Software Project Management for the Defense Advanced Research Projects Agency Spectrum Collaboration Challenge

Andrew S. Freeman, Anthony T. Plummer Jr., Jordan N. Kraus, Mark L. Plett, Kevin E. Parker, and David M. Coleman

ABSTRACT

Development and management of the Colosseum, the wireless communications research test bed for the Defense Advanced Research Projects Agency (DARPA) Spectrum Collaboration Challenge (SC2), was a complex undertaking. With its world-class expertise in communication systems and experience in information technology infrastructure, the Johns Hopkins University Applied Physics Laboratory (APL) was well positioned to design, host, and maintain the Colosseum on its Laurel, Maryland, campus from 2016 to 2019. The effort required close coordination among members of the APL team and between APL and DARPA. To effectively and efficiently manage the design and maintenance of the Colosseum, APL applied tested project management tools and techniques, a development and operations approach, and an agile framework. This article focuses on the early planning and initial development efforts and documents the project management attributes, including the composition of the APL team as well as the software tools, that contributed to the success of the effort.

INTRODUCTION

In 2016 the Defense Advanced Research Projects Agency (DARPA) launched the Spectrum Collaboration Challenge (SC2) program, seeking a new paradigm for efficiently allocating and sharing the increasingly crowded radio frequency (RF) spectrum. Competitors would develop adaptable radio technology powered by artificial intelligence (AI) and participate in events to test those solutions. At the foundation of the research, development, and testing of these solutions would be a remotely accessible test bed for wireless communication research. This test bed would be called the Colosseum, and it would include 128 software-defined radios (SDRs) connected through a wireless channel emulator[1] and would serve as a proving ground for machine-learning algorithms for collaborative spectrum access. The Colosseum would operate 24 hours a day, 7 days a week without an operator in the loop, and it would enable three formal competition events that would culminate in DARPA awarding over $17 million in prize money across an international field of entrants. This was DARPA's vision for the SC2 program in 2016.

To draw the eye of the research community, DARPA announced the SC2 schedule in July 2016. The competition included three phases. The first phase (September 2016 to December 2017) was dedicated to building the Colosseum and establishing a baseline of performance across the competitors. In the second phase (January 2018 to December 2018), the Colosseum's capabilities would be increased to push spectrum-sharing algorithms. The competition would culminate

in a final event at the end of phase 3 (January 2019 to October 2019).

APL, selected as the lead in development, integration, and operation of the Colosseum, was particularly well suited for this effort. As the nation's largest university-affiliated research center, APL has a long history with the US government as an independent trusted agent addressing hard problems facing the nation. DARPA's SC2 program demanded a team with a diverse skill set to innovate and integrate cutting-edge technologies to provide a testing ground for AI in next-generation communication systems. APL was able to contribute multiple skill sets across the life of the project. APL systems engineers, software developers, RF experts, field-programmable gate array specialists, and hardware and networking experts all came together to form one team to deliver the Colosseum. APL also has an extensive history in testing and evaluating large complex systems, making it an ideal organization to verify and validate the performance of third-party Colosseum components, such as the wireless channel emulator developed by National Instruments.[1] Moreover, by leveraging internal (yet separate) staff for verification and validation of end-to-end Colosseum operations, APL was able to reduce timelines and meet the competition schedule.

For SC2, APL faced two main project management challenges: adhering to a tight schedule and ensuring continuous Colosseum access for competitors. To address the schedule challenge, APL broke phase 1 into four major periods—procurement, development, integration, and test—as shown in Figure 1. The procurement period was scheduled to last from September through December 2016. As components arrived, development, integration, and testing started in January 2017, and the APL team maintained pace until the Colosseum officially opened in May 2017. This decomposition allowed APL to keep up with schedule demands while developing the brand-new remotely accessible research test bed.

Tasked with continuously delivering features and working toward hard dates yet facing "soft" and evolving requirements, APL applied a modified version of a scaled agile paradigm across multiple teams to develop, integrate, and test the Colosseum's capabilities and deliver new features on a continuous basis throughout the program. As shown in Figure 1, the development period continued in each phase of the program. The agile framework applied to the SC2 program is discussed in more detail in the next section.

APL was also required to provide customer support to the research teams and maintain continuous operations while developing and evolving the capabilities of the test bed. To meet these requirements, APL employed a development and operations (DevOps) approach. (See the article by Plummer and Taylor in this issue for more on DevOps.) Using DevOps, the APL team established virtual environments with replicated software libraries and testing components, enabling software developers to rapidly integrate and test new features before their release to users. The Colosseum remained online and open to competitors, with new feature releases (and bug repairs) deployed during regular announced weekly maintenance periods. APL established a help desk for customer support and, from 2016 to 2019, resolved over 5000 help requests.

This article describes how APL applied the agile development process to the SC2 program and discusses the tools the 50+-member team used to enhance productivity.

AGILE PROJECT MANAGEMENT

Starting in September 2016, APL adopted an agile methodology[2] that satisfied the rigorous SC2 program schedule while allowing for continuous requirement derivations with DARPA. Agile focuses on delivering

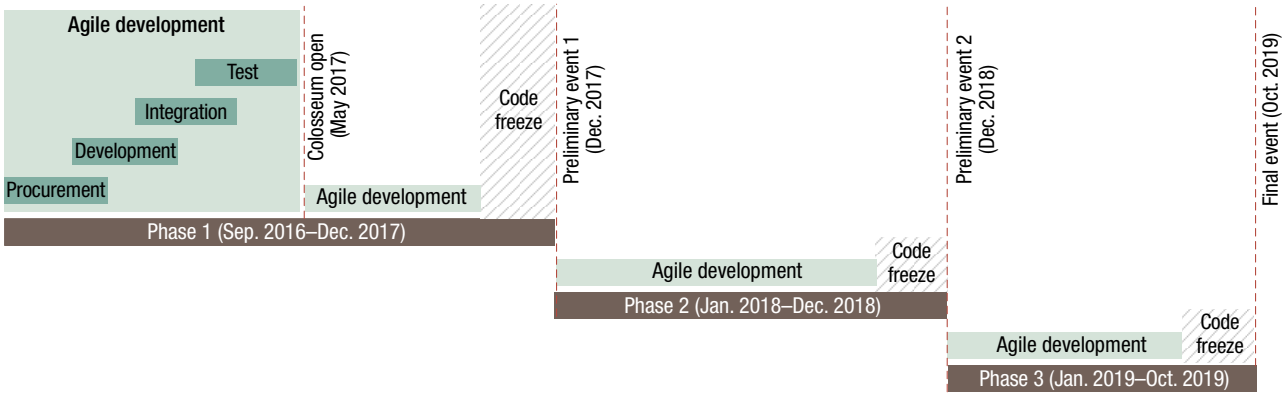


Figure 1. The SC2 program included three phases: In phase 1, the Colosseum was built and competitors' baseline performance was established. In phase 2, the Colosseum's capabilities grew to push spectrum-sharing algorithms. The competition culminated in a final event at the end of phase 3. APL further decomposed these phases to ensure that it remained on schedule and met program requirements while developing the Colosseum.

capabilities early and continuously, while allowing for changing requirements; this vision is achieved through constant communication among APL team members and collaboration with the customer.[2] This approach was best for SC2 since the capabilities of the Colosseum would have to rapidly evolve over the course of the program to meet the changing needs of the research community. This agile approach enabled DARPA to focus on the capabilities required at each stage of the competition rather than having to define all the capabilities at the onset of the program.

APL Team Composition

The APL team was organized into five functional teams, each with an eye to the test bed's end users. Each team had a shared vision, an appointed team representative for cross-functional team meetings, and an agile lead. The local product owner was appointed by DARPA to maintain the vision and produce ongoing requirements for the team. The five teams, illustrated in Figure 2, include:

1. **Competitor Experience (CE) team**—The 16-member CE team focused on developing the system components involving interaction with the competitors, such as the standard radio node (SRN) controller, the Resource Manager, the public-facing website, and the Traffic Generation System. (For more information on the SRN controller, the Resource Manager, the website, and the Traffic Generation System, see the articles by White et al., Mok et al., Coleman et al., and Curtis et al., respectively, in this issue.)

2. **Event Management (EM) team**—The 5-member EM team focused on planning SC2 events such as scrimmages and formal competitions. Conducted by APL before each preliminary event, scrimmages gave competitors the opportunity to build their radios up to the specifications required for participation in the preliminary events. Preliminary events were the formal events during which competitors executed their designs against each other within the parameters of scenarios designed to mimic real-world challenges a network of collaborative autonomous radios would have to overcome. (See the article by Coleman et al. in this issue for details on scrimmages, preliminary events, and scenarios.) The team also planned scenario development and management (including GPS reporting) and data management and distribution to users.

3. **RF Emulation System (RF-ES) team**—The 12-member RF-ES team focused on developing the RF Emulation System, which included integrating the wireless channel emulator from National Instruments.[1] (See the article by Barcklow et al. in this issue for details on this system.) Although competitors interacted with the RF Emulation System, this system was not part of the CE team's tasks; instead, it had a dedicated team because of its heavy dependence on external teams and the extensive wireless channel emulator integration effort.

4. **Colosseum Operations (CO) team**—The 4-member CO team focused on all aspects of operations, including cybersecurity, infrastructure monitoring, facilities, and networking. Additionally, the team was responsible for introducing the development tools, techniques, and automation needed to facilitate the DevOps process. (See the article by Plummer and Taylor in this issue for more detail on DevOps.)

5. **System Test and Competitor Support (ST-CS) team**—The 10-member ST-CS team focused on supporting competitor use cases on the Colosseum, including the competitor help desk and wiki, as well as testing new features before their release and verification testing of the Colosseum. ST-CS team members acted as internal beta testers and created incumbent systems for developers and competitors to use in the Colosseum for self-testing. (See the article by Yim et al. in this issue for details on incumbents.)

Because the Colosseum was an agile project, this team structure was subject to modification at each phase of the project. New team members were brought on as different skill sets were needed, and subteams were reorganized to facilitate the best application of skill sets to the current need.

Furthermore, working within the agile framework, the APL team adopted its own four core principles for success: colocation, self-management, cross-functionality, and commitment. First, all team members were colocated in a collaboration space. This allowed for personal and quick interactions among developers, increasing critical communication and reducing laborious formal documentation, which allowed for quick integration of simple and working code. Second, each team was self-managed with team representatives and team leaders acting as peers as well as focusing on cross-team synchronization and issues. In this approach, each team member was empowered to raise issues immediately, and each team member understood the overall vision for their team so that cross-team interactions occurred seamlessly (and not on the schedule of an individual). The team leads met daily to address cross-team issues on a regular cadence. Third, the team was cross-functional in the skill sets necessary to complete a task. For example, the RF-ES team comprised software developers, RF systems engineers, and embedded programmers, ensuring the wireless channel emulator's successful integration into the Colosseum. Fourth, the team attempted to get team members that

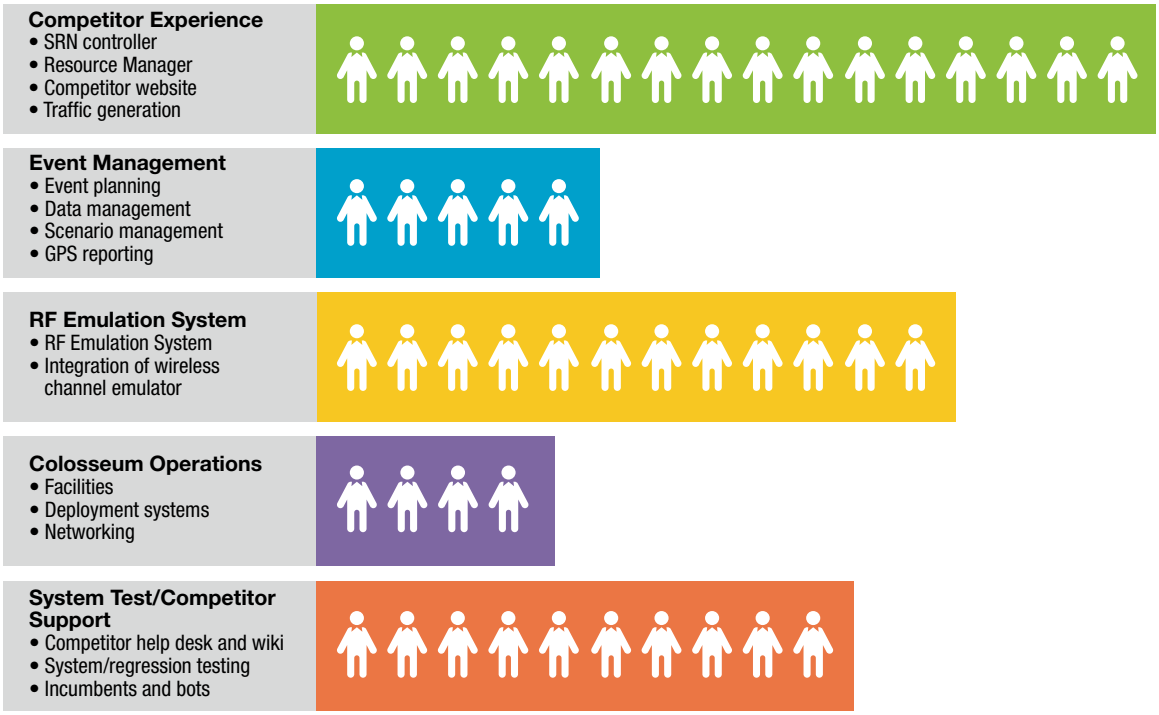


Figure 2. Team organization and overview of tasks. The larger APL team was organized into five functional teams, each with defined responsibilities to ensure a positive experience for Colosseum users and successful execution of SC2 events.

were assigned to the SC2 project as full-time staff and committed to the project for an entire phase whenever possible. This resulted in a cohesive team, where team members built trust in each other and were always available for critical project discussions.

Task Composition

The APL team defined a framework within the agile methodology to organize the work products for the functional teams. The SC2 project included four levels of task decomposition: initiative, epic, story, and subtask. Initiatives were defined by the capability, feature, or requirement provided by DARPA. Initiatives typically spanned multiple teams and months of work, so the APL team would decompose each into smaller epics. Epics were expected to define the tasking within each team, were required to satisfy the higher-level initiative, and were typically achievable within 12 weeks. Each epic was further decomposed into stories representing up to 2 weeks of work, fitting within a standard agile "sprint." Subtasks were the lowest level of decomposition, representing a short duration (1–2 days) of work. While there were many cross-team discussions regarding initiatives, epics, and stories, each team was free to use (or not use) subtasks to define their daily tasking, balancing the need to plan work versus to get work done. Tasks were tracked in Atlassian Jira. (See the section on Project Management Support Tools for more detail on Jira.)

Task Planning

The APL team devised its planning process for the Colosseum development, integration, test, and deployment in concert with the DARPA SC2 team. Planning was split across three phases: high-level planning, sprint planning and execution, and a sprint retrospective. High-level planning (level 0 and level 1) and sprint planning (level 2) occurred before the start of a task. The work was performed during sprint execution. And, finally, the sprint retrospective was a debriefing meeting that enabled the APL team to assess performance during the sprint, modify internal team practices, and set expectations for the next sprint. A diagram of these meetings is shown in Figure 3.

High-Level Planning

At the high-level meetings (level-0 meetings), DARPA and APL team leads discussed new Colosseum capabilities. During these meetings, each capability was defined as an initiative since it could span multiple months and require support from multiple APL teams. For example, as shown in Figure 4, one of the initiatives in phase 1 of the program was "SC2-3943: The Colosseum will support wireless channel emulation with a channel update rate of 1000 Hz." (SC2-3943 is a tracking number automatically assigned by Jira.) DARPA maintained the collection (i.e., project backlog) of initiatives and continuously reprioritized them during level-0 meetings. Also during these meetings, APL project manage-

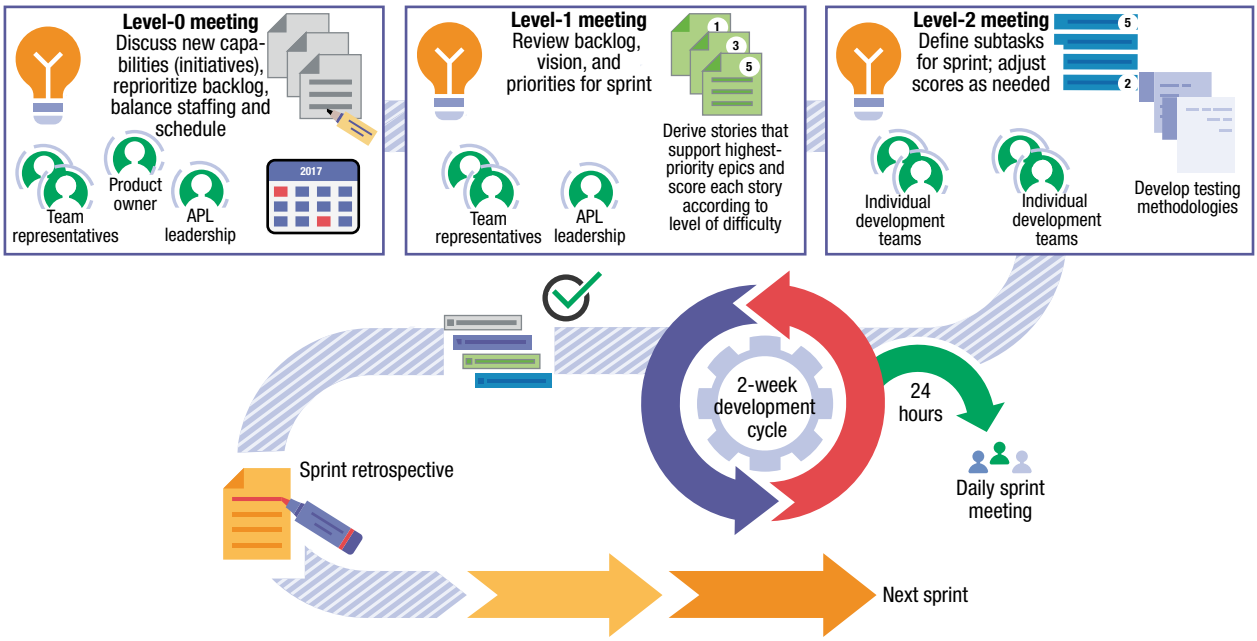


Figure 3. Agile approach used for SC2. The agile methodology based on a series of short fixed-length development cycles, called sprints, of promised scope and delivered features. Sprints were planned in a series of meetings, and they were evaluated in a sprint retrospective meeting to capture lessons before the start of the next sprint.

ment rebalanced the initiative's staffing and schedule (i.e., expectations) with DARPA.

At level-1 meetings, the APL team leads decomposed the initiatives into epics that were defined by cross-team efforts. For example, initiative SC2-3943 described above was decomposed into several epics including "SC2-756: As a Colosseum Administrator, I need to manage the wireless channel emulator" and "SC2-4383: As a System Test Engineer, I want to measure the performance of the wireless channel emulator." Each epic was placed on the project backlog for tracking. Since each epic depended on multiple team efforts, epics were further decomposed into individual team efforts, known as stories. Each story encompassed only work performed by one team and was assigned points based on difficulty and level of effort. For example, SC2-756 was supported by stories "SC2-4619: Define the interfaces on the wireless channel emulator" and "SC2-3759: Develop a software update procedure with external team members." SC2-4619 was assigned to the RF-ES team with 10 points, while SC2-3759 was assigned to the CO team with 5 points.

After stories were assigned during level-1 meetings, APL team leads met with their individual technical teams to define subtasks for the sprint. These meetings were referred to as level-2 meetings and gave team members opportunities to ask probing questions and express concerns about scope, requirements, integration, and testing for the stories. The team was free to adjust the

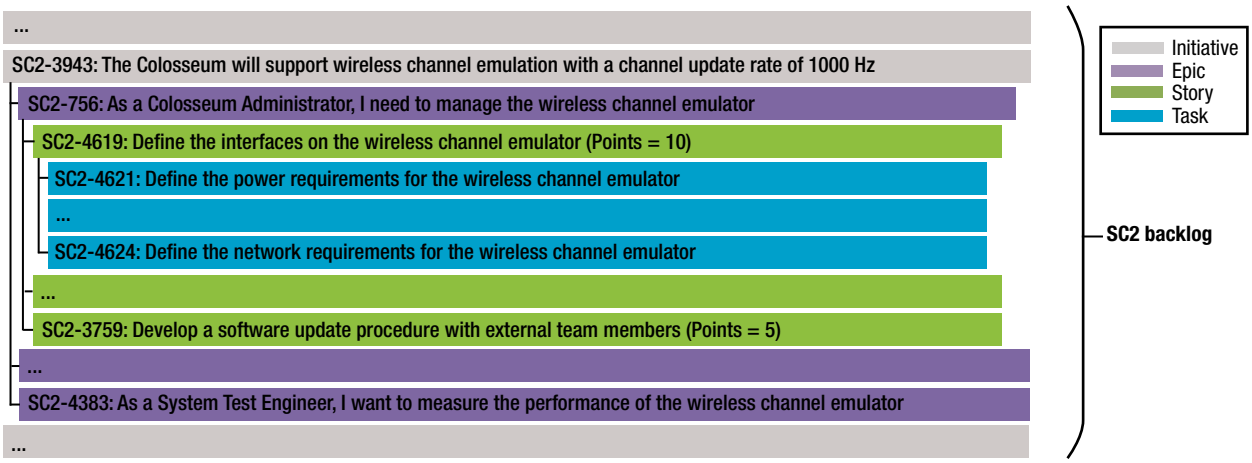


Figure 4. Task decomposition. The SC2 project included four levels of task decomposition: initiative, epic, story, and subtask.

expected level of difficulty (points) for the story and offer a collective estimate of the scope of each subtask. Since the technical team was represented in the level-1 meeting, large discrepancies in scores rarely occurred and, when they did, they were quickly mitigated since APL project leadership was colocated with subteams. Additionally, the output of level-2 meetings included testing methodologies for new features. The ST-CS team used the test methodology to evaluate features prior to release and the results were included in release notes for competitors. For example, SC2-4619 was decomposed into several individual subtasks, including "SC2-4621: Define the power requirements for the wireless channel emulator" and "SC2-4624: Define the network requirements for the wireless channel emulator." The individual subtask was then assigned to one team member and tracked for a finite time known as a sprint. Individuals were also assigned bugs to diagnose and fix during the sprint.

Sprints

Because of the size of the teams and the diverse nature of the skills needed to accomplish the tasks, APL used a large-scale Scrum (LeSS)[3] methodology to plan and execute work. Scrum is an agile methodology based on a series of short fixed-length development cycles, called sprints, of promised scope and delivered features. It uses the repeatability of the fixed delivery cycle to establish a team "velocity" that makes the amount of completed tasking more predictable. APL selected 2-week sprint intervals, and, therefore, the largest amount of work completed within a sprint would be defined by stories.

By having a 2-week development cycle, the teams were able to rapidly respond to changing requirements, introduce new aspects of the maturing vision, and address competitor feedback. Sprints were planned during level-2 meetings within each team and tracked in the daily scrum meetings.

Sprint Retrospective

The achieved velocity for the sprint was determined at the end of the sprint during the sprint retrospective. The retrospective was pivotal to identifying process issues in the previous sprint so that the team could adjust as needed in the next sprint. Example issues included latency in cross-team communication, unexpected integration challenges, and oversubscribing to difficult subtasks (i.e., poor story decomposition). During the retrospective, the team evaluated the sprint burn-down chart for unfinished work and scope creep (i.e., unplanned work). Figure 5a illustrates a poor sprint burn-down (sprint 7, December 7–21, 2016), where the sprint included many unplanned events and unsuccessful tasking. This is typical for teams first learning the Scrum approach and was remedied as the program continued. Conversely, Figure 5b shows a good sprint burn-down in sprint 14 (April 4–18, 2017). This sprint began with 435 points and successfully completed 403 points within the 2-week duration with a near-linear burn rate. The unfinished 32 points would be carried over to the next sprint if the priority was highest among the remaining items on the product backlog.

PROJECT MANAGEMENT SUPPORT TOOLS

There are many software tools available online for project management, including tools that support subtask tracking, team documentation and communications, and system administration. This section describes the tools used at APL for the SC2 program.

Atlassian Jira

Jira is a flexible browser-based application that enables project leaders to manage the development and release of capabilities by collaborating on user stories, sprint plans, and the distribution of subtasks across the software team.[4] The APL team used Jira during sprint

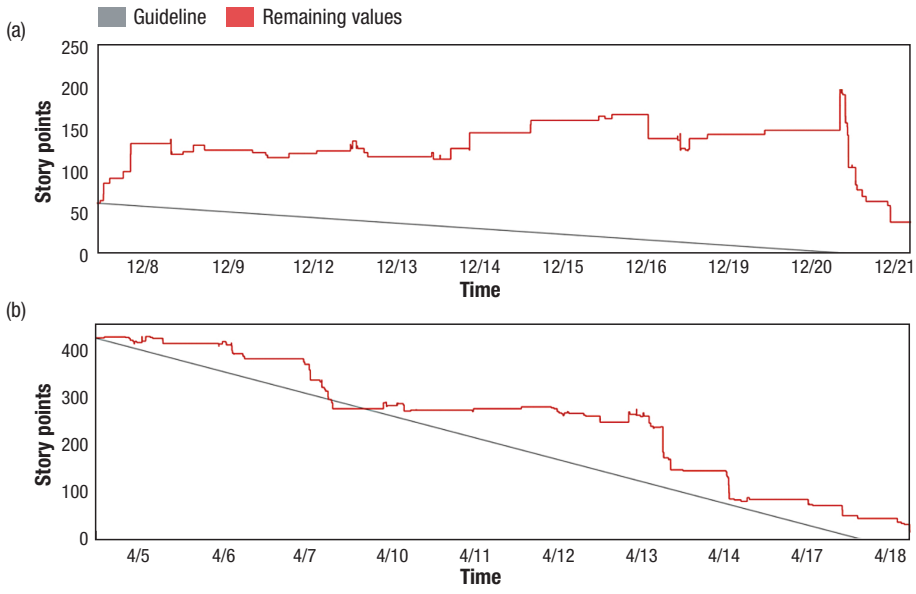


Figure 5. Sprint burn-down charts. Panel a shows a poor sprint burn-down, with the sprint including many unplanned events and unsuccessful tasking. Conversely, panel b shows a successful sprint burn-down.

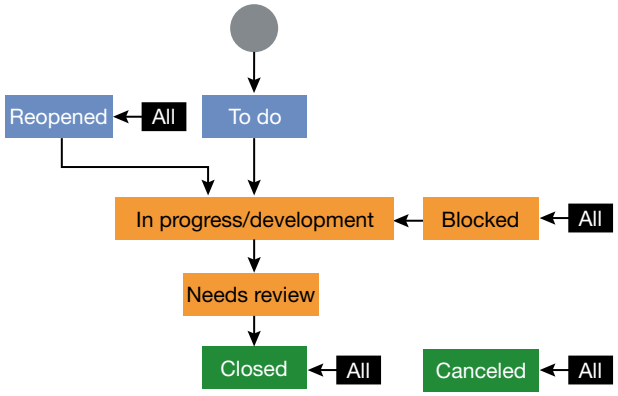


Figure 6. Jira workflow. Jira enabled project leaders to collaborate on features, sprint plans, and the distribution of subtasks across the software team.

planning meetings to create initiatives, epics, stories, and subtasks based on desired Colosseum capabilities, to track daily software developer tasking, and to schedule release dates for new features. Each story (i.e., feature) stepped through a rigorous workflow, outlined below, so that the team could make sure each feature was completely developed and tested before being released (see Figure 6):

- **To do/reopened**—Subtasks waiting to be worked on, in order of priority or dependency
- **In progress/development**—Subtask being worked
- **Needs review**—Subtask completed but needs approval and/or testing before acceptance
- **Blocked**—Subtask that cannot be worked on because of an outside dependency
- **Closed**—Complete tasks
- **Canceled**—Subtasks that are incomplete but no longer need to be completed

When a user enters information in Jira, the system creates a "ticket." Tickets are assigned a type that allows them to be grouped hierarchically. Similar to project decomposition, the types for tasking include story, task, bug, improvement, and feature. Each of these types can have a number of subtasks assigned to them, and these subtasks need to be completed before the parent task can be closed. Above the tasking types in the hierarchy is an epic ticket, which is used to categorize the other tickets; each task type can only be assigned to a single epic. While Jira tickets can be queried directly, the main advantage of the software is its ability to display the current tasking in agile boards that support Scrum task management (Figure 7).

So that it mirrored the project planning structure, Jira was modified with an additional ticket type, initiative, which was used as a parent type to epics so that they could be grouped together. Jira add-on software, called Structure, was used to display the hierarchical representation of the tasking, how child tasks rolled up, and the completion status of the tasks' parent tasks. Figure 8 shows an example hierarchical view of Structure.

From 2016 to 2019, the SC2 project board in Jira contained 66 initiatives; 501 epics; 5189 tasks, bugs, stories, features, and improvements; and 1687 subtasks. With its

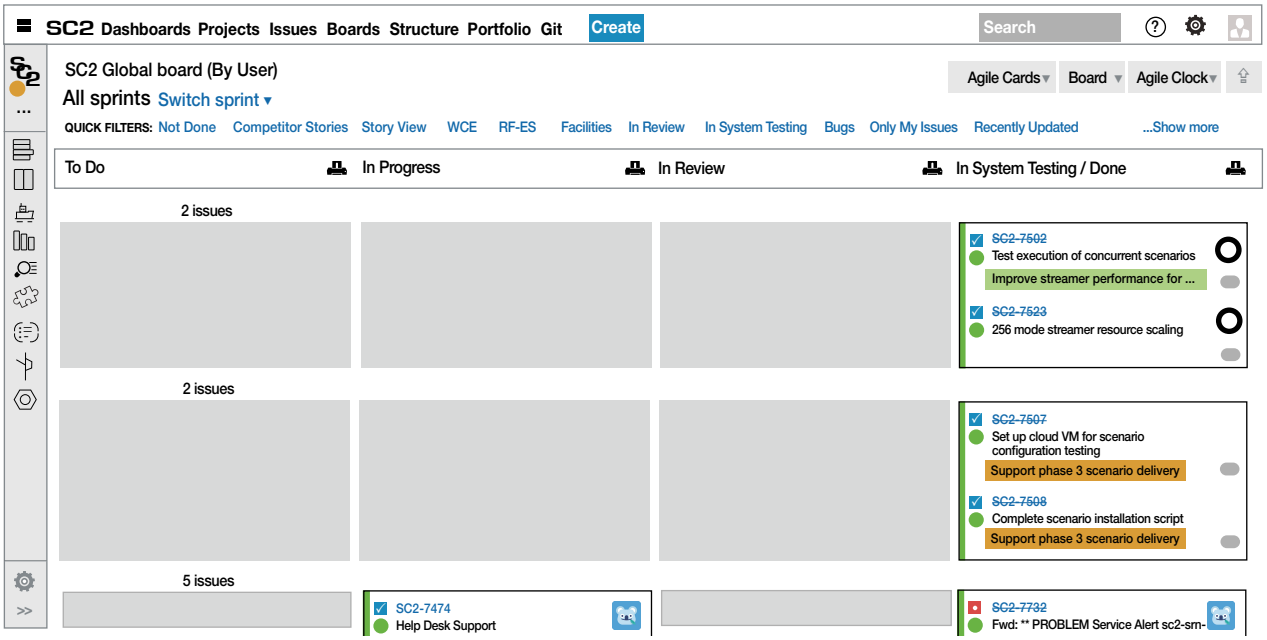


Figure 7. Jira Scrum board. One of Jira's main advantages is its ability to display the current tasking in agile boards that support Scrum task management.

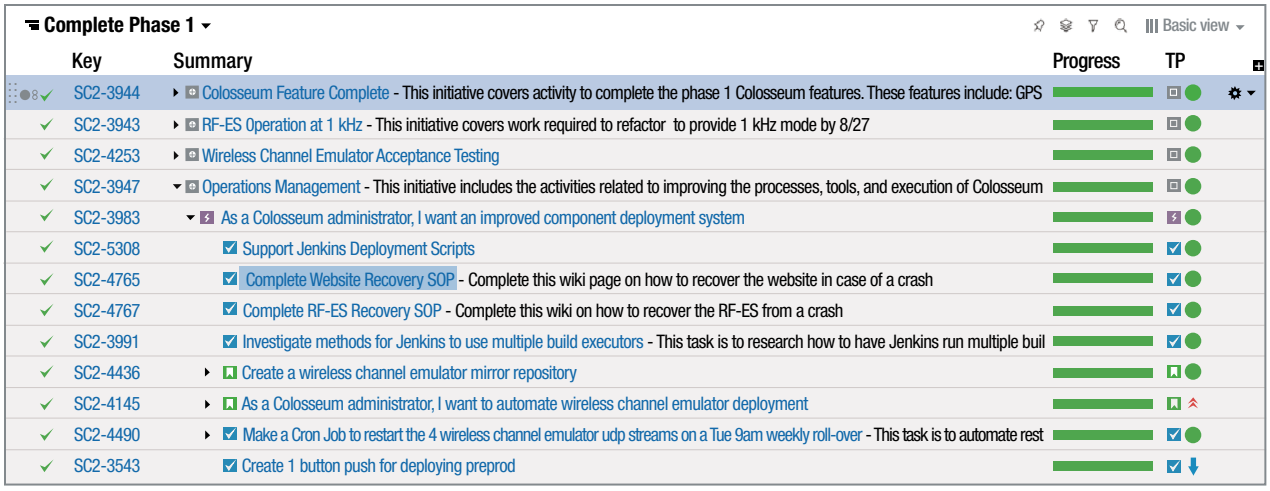


Figure 8. Structure hierarchical view. This Jira add-on displays the hierarchical representation of the tasking, how child tasks rolled up, and the completion status of the tasks' parent tasks.

flexibility and tracking of project activities, Jira was integral to managing and executing the LeSS agile methodology on the SC2 program.

Atlassian Confluence

Confluence[5] is a wiki system used to quickly share data among a team. Confluence was the tool used to capture all the high-level requirements, architecture designs, and level-1 planning. It was both a collaboration space for the team to communicate and an archive of captured domain and institutional knowledge.

The APL team also generated program documentation on the Confluence wiki. This method provided a single source for all project documentation, reducing integration time since developers had continuous access to the latest protocols and standards. The team used Confluence to generate and store interface documents, technical documents (e.g., software architecture design documents, facilities layout information, and networking documents), test plans and reports (e.g., verification and validation test plans, software functionality test plans, and competition readiness plans), and user manuals (e.g., how-to guides, shut-down and start-up procedures). Confluence integrates directly with Jira, allowing for quick construction of Jira status pages and release notes based on completed Jira tickets.

Slack

Slack is an online chat application featuring persistent chat rooms organized by topic, called channels. It also offers private groups and direct messaging. The APL team used Slack to stay in contact with team members,

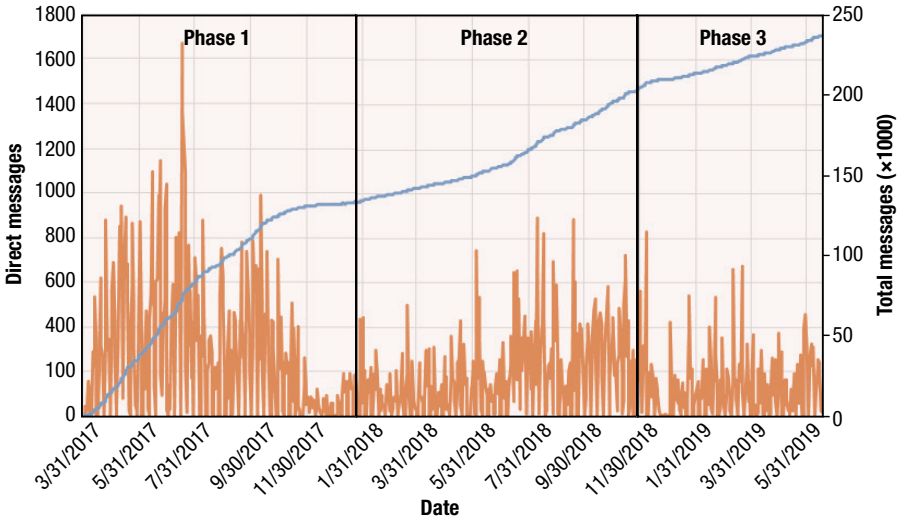


Figure 9. Slack statistics on direct messages. Of the 230,000+ messages posted in Slack, most were direct messages between developers. This communication between developers was critical to the success of the SC2 integration process.

rapidly address concerns, and notify team members of impending system changes. To streamline team communication, channels were created for major development components and different operational aspects. This structure allowed individual team members to subscribe and unsubscribe from different channels as their roles changed over the SC2 development cycle. Themed channels helped to cut down on miscellaneous notifications for sections of the project that a team member was not working on, as well as to focus conversations to a single topic at a time. While channels were used mainly for notifications, direct messages between members of the APL team were more conducive to active development. Over the course of the program (2016–2019), the majority of the 230,000+ messages posted in Slack were direct messages between developers (see Figure 9). Rapid, consistent, and direct communication between developers proved vital in the success of the SC2 integration process.

Jenkins

Jenkins is an open-source automation server for continuous integration and continuous delivery of software. By integrating Git[6] and Slack, APL was able to execute the process of building and deploying SC2 software with the push of a single button. By design, the internal SC2 network was configured to be isolated from the rest of the APL network and internet, with only one gateway between them. This architecture prevented direct access between the development Git repositories and the system hardware. Testing and deploying code across this multihop boundary is ideal for automation software such as Jenkins. Pipelines were created for each software component, which allowed APL team members to customize deployment and tests to meet their needs. In a pipeline that deploys SC2 software, the first step is for Jenkins to pull a specified branch from the repository onto the Jenkins server. The pipeline transfers the codebase across the network boundary and to the specific hardware destination. Once the codebase is in the final destination, the pipeline remotely runs a shell script configured to stop the old running processes, archive the old codebase, build the new codebase, and start the new processes or tests. The output of each stage in the pipeline is logged, and the final result is clearly displayed on the Jenkins website. A history of all pipeline executions is archived, making it easy to track down bugs and errors introduced in new code.

The tight integration of the project management tools, the team structure, and the LeSS agile framework created a successful development environment for the APL team. The flexibility of Jenkins and the integration of Git and Slack allowed for the rapid testing and deployment required to meet project deadlines, such as new code deployment during maintenance windows. The wiki implementation created a single knowledge base that aided team members during their ramp-up phase and was critical for the help desk team members. The wiki's integration with Jira made for easy-to-track development efforts and condensed new capabilities into release notes for competitors.

CONCLUSION

The construction and management of the Colosseum for DARPA's SC2 program presented many challenges for the APL team, including having to meet tight deadlines and adapt to rapidly changing requirements while needing to maintain a continuously operational system. The project management decision to implement the agile paradigm and Scrum methodology contributed to the team's ability to successfully meet evolving requirements on schedule. The team structure and time spent in sprint planning enabled the subteams to focus on development and to continuously add capabilities. The integration of project management tools aided rapid development and allowed the APL team to meet all its hard deadlines. The project management tools were integral for facilitating internal team communications and for quickly resolving competitors' requests for support. The strong project management construct leveraged throughout the three phases of SC2 was critical to the successful completion of the Colosseum, the world's largest wireless test bed.

ACKNOWLEDGMENTS: We thank Paul Tilghman (DARPA SC2 program manager) and Craig Pomeroy and Kevin Barone (Systems Engineering and Technical Assistance at DARPA) for their invaluable collaboration and support. We also thank the many APL SC2 contributors, whose names are listed on the inside back cover of this issue of the *Digest*. This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the US government.

REFERENCES

[1]A. Chaudhari, D. Squires, and P. Tilghman, "Colosseum: A battleground for AI let loose on the RF spectrum," *Microwave J.*, vol. 61, no. 9, Sep. 13, 2018.
[2]K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, et al., 2001. "Manifesto for agile software development." http://agilemanifesto.org/ (accessed Jul. 19, 2018).
[3]"LeSS." http://less.works/ (accessed Jul. 19, 2018).
[4]"Jira software." Atlassian. https://www.atlassian.com/software/jira (accessed Jul. 19, 2018).
[5]"Confluence software." Atlassian. https://www.atlassian.com/software/confluence (accessed Jul. 19, 2018).
[6]"GitLab." https://gitlab.com/ (accessed Jul. 19, 2018).

Andrew S. Freeman, Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Andrew S. Freeman is a software engineer and agile development coach in APL's Asymmetric Operations Sector. He has an MS in computer science from North Carolina State University. Andrew has been developing software in various domains for 20+ years. To the SC2 team he brought his experience establishing and running medium- to large-size teams using agile management techniques. His current focus is on creating workflow systems around human language technology (HLT) tools for batch audio analysis. He is a Certified ScrumMaster (CSM) and holds a CISSP (Certified Information Systems Security Professional) from the International Information System Security Certification Consortium, also known as $(ISC)^2$. His email address is andrew.freeman@jhuapl.edu.

Anthony T. Plummer Jr., Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

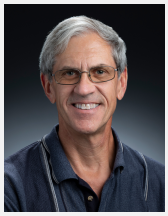
Dr. Anthony T. Plummer Jr. is the supervisor of the Spectrum Analysis Section in the Tactical Communications Systems Group in APL's Asymmetric Operations Sector. He received a BS in electrical engineering from Morgan State University in 2005 and an MS and a PhD in electrical engineering from Michigan State University in 2007 and 2011, respectively. His interests include the design and implementation of software systems and researching approaches to applying machine learning to communication and networking applications. His email address is anthony.plummer@jhuapl.edu.

Jordan N. Kraus, Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Jordan N. Kraus is an electrical engineer in APL's Asymmetric Operations Sector. He earned a BS in electrical engineering from Lehigh University and an MS in electrical and computer engineering from Johns Hopkins University. At APL, Jordan has worked with the GNU Radio software-defined radio framework to develop custom modems for interoperability with existing commercial off-the-shelf (COTS) and non-COTS radio equipment. He has worked with a variety of wireless waveforms and protocols such as HF, Wi-Fi, Zigbee, and SATCOM implementations. Jordan contributed to SC2 by developing and analyzing validation tests for the wireless channel emulator and the overall system. His email address is jordan.kraus@jhuapl.edu.

Mark L. Plett, Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Mark L. Plett was supervisor of APL's Wireless Cyber Capabilities Group from 2013 until 2019. He earned BS degrees in electrical engineering and physics, an MS in electrical engineering, and a PhD in electrical engineering, all from the University of Maryland. He is an expert in communications and radio frequency wireless systems. He joined APL in 2009, and throughout his time at the Lab, he made critical impacts on the development of wireless systems leveraging advanced technologies such as software-defined radio. His technical breakthroughs in wireless systems enabled several sponsor-critical cyber operations missions. Before joining APL, Mark worked as a senior hardware developer for Microsoft and as a senior systems analyst for Terabeam.



Kevin E. Parker, Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

Kevin E. Parker is a project manager in APL's Asymmetric Operations Sector. He holds a BSEE from Virginia Tech and an MSEE from Johns Hopkins University. Kevin started his career by serving on submarines in the US Navy. After 7 years of active duty, he came to APL while continuing in the Navy Reserve. Over his 31-year career at the Lab, Mr. Parker's technical work has focused primarily on communication systems engineering and receiver signal processing. More recently he has served in a variety of technical, project, and line leadership roles to help advance support to the cyber operational forces. Kevin served as the project manager for the DARPA SC2 project. His email address is kevin.e.parker@jhuapl.edu.

David M. Coleman, Asymmetric Operations Sector, Johns Hopkins University Applied Physics Laboratory, Laurel, MD

David M. Coleman is research area lead for spectrum operations and supervisor of the Communication Systems Concepts Section in APL's Asymmetric Operations Sector. He earned a BS in computer engineering from Elizabethtown College, an MS in computer engineering from the University of Arkansas, and a PhD in electrical engineering from University of Maryland, College Park. For the DARPA SC2 effort, Dr. Coleman led the software development for the Colosseum's RF Emulation System as well as the Interface Control Document definition between this subsystem and the Colosseum. Dr. Coleman has presented at many conferences and has been published in journals and proceedings. His email address is david.coleman@jhuapl.edu.