ERDC/GRL TN-22-2
September 2022

Adapting Agile Philosophies and Tools for a Research Environment

By Nicole M. Wayant

PURPOSE: There exist myriad project management methodologies, but none is focused solely on scientific research. Research projects are unique compared to other types of projects, including software development, manufacturing, and drug trials; research projects inherently have unplanned risks. These risks provide a challenge to managing resources, developing schedules, and providing team ownership while still achieving project goals. To help mitigate the risks and the challenges associated with scientific research, a methodology to manage research projects needs to be developed.

BACKGROUND: Most management of scientific research can be distilled into two distinct frameworks: sequential increments and continuous research. Sequential increments is the most straightforward of these frameworks. The sequential increments framework relies on breaking up research projects into a series of sequential phases with clear objectives and detailed requirements. For example, phase one of a project consists of all field work. Phase two is data processing, phase three is experimentation, and the last phase is the summarizing and writing of results. While these sequential phases logically make sense, issues arise if the focus of the work needs to be tweaked or a problem is identified later in the process, which requires revisiting an earlier step. For example, if during either the experimental phase or the writing phase it is discovered that the data were collected incorrectly, the whole project must start over.

Continuous research is the opposite of sequential increments. The posed research question is broad, and there are no clear milestones. This gives a researcher flexibility to adapt research directions based on experiment results, correct mistakes, and have a thorough understanding of the phenomenon being explored. However, following this framework can lead to researchers having difficulties answering the original question and enlarging the scope of the project. The lack of concrete milestones in continuous research makes progress hard to track and the management of the project almost impossible.

The problems that often plague scientific research projects (e.g., scope creep, rigid timelines) also cause issues for software development projects. To overcome these common challenges, a group of software developers came up with a set of values and principles for software development, which they dubbed Manifesto for Agile Software Development, or Agile for short. Today Agile is used by more than 70 percent of American companies, with approximately four out of five businesses using some form of Agile (Panditi 2018). While the relationship between scientific research and software development isn't one-to-one, the philosophy of Agile—rapid prototyping, collaboration, responding to change—may greatly benefit the scientific community. This paper explores how Agile can be adapted for a scientific research environment.

AGILE: Agile project management was born out of the Manifesto for Agile Software Development, which is a collection of values and principles that serve as a foundation for how a team can make decisions about developing software. The Manifesto reads as follows:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> Individuals and interactions over processes and tools
>
> Working software over comprehensive documentation
>
> Customer collaboration over contract negotiations
>
> Responding to change over following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more. (Beedle et al. 2001)

As stated in the text, values listed on the left have greater importance than the values on the right. That does not mean that the values on the right are not important, only that a greater emphasis is placed on the values on the left.

In addition to the manifesto's values, there are also twelve principles. These principles support the philosophy laid out in the manifesto.

1. Satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
4. Business people and developers must work together daily throughout the project
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
6. Face-to-face conversations are the most efficient and effective method to convey information within a development team
7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. (Beedle et al. 2001)

Based on these values and principles, the life cycles of Agile and software development differ greatly. Within software development, the typical life cycle walks through the following steps (Figure 1):

1. Analysis—determination of scope, timeline, and quality assurance
2. Design—define features and determine dependencies, database schema, and technologies
3. Implementation—determine coding standards, break work into tasks, assign tasks
4. Testing—test developed code, assign bugs to developers, re-test fixes
5. Deployment—perform final tests, deploy to production, and test on production
6. Maintenance—manage customer bugs, develop new enhancements, and release software upgrades

If a customer changes their mind about the final products or there are significant bugs in the code, the life cycle has to start over.

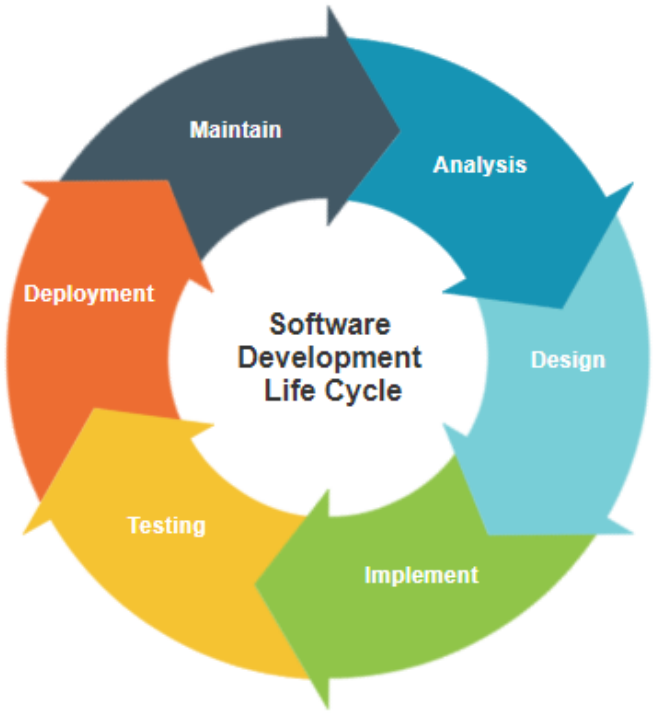

Figure 1. Depiction of the software development life cycle (Bigwater Consulting 2019).

The Agile life cycle differs from the typical life cycle of software development in that each life cycle, defined as a sprint, only lasts two to four weeks. During a sprint, enough information is gathered to start or continue the project, a plan is developed to create the next feature in the product, the feature is tested, everyone on the team provides a demo or update on what they completed in the sprint, and the customer views and provides feedback on the current product version. If the customer decides to tweak the desired outcome or add more requirements, the changes can be easily incorporated into the next sprint. Within each sprint, several different tools are used to enact the philosophy of Agile. Some of the most widely used, and the ones described in this paper are the daily scrum, story points, and Kanban boards.

One of the most commonly used tools within the Agile framework is the daily scrum. During a sprint, team members meet daily for 15 minutes or less to listen to each individual discuss what they accomplished in the past day, what they plan to do during the current day, and any impediments standing in their way. Sometimes referred to as stand-ups, the meetings are so short

that some teams remain standing for the meeting's duration. The purpose of the daily scrum is to offer a status update, keep people accountable, and identify any issues. If more in-depth meetings are needed to tackle a problem, separate meetings are scheduled, ensuring that meetings are focused and attended only by pertinent individuals.

Another often-used tool within Agile is story points. Story points are an estimation of the amount of time it will take to complete a task. The more complex a task, the more story points assigned to it. During sprint planning sessions, where tasks are determined and assigned to teammates, everyone on the team must agree on the definition of a story point and how many story points a task should be assigned. This helps to ensure that no one individual is given too many difficult tasks or that others aren't challenged.

One way to keep track of sprint tasks is to place them on a Kanban board. A Kanban board is a visual division of tasks, identifying the status and progression of tasks as well as the metric required to consider tasks successfully completed. An example of a Kanban board can be seen in Figure 2. In the figure, one of the columns is labeled "Backlog." The backlog refers to all tasks for the current sprint that have not yet been started. The Backlog does not mean that a task is late or incomplete, only that it still needs to be carried out. The "In Progress" column refers to tasks actively being worked on. Tasks are only moved to the completed column once they have met their metrics of completion. The metrics of completion are binary and provide a standard for each task. If the task product cannot pass a metric of completion, then it is not considered complete.

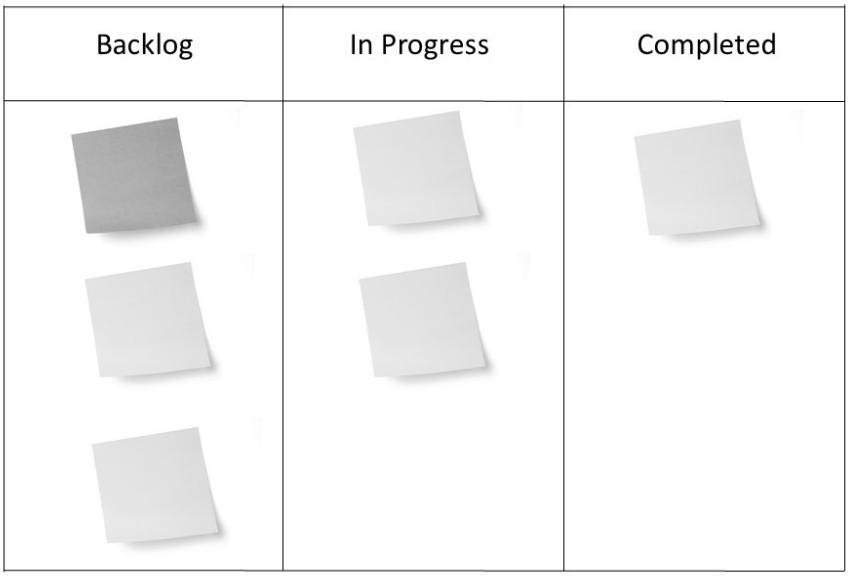

Figure 2. Kanban board example. The task progress goes from left to right.

ADAPTING AGILE FOR RESEARCH: Forms of Agile have been utilized within individual research projects (Pirro 2019), at universities, and at government-funded and industrial labs (Hicks and Foster 2010). While these laboratories utilize certain aspects of Agile, they do not embrace its philosophies. Instead, they focus on utilizing a common Agile tool: the scrum (Hicks

and Foster 2010). The scrum status meeting, whether held daily, weekly, or a few times a week, is a powerful tool (Hicks and Foster 2010). However, the use of scrum does not encompass the values and principles of Agile. In order for scientific research to take advantage of all Agile has to offer, the essence of Agile must be adopted and adapted.

Agile Philosophy and Research. Adapting Agile to research, whether biological or computer science, begins with modifying Agile's values and principles. Listed below are possible modifications to the Agile philosophy. The original statements are crossed out and the modified concepts are in bold.

> We are uncovering better ways of ~~developing software~~ **conducting scientific research** by doing it and helping others do it. Through this work we have come to value:
>
> Individuals and interactions over processes and tools
>
> ~~Working software~~ **Answered questions** over ~~comprehensive documentation~~ **journal articles**
>
> Customer **and Cohort** collaboration over ~~contract negotiations~~ **obtaining additional funding**
>
> Responding to change over following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more.

The first set of values to be modified refers to placing more value on producing working software than comprehensive documentation. For scientists, comprehensive documentation can be considered journal articles. In research, peer-reviewed publication is not only important, an individual's career can rest on their publication history. However, in order to produce quality publications that are often cited, one needs to have solid science about which to write because "quality begets quantity" (Schimel 2012). Therefore, the scientific Agile philosophy values answered questions over journal articles and detailed notes. That does not mean that peer review publication is not important, only that answered questions are valued more. In fact, by focusing on answering scientific questions, writing journal articles could become easier.

The second set of values to be modified refers to valuing customer collaboration more than contract negotiations. The basis of this value hierarchy is as true for research as it is for software development. However, for large research projects, scientists are not always able to regularly collaborate with their customers or funding source. Project managers and lead researchers may interface with individuals outside of the project, but the majority of the researchers do not. Instead, researchers interact more with their lab cohorts. Therefore, more value should be placed on customer and *cohort* collaboration over obtaining additional funding for their projects or protecting their research.

Within the twelve principles of Agile, six need to be modified, as listed in the text below.

1. Satisfy the ~~customer~~ **funding party** through early and continuous delivery of valuable ~~software~~ **research.**
2. Welcome changing requirements, even late in development.
3. ~~Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.~~ **Deliver updated research every few months.**
4. ~~Business people and developers must work together daily throughout the project~~ **Management and researchers must collaborate regularly throughout the project.**
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. Face-to-face conversations are the most efficient and effective method to convey information within a development team.
7. ~~Working software is the primary measure of progress.~~ **Answered research questions/completed objectives are the primary measure of progress.**
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good **scientific research** design enhance agility.
10. Simplicity—the art of maximizing the amount of work not done—essential.
11. The best architectures **research**, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The modifications focus on switching software production to research progress and changing time scales from weeks to months, as scientific progress is rarely completed in leaps and bounds. Surprisingly, the majority of the twelve principles of Agile can be applied to scientific research without modification. Simplicity, regular communication, attention to technical excellence, and the promotion of a positive work environment are tenets any project can be built upon.

Adopting Agile Tools to Research. The sprint cycle, in which the next product version is created and tested, is very powerful. Within software development, sprints are typically 2 weeks long and provide developers with just enough time to make visual progress on a product. For research, 2 weeks may not be enough time to complete the scientific task at hand. When working in a research environment, it is recommended to extend the duration of sprints and change the cycle to fit research.

A sprint duration of 1 to 2 months is usually an adequate amount of time to fully research and test an idea or complete a research objective (e.g., data collection). At the beginning of the sprint, researchers should determine the question they are trying to answer or the objective they want to meet. They will then build out the tasks that will need to be completed to accomplish the overall sprint goal, such as data to be gathered and tests to perform. A tangible deliverable and yes–no metrics of completion should be established for each task. These criteria help keep the tendency toward continuous research at bay. Lastly, if something occurs that changes project resources or schedule, tasks and goals can be adjusted at the beginning of a sprint cycle.

A sprint does not need to be focused on data collection or experimentation. For example, a sprint can also be a literature review. Literature reviews are an important part of research and can take

anywhere from a few days to a few months to complete. When planning a literature review sprint, it is important to set precise goals for what you are wanting to learn to ensure you don't fall into the trap of continuously reading and never completing any of your own research.

Scrums can still be completed during sprints, but daily scrums tend to be excessive (Hicks and Foster 2010). A weekly scrum is beneficial for setting the tone of the week and allows teammates to understand what their peers are doing. While meeting in person is preferable to build comradery (Hicks and Foster 2010), this is not always feasible, as many researchers are on multiple projects and have several meetings throughout the week. When in-person meetings are not possible, digital or written scrums can suffice. During scrum updates, each teammate should answer the following questions, as done during traditional stand-up meetings: (1) What did you accomplish since the last update? (2) What do you plan to accomplish before the next scrum? (3) What, if anything, is standing in the way of you accomplishing your tasks?

Additionally, for larger projects, even weekly scrums can become daunting, with weekly meetings lasting 30 minutes to an hour or written scrums being several pages long, with most teammates ignoring the updates. For large teams with more than six individuals, breaking the project into smaller teams makes scrums a more user-friendly tool. In the author's experience, the ideal team size for utilizing scrums is three to six people.

For Kanban boards, not much is needed to change besides a team's preference for the visual division of tasks. However, the story points typically assigned to tasks and placed on Kanban boards have been proven, on a limited basis, to not be useful within scientific research. In the author's experience, researchers have a difficult time estimating how long a task will take them to complete. They are able to gauge what is doable within a sprint, but the nuance of setting individual task deadlines within the sprint is more difficult. Hypothetical examples include what was thought to be a 1-week literature review really took 2 weeks, a simulation was developed faster than originally thought, and a new theorem was published that changes the scientific understanding of a topic. To overcome the challenges in scheduling task time, researchers should develop yearly, quarterly, and monthly goals and develop individual tasks at each sprint that push toward these overall goals. Additionally, at each sprint planning meeting, the monthly and quarterly goals should be reviewed to determine if the scope of the research or the goals need to be changed.

As in software development, at the end of a sprint each team member should demonstrate or discuss what they accomplished. For larger projects with multiple teams, there should be additional demos, at least twice a year for each team. This allows the project as a whole to see individual components, understand how everything fits together, learn about teammates' skills, and invite the customer and stakeholders to see a project's progress. While customers and stakeholders are part of every software development sprint demo, this is not always feasible for large research projects. Monthly meetings highlighting one of a project's teams allows the customer and stakeholders to grasp the scientific progress being made without overwhelming them with numerous updates.

SCIENTIFIC AGILE IMPLEMENTATION: The implementation of Agile within a research project can face many challenges. Perhaps one of the most difficult challenges is defining tangible research deliverables and binary metrics of completion, which is key for stopping continuous research. For example, how does one know when a literature review is complete? What if the

deliverable and the metric of completion are the same? To what level of detail do tasks need to be broken down? With time and experience, these questions become easier to answer.

Besides difficulties in defining metrics of completion, Agile can also be confusing. The agile framework was originally developed for computer programmers, not research scientists. Adapting to new project management vocabulary, expectations, and philosophies takes time. Those using Agile must be convinced of its usefulness. However, once researchers have been using Agile on a regular basis, they notice many benefits.

From limited implementation at the author's laboratory, researchers have found that they spend less time in project progress meetings and more time discussing science. Progress is made faster and it is easier to identify both scientific and personnel issues. Everyone on the team has a clear idea of their tasks and how they fit into overall project goals, providing ownership to everyone on the team. When new team members are added to a group, they are better able to grasp the scope of the project and understand their specific roles.

CONCLUSION: Agile is a popular framework for the management of software development. The same philosophies and tools used within Agile can be modified to work for scientific research projects. One of the main tenets of the Agile philosophy is prioritizing responding to change over following a plan. Therefore, modifying Agile methodologies to work with research is in keeping with the Agile philosophy. Indeed, the framework presented in this paper can also be changed to fit specific types of projects.

Following the Agile philosophy and adapting the methods presented here can help mitigate the unplanned risks found in scientific research. Regularly occurring sprints allow team members to reassess the direction of their research and the project's objective. Binary metrics of completion define when tasks are complete, and scrums and Kanban boards help to keep team members accountable. Additionally, Agile makes it easier to incorporate new teammates, helps provide individual ownership and group leadership, and provides the necessary flexibility needed within research.

REFERENCES

Beedle, M., A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Highsmith, A. Hunt, et al. 2001. Manifesto for Agile Software Development. https://agilemanifesto.org.

Bigwater Consulting. 2019. Software Development Life Cycle (SDLC). https://bigwater.consulting/2019/04/08/software-development-life-cycle-sdlc/.

Hicks, M., and J. S. Foster. 2010. "Adapting Scrum to Managing a Research Group." *Communications of the ACM*.

Panditi, S. 2018. "Survey Data Shows That Many Companies Are Still Not Truly Agile." *Harvard Business Review*. Brighton, MA: Harvard Business Publishing.

Schimel, J. 2012. *Writing Science*. New York: Oxford University Press.