The government seeks individual input; attendees/participants may provide individual advice only.

Middleware and Grid Interagency Coordination (MAGIC) Meeting Minutes[1]
September 1, 2021, 12-2 pm ET

Virtual

Participants

Alison Derbenwick Miller (Oracle)	Kaushik De (UTA)
Anshu Dubey (ANL)	Keith Beattie (LBL)
Arjun Shankar (ORNL)	Kjiersten Fagnan (LBL)
Ben Brown (DOE)	Kristin Davis (NTIA)
Bev Corwin	Ludovico Bianchi (LBL)
Bill Spotz (DOE)	Mallory Hinks (NCO)
Dan Gunter (LBL)	Mike Heroux (Sandia National Lab)
Dan Katz (Illinois)	Miron Livny (OSG)
David Bernholdt (ORNL)	Richard Carlson (DOE-SC)
David Martin (ANL)	Sharon Broude Geva (U of Michigan)
Donald Petravick (Illinois)	Terrill Frantz (Harrisburg Univ)
H Birali Runesha (U of Chicago)	Tevfik Kosar (NSF)
Hal Finkel (DOE/SC)	Todd Tannenbaum
Jeff Carver (U of Alabama)	Valerie Taylor (ANL)

Introductions: This meeting was chaired by Richard Carlson (DOE/SC) and Tevfik Kosar (NSF)

Sustainable Software Speaker Series: Best practices in designing and developing sustainable research software

Contemporary Peer Code Review Practices in Research Software

Jeffrey C. Carver, University of Alabama

- Peer code review is one of the approaches we can use to check the quality of software as we're developing it.
- Showed some examples of code and asked the audience if they could figure out what was wrong with them.
- Showed some examples of code that had security issues.
- Compared peer code review with peer editing.

[1] Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Networking and Information Technology Research and Development Program.

- Mentioned that adding code review to the process allows you to reduce some of the time you spend in other activities (testing and debugging)
- Benefits:
    - Team building
    - Team cohesion
    - Code quality
    - Different perspectives
    - Consistency
    - Learning
- Discussed the mindset you need to have to make code review process work well
- Code Review Techniques: Let people focus on the things they're good at and computers do what they are good at.
    - Readability, etc.
- Research code and scientific code review
    - Testing is more difficult when the results are unknown
    - Code review may be helpful to find issues
- Mentioned a survey he and one of his former students conducted
    - People in research software field
    - Found when software developers do code review it's informal but it's generally beneficial
    - People commented on the lack of time available to perform code review
    - Suggestions were to make the process more formal, more training, more compensation
- Jeff said if anyone here is interested in having a larger tutorial done in the future, they should contact him.

Q&A

- Kaushik asked about artificial intelligence as the first step in code review.
    - Jeff said that as AI techniques mature the line between where the computer ends, and the person begins may be shifting. He said he thinks there's a lot of potential to do more automatically.
- David said that the issue they have is that code is written by postdoc or people who are rewarded for publishing, not necessarily documenting code or spending time in code reviews. He asked if Jeff had thought about what we could do in that environment.
    - Jeff said he doesn't have an answer to that other than changing the culture. He reiterated the benefits of code review.
- Keith asked if Jeff had any perspective on the relative strengths and advantages or different approaches between doing code review as part of an acceptance of changes to the repository as opposed to looking at code that already exists in the repository.
    - Jeff said there is a lot of value to both of those. He said he would argue that doing code review at any point is valuable. The earlier you do it in the process, the better.
- Anshu said that one thing that she finds most challenging to peer code review is the fact that there is no redundancy of expertise in the team. So other than a generic quality, there isn't a mechanism for doing a reasonable code review. She asked if he had thought about that problem.

    - Jeff said that a code review can still be beneficial because there's code quality and things like that to look for. He said it depends on the situation, depends on the team and the expertise as to what level of detail you can go down in the peer review.
- Hal mentioned badges or other certifications to put on research papers. He asked if that could be applicable to code review.
    - Jeff said that he doesn't know if anything like that for code review. He mentioned that there also could be more internal incentives (financial, recognition, promotion). He said there could be both community kind of things like a badge on a paper, but also more internal to teams and organizations that tell people that it's valuable to spend time on doing this kind of thing.
    - Hal said that one thing that he's observed as an issue is getting over the mentality of "it's just an experiment". The idea that I'm working on something short lived, so it's not worth spending all this time on process. He said understanding what the various trade-offs and expected life cycle of various pieces of software is key.
    - Hal commented on Anshu's comment about the lack of redundancy of expertise. In his experience, this is a critical piece of the puzzle. He said it's really important that you view the code review process as an opportunity to educate your peers about what you've done. He said it is important to propagate your knowledge beyond yourself.
    - Jeff said understanding where in the stage of the software it becomes worth adding process is definitely an open question.
- Ludovico asked if Jeff had any insight for how to determine what parts of the code people should review.
    - Jeff said one of the things to consider is what is it that you want to get from the code review (how it will interact with other codes, how other people might use the code, etc). He's good to tell the person reviewing the code what perspective of theirs is important for the code to focus their efforts.

Software Design Insights for Longevity of Scientific Software

Anshu Dubey, Argonne National Laboratory

- Noted that this presentation was taken from a tutorial that the Ideas Exascale Computing Project conducts on better software at various venues
- Anshu said that the big problem she sees with the scientific computing and research software for scientific computing is that there is a positive feedback loop.
- Discussed general design principles for HPC scientific software. She said many of them are at odds with what we need to do in scientific computing.
- Shared a schematic of what you should do for code design. Design first, then apply programming model to the design instead of taking a programming model and fitting your design to it.
- Gave an example of multiphysics partial differential equation-based code for a distributed memory parallel machine.
- Described takeaways
    - Differentiate between slow changing and fast changing components of your code
    - Understand the requirements of your infrastructure

    - Implement separation of concerns
    - Design with portability, extensibility, reproducibility, and maintainability in mind
    - Do not design with a specific programming model in mind
- Described features and abstractions that must come in
- Described the underlying ideas
    - Make the same code work on different devices
    - Assigning work within the node
    - Look at what is needed, design for commonalities
    - Even when using third party abstraction tools, understanding the code's structure and needs is critical for performance portability that translates to investing in design.
- Explained how abstraction layers work
    - Infer the structure of the code
    - Infer the map between algorithms and devices
    - Infer the data movements
    - Map computations to devices
    - These are specified either through constructs or pragmas

Q&A

- Valerie asked at what point, as you have those components that are flexible and composable, are you going back to consider the programming model so that you can get the performance.
    - Anshu gave an example of one of the things that started coming into play was task-based asynchronous programming. Most people seemed to pick up one of the task-based systems and design their code so that it would work with that task-based system and would become so wedded to it that if they wanted to try another one it was a huge amount of work. What they should have done was thought about it independently of one particular package.
- Ludovico asked if Anshu had any advice about refactoring existing code.
    - Anshu said that as a part of the SSW tutorial she has a 45–60-minute presentation on refactoring. She said she has undergone three major refactorings of Flash. She has learned that you have to make sure that you have adequate testing infrastructure in place, you need to have coverage for all of the code that you're touching for refactoring, you want to do it incrementally, and you need to get the stakeholders on board.

Open Discussion

- Anshu brought up culture change and lack of redundancy of expertise in the team. She said that may be tied to the way that funding models work. She said that if it wasn't for the fact that there was a very sustained funding available for the development of Flash in the early days, it probably would not have come into being. Any problems that they ran into could be rectified because there was stable funding. She said she thinks there is less slack built into the funding system now.
- Dan Katz said that sometimes it seems like there's a need for free exploratory time to let people try things that aren't tied to project milestones. He agreed that culture seems tied to funding practices.
- Dan Gunter said that there's additional challenge sometimes if you're working as part of a larger project where the focus of that project is not software; the focus is getting some sort of

deliverable in terms of the science or engineering. The thought of calling out additional time to try out things that may fail within the software seems difficult.

- Sharon said another of the problems is that although software is integral for the actual research results, there are many people who are staff (not students or faculty) that are working on the software and end up getting no credit. Another cultural change that needs to happen has to do with career professionalization for RSEs and others. She pointed out Orchid's credit model that gives credit to people who don't get mentioned in the publications and grants. https://info.orcid.org/credit-for-research-contribution/
- Anshu described the story of how she got into the business of writing papers and software. She said there was a science campaign run. She and her team were an integral part of it. But when the paper was written, not one member of her team was a co-author. So, she started writing papers on the work they were doing with the software itself.
- Alison asked how often we are doing software innovation as part of a research project where the outcome is looking for different scientific results or something where the focus is not on the software. She also asked how often we are rewriting things because people are not as familiar as they could be with tools that are available or optimized configurations they could be using.
- Anshu said some of both.
- Alison asked if it would be helpful to have some sort of resource library of known images, known configurations, known ways of approaching certain common scientific problems from a software perspective.
- Anshu said that in the astrophysics community the idea of open code took hold quite early. When a new grad student comes in, they don't start to write their own code, they look to see what's available and start working that. She said other communities can learn from this.
- Dan Gunter added that one of the most painful tasks in programming is software integration.
- Anshu said that that leads to innovation in many cases.
- Dan Katz said this is also where peer review can come in, particularly in the case of software publication.
- Birali said we should think about changing the ways we train our students and even curriculum.
- Dan Katz said that management and leadership are certainly needed, but a lot of the issues are baked into institutional practices
- Kjiersten said agreed but she said there are also challenges just maintaining staff within the National Lab system. She said appreciating the people who are engaged in your projects could potentially be useful for generating numbers, but it feels like another metric and culture change at the organization level is challenging. She said even identifying the characteristics that we want when hiring managers and leaders for teams and projects could also help, but those tend to be ignored in favor of hiring a person with the best publication record.
- Anshu said in the last few years she's been encouraged. She said what we really need to think about is how do we accelerate the changing course.
- Sharon said the problem in academic institutions is there is a separation between staff and faculty. And faculty aren't used to giving credit to staff. This is something that universities can do about. Funding models can do something about that as well.
- Dan Katz suggested that authorship and credit have to be talked about when you start a collaboration. This is harder to do at the institutional level.

- Dan Katz said that in the UK there have been some interesting actions by the funding agencies to recognize technical work.
- Anshu described her experiences at the University of Chicago and NNSA.
- Sharon described her experience helping manage a group of consultants at the University of Michigan. She said they don't always get credit for what they're doing. She said there was a large push from the university administration to have them on soft money. She said discussions between funding agencies and universities would go a long way.
- Anshu asked if it would help to look at large installations and apply their lessons to software.
- Sharon emphasized how important it is to be seen as part of the "team".
- Dan Katz asked if there is something this discussion should turn into.
- Rich said Mallory should come up with a summary of the challenges that the group identified as being impediments to being able to have sustainable software. Then we can have people come in to talk about how to address this.
- Tevfik said that he thinks these discussions help in multiple ways. One is identifying the problems. Another is to share best experiences. He said as program directors at funding agencies, they are listening to these problems and they provide this as feedback to their internal discussions in order to potentially address those problems in the future.
- Tevfik said that NSF recently released a new blueprint document on workforce development. He said some of the issues discussed are very related to some of the recognition and career pathway to augment issues of the software developers. He suggested everyone read it. https://www.nsf.gov/cise/oac/vision/blueprint-2019/CI-LWD.pdf He said that if there are other issues that it should touch on to let him know.
- Dan Katz said that he was thinking about a community workshop, but as an interagency workshop or meeting it may have a different purpose.
- Rich said there is an opportunity for MAGIC to decide that we want a focused workshop on some activities here.

Round Table

- Dan Katz mentioned the Workshop on Sustainable Software Sustainability. October 6-8. https://wosss.org/wosss21-home
- Anshu mentioned the ongoing webinar series by the Ideas project. There's typically one per month. http://ideas-productivity.org/events/hpc-best-practices-webinars/
- Anshu said at SC21, they will be giving another tutorial on better scientific software.
- Dan Gunter said that DOE has the Leadership Scientific Software Sustainability town halls coming up. Mike Heroux gave the link. https:/lssw.io They will have their first town hall on September 16 from 3 to 4:30 pm ET.
- Rich said that at the October MAGIC meeting we would be discussing what kind of activities we will be planning for the next year.

Next Meeting

October 5 (12 pm ET)