Challenges of and Opportunities for a Large Diverse Software Team

U. M. Yang[1], C. Balos[1], P. Luszczek[2], S. Osborn[1], J. Willenbring[3]

What is a software team? It is a group of several people (at least two) who work together towards the common goal of creating/improving/maintaining a software product. So, the main ingredients of a software team are first people, second collaboration, third a software product, fourth a common goal. To achieve success, such a software team requires members with the appropriate skills, an agreement on how to collaborate, e.g., a predetermined set of rules, equipment and tools for creating a useful software product (e.g., a library), which generally includes computers, debugging and performance tools, etc., and finally a goal that motivates the team to charge ahead, such as customers that need the product to accomplish their own larger goal, e.g. a simulation that advances science. The more inspiring the goal the better.

When all these things are in place great things can happen. One of the authors specifically remembers such a team: “There were four of us with the appropriate skill sets. We worked together towards the common goal of mathematical software needed by application teams for a new computer architecture, and we made rapid progress. What made this collaboration so successful and rewarding? There was mutual respect and trust. This enabled us to have open discussions without fear that there would be any putdowns. Each input was listened to and considered. Although we were a diverse team, we understood each other well. Each person had a specific task and skill set that complemented the rest of the team and was crucial to get the work done. We knew that we needed each other. While the members were highly skilled, there were no `big egos’, nobody tried to get ahead of the rest. We all worked for the whole team to be successful. Finally, we also worked hard, but because we enjoyed the collaboration and pursuit of the goal, hard work was not a burden.”

In the context of this white paper, we will consider a large diverse software team. While the definition of a software team still applies, we are now looking at a team consisting of at least thirty people. Many of the issues we will discuss here will also apply to smaller teams, but a large team has to deal with additional issues. In this case, certain team dynamics are also more complicated because the team is really a team of teams, with each sub-team working primarily on a different, largely independent software product, but also working together on a larger project that all the individual software products are part of. We will discuss challenges and opportunities arising for such a large team using the example of the Extreme-scale Scientific Software Development Kit (xSDK) project (xSDK.info), focusing on first technical and then cultural challenges.

The xSDK team works toward the seamless build and use of a variety of independently developed highly efficient interoperable math libraries to support scientific applications. Each of these libraries have been developed by smaller independent teams using their own software strategies and styles. Achieving a common build and sustaining and creating interoperabilities among these

[1] Lawrence Livermore National Laboratory. Prepared by LLNL under Contract DE-AC52-07NA27344.
[2] University of Tennessee at Knoxville
[3] Sandia National Laboratories

libraries where appropriate and useful is very challenging. To achieve this goal the team created a set of guidelines, the xSDK community policies, that each member package has agreed to follow. These rules consist of mandatory policies that address topics such as building, installing, testing, portability, licensing, namespacing, repository access and more, and help to improve software quality, usability, and sustainability of the individual libraries and ultimately the whole kit. There are also a few recommended policies that are not currently enforced but might be at a later time. These rules address a lot of potential issues that can interfere with the common build and interoperability of the software but are not too heavy-handed to interfere with the software strategies of the individual libraries. These policies are regularly reviewed with input of the broader community, so they do not become outdated. They have also served as a model for similar policies for other software development kits, e.g., the E4S software stack (E4S.io).

Dealing with a large set of math libraries and producing regular xSDK releases comes with its own set of challenges. The development and testing of the whole xSDK product is very challenging since the individual libraries continue to be improved and developed. So, the overall software stack is continually in flux. We are now taking steps to work toward a faster, and more people-efficient workflow for development and testing within the xSDK. Currently Gitlab CI is employed to test the entire xSDK on several different platforms. Additions are being made to the testing capability by adding new tests, and layers of testing, e.g., only testing a subset of xSDK packages to easier diagnose potential issues. The primary goal of the CI testing is to ensure the current release version of the xSDK is stable. However, an important secondary opportunity is to test with the development versions of various libraries. This will enable potential issues, like API changes, to be identified sooner and is critical for long-term sustainability. Since we have not found an optimal solution yet, we continue to discuss viable approaches for this problem. We have also developed an example code suite exhibiting interoperabilities between libraries. It serves both as a testing tool as well as a training tool for users of the xSDK. Extending this suite through new examples brings additional complications due to the continuous, mostly autonomous development of xSDK member packages. Due to the challenges associated with keeping development versions of all xSDK member packages working together, there is not currently a development version of the example suite. Using this model, when new interoperability features have been added to individual xSDK libraries, but those features are not yet in a released version of xSDK, either the examples must be built separately (not with the Spack package for the xSDK-examples) or more frequent releases of the xSDK are necessary. This is problematic as synchronizing releases with interoperating libraries that have API changes is difficult and time-consuming. Additionally, the individual libraries have independent release schedules that make coordinating for a new xSDK release challenging. Another option could be to modify the example test suite to use Spack variants associated with every xSDK member package, but this approach would increase the complexity of the example suite significantly.

The xSDK team consists of over fifty software developers, computer scientists and computational mathematicians from various national laboratories, universities, and industry. The size of the team and the location of the members brings challenges, but also opportunities. A team of this size brings with it a large network of contacts to outside experts, which is helpful to get additional input and advice and can result in improved quality. The diversity of the team leads to new insights,

allows members to evaluate different perspectives, to think outside the box and to ultimately come up with new solutions that uniform teams might miss. Thus, we should celebrate our differences.

Since the xSDK team is spread across the whole United States and even includes members from Europe, we need to overcome distances and deal with varying time zones. Frequent in-person meetings are generally not possible due to cost and logistics. Regular meetings need to now be done virtually, using applications like Zoom, WebEx, Blue Jeans, etc. This is generally easier if the team was able to work in-person before, since they still have the visual image of in-person meetings in their minds, but it is much harder for a person that was unable to attend the past in-person meetings or a team that is newly created under these circumstances. It is helpful to be able to use cameras so one can see facial expressions and just to know what a team member looks like, for when we run into them at a conference or workshop: we can recognize them without official introductions. It also provides a visual in our mind of who the other team members are. However, it is not always possible to turn on cameras, be it due to bandwidth issues, or if the team is very large, as in the case of the xSDK team, there is not sufficient room on the screen to see everybody at a reasonable size. Nevertheless, regular virtual meetings are necessary to keep the team on track, motivated and provide cohesion.

Having been involved in several very large teams, we have found that having at least an initial meeting of all team members at a specified location over a few days is very helpful to get to know each other and start building trust. While this was not the case for the current xSDK team, it was the case for the IDEAS team, from which the xSDK team originated. The IDEAS team started off with a large meeting of the whole team in San Francisco, which lasted several days. This was a very important meeting since it allowed the team members to get to know each other outside of the online meetings and collaboration that dominated the work since then. Not everything went smoothly at this meeting. One of the challenges in bringing a variety of people from different backgrounds together is to learn to communicate. Misunderstandings can easily happen. It is important to develop a mutual vocabulary. It is easy to assume that another person understands our words the way we would, however their background and upbringing are important factors since they affect their view and understanding of what is being said. It is crucial to overcome these communication barriers. Team members need to get to know each other's characters. Ultimately, the San Francisco meeting was a fruitful meeting. It started the process of developing a common vocabulary and building trust and respect towards each other.

It is also good practice to take advantage of opportunities at conferences and workshops, which are attended by most of the team members to arrange for small-group meetings and dinners to continue building in-person relationships between members in addition to virtual meetings. Building such familiarity is important to create a culture of openness and honesty. Creativity is best when people can take their guards down and there is no need to feign engagement. People can speak freely without fear of judgement and are invited to substantive conversation.

Working together on a common goal will also strengthen professional relationships and build up the team. Overcoming technical challenges requires collaboration on a technical level. Recognizing that we often need each other's help keeps us humble and generates respect towards those who have the necessary skill and make our own work faster and/or easier. Seeing

progress being made and taking note of smaller achievements helps to keep the team motivated. Finally, we should not forget the overarching goal to stay focused. For the xSDK team, this goal is to support application teams to reach exascale performance.

In summary, while technical advancements are essential for a successful software team, the human factor is just as important. It can be the key that leads to final success or the hindrance from reaching the goal. A thriving team where all are on the same page striving towards an excellent software goal, working together to make it happen, can lead to great results. However, elements like personal ambition, envy, boredom, burn-out, etc. can make such a team ineffective or even fail. While working together toward a common goal, we share successes and new discoveries that help us refine our software. Helping to mutually improve each other's libraries while giving credit where credit is due increases the quality of the whole product and moves us closer to the goal of achieving great science at exascale.