# Quantitative methods for research software community management

Kyle Harrington[1,2]

[1]Image Data Analysis Group, Max Delbrueck Center for Molecular Medicine, Berlin, Germany
[2](Joining Dec 6, 2021) Multimodal Data Analytics Group, Oak Ridge National Laboratory

## Challenge

There are multiple problems that commonly arise in research software communities that inhibit their growth and productivity: ad hoc governance, redundant efforts, and hostile behaviors. These problems are social and can be addressed with community management and tools that support community management.

Community management is an important aspect of successful research software projects, but there are no well defined standard of practice and community management support tools are lacking. Some large projects (especially in industry) have realized the benefits of having dedicated community managers. However, in most cases community management becomes the responsibility of research software engineers (RSEs), RSEs are often unaware of the impact of their behaviors and interactions on community growth and maintenance (although there are exceptions). In positive cases, RSEs encourage users to take ownership of tasks like bug reporting, and may assist users with contributing their first software fixes. In detrimental cases, RSEs may be dismissive of users and other developers for being less knowledgable, and sometimes may simply ignore community engagement due to lack of resources to invest in community level activities. These behaviors can: grow or shrink the community, facilitate conversion of users into contributing developers, contribute to developer attrition, reduce community diversity, and lead to duplicate efforts. While these projects can attract millions of dollars worth of investment, the failure to fully appreciate their respective research software communities leads to significant unrealized potential.

This paper uses the open source scientific image analysis community as an example of a healthy, growing research software community. In this community, a few large-scale projects that have hundreds of contributors coexist (e.g., Fiji/ImageJ, scikit-image/napari, ITK/VTK). The funding strategies followed by these three example projects represent notably different mechanisms, where Fiji and ImageJ was originally supported by a single NIH researcher followed by numerous grants at institutions across the world, scikit-image and napari are now largely supported by the non-profit foundation: the Chan Zuckerberg Initiative, and ITK and VTK are supported by an open-source company: Kitware. Funding strategies have an impact on the capacity to directly support community management due to financial flexibility, institutional continuity, and the potential necessity to focus on developing new features and projects. This example community can be used to establish a broader model of behavioral dynamics in research software communities that can be used to evaluate management policies, the impact of individual behaviors, and determine useful quantitative measures for assessing community health, success, and productivity.

## Opportunity

This paper proposes to use computational models and social coding metrics to guide and assist in the identification of individual strategies and community policies that can reduce existing problems and make research software communities more robust and productive. Consider that each community is a population of individuals (including users and developers) with skills (application of tools, programming, ...), needs (analyzing data, getting credit for tool creation, ...), and behaviors (collaborative, lone wolf, codependent, ...), and that this community is governed by a code of conduct. By simulating population models of research software communities with different parameters, it is possible to see how interactions between individuals and community governance impact the dynamics of the community. Does the user base grow? Is there developer attrition? Is the community robust to the loss of key individuals? This approach can ultimately be used to identify the impact of individual behaviors and help shape community policies to maintain positive engagement and grow the number of users and contributing developers.

Establishing this model of research software communities involves a range of fields, including software engineering, agent-based modeling, game theory, cognitive science, and even project management. We will draw upon all of these disciplines and utilize the latest technologies in collaborative open source software engineering to enable the initialization of simulations based upon measures taken from real-world communities. This will make it possible to answer many important questions:

- What type of policies can we adopt to improve engagement in research software communities?
- What is the role of community member interactions in facilitating the conversion of users into contributors?
- Can we detect and quantitatively measure beneficial and pathological behaviors of developers based upon their social interactions (e.g. Github issues, mailing lists, code review, etc.) and other activities?
- What factors are relevant to defining the success of a software community: growth of user base, increase in number of developers, greater number of code contributions, etc.?
- Can we eventually: predict the success of a software community, incentivize behaviors that lead to positive community growth, and other future forward strategies?
- Can we quantify major differences between research software communities across domains?
- When creating a new research software project, how heavily should one weigh each choice (e.g. programming language, social coding platform, communication mechanisms, amount of community engagement, etc.)?
- Can we identify macroscale commonalities between communities? Can we find shared goals or subprojects that could be achieved by two or more communities cooperating?

## Timeliness

There have been many recent advances that contribute to the possibility of this research direction, including: (a) advances in modeling and simulation have laid a foundation for the establishment of models of research software communities, (b) new social analytics tools can quantify existing research software communities within modeling frameworks, and (c) a large number of open source research software projects have significant presences on social coding platforms. While general progress in these areas has established a groundwork for the proposed research, there are some key technological advances and social developments that are particularly important.

Social coding platforms and continuous integration tools now dominate the software engineering industry. Sites like Github and Gitlab are built upon principles of social coding, where source code hosting, issue management, community discussions, and more, all reside within a single platform. These platforms have open APIs that make it possible to retrieve data and statistics related to projects and users. These data can serve as a quantitative basis for establishing models of research software communities based upon the latest statistics. Continuous integration tools and pipelines are triggered based upon events within projects, like code updates, which can be used to trigger subsequent analyses of community interactions, such as community management recommendations.

Natural language processing (NLP) has significantly advanced with tools like GPT-3 that can generate realistic natural language. The NLP subfield of sentiment analysis serves as a way to algorithmically assess valence and tone of user interactions, such as detecting bullying behavior. This opens the possibility for supporting community managers at monitoring large volumes of community interactions. Additionally, tools like Codex can now explain code, providing an easy way to facilitate the transition from non-coding users to code contributors.

Some tech communities and individuals have developed notorious reputations when it comes to diversity, equity, and inclusion (DEI). Those who are sick fall behind; people who take parental leave lose opportunities; women are harassed. While DEI activities and agendas are finally being actively pursued, there are still many open questions in the scope of community management. If there is only private and/or in-person information about negative interactions between individuals, then how can these interactions be resolved within the community? How can disadvantaged individuals be supported when they have long periods of inactivity or require accommodations?

The maintenance and growth of sustainable research software communities should be directly addressed through the science of scientific software development. The proposed research will enable the assessment of strategies and policies, and development of management support tools to facilitate the advancement of research software development. These studies and tools can also inform and assist new projects when they are being established by suggesting a set of best practices. Ultimately, this work will empower research software communities to eliminate waste from redundant efforts and loss of skilled developers, and ensure that these communities can focus on the development and application of scientific software.