.[5] Moreover, service-based scientific applications often consist of deep, bespoke software stacks, developed by domain experts who become integral to their sustainability. The development teams are small and, once

in production, may consist of only a single expert developer who knows the system well-enough to provide support for issues. This lack of scalability causes risks for scientific applications that serve large communities of scientists. These issues have led to a "chasm" between scientific computing and formal software development.[8] However, it has been suggested that this shortcoming can be overcome by building collaborative bridges between the scientific computing and software development communities.[8,9]

In this article, we describe the experience of scientific computing developers[10] who collaborate with a professional Agile software development team[11] to create a scientific application[12] that depends on a cloud-based service created independently by the software developers.[13] This collaboration allowed the scientific team to mitigate the risk of legacy software[14] by establishing a dependency on an upstream development project that is committed to supporting external users for its cloud-based services. However, the upstream project itself was being developed simultaneously, which increased the need for effective coordination between the two projects.

To implement this coordination, the collaboration established an Agile Scrum process for the development of the scientific software that tracked the ongoing Agile development of the upstream cloud-based service. We describe the coordination required to successfully manage these two parallel, but dependent, Agile processes, where the downstream process necessarily trailed the upstream one. We focus primarily on the downstream Agile process that this collaboration created and outline the effects this had on the ongoing, upstream Agile process. We also discuss the software engineering procedures we employed to ensure that the scientists could trust and depend on the in-progress upstream cloud-based service.

We review traditional and Agile software development procedures and provide an introduction to Agile abstractions in Section 2. We provide an overview the two groups in Section 3. We describe the creation of our collaboration and the risks involved in Section 4. Our approach to using Agile Scrum to resolve these risks is described in Section 5. We assess the collaboration's Agile Scrum framework and the outcomes for each organization in Section 6. We provide connections between this case study and other related works in Section 7. Section 8 summarizes the takeaways of this case study for the scientific computing community.

2 | SOFTWARE DEVELOPMENT METHODOLOGIES

The integration of commercial software engineering best practices into scientific application development can be performed via a collaboration between scientists and professional software developers.[9] The goal of such a collaboration is for the scientists on the development team to use their domain expertise to explain software requirements to the software developers, allowing the two groups to utilize their collective knowledge to generate software requirements from the scientific use-case.[15]

Such a partnership may introduce productivity risks for the professional developers, in addition to risks in the scientific development process. In this project, the professional development team had to maintain core development timelines and goals for its original stakeholder base, while at the same time developing new features and mechanisms to support the evolving scientific software development.

These issues can be aggravated by traditional software development frameworks such as the "Waterfall" approach, wherein the stakeholders and developers establish software requirements at the beginning of the project and then develop to meet those requirements.[16] If the requirements change during the development period, the software is obsolete when delivered. To avoid this, Waterfall developers must correct the requirements and refactor the software during a release. If the requirements change rapidly enough or the refactoring time is too long, software delivery can be disrupted or delayed.

One approach to mitigate the issues of the Waterfall approach is to instead utilize an Agile Scrum software development framework.[9,15] The Agile software engineering process (described in Section 2.1) is designed to be flexible to rapidly changing requirements. It is an iterative approach to software development, wherein week- to month-long development tasks are favored over large, long-term goals for the end product.[17,18] Moreover, the goal of an Agile development effort is to have a releasable product at the end of each short development period, called a "sprint." Thus, as requirements change, the software is never more than a sprint's duration away from a partially featured releasable state.

The Agile framework's emphasis on short-term goals might not seem reasonable for large-scale scientific projects that often have long-term science goals on the timescale of decades. The key observation in this context is that while the long-term science goals may remain relatively fixed, the requirements for the software that is developed to meet these

science goals may change as scientific insights emerge or as personnel change. Agile software engineering encourages such incremental progress by organizing development into "User Stories," which are individual tasks created and completed as emergent software requirements are discovered.[18,19] While some studies have combined the Agile and Waterfall methods[20] or utilize nested Agile sprints within a single team,[21] the use of a single sprint process is typical of Agile practices.

The Scrum approach is one method to adopt Agile practices into a project.[15,18] Scrum is a process framework that structures a project into a planning phase, design phase, and development phase to provide an incremental and emergent approach to Agile development.[22] These elements are particularly relevant in the scientific computing communities given their tendency of rapidly evolving software requirements.[15,23-27] Despite these potential advantages, Agile practices are not widely implemented[8,9,28] or incorrectly adapted[15] in the scientific computing community.

2.1 | Review of Agile abstractions and personnel roles

An Agile development effort requires the participants to fulfill specific functional roles that manipulate a set of abstractions representing the software throughout its development life-cycle. We briefly summarize definitions of these abstractions and roles for those who may be unfamiliar with the terms.

2.2 | Scrum

The overall organizational abstraction for an Agile project is the Scrum, typically comprising several components: *sprint planning*, *sprints*, and *sprint retrospectives*. During a sprint planning meeting, the specific development tasks are discussed and decided upon. A sprint is a discrete period of software development during which the developers carry out the tasks identified during the sprint planning meeting. Finally, during a sprint retrospective meeting, the team reviews the previous sprint to identify uncompleted tasks and review team productivity.

A key element to this approach is that each sprint leads to the completion of the tasks identified during sprint planning even if the requirements change mid-sprint. That is, adapting the development in response to changing requirements occurs during sprint planning and sprint retrospectives. This is facilitated via daily check-in "standing" meetings involving all Scrum participants. During a check-in, each project member gives a short statement of what they worked on the day before, what they will work on in the current day, and any blockers where other tasks are limiting their progress.

2.3 | Abstractions

The principle abstractions in the sprint process are *Epics*, *User Stories*, *tasks*, and the *Backlog*. An Epic is an overarching functionality road-map for the software. It is expressed in terms of User Stories and the relationships between them. A User Story is a description of a specific functionality a user would like to have the software fulfill. Typically, a User Story consists of the identification of the user's role, the functionality, and the expected outcome. The collection of User Stories for the Scrum describe the expected user experience for the software. A task is a specific software development item that is necessary to meet the functionality desired by one or more User Stories. Each task is scoped so that it can be completed within the time allocated to a sprint and each developer takes responsibility for completing one or more tasks. The Backlog is the collection of tasks that have yet to be claimed by a developer in a given sprint. The Backlog is created during each sprint planning meeting and contains all tasks to be completed during that sprint. Thus, while each developer will work on their assigned tasks independently, the developers share the same Backlog. At the conclusion of a sprint, any uncompleted tasks remaining in the Backlog are analyzed during the sprint retrospective.

2.4 | Personnel roles

A Scrum involves development personnel that fulfill one of three roles: *Scrum Master*, *Product Owner*, or *Team Member*. Additionally, the project identifies *Stakeholders* who represent those invested in the success of the software, but do not

take an active role in the Scrum meetings. The Scrum Master runs the Scrum, chairing all meetings, ultimately creating the Backlog for each sprint, and managing the analysis of uncompleted tasks in a sprint retrospective. The Product Owner represents the interests of the Stakeholders and users. Finally, the Team Members comprise the developers who implement tasks during each sprint.

3 | ORGANIZATIONS AND BACKGROUND

The SuperNova Early Warning System (SNEWS) is an open, public alert system that is built to provide an early warning for core-collapse supernovae in the Milky Way galaxy.[29] The first indication of a potential stellar explosion will be the arrival of a burst of neutrinos.[30] If several detectors report a potential supernova within minutes of each other, SNEWS will issue an alert to its subscribers, which include astronomical observatories, neutrino detectors, and amateur astronomers and citizen scientists.

There are seven neutrino experiments participating in the current SNEWS framework and about 20 neutrino detectors worldwide that are planned to participate in the new SNEWS 2.0 framework, which is the basis of this work. From these neutrino experiments, over 100 scientists from all over the world are part of the SNEWS collaboration. Each neutrino detector has different data formats that the SNEWS framework must be able to combine to identify neutrino burst coincidences. Additionally, these neutrino detectors will be active within SNEWS at different times. With these specific requirements, SNEWS is one of the few successful examples, to date, of cyberinfrastructure spanning major neutrino experiments.[10] In the current era of multi-messenger astrophysics (MMA), there are new opportunities for SNEWS to optimize its science reach for the next Galactic supernova. To achieve this goal, SNEWS depends on developing and sustaining software with a number of integrated components.[10] SNEWS received a 3-year grant to complete this upgrade with the potential to extend the funding period.

SNEWS represents one agent in an ever-growing community of MMA organizations that rely heavily on cyberinfrastructure to support their science requirements.[31-38] These requirements include prompt, reliable, and efficient alerting in addition to seamless support for different message formats and conventions.[38-41] The requirements also pertain to multiple detectors, institutes, and nations, and would be unrealistic for any individual group to address.[42]

These challenges are not unique to the SNEWS effort. The MMA science community has developed various technologies, usually focused on a single project or experiment, that promote the sharing of results and collaboration among projects. However, MMA requires integration between individual systems and teams distributed across the world, highlighting a need for substantial cyberinfrastructure. For example, the landmark multi-messenger detection of the binary neutron star merger "GW170717" involved two gravitational wave detectors in the United States (the Laser Interferometer Gravitational-Wave Observatory detectors at Hanford and Livingston), one gravitational wave detector in Italy (Virgo), a gamma ray satellite (the Fermi Gamma-ray Space Telescope), and over six independent teams of astronomers.[43] The coordination of such distributed teams and instruments requires a prompt, robust alerting system to facilitate communication between individual organizations. This requirement, and the general need for a common set of functionalities across otherwise separately developed MMA systems, prompted the creation of the Scalable Cyberinfrastructure for Multi-Messenger Astrophysics (SCiMMA) project,[11] a collaboration involving approximately 50 faculty, research scientists, and computing staff in physics, astrophysics, and computing from over 10 universities. SCiMMA has been addressing the cyberinfrastructure needs of multiple MMA organizations and science efforts by engaging with the user community[44,45] and developing a suite of MMA services (Figure 1) through a 2-year community planning grant and 3-year cyberinfrastructure grant. SCiMMA's development has focused initially on addressing the general need for the coordination of alerts and data between MMA organizations, such as the Vera C. Rubin Observatory's Legacy Survey of Space and Time (VRO LSST),[31] the advanced Laser Interferometer Gravitational-Wave Observatory (LIGO),[46] and the IceCube Neutrino Observatory.[47] These services include the shared, openly available, multi-format data distribution software `hop-client`,[48] a customizable `hop-client` application template,[49] and a cloud-based system of data streams with extensive identity/access management controls known as `HOPSKOTCH`.[13,50]

To develop these services, the SCiMMA collaboration employs an Agile Scrum framework staffed by 5–10 part-time scientific software developers, many of whom are already involved in the computing efforts of LIGO, VRO, and IceCube. The Agile team is led by two senior scientists and investigators from VRO and the North American Nanohertz

Observatory for Gravitational Waves.[51] SCiMMA is thus already connected to the MMA ecosystem, allowing it to deploy a set of successful practices and systems for the MMA community.[48,52,53]

3.1 | Collaboration opportunity

SNEWS was preparing to upgrade its alert software framework from the current version of SNEWS to SNEWS 2.0. SNEWS 2.0 would require a scalable and reliable publish-subscribe system, potentially based on cloud-hosted web-services. The SNEWS team had already created the requirements for their first alert software as part of a Waterfall methodology and was exploring technology options for a purely in-house development effort.[10] Initially, SNEWS considered two options for their software development upgrade. SNEWS could either (1) build these systems from scratch, or (2) explore existing software. Each option would offer both advantages and challenges. Option 1 would allow SNEWS to maintain direct control over the development and ensure their needs were met, but it would impose the burdens of long-term development, support, and maintenance entirely on SNEWS developers. This could ultimately become problematic as SNEWS most likely would not have the resources and personnel to sustainably maintain the software long-term. Option 2 would avoid these burdens by offloading the core programming to an external software group, but it would reduce SNEWS's control in the process and potentially limit their representation and communication during development.

At the same time that SNEWS was assessing the upgrade process, SCiMMA was in the midst of its first release cycle for new cyberinfrastructure systems, `HOPSKOTCH` and the `hop-client` (Figure 1), using an Agile Scrum methodology.[13,48] These systems were developed as part of SCiMMA's goal of supporting the MMA community's need for sustainable, industry-standard software tools and cyberinfrastructure for alert data management and sharing, in order to support the use-cases of its current stakeholders from LIGO, VRO, and IceCube.[11]

The connection between SNEWS and SCiMMA began when a principle investigator from SNEWS met with leadership from the SCiMMA team at an in-person National Science Foundation "Harnessing the Data Revolution" grant meeting.[54] During the discussions, it became clear that `HOPSKOTCH` either provided or would provide the functionality required by the SNEWS 2.0 system, and that SNEWS could serve as SCiMMA's first true user group. While none of the developers from SNEWS and SCiMMA had met, the members shared a common language of scientific software development and the principle investigator discussions indicated that cooperation could lead to potentially mutual benefits. Given this clear overlap of interests, the two groups proposed an informal, remote cooperation between SNEWS and SCiMMA.

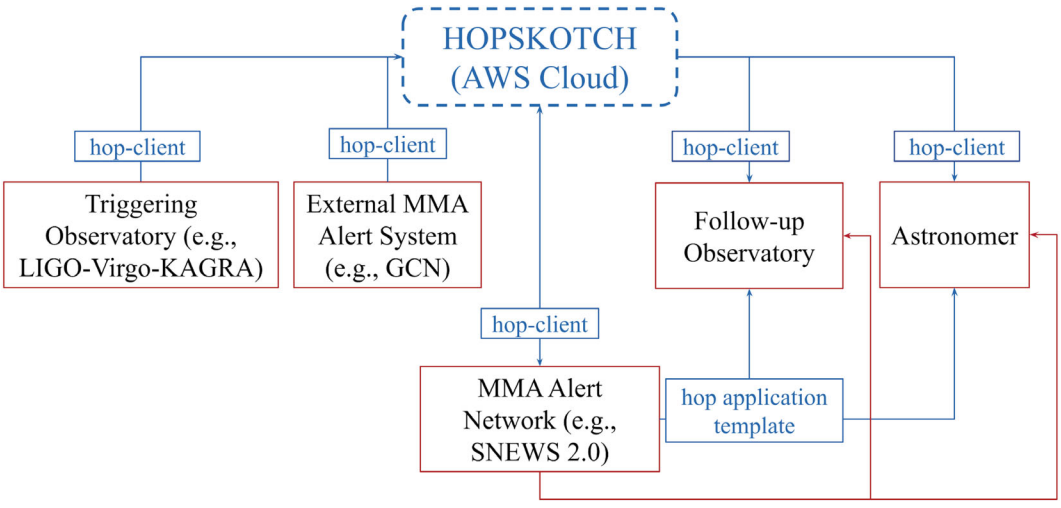

FIGURE 1 Schematic of the services developed by SCiMMA (blue) and their integration into the MMA community (red). The `hop-client`[48] provides MMA observatories, systems, and users with authenticated publish/subscribe access to the `HOPSKOTCH` cloud[13,50] of Kafka topics.

Initially, the cooperation began experimentally without direct collaboration between the groups. To test the hypothesis that HOPSKOTCH would meet SNEWS's requirements, the SNEWS implementation team carried out their own development using publicly available code from SCiMMA[48,49] without directly engaging the SCiMMA client development team. This allowed SNEWS to explore the potential benefits of SCiMMA's systems without significant financial investment, as this required only partial effort from a handful of individuals without the need for a separate grant or funding source. To become more familiar with the code, the SNEWS team engaged with one of SCiMMA's virtual community workshops to further assess the value of a collaboration.[45] This phase was a traditional example of a user-group adopting a piece of open-source software, wherein developers modify another group's existing codebase to fit their own needs without coordination or direct interaction between the groups.

Once SNEWS developers began integrating HOPSKOTCH into their early prototype of SNEWS 2.0, they found that they required more extensive features that were not available in the current version of SCiMMA's systems. SNEWS already knew exactly what features they needed in SCiMMA's software, but did not have the developer effort necessary to implement them. SCiMMA had ample professional software development effort to provide for this need, but did not have an understanding of SNEWS's specific scientific requirements and software needs. Given this split of domain knowledge and software development talent between the organizations, the traditional open-source development strategy of making feature requests or independently customizing the software would not meet SNEWS's needs. Therefore, SNEWS principle investigators contacted SCiMMA principle investigators to discuss the possibility of a focused collaboration to develop the desired features together. This prompted the creation of a partnership that evolved into a dedicated bilateral collaboration involving members from both the SNEWS implementation team and the SCiMMA client-development team (Figure 2). This combined effort allowed a collaboration to form via piece-wise funding contributed by both SNEWS and SCiMMA, without the need for a formal joint funded proposal between the organizations. This direct engagement ultimately catalyzed development for both organizations, but both teams were concerned with the inherent risks due to the collaboration's novel structure and the fact that each organization's development was at an early, prototyping stage. The main concerns were:

- Can SNEWS rely on pre-alpha, externally supported systems to sustainably accomplish the scientific goals for SNEWS 2.0? Would the scientific requirements be communicated and addressed adequately on a short time-frame in a collaboration?
- Can SCiMMA temporarily prioritize a single user-group without derailing the ongoing development needed to support the broader MMA community? Would early user-engagement be feasible and beneficial in the long-term?

These risks could not be confronted until the SNEWS developers first ensured that SCiMMA's work would be useful in their own efforts. Since SCiMMA had made its codebase openly available and promoted its development efforts

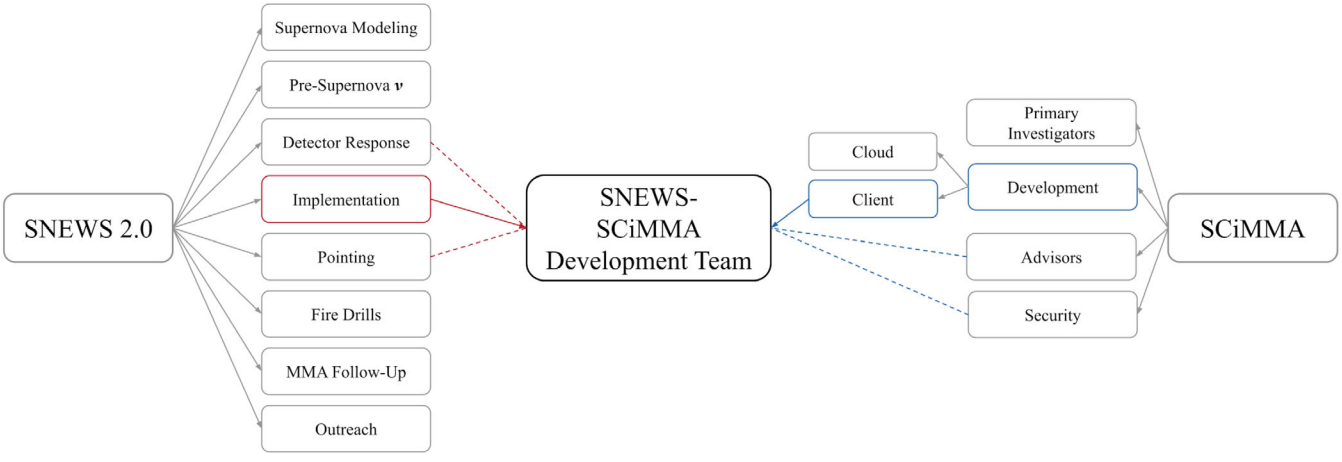

FIGURE 2 Collaboration ecosystems for SNEWS and SCiMMA. The SNEWS collaboration is broken into eight working groups. The SCiMMA collaboration is broken into four working groups. The SNEWS implementation group worked with the SCiMMA client-development group to form the development team described in this work. The dotted lines indicate teams which provided functional input to the development, but did not participate in the development itself.

via community workshops, SNEWS could assess an active collaboration with SCiMMA prior to committing to such an engagement. We believe that this initial trust-establishing step was key to the ultimate success of the effort. Without it, much more of the collaboration itself would have focused on trust building and maintenance rather than on productivity and risk management.

4 | COLLABORATION APPREHENSION

While the collaboration presented a unique opportunity to accelerate development, it brought risks to both teams. The concerns of each group provided an outline for the nature of the collaboration, which evolved through several stages of bilateral development effort and progress evaluations. To address these concerns (Section 5), the collaboration adopted its own Agile Scrum framework in parallel with SCiMMA's own Agile Scrum process. This new weekly sprint trailed SCiMMA's ongoing bi-weekly sprint. This promoted communication and drove software development in response to changes introduced by both the SNEWS requirements and the original SCiMMA stakeholders.

4.1 | Concerning the reliance on external software

Upon learning about SCiMMA and `HOPSKOTCH`, SNEWS considered using these services for their own development prototyping. This would allow SNEWS to actuate its initial development using the already-existing systems and resources provided by SCiMMA, which would be supported and maintained by professional software developers long-term. However, the main concern of the SNEWS team was that the development priorities of SCiMMA may not align with SNEWS's feature requests and bug reports, which could cause a potential latency or disconnect between SNEWS's requirements and SCiMMA's implementation. Moreover, the software could become unmaintainable if the SNEWS team was not engaged in the development process. Nonetheless, SCiMMA's available features and end goal appeared to fit all of the initially identified needs, so SNEWS decided to proceed.

SNEWS carried out the initial development which began in Phase 0 (Figure 3). During this stage, SNEWS developers attended the SCiMMA 2020 Virtual Workshop software demonstration session.[45] This allowed the SNEWS developers to utilize the SCiMMA `hop-client` application template, a customizable interface to `HOPSKOTCH`, as the basis to prototype SNEWS 2.0 using SNEWS's previous software requirements. They extended the template with additional SNEWS-specific functionality, maintained the codebase in a private GitHub repository, and ran local tests in Python. During development, SCiMMA added the SNEWS developers to the SCiMMA Slack[55] channel to provide user support and debugging as SNEWS developed the prototype, though the development itself was driven entirely by SNEWS. This phase resulted in a successful redeployment of SNEWS's basic functionality using SCiMMA's `hop-client`.

Once SNEWS developed their prototype to the extent possible with `hop-client`, SNEWS presented a status update at SCiMMA's July 2020 Public Teleconference.[56] This presentation highlighted SNEWS's current implementation, usage requirements, and desired features as realized during Phase 0 (Figure 3). Some requested features were not currently in SCiMMA's software, such as a database for message storage or a customizable message format. The SNEWS team considered three options: develop the missing requirements themselves, wait until SCiMMA developed these requirements in their own timeline, or seek a collaboration with SCiMMA to develop the requirements together. After this status update, it was not clear which path forward SNEWS should take.

4.2 | Concerning available development effort

The SNEWS status update prompted a period of internal assessment for SCiMMA to evaluate the feasibility and utility of collaborating with SNEWS. The potential benefits of a collaboration were promising, as SNEWS would serve as SCiMMA's first development partner and provide a practical use-case for its cyberinfrastructure. This could accelerate the development and demonstrate wider community impact for SCiMMA.[11] SCiMMA therefore proposed a short period of focused development with SNEWS after the SNEWS status update near the end of Phase 0. However, this collaboration presented a risk to SCiMMA's core goals due to its limited funding: SCiMMA's cyberinfrastructure grant

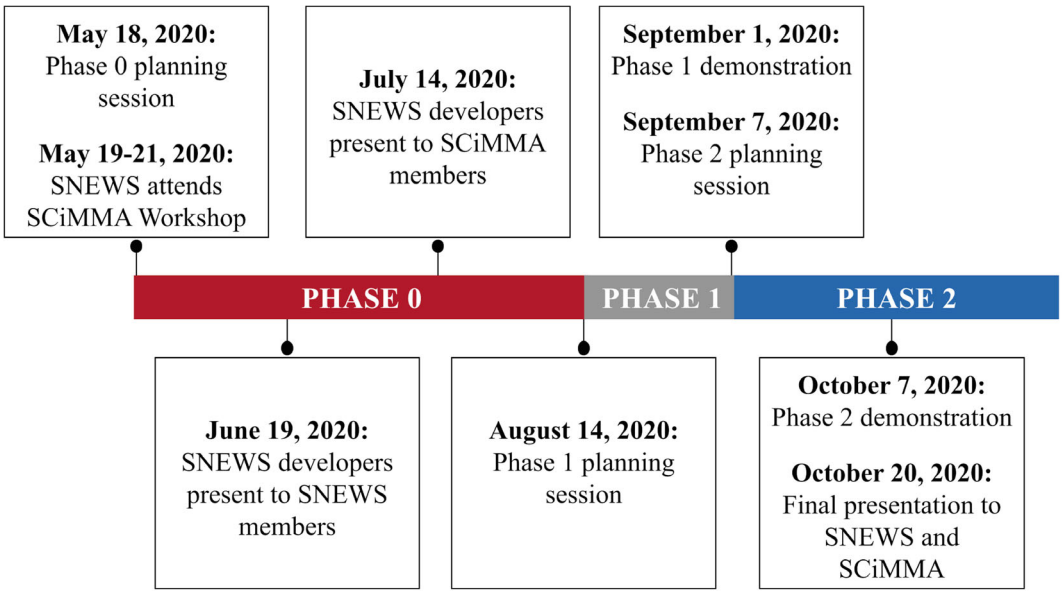

FIGURE 3 Timeline of events that led to the development of the SNEWS-SCiMMA collaboration and the SNEWS 2.0 prototype. Indicated on the timeline are the three phases of the collaboration. The conclusion of each phase marked an evaluation period for each organization, which provided a decision point on whether to continue the collaboration.

had to support HOPSKOTCH cloud costs in addition to the developers. To mitigate this risk and ensure that development would not be derailed, the SCiMMA team formally integrated the collaboration with SNEWS into its own Agile sprint structure. The team created an Epic and User Stories specifically for SNEWS development support, and assigned this project to two developers. These SCiMMA developers remained active in core SCiMMA tasks and assessed the SNEWS codebase, but did not commit any time toward SNEWS software development during the remainder of Phase 0. After SNEWS's presentation to SCiMMA and the SCiMMA developers' internal sprint presentation, it became clear that a collaboration would support SNEWS's development goals while providing an informative use-case for SCiMMA's own development. While there were still concerns about allocating effort away from core development, SCiMMA proposed a short period of focused collaboration with SNEWS (Phase 1 in Figure 3) given the many synergies between the projects. Both groups then decided to actively work together to further the development of the SNEWS 2.0 prototype.

4.3 | Concerning communication during a collaboration

The assessments during Phase 0 confirmed both that HOPSKOTCH was a valuable technology for supporting SNEWS 2.0 and that SCiMMA was willing to work with SNEWS to meet their requirements and timeline, suggesting that a collaboration would be mutually beneficial. However, given the concerns of each organization, the structure of the collaboration would need to meet several constraints. Foremost, the workload would need to be shared between organizations. This would ensure that SNEWS would have the understanding required to sustain the codebase after SCiMMA's involvement, allow the SCiMMA developers to understand the existing SNEWS codebase on a short timescale, and avoid software duplication. This collaboration would also change SCiMMA's previous community engagement model, wherein MMA organizations served only as end-users rather than collaborative developers. This would increase the importance of reliable communication between SNEWS and SCiMMA to identify and evaluate ideas emerging from the collaborative development. The emergent nature of these constraints, along with the short timescale, suggested that an Agile Scrum framework would benefit the collaboration. However, the addition of a second Agile Scrum process to SCiMMA's effort would require careful implementation to avoid complicating the separate, but dependent, development efforts. Moreover, trust would have to be built in the Agile process itself since none of the SNEWS team had utilized Agile methods for software development nor heard of its success in scientific computing.

5 | USING AGILE SCRUM TO MITIGATE APPREHENSION

The collaboration was structured to address the concerns of SNEWS and SCiMMA to ensure that the combined effort would benefit each individual group. While the shared funding commitment from SNEWS and SCiMMA reduced some of the up-front risk of collaborating, mutual trust was still required in order to address the concerns of each organization. The success of one group was dependent on the success of the other, so each team needed to consider the other's requirements in addition to their own. This promoted mutual interest between the two projects and provided a foundation of investment and trust despite the lack of in-person meetings, pre-existing social connections, or formal joint funding proposals. The precollaboration planning (Phase 0) and the initial 2-week period of focused collaborative development (Phase 1) provided informative requirements for the collaboration structure, which evolved into a more formal Agile framework in the 4-week Phase 2. Each phase proceeded only after a consensus between organizations was reached, so that the conclusion of each phase allowed each organization to build trust and evaluate whether to continue the collaboration. This approach ensured that the collaboration was addressing the evolving needs and perspectives of each organization.

This collaboration was composed of two members of SCiMMA and four members of SNEWS, from the working groups pictured in Figure 2. In this collaboration, the Scrum Master was a part-time SCiMMA software developer who primarily worked on gravitational-wave computing cyberinfrastructure. There were two SNEWS Product Owners, one primary investigator and one graduate student from particle astrophysics domains. The Scrum Master and Product Owners identified the specific User Stories that were to be tasked during each sprint. The entire team included the Scrum Master and Product Owners, a second software developer from SCiMMA, an undergraduate student developer from SNEWS, and an academic IT specialist from SNEWS. This composed the core Scrum team, which coordinated sprints via sprint planning sessions, sprint retrospective meetings, and daily check-ins. The Stakeholders included six primary investigators in total, three from SCiMMA and three from SNEWS, who served to evaluate the phases and decide if a new phase should be generated. Due to the geographically distributed, remote nature of the collaboration, all Scrum and Stakeholder meetings were conducted via Zoom,[57] check-ins were facilitated via the online real-time messaging system Slack,[55] and User Story management was conducted via a GitHub Project Board (Figure 4).[58]

5.1 | Addressing simultaneous development efforts

As the Phase 1 collaboration began, SCiMMA created an Epic in its Agile process for a collaboration with SNEWS that would run in parallel to SCiMMA's main development efforts and dedicated two SCiMMA developers on this Epic. Sprint deliverables from this SCiMMA Epic triggered the addition of User Stories to the SNEWS Backlog. SCiMMA integrated the SNEWS collaboration progress into the regular SCiMMA Agile Scrum status updates during daily check-ins and sprint

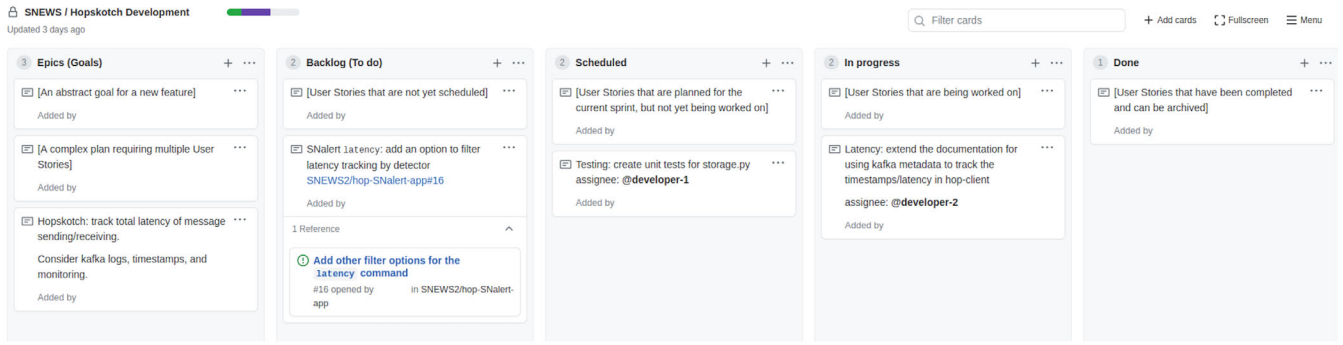

FIGURE 4 An example of the GitHub Project Board used for the SNEWS-SCiMMA development collaboration. At the beginning of Phase 1 and 2, developers would brainstorm Epics in the first column based on feedback from the two organizations. During each sprint, the Scrum Master would work with developers to generate new User Stories from the Epics. These User Stories would remain in the *Backlog* column until they were assigned to a specific developer and later moved to the *Scheduled*, *In-progress*, or *Done* columns over the course of the sprint.

retrospectives, ensuring that the SNEWS collaboration would be synchronized with SCiMMA core development. The regular status updates provided an active dialogue throughout Phase 1, allowing SCiMMA to remain in-sync with emergent feedback and ready to adjust its development plans in response. This dialogue involved multiple new User Stories generated from bugs and feature requests during Phase 1. In addition, SCiMMA noted that many of these User Stories were of general MMA interest and decided to transfer some of these stories from the SNEWS Backlog to their own Backlog. This synergy suggested that dedicating more effort for a focused collaboration could continue benefiting core SCiMMA development, even if it reduced total effort available in the short-term. Since these potential benefits were within SCiMMA's primary scope, the developers could transition to collaborative development in support of SCiMMA's preexisting Agile process with little concern about formalizing a commitment of time and funding away from core efforts. Moreover, core development could still proceed even if several developers were focused exclusively on an external collaboration, as the rest of the team could continue working on components that were independent of the SNEWS client development, such as the `HOPSKOTCH` cloud servers.

5.2 | Addressing communication and reliance on external software

To implement Agile methods for Phase 1, the Scrum Master coordinated meetings and the creation of User Stories based on the Product Owners' scientific requirements. Developers from each team actively engaged during Phase 1, communicating regularly via Slack check-ins, Zoom meetings, and by using a shared Github project board. The two teams conducted one-week sprints, consisting of a review meeting each week to informally check progress and create User Story task cards on the project board. These meetings continued in-parallel with SCiMMA's ongoing bi-weekly sprint meetings, with additional virtual impromptu pair-programming sessions for code debugging and technical questions. The Scrum Master regularly consulted with the SCiMMA Scrum Master and Product Owners during their ongoing sprint process. The SNEWS Product Owners were responsible for ensuring that the SNEWS 2.0 prototype was within specifications, communicating the needs of SNEWS to SCiMMA during sprints, and reporting the collaboration progress during SNEWS's internal meetings. The other developers worked on improving the prototype and deploying it on test systems. This framework ensured bilateral effort for the collaboration and provided regular communication between the organizations. It gave SNEWS an active role in the development, allowing for feature requests and bugs to be relayed through the SCiMMA developers to the ongoing SCiMMA sprint. This also allowed for SNEWS to become familiar with the software, reducing the risk of unsustainable software after the collaboration. The Agile Scrum framework also supported timely development by allowing large goals to be translated into smaller tasks, promoting iterative development.

5.3 | Addressing timeline constraints

Phase 1 concluded with a demonstration of the software to the SNEWS and SCiMMA stakeholders. After reviewing the collaboration progress, it was clear that there was mutual benefit and that the initial concerns appeared to be ameliorated in a manageable way. Both organizations agreed to a second, 4-week period of focused development (Phase 2 of Figure 3). However, at the end of Phase 1, there were clear timeline constraints: SNEWS was seeking to begin the transition to production operation of the software and SCiMMA expected to focus more effort on the internal development tasks that had resulted from the collaboration. These timelines were constrained by the funding available for each project independently. These new constraints prompted several changes to the collaboration structure.

Phase 2 continued to use the GitHub Project Board and daily check-ins on Slack as in Phase 1, but the sprints were restructured to better reflect timeline constraints. Four 1-week fnsprints were set up, with a weekly sprint meeting consisting of review and planning sections, in addition to the ongoing daily check-ins. In the review section, the developers gave short, round-table updates on their progress and blockers from the previous week based on the cards they were assigned. In the planning section, the Scrum Master evaluated the status of the *Scheduled* and *In-progress* task cards in the project board (Figure 4), marking them as *Done* or *In-progress*. The developers then created new Backlog cards by discussing the existing Epics and self-assigned them to work on in the upcoming week. This revised format matched the Agile Scrum development model, as the short-term iterative development progress could be checked against the longer-term Epics during each week.

While the 1-week sprint cadence differed from SCiMMA's 2-week sprint schedule, this allowed the SNEWS effort to stage its planning and evaluation meetings both between SCiMMA sprint meetings and also immediately after them. In this way, the SNEWS effort was able to respond to requirement changes introduced by the core SCiMMA effort. If the Agile timelines had been the same, there was a risk that the SNEWS effort would respond during a 2-week sprint only to find that the core effort had already introduced specification changes, and thus SNEWS would never be able to remain in-sync with the SCiMMA development. This allowed SNEWS to keep a separate timeline for its own deliverables without depending on the timeline for core SCiMMA development.

The second purpose allowed SNEWS to pick a specific release date set for the end of Phase 2. SNEWS's goal for Phase 2 was a product that could be effectively released to SNEWS for further enhancement and operational deployment. After this release, SNEWS would contribute to SCiMMA as a community member through existing interaction mechanisms such as workshops, but not through direct collaboration. Thus the developers on both teams decided on multiple week-long sprints to achieve the end goal for the software. As with Phase 1, Phase 2 concluded with a demonstration of the software to the stakeholders, which was recorded as an endpoint marking the conclusion of the dedicated software development period.[59]

6 | DISCUSSION

Each phase of the collaboration was structured around technical demonstrations to iteratively evaluate progress, assess current goals, and plan out goals for the next phase. These demonstrations allowed for trust-building and re-evaluation of personnel commitments and funding for each organization. Phases 1 and 2 utilized weekly meetings coordinated via a GitHub Project Board (Figure 4) for task and project management. The evolution of the codebase and development metrics throughout the collaboration is outlined in Figures 6 and 7, allowing for the success of the collaboration to be tracked throughout the phases.

6.1 | Measurements of success

The main goal of Phase 0 was to rewrite the current SNEWS publish-subscribe system using the `hop-client`[48] (Figure 5). Though an Agile framework was not used during this phase, Phase 0 served a similar role to the Agile Inception Phase: planning and deciding on software requirements was the primary goal. Phase 0 was marked successful by the demonstration of a working prototype that included sample publishing and subscribing of

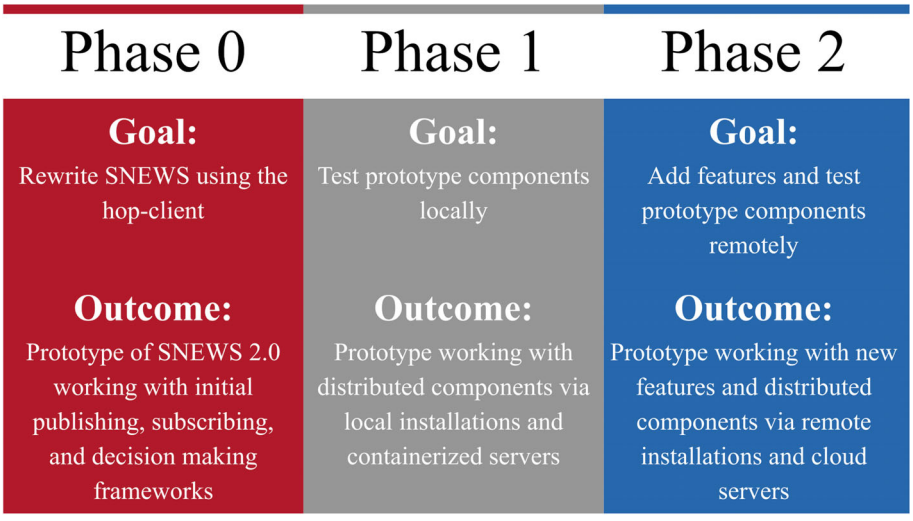

FIGURE 5 Illustration of the goals and outcomes for each phase of the SNEWS-SCiMMA collaboration. Each phase proceeded after a progress review and consensus within each organization.

messages with the SNEWS decision making framework.[56] This established the system's baseline prior to any collaborative development and gave SNEWS the confidence to rely on external software.

Phase 1 was then planned around a goal to develop a prototype of the system running locally (Figure 5). To capture SNEWS's large goals from Phase 0, the developers created five broad Epics. These were translated into 10 User Stories over the course of Phase 1. Some of these Stories were relatively broad and took more than one sprint or developer to complete since the teams were learning how best to collaborate and manage the project. Nonetheless, eight of the Stories were completed by merging five pull-requests and resolving four issues during Phase 1, as illustrated in Figures 6 and 7. The teams assessed the progress of this phase via a demonstration of a prototype that used local containerized services and a local installation of the SNEWS 2.0 app. The prototype at this stage replicated most of the current SNEWS functionality. Both teams approved of the collaboration's progress: SNEWS had a new implementation of their server and SCiMMA improved the `hop-client` features beyond their initial scope. This phase and its success enhanced the trust between the two teams, encouraging the teams to agree to a second round of collaboration, Phase 2.

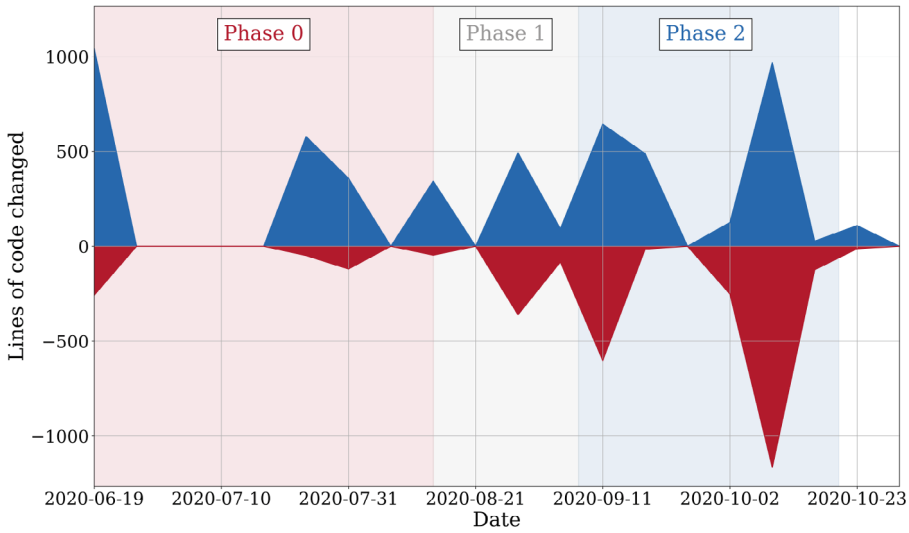

FIGURE 6 Lines of code changed during each phase of this case study. Positive values (blue) are the number of lines of code added to the SNEWS 2.0 codebase. Negative values (red) are the number of lines removed from the codebase.

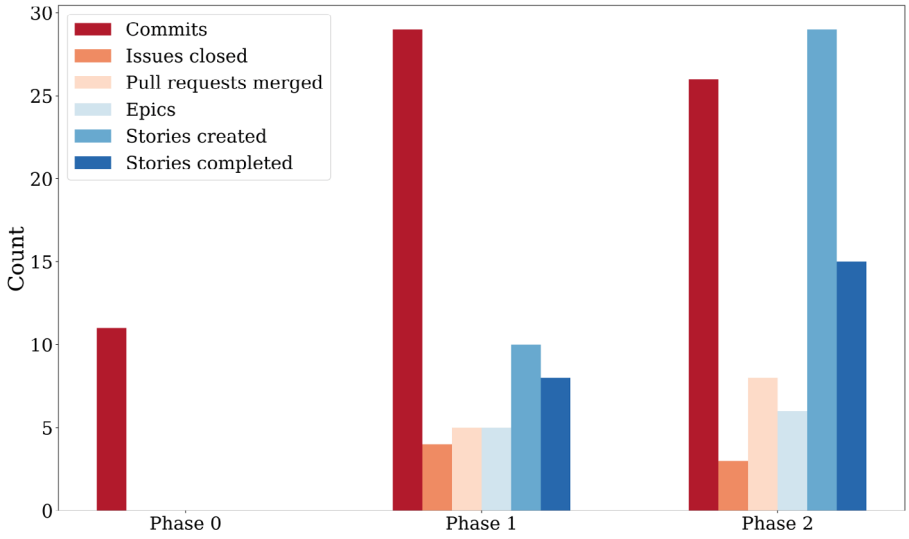

FIGURE 7 Figure displaying the software development and Agile framework metrics during each phase. We show the number of commits (dark red), issues resolved (orange), and merged pull requests (light red). Additionally shown are the number of epics created (light blue), user stories created (blue), and number of user stories completed (dark blue).

Phase 2 built on the Phase 1 goal of a local prototype by extending the prototype using distributed systems, with cloud servers to demonstrate robustness of remote but integrated components (Figure 5). The developers reviewed the remaining Epics and User Stories from Phase 1, and determined that two Epics were no longer in-scope with the teams' goals after the Phase 1 demonstration. The remaining three Epics and two User Stories were determined to still be relevant and were carried over into Phase 2. This phase began with six new Epics based on the assessment of the Phase 1 demonstration, focusing on measuring the prototype's performance and improving client authentication. This phase allowed SCiMMA to test a service that used their cloud servers and SNEWS was able to create a fully functioning prototype. The Epics were translated into 29 User Stories over the course of Phase 2. Of these, 15 were completed by merging eight pull-requests and resolving three issues during Phase 2. Both organizations reconvened for a demonstration of the remote prototype, which met the initial goal and concluded Phase 2.[59] After the demonstration, seven of the nine Epics were completed. The remaining two were either ill-defined or out-of-scope for the collaboration. Fourteen of the 29 User Stories were incomplete: five *Backlog*, six *Scheduled*, and three *In-Progress*. The developers continued working on four of these User Stories independently after the collaboration; the remaining 10 User Stories were considered out-of-scope for the collaboration period since the desired Epics had already been met.

The three phases resulted in a variety of development metrics that progressively evolved during the collaboration. In particular, while the SNEWS developers were able to create a prototype independent of SCiMMA by making moderate additions to the codebase (Phase 0 of Figure 6), the collaboration with SCiMMA (Phases 1 and 2 of Figure 6) demonstrably accelerated development. In particular, Phase 1 resulted in roughly half the amount of the codebase changes made during Phase 0 in a quarter of the time (2 weeks, compared to eight); Phase 2 then led to even more changes than either Phase 0 or Phase 1. While this may in part be due to the simple increase of the number of developers, Figures 6 and 7 suggest that the Agile practices and cooperation could offer added benefits.

The progression of the phases reflects the evolving Agile coordination in this collaboration. Phase 1 was fruitful, but remained a short-term trust-building exercise. The demonstrated success of Phase 1 encouraged both teams to invest more resources, effort, and time into the collaboration and provided a basis for Agile methods that Phase 2 continued to develop. This is shown in the evolution of Agile metrics across each phase in Figure 7. The results of Agile practices remained steady during Phase 1 and 2, wherein the 4-week Phase 2 yielded roughly twice the number of Epics and User Stories compared to the 2-week Phase 1. This demonstrates that the short-term success of Phase 1 could be extrapolated to a longer engagement; note, however, that Phase 2 involved more organized Agile practices compared to Phase 1, which may suggest that the less-formal implementation of Agile in Phase 1 may not have been adequate for a longer phase.

6.2 | Future directions

After the demonstration and evaluation of Phase 2, the collaboration had successfully produced a prototype of SNEWS 2.0 using SCiMMA's `hop-client` publish-subscribe system.[12] A schematic of the prototype is shown in Figure 8. The prototype established directions for future development for each group, allowing the organizations to remain in contact even after the phases of focused collaboration had concluded. It has provided direction for SCiMMA's authentication and permission management infrastructure, and has helped SNEWS begin prototyping their detector network using the SCiMMA `hop-client`. The next steps after the development of the prototype include having SNEWS members from outside the development team test the software using tutorials provided by the SCiMMA developers to ensure that the software is user-friendly. After this testing, SNEWS plans to integrate all of its neutrino experiments into the SNEWS 2.0 network. This will happen during a dedicated workshop wherein SNEWS will implement Agile methods, similar to those utilized during Phase 2 to drive the development. Specifically, the workshop will take place over a 4-week sprint and will include daily check-in meetings to discuss progress. Additionally, Epics and User Stories will be defined to keep the integration plans on track.

The collaboration promoted the general development efforts for each group. SNEWS utilized SCiMMA's systems and software engineering expertise to quickly develop their prototype, allowing SNEWS to avoid the difficulties of scientific computing and software development. The collaboration also unexpectedly accelerated the pace of SCiMMA's core development since SNEWS' use-case contained features that were of broader MMA interest. Additionally, it provided an exemplar for future engagements: SCiMMA plans to integrate Agile in future collaborations within the MMA community.

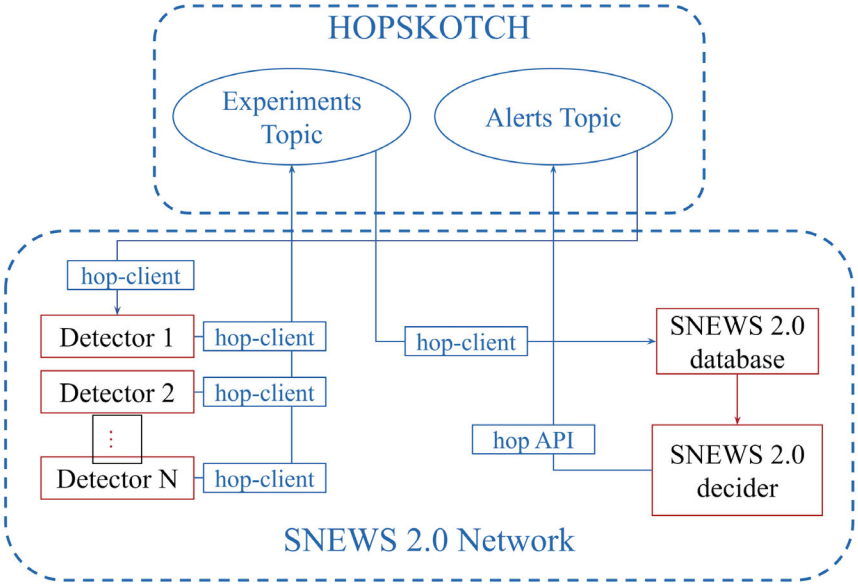

FIGURE 8 Schematic of the SNEWS 2.0 prototype application. The application was developed as an extension of the hop application template to contain a database and coincidence-checking system.

6.3 | Lessons learned

The success of this collaboration relied heavily on the Agile Scrum framework. The implementation of Agile practices in our work provided several key elements that ameliorated the difficulties of scientific software development as well as the concerns that emerged from each organization, which suggests that some of the Agile practices specific to our collaboration may be broadly relevant to the scientific software community.

First, the daily Slack check-ins and weekly sprint meetings facilitated constant communication between otherwise independent developers. This ensured that the emergent concerns and evolving scientific requirements of SNEWS and SCiMMA were brought up and addressed. The team found that the week-long sprints and User Story planning helped identify achievable tasks within broader goals of the collaboration. In particular, the sprint and phase retrospective meetings were crucial to the success of this collaboration by allowing the team to frequently evaluate its progress and the resulting software to ensure that the needs of both SNEWS and SCiMMA were being addressed. The progress reports at the ongoing SCiMMA sprint within phases and the collaboration-wide leadership meetings between phases ensured that the separate sprints remained on track. Having persistent communication channels was also essential since the development team was geographically distributed and entirely remote for the duration of the collaboration.

Second, implementing an Agile framework accelerated the trust establishment. Trust in the Agile process itself was also essential to the collaboration, since while the SCiMMA developers were already familiar with Agile methods, the Scrum Master did not have prior experience with running a sprint and the SNEWS developers were entirely unfamiliar with Agile. The sprint meetings and required daily communication ensured that each member had regular correspondence with the others, which allowed the otherwise unacquainted teams to quickly meet and trust the other remotely. This demonstrated the advantages of Agile to the collaboration, enhancing trust in the sprint process itself. Additionally, the trust establishment from the Agile framework built upon the shared scientific backgrounds that the two teams had: nearly all of the developers were already involved in scientific computing and/or MMA. The Agile sprint process capitalized on this mutual scientific language to further enhance trust and communication regarding the scientific software development effort. In particular, the SNEWS Product Owners were able to rapidly convey the technical alerting requirements of the application to the SCiMMA developers since the developers were already familiar with alert software in another area of astrophysics. Likewise, the SCiMMA developers brought technical software engineering expertise that allowed them to work with SNEWS computer scientists and system administrators. In particular, they worked with the computer science student developer to enhance the packaging, documentation deployment, and unit testing systems for the software, as well as the academic IT specialist to deploy the software on local hardware.

Third, the coordination of two Agile processes was novel to both organizations. We believe that adequate trust evolved nonetheless due to two reasons. First, the Agile process continuously generated measurable artifacts in the form of User Stories. The Product Owners and various stakeholders auditing the process could see, in real time, the progress and thus were willing to increase their trust. The second reason is that the coordination increased communication between the different organizations, which was particularly important since the development teams were involved in separate sprints and Agile alone is intended to orchestrate a single sprint team. By coordinating the downstream process via User Stories generated from the upstream process and holding cross-sprint check-ins, we created a method to communicate efficiently between the development teams.

7 | RELATED WORK

Our work serves as another case study within the rich ecosystem of scientific software development,[2,25] Agile methods utilized by scientists,[19,23] and nontraditional Agile processes for software development.[20,21,60]

Our work shares much with the work described in Reference 25 in that both describe experiential case studies involving interactions between scientists and software engineers. Our work differs in that the teams involved do not belong to a single organization nor are the teams focused on the completion of a single goal. Further, the author of Reference 25 cites difficulty in creating effective specifications as a key impediment to a successful collaboration, while for our project, the adoption of Agile alleviated this difficulty. However, the collaborative element of our Agile process introduced additional risks to the success of our work that were not covered in these case studies.

In Reference 2, the authors conduct and report on a survey of scientific development teams and organizations to determine how these teams build software. They report features of scientific development described in their survey that our collaboration also embodies, such as the "trust" in community-maintained software when the community is large, such as SCiMMA's use of public Kafka libraries.[61] Our work is largely complementary, however, as we adopted a specific approach without a controlled comparison with other alternative approaches. As such, our work would constitute a single, successful data point within this previous survey.

In Reference 19, the authors survey climate scientists to determine their common software development practices. They find that, philosophically and culturally, many development teams resonate with the tenets of Agile development. Our work seems to support this hypothesis, albeit in a different scientific domain, in that the teams involved in our study successfully adopted an Agile framework without much difficulty.

Our work also proposes a coordination mechanism between separate but dependent Agile projects wherein development of the "upstream" project (SCiMMA) must proceed ahead of the "downstream" project (SNEWS 2.0) that depends on it. Existing uses of this approach were not found in the literature, but there are several similarly nontraditional implementations of Agile processes involving novel sprint structures. For example, the authors of Reference 60 propose the "Scaled Agile Framework" (SAFe) as an additional Agile abstraction to extend Scrum for the coordination of multiple Agile teams. Our work differs in two key ways. First, we articulate a specific methodology based on the existing Agile abstractions of User Stories, Epics, and Backlog. In our case study, the upstream project developed Epics and User Stories that were incorporated into its own Backlog, but the completion of those Backlog tasks would then prompt the creation of User Stories in the downstream Backlog. Additionally, some User Stories generated by the downstream project were transferred to the Backlog of the upstream project based on their broader relevance to the upstream project's users. Second, Reference 60 implicates a more complex structure of organizational dependencies between multiple teams, whereas our case study examines only a single dependency in a bilateral context.

In Reference 20, the authors propose a combination of Waterfall and Agile methods for software development. This is similar to our case study: while we did not formally implement Waterfall methods, the phase-driven approach of our collaboration is comparable to the phase-gate Waterfall process discussed in Reference 20. After each phase, a meeting with leadership members was held in order to assess the status of the project and whether to proceed or halt the collaboration, akin to the "Go/Kill/Hold/Recycle" options after each phase-gate. This functioned to steer the outcomes of the Agile development phases without predefining the software specifications or development tasks that evolved out of the Agile process, similar to the proposed Project Plan in Reference 20.

In Reference 21, the author proposes a "MetaScrum" process in the context of a single Agile development team involving multiple overlapping sprints. While this implementation differs from our case study by considering only one Agile team, it bears similarities in terms of introducing novel Agile processes to coordinate multiple overlapping sprints. Specifically, our case study required coordination and reporting of progress from the downstream sprint (1-week cadence) to the

upstream sprint (2-week cadence). While a collaboration-wide MetaScrum meeting was not implemented to manage this coordination, the progress of the downstream project was discussed at meetings involving leadership and management for each organization. In between the collaboration phases, the Agile team met with leadership from each organization to review progress; during phases, the Scrum Master reported progress during SCiMMA sprint reviews and weekly collaboration meetings, and the two Product Owners reported progress at SNEWS monthly team meetings. These processes allowed for broader coordination of the sprints between the organizations, but differed from MetaScrum in that they were not specifically designed with sprint coordination in mind, nor structured to directly solicit feedback from organization leadership.

8 | CONCLUSION

The development of software to support collaborative science projects requires risk mitigation between the teams involved in order to bridge the chasm between scientific computing and software engineering. This case study describes the risks within a collaborative scientific software development effort and the ways in which the collaboration members confronted them. In this case study, the emergent concerns and needs of each organization were addressed by adopting Agile Scrum practices with an active dialogue between organizations. By working in week-long sprints, the high-level goals of each organization were translated into manageable development tasks. By demonstrating the final product after each phase, the Product Owners could re-evaluate the collaboration's status and the resources required for personnel time and effort; this resulted in enhanced trust in the collaboration. The software developers utilized the practical use-case from the science team to further its software and infrastructure to support the broader MMA community. The science team gained a functional and maintainable prototype of their desired software while also gaining valuable software development experience. The collaboration allowed both organizations to gain useful experience and furthered their science programs in ongoing and future work. The experience also demonstrated several themes of broader interest for the scientific computing community:

- Focused, bi-lateral collaboration can streamline development and improve communication of software requirements between users and developers, beyond the traditional open-source pull-request development model.
- The constant communication and incremental tasking promoted by Agile practices allow for software development to be flexible to rapidly changing requirements, which is particularly important when creating scientific applications.
- Collaborations between scientists and software developers can accelerate the creation of scientific software applications while also promoting knowledge transfer between domains.
- Coordination between separate, but dependent, Agile processes can be done without additional Agile abstractions, but this may require additional planning and communication methods to ensure progress for both teams involved.

The framework described herein proved to be beneficial in this study, which involved $\mathcal{O}(10)$ people across two teams and time frames $\mathcal{O}$(months). Note that these strategies may not generalize to larger-scale or well-established development projects, which may struggle to either implement Agile practices or respond to rapidly emerging software requirements. However, these practices could still alleviate some of the general difficulties of scientific software engineering, such as interdisciplinary communication, creating sustainable software without relying on external code, and constrained resources and timelines. These emergent and iterative frameworks can additionally support the need for rapid results and ever-changing requirements of the scientist user-group community. This case study indicates that a collaborative development approach centered on an Agile Scrum framework can benefit both the scientific user-group and the software developer team while mitigating some of the difficulties facing the scientific software engineering community.

AUTHOR CONTRIBUTIONS

The alphabetic-order author list reflects a variety of contributions from members in the SCiMMA and SNEWS organizations. Particular contributions to this article and the SCiMMA-SNEWS collaboration are noted as follows. BC, ALB, and RW made primary contributions to the text. BC and ALB led the demonstrations after each phase of development. BC was the Scrum Master who organized and led the development sprints, supported software development during Phases 1 and 2, and implemented the software on cloud infrastructure. ALB served as a SNEWS product owner, supported software

development during Phases 0, 1 and 2, and contributed substantially during development sprints. YX led the SNEWS software development process during Phase 0, 1 and 2. PG led the SCiMMA software development during Phases 1 and 2. Together, PG and YX contributed substantially during development sprints. MWL supported software development during Phases 1 and 2, and implemented the software on prototype infrastructure. AH served as a SNEWS product owner and provided substantial input during development sprints. AB, AH, MJ, RFL, CDT and RW each provided substantial input during progress reviews after each phase. AB, MJ, and CDT proposed the initial collaboration at a National Science Foundation "Harnessing the Data Revolution" grant proposal meeting.

The SCiMMA Agile Scrum team (AB, DC, BC, VE, SE, PG, DAH, MJ, WL, CM, SN, DP, RT, CW, and RW) supported core SCiMMA software development prior to and during the collaboration. Additional SCiMMA members (MWGJ, CK) have continued support of the SCiMMA project. The SNEWS Signal Prediction, Alert Formation, and Follow-up working group members (ALB, SYB, WB, MC, AC, MCM, ADO, DD, AGR, SG, AH, RH, SH, JPK, AK, VK, ML, RFL, SL, ML, MWL, JM, DM, RN, EO, BWP, NJ, AR, JR, TS, JCLT, CDT, CFV, CJV, KEW, LW, XJX, YX) supported SNEWS software development prior to and during the collaboration.

AFFILIATIONS

[1]Department of Physics and Astronomy, Purdue University, West Lafayette, Indiana, USA

[2]Department of Physics and Astronomy, University of Rochester, Rochester, New York, USA

[3]INFN Sezione di Cagliari Istituto Nazionale, Complesso Universitario di Monserrato, Monserrato, Cagliari, Italy

[4]Center for Advanced Computing, Cornell University, Ithaca, New York, USA

[5]Université de Paris, CNRS, AstroParticule et Cosmologie, Paris, France

[6]Las Cumbres Observatory, Goleta, California, USA

[7]Instituto de Física Corpuscular (CSIC – Universitat de València), Paterna, Valencia, Spain

[8]Institute for Computational and Data Sciences, The Pennsylvania State University, University Park, Pennsylvania, USA

[9]Department of Physics, The Pennsylvania State University, University Park, Pennsylvania, USA

[10]Instituto de Físic, Pontificia Universidad Catolica de Chile, Santiago, Región Metropolitana, Chile

[11]Aix Marseille Univ, CNRS/IN2P3, CPPM, Marseille, France

[12]National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA

[13]Computer Science Department, University of California Santa Barbara, Santa Barbara, California, USA

[14]Department of Physics, Laurentian University, Sudbury, Ontario, Canada

[15]Institute for Gravitation and the Cosmos, The Pennsylvania State University, University Park, Pennsylvania, USA

[16]Department of Physics and Astronomy, University of Minnesota Duluth, Duluth, Minnesota, USA

[17]Center for Neutrino Physics, Department of Physics, Virginia Tech, Blacksburg, Virginia, USA

[18]DiRAC Institute and the Department of Astronomy, University of Washington, Seattle, Washington, USA

[19]Department of Physics, NC State University, Raleigh, North Carolina, USA

[20]Department of Physics, University of California San Diego, La Jolla, California, USA

[21]Department of Physics and Astronomy, Michigan State University, East Lansing, Michigan, USA

[22]INFN Sezione di Genova, Genova, Italy

[23]Dipartimento di Fisica, INFN Sezione di Padova & Università di Padova, Padova, Italy

[24]Department of Physics, King's College London, London, UK

[25]Baksan Neutrino Observatory, Institute for Nuclear Research of Russian Academy of Sciences, Neytrino, Kabardino-Balkarian Republic, Russia

[26]Department of Physics and Astronomy, Uppsala University, Uppsala, Sweden

[27]Department of Physics, British Columbia Institute of Technology, Burnaby, British Columbia, Canada

[28]TRIUMF, Vancouver, British Columbia, Canada

[29]Department of Physics, University of Houston, Houston, Texas, USA

[30]SNOLAB, Sudbury, Ontario, Canada

[31]Department of Physics, University of Oxford, Oxford, UK

[32]Department of Computer Science, Rice University, Houston, Texas, USA

[33]Department of Physics and Astronomy, Rice University, Houston, Texas, USA

[34]Department of Physics, University of Torino & INFN, Torino, Italy

[35]Institute for Cyber-Enabled Research, Michigan State University, East Lansing, Michigan, USA

[36]Department of Physics, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

[37]Service de Physique Théorique, Université Libre de Bruxelles, Bruxelles, Belgium

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation "Windows on the Universe: The Era of Multi-Messenger Astrophysics" program: "WoU-MMA: Collaborative Research: A Next-Generation SuperNova Early Warning System for Multimessenger Astronomy" through grants 1914448, 1914409, 1914447, 1914418, 1914410, 1914416, 1914426; and via the "HDR-Harnessing the Data Revolution" program, grant 1940209; and with the "Office of Advanced Cyberinfrastructure" through grant 1934752.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available on Github at https://doi.org/10.5281/zenodo.4437579.[12]

ORCID

Amanda L. Baxter https://orcid.org/0000-0003-0048-6402
Bryce Cousins https://orcid.org/0000-0002-7026-1340

REFERENCES

1. Brooks FP. The computer scientist as toolsmith II. *Commun ACM*. 1996;39(3):61-68. doi:10.1145/227234.227243
2. Hannay JE. How do scientists develop and use scientific software? SECSE'09; 2009:1-8; IEEE Computer Society.
3. Merali Z. Computational science: error. *Nature*. 2010;467:775-777. doi:10.1038/467775a
4. US RSE: The US research software engineer association; 2021. https://us-rse.org
5. Wilson G. Best practices for scientific computing. *PLOS Biol*. 2014;12(1):1-7. doi:10.1371/journal.pbio.1001745
6. Goble C. Better software, better research. *IEEE Internet Comput*. 2014;18(5):4-8. doi:10.1109/MIC.2014.88
7. Gousios G, Pinzger M, van Deursen A. An exploratory study of the pull-based software development model. Proceedings of the 36th International Conference on Software Engineering; 2014:345-355.
8. Kelly DF. A software chasm: software engineering and scientific computing. *IEEE Softw*. 2007;24(6):120-119. doi:10.1109/MS.2007.155
9. Storer T. Bridging the chasm: a survey of software engineering practice in scientific programming. *ACM Comput Surv*. 2017;50:1-32. doi:10.1145/3084225
10. Al Kharusi S. SNEWS 2.0: a next-generation supernova early warning system for multi-messenger astronomy. *New J Phys*. 2021;23(3):031201. doi:10.1088/1367-2630/abde33
11. Chang P. Cyberinfrastructure requirements to enhance multi-messenger astrophysics. *Bull Am Astron Soc*. 2019;51(3):436.
12. Godwin P. SNEWS2/hop-SNalert-app: release 1.0 for Zenodo reference. 10.5281/zenodo.4437579; 2021
13. SCiMMA Collaboration. HOPSKOTCH; 2021. https://hop.scimma.org
14. Bennett K. Legacy systems: coping with success. *IEEE Softw*. 1995;12(1):19-23. doi:10.1109/52.363157
15. Segal J, Morris C. Developing software for a scientific community: some challenges and solutions. In: Leng J, Sharrock W, eds. *Handbook of Research on Computational Science and Engineering: Theory and Practice*. IGI Global; 2012:177-196.
16. Adenowo AA, Adenowo BA. Software engineering methodologies: a review of the waterfall model and object-oriented approach. *Int J Sci Eng Res*. 2013;4(7):427-434.
17. Beck KM. Manifesto for agile software development; 2013. https://agilemanifesto.org
18. Kane D. Agile methods in biomedical software development: a multi-site experience report. *BMC Bioinform*. 2006;7:273. doi:10.1186/1471-2105-7-273
19. Easterbrook SM, Johns TC. Engineering the software for understanding climate change. *Comput Sci Eng*. 2009;11(6):65-74.
20. Schuh G, Rebentisch E, Riesener M, Diels F, Dölle C, Eich S. Agile-waterfall hybrid product development in the manufacturing industry—Introducing guidelines for implementation of parallel use of the two models. Proceedings of the 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM); 2017:725-729; IEEE.
21. Sutherland J. Future of scrum: parallel pipelining of sprints in complex projects. Proceedings of the Agile Development Conference (ADC'05); 2005:90-99; IEEE.
22. Schwaber K. *SCRUM Development Process*. Springer; 1997.
23. Sletholt M. A literature review of agile practices and their effects in scientific software development. Proceedings of the International Conference on Software Engineering; 2011. doi: 10.1145/1985782.1985784
24. Segal J. Models of scientific software development. Proceedings of the 1st International Workshop on Software Engineering in Computational Science and Engineering SECSE 08, Workshop Co-located with ICSE 08; 2008. http://oro.open.ac.uk/17673/
25. Segal J. When software engineers met research scientists: a case study. *Empir Softw Eng*. 2005;10:517-536. doi:10.1007/s10664-005-3865-y

26. Crabtree CA, Koru AG, Seaman C, Erdogmus H. An empirical characterization of scientific software development projects according to the Boehm and Turner model: a progress report. Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering; 2009:22-27.
27. Lane PC, Gobet F. A theory-driven testing methodology for developing scientific software. *J Exp Theor Artif Intell*. 2012;24(4):421-456.
28. Umarji M. Software engineering education for bioinformatics. Proceedings of the 2009 22nd Conference on Software Engineering Education and Training; 2009:216-223.
29. Antonioli P. SNEWS: the supernova early warning system. *New J Phys*. 2004;6:114. doi:10.1088/1367-2630/6/1/114
30. Nakamura K. Multi-messenger signals of long-term core-collapse supernova simulations: synergetic observation strategies. *Mon Not Roy Astron Soc*. 2016;461(3):3296-3313. doi:10.1093/mnras/stw1453
31. Ivezić Ž. LSST from science drivers to reference design and anticipated data products. *Astrophys J*. 2019;873(2):111. doi:10.3847/1538‐4357/ab042c
32. Abbott BP. Multi-messenger observations of a binary neutron star merger. *Astrophys J Lett*. 2017;848(2):L12. doi:10.3847/2041-8213/aa91c9
33. Graham MJ. The Zwicky transient facility: science objectives. *Publ Astron Soc Pac*. 2019;131(1001):078001. doi:10.1088/1538-3873/ab006c
34. Smith M. The astrophysical multimessenger observatory network (AMON). *Astropart Phys*. 2013;45:56-70. doi:10.1016/j.astropartphys.2013.03.003
35. Ageron M. The ANTARES telescope neutrino alert system. *Astropart Phys*. 2012;35:530-536. doi:10.1016/j.astropartphys.2011.11.011
36. Huerta E. Enabling real-time multi-messenger astrophysics discoveries with deep learning. *Nature Rev Phys*. 2019;1:600-608. doi:10.1038/s42254-019-0097-4
37. Aartsen MG. The IceCube realtime alert system. *Astropart Phys*. 2017;92:30-41. doi:10.1016/j.astropartphys.2017.05.002
38. International astronomical union R. The transient name server; 2015. https://www.wis-tns.org/
39. Barthelmy SD. GRB coordinates network (GCN): a status report. *AIP Conf Proc*. 2000;526(1):731-735. doi:10.1063/1.1361631
40. Gabriel C, Arviset C, Ponz D, Enrique S. Astronomical data analysis software and systems XV; 2006.
41. Rutledge R. The Astronomer's telegram: a web-based short-notice publication system for the professional astronomical community. *Publ Astron Soc Pac*. 1998;110:754. doi:10.1086/316184
42. Allen G. Multi-messenger astrophysics: harnessing the data revolution; 2018.
43. Abbott BP. Multi-messenger observations of a binary neutron star merger. *Astrophys J Lett*. 2017;848(2):L12. doi:10.3847/2041-8213/aa91c9
44. SCiMMA Collaboration. SCiMMA google groups; 2020. https://groups.google.com/g/scimma
45. SCiMMA Collaboration. SCiMMA virtual workshop; 2020. https://scimma.org/events/virtual2020
46. Aasi J, Abbott B, Abbott R, et al. Advanced LIGO. *Class Quant Grav*. 2015;32(7):074001.
47. Aartsen MG, Ackermann M, Adams J, et al. The IceCube neutrino observatory: instrumentation and online systems. *J Instrument*. 2017;12(3):P03012.
48. Godwin P. scimma/hop-client: v0.2 release of scimma/hop- client; 2020. 10.5281/zenodo.4033483
49. SCiMMA Collaboration. SCiMMA hop app template repository; 2020. https://github.com/scimma/hop-app-template
50. SCiMMA Collaboration. SCiMMA infrastructure repository; 2020. https://github.com/scimma/scimma-admin
51. Nanograv Collaboration. The NANOGrav 12.5 yr data set: search for an isotropic stochastic gravitational-wave background. *Astrophys J Lett*. 2020;905(2):L34. doi:10.3847/2041‐8213/abd401
52. Heroux MA, Willenbring JM. Barely sufficient software engineering: 10 practices to improve your CSE software. SECSE'09; 2009:15-21; IEEE Computer Society.
53. SCiMMA Collaboration. SCiMMA containerization repository; 2020. https://github.com/scimma/scimma-server-container
54. National Science Foundation. Harnessing the data revolution at NSF; 2020. https://www.nsf.gov/cise/harnessingdata/
55. Slack Technologies. Slack; 2021. https://slack.com
56. Depoian A Using SCiMMA architecture for SNEWS 2.0; 2020. https://www.youtube.com/watch?v=GJLTCycvknE
57. Zoom video communications. Zoom; 2021. https://zoom.us
58. GitHub. About project boards; 2021. https://docs.github.com/en/github/managing-your-work-on-github/about-project-boards
59. Cousins B, Depoian A. SNEWS 2.0 prototyping with SCiMMA; 2020. https://www.youtube.com/watch?v=ZZO4CIqUTcY
60. Brenner R, Wunder S. Scaled agile framework: presentation and real world example. Proceedings of the 2015 IEEE 8th International Conference on Software Testing, Verification and Validation Workshops (ICSTW); 2015:1-2.
61. Confluent Inc. Confluent's python client for apache Kafka; 2021. https://github.com/confluentinc/confluent-kafka-python

How to cite this article: Baxter AL, BenZvi SY, Bonivento W, et al. Collaborative experience between scientific software projects using Agile Scrum development. *Softw Pract Exper*. 2022;52(10):2077-2096. doi: 10.1002/spe.3120