Experiences with a Flexible User Research Process to Build Data Change Tools

Drew Paine, Devarshi Ghoshal, Lavanya Ramakrishnan
Lawrence Berkeley National Laboratory
{pained,dghoshal,lramakrishnan}@lbl.gov

Abstract

Scientific software development processes are understood to be distinct from commercial software development practices due to uncertain and evolving states of scientific knowledge. Sustaining these software products is a recognized challenge, but under-examined is the usability and usefulness of such tools to their scientific end users. User research is a well-established set of techniques (e.g., interviews, mockups, usability tests) applied in commercial software projects to develop foundational, generative, and evaluative insights about products and the people who use them. Currently these approaches are not commonly applied and discussed in scientific software development work. The use of user research techniques in scientific environments can be challenging due to the nascent, fluid problem spaces of scientific work, varying scope of projects and their user communities, and funding/economic constraints on projects.

In this paper, we reflect on our experiences undertaking a multi-method user research process in the Deduce project. The Deduce project is investigating data change to develop metrics, methods, and tools that will help scientists make decisions around data change. There is a lack of common terminology since the concept of systematically measuring and managing data change is under explored in scientific environments. To bridge this gap we conducted user research that focuses on user practices, needs, and motivations to help us design and develop metrics and tools for data change. This paper contributes reflections and the lessons we have learned from our experiences. We offer key takeaways for scientific software project teams to effectively and flexibly incorporate similar processes into their projects.

Author's version. Please view and cite the version of record:
Paine, D., Ghoshal, D. and Ramakrishnan, L., 2020. Experiences with a Flexible User Research Process to Build Data Change Tools. Journal of Open Research Software, 8(1), p.18. DOI: https://doi.org/10.5334/jors.284

1 Introduction

Scientific software development is noted for its practices which are often distinct from commercial software engineering practices [12]. Building successful software for scientists is challenging due to the emergent, rapidly evolving and fluid nature of the problem spaces and associated research [3, 10]. In addition, scientific collaborations often involve multiple institutions, and sometimes even communities, and software transcends traditional organizational and collaborative boundaries. Qualitative user research methods and user-centered design (UCD) processes are widely used in commercial software development to generate insights into the ways humans work and their experiences using developed products. User research practices in commercial software development span a wide range of approaches and commonly include usability evaluations of systems to identify problems [13,14], as well as ethnographic inquiries employing interviews and observations to better understand the people and problems of interest [5, 15]. Some scientific software projects have incorporated these processes to evaluate the products they are producing [1, 6, 9, 11]. There has been limited discussion and reflection about the incorporation of user research and design as an integral element of scientific software projects [10]. Prior work has largely focused on the evaluation of user interfaces and prototype tools. Science software teams should be incorporating user research more broadly to improve their ability to build and deliver useful, meaningful products to their collaborators.

In this paper, we reflect on our experiences conducting user research in a project investigating scientific data change and developing tools, metrics, and methods for calculating such changes. Changes in scientific

data often arise from instrument configurations, software updates or quality assessments. Data change can be as simple as changes to the organization of data within files or files within a directory breaking an automated workflow tool's ability to process large volumes of data. Frequently, change is more complex and varies by domain, data values frequently change resulting in unexpected impacts to downstream data analyses that rely on the changed data. For example, sensor data from an environmental site is gathered and processed by a team then shared with end users as comma separated value (CSV) files. Data changes could be because of changes to headers (e.g, temp to temperature) or data values (e.g., 100.009 to 100.01). Users of the data have to understand and ensure their analysis pipelines continue to function and that results are not impacted by major adjustments to data points. Over time new processing approaches can require all older data be reprocessed.

Our user research efforts focused on the concept of data change and established an understanding of the scientific process and work practices to help our project develop metrics, tools and interfaces. We explored and used user research methods to investigate the larger concept of data change rather than just to evaluate user interfaces or the functionality of a prototype tool. We employed foundational, generative, and evaluative qualitative methods to characterize the problem space, design solutions to refine and clarify this space, and assess the solutions produced. Successfully accomplishing such work is challenging and our reflections explore the ups and downs of our experiences, potential biases to qualitative research, and the lessons we learned for teams striving to balance an idealized application of user research methods pragmatically with the constraints scientific software projects face.

2 Background

The Deduce (Distributed Dynamic Data Analytics Infrastructure for Collaborative Environments)[1] project is exploring the challenges and methodologies for enabling usable and efficient scientific data change management. The project uses a mix of techniques to address the data change problem space. We are working with use cases from multiple scientific domains (optical astronomy, earth sciences, and x-ray light sources) that each have datasets with varying format, scale, and structures. We are using user research methods to understand the concept and evaluate our techniques and statistical analyses to detect and measure the impact of changes in datasets. We are also developing Dac-Man[2] [4], a change detection and management framework capable of comparing massive datasets at scale. Dac-Man is designed to run in a variety different computing environments from desktops to high performance computing (HPC) systems.

Our user research processes are fluid, iterative, and evolve to match the rhythms of research and development. Foundational methods help to characterize problems while generative methods create design ideas and evaluative methods test the resulting product's fit with a problem and user needs. Outlined in Figure 1, our process employs foundational semi-structured interviews, generative user interface mockups, and evaluative usability testing to study data change from multiple perspectives. We are gathering foundational insights while also generating and evaluating our own research products to advance our understanding of data change. Our experiences illustrate how applying these methods requires maintaining flexibility and the ability to handle risks and uncertainty that come with new research directions and small teams with limited funding, since scientific R&D environments are typically exploring new paradigms that are not necessarily fully characterized. Detailed findings from these methods are available in [7]. This paper focuses on reflections of our experiences conducting this work and here we describe why and how we used these methods.

Semi-Structured Interviews. We use semi-structured interviews in each phase of our user research to gather foundational insights about data change and associated work practices. Semi-structured interviews are conducted with a set of guiding questions while leaving room for the interviewer to follow up strategically based on a subject's responses. This form of interview enables researchers to develop thorough descriptions, hear and see multiple perspectives on processes, and develop descriptions of problems [15]. We conducted four exploratory interviews with astronomy and earth scientists. These interviews revealed that scientists were undertaking ad hoc, manual work to identify changes between data releases that typically required in-

[1] http://deduce.lbl.gov/

[2] https://github.com/deduce-dev/dac-man

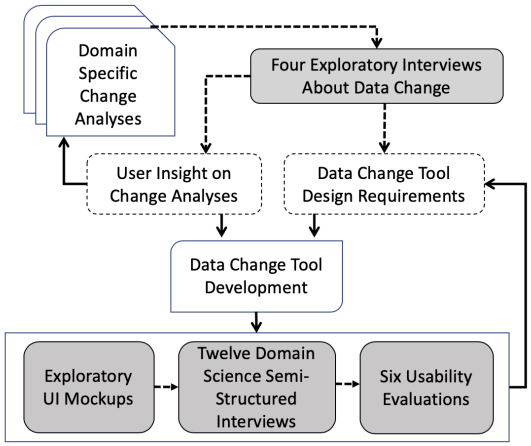

Figure 1: *Our user research process investigating and characterizing scientific data change. We conducted four exploratory semi-structured interviews with astronomers and earth scientists. We learned about data change in their work and gathered feedback on statistical analyses created by our team. Findings were used to generate design requirements for a data change tool. Next we conducted twelve domain science semi-structured interviews to better understand how scientists and projects handle data change, created user interface mockups to explore representations of change analyses, and evaluated a prototype of our tool through six usability evaluations.*

depth domain knowledge and the ability to work across multiple computing systems. We conducted twelve domain science interviews to gather both foundational and evaluative insights from astronomers, earth scientists, and x-ray light sources. We probed the work scientists do choosing data releases for analyses and the effects of project rhythms and structures on their decisions [8]. We also took the opportunity to evaluate user interface mockups and metrics our team had generated using outputs from our prototype tool. This evaluation came during the interview after our initial discussion about the participant's research work and experiences with data change, but before they learned about Dac-Man.

Usability Testing. We have evaluated our team's products by conducting usability testing of the Dac-Man prototype with six individuals. Usability testing is traditionally a method of laboratory study to measure user's experience and satisfaction with a system based on completing carefully defined tasks [13]. A traditional hallmark is the researcher's control of the test environment and format, in part to bring about quantitative rigor. However, our tests with six individuals were intentionally semistructured due to the diversity across sciences and to better understand the underlying problem of data change. This afforded our participants the flexibility to explore the Dac-Man prototype and elucidate on the concept of data change in their work.

We organized the evaluations with a broad script of reading about the tool's features in its README, installing the prototype in the tester's preferred computing environment (local machine or HPC), and trying out its commands with some sample data as well as their own scientific data. The tests were semi-structured since we encouraged users to explore the tool as they saw fit, investigating the features intriguing to their own work rather than constraining their test to complete rigidly specified tasks. We used the outputs from the tools to design visual mockups of a web-based interface for interacting with these calculations. We did not do a usability study on the mockups but used it more as a conversation starter to dig deeper into the foundational aspects of data change. These conversations led to insights that the users wanted domain-agnostic analyses to be complemented by specific custom analyses.

Six participants were chosen for their experience working with scientific datasets, tools, and computing systems from across our three domains. Four were first contacted for a domain science interview and we then followed up to have these individuals participate in a usability session. The two other testers were computer scientists who build tools and systems for data science. These participants were selected to provide a range of experiences with data and computing. Inevitably there is some form of bias that is inherent to qualitative research and during analysis the team had to account for this when interpreting the feedback gathered. Feedback helped us refine the prototype's terminology and outputs for its initial release and

offered insights for further evaluative and generative work.

Our approach enables us to accommodate the diversity of our users and scientific domains and the dynamic nature of scientific environments. Next, we detail our experiences and the generated insights, both unique and complementary to past work, that we believe can help other scientific software teams as part of their own work.

3 Reflections on our User Research Process

We highlight three overarching reflections and lessons learned from our experiences conducting user research in the Deduce project while discussing different facets that arose in each. Takeaways from our experiences are summarized at the end of each section.

3.1 Adopting broader goals for product evaluation

Traditionally, usability testing evaluates well defined aspects of a product to identify simple key issues to rapidly address and generate insights that affirm a team's design goals or requirements [6, 13]. Tests may assess how long it takes users to complete tasks or attempt to identify issues that induce frustration. In our case Dac-Man, addresses an initial set of design requirements, but the problem it is addressing is also not well-defined since change means different things depending on the particular moment, discipline, and even project at hand. Consequently, we also assessed our conceptualization of data change and documentation. Scientific software teams may benefit from adopting this type of broader approach to evaluation in their own work.

Evaluating tool functionality and overall concepts. Testing with six users generated a variety of useful general insights about the tool and our conceptualization of data change while pinpointing areas that needed to be refined. Our primary focus was to validate our Dac-Man prototype with respect to our astronomy use case since the tool had been motivated by needs from this domain. Conducting a test with a core use case is an important baseline for tools addressing nascent problem spaces since the team should re-evaluate its use cases and design goals if a large mismatch is identified.

Dac-Man is designed to help identify changes between datasets regardless of what qualifies as big data to a given scientist. Our astronomy testers were data producers in a community that produces large datasets and places them in open repositories. This work takes place with fairly well-defined processes with standardized metadata in a data format that is used by the majority of the community. Our earth sciences participants were data users in a community where central data repositories are becoming more common but are not as established. The experiences of these scientists demonstrated a more fluid data production process before a derived product is prepared for a community repository. These users primarily work with Excel spreadsheets, or CSV files, and the scale or scope of this data is smaller than our astronomy cases. At the same time the increased volume of spreadsheets and CSV files makes the scale large for these earth scientists.

The two astronomy testers affirmed that our initial design addresses their general needs, identifying changes among large sets of files and across multiple computing facilities. Both astronomy participants not only grasped how the tool works but were also able to derive some insights about differences between two versions of their project's releases and provided valuable feedback for future work. Similarly, the four other testers who tried the prototype were able to understand this characterization of data change and discuss how it would align or deviate with issues they face in their own work. We did not expect that these testers would have well formed notions of what data change means, yet their understanding and feedback informed our development of clearer potential use cases in different domain-specific contexts.

Scrutinizing mockups during interviews. Our semi-structured interviews were primarily designed to gather foundational information about scientist's work with changing datasets. We also elicited feedback on mockups of a web-based user interface to help evaluate our portrayal of data change since we used actual calculations from Dac-Man and the project's statistical analyses. Feedback emphasized that facilitating exploration of large amounts of changes requires the ability for users to conduct domain-specific analyses. Generalized overviews were primarily useful for sorting and filtering large lists of outputs.

Assessing documentation. A key aspect of a tool's usefulness is its documentation. Writing documentation is not necessarily a simple task. Research prototypes often result in a basic README with minimal instructions for the users to be able to use the tool. The initial documentation written for Dac-Man primarily consisted of a README and command line help. Our usability testing showed that users with different levels of expertise often follow the README instructions differently, some skimming quickly to just start running the tool while others tried to digest the overview in greater detail. Testing clarified areas to focus on immediately for an upcoming release, including information needed to quickly try the tool. Feedback noted that documentation for advanced users outlining how to integrate the tool into more complex workflows or computing environments was likely a necessity for wider adoption. These insights illustrated that scientific software projects need to offer multiple forms of documentation for different types of users and contexts.

- Test tools with stakeholders from the core design use case to ensure key features work
- Discuss UI mockups during interviews, even if a UI is not going to be built, to gather design feedback and simultaneously assess the conceptualization of the design problem.
- Offer multiple forms of documentation and test with different types of users in varying contexts (e.g., web-based, command line).

3.2 Organizing and conducting evaluations

Historically usability tests are conducted with some degree of artificiality, including well structured tasks for users to complete and lab-based environments with the software pre-installed and configured (although recent remote and mobile testing is more organic). Providing a tester with some flexibility is a way to learn more about their everyday computing practices and lessens the artificiality of the test session. Our mix of usability evaluations that had some flexibility in tasks, discussion around mockups, and interview format allowed users to draw upon a variety of their work practices and reduced some of the artificiality of the process.

Setting the context for an evaluation. Setting the context for interviews and evaluations is something teams need to consider on an ongoing basis. Users will not normally sit down to work with a new tool while someone is present asking them questions or pushing them to complete certain tasks. We conducted usability tests by having either an in-person or video meeting with our six testers, providing some tasks but primarily letting users explore features that caught their attention in the context of their own datasets after installing the tool on their preferred system, whether desktop or HPC. We did this after thinking about how to best setup the context for test users so that they have appropriate expectations for the tool at the time. In this case, we wanted to see how they incorporated the prototype into their regular routines.

For tools that are meant to function in a range of computing environments, having testers install the software in their preferred computing environment helps both surface bugs and identify potentially diverse systems the tool may end up running on. Testing prototypes and asking the user to install it on their own computing system requires deciding how to package materials for testing. In an effort to simplify the overhead we gave users access to our development repository[3]. This resulted in some confusion and distraction among our testers since developer resources were also available in the repository. These are artifacts that we should have cleaned and removed for a more polished test package. One tester took the opportunity to file a ticket identifying a bug. The ability to do this is less common in usability tests where the situation is fully controlled and feedback directed through the team members conducting the session. Teams developing tools openly are likely to find users providing feedback through issue trackers and will need to determine how to assess the insights offered. Information obtained this way should be considered when designing more systematic usability tests or other user research activities.

Most of our users ended up installing the prototype on their laptop in the session, noting that they like to try tools locally at first, however a couple of astronomy testers did run the tool on an High Performance Computing (HPC) system that stores astronomy data releases. Relying on users everyday computing systems during testing can surface key issues outside of the researcher's control regarding the configuration, stability, and consistency of these environments. In a couple of cases, we encountered issues which affected

[3] Dac-Man is open source but at the time of testing had not been made public due to institutional review requirements.

tester's ability to complete tasks. In one case, an HPC machine's entire file storage system unexpectedly became unavailable during the testing, entirely disrupting the task the user was working on. In another case, the environment didn't support the tools we expected for a more advanced feature that a user decided to try out. Scientific software teams having users test in less controlled computing environments should expect issues and anticipate that they may impact the process and/or results.

Balancing user exploration and task completion. We approached our usability evaluations with flexibility in mind so that testers could explore the features of Dac-Man while we also probed their perceptions of data change. Taking this approach required we balance consistency of task completion among our testers, sacrificing some potential rigor in the user research process, with their explorations and the knowledge we gained.

Our evaluations were designed with four overarching tasks during an hour long session, ranging from installing the tool to running it on a sample dataset then the user's own dataset. We ended up encountering a fair amount of inconsistency with how many tasks users could actually accomplish in an hour since they would often deviate and explore different features or play with their datasets. This inconsistency was an expected challenge since we wanted testers to explore the problem space, and using complex tools can require more time to examine and play with compared with more everyday consumer systems. A way to briefly get a sense of the potential impact of this issue is by piloting the test protocol with a team member – who is not developing the tool or designing the study to judge

whether there is enough time for both exploration and task completion in the session.

A highlight of our evaluation process's flexibility was one tester taking the opportunity to explore what data change means in their specific scientific context by trying to build a domain specific plugin. Rather than test with our sample dataset task they ran Dac-Man commands on their own CSV files then took a copy of one file and began making changes to see how it affected the change summary outputs our tool produces out of the box. This tester's exploration was simple, just copying a CSV file and opening it to systematically change some values such as a header or data point. This individual then used Dac-Man to compare the original CSV file to the newly modified CSV file. As this individual worked through the outputs and reflected on the changes made, they were able to develop a stronger grasp on the types of outputs the tool could produce. Noticing an option to use plugins in the README this individual decided to try writing a simple script to compare particular data values. The README offered no information about how to construct plugins, only instructions on having the tool use one provided at run time. The script this user wrote failed to run, but their curiosity and the opportunity to try producing a change analysis allowed them to develop a stronger grasp on data change and a better understanding of our prototype.

Our approach valued exploration at the cost of some task consistency, but we were not trying to measure how long achieving a task took or other traditional usability measures. We found such experiences valuable and overall worthwhile. We ended up experiencing different computing environments and practices that our potential users work with providing a broader view on our tool and its utility. We faced situations where circumstances outside our or the tester's control arose (e.g., an HPC filesystem going down in the middle of a test) and still obtained useful data by discussing the larger concepts at hand. Depending on a team's goals for evaluative testing they should balance the need to have users consistently complete all desired tasks with the opportunity for them to explore the system in question and the problem space at hand. This may vary based upon the nascence of the project and the team's ability to gather different types of data.

- Pilot a test or interview protocol with a team member to assess how much time is needed to explore the tool and complete tasks during a session.
- Evaluate the context for evaluations repeatedly over time, balancing tester's ability to use their regular computing environments with reliability & structure of a system.
- It is often informative to have users test in their natural work environments. However, having users test in their own computing environment can result in unexpected issues & may impact results or the ability to complete all tasks.
- Provide testers with flexibility to explore a tool balanced with their ability to complete a sufficient number of tasks to have consistent and rigorous enough results across the study.

3.3 Cross cutting challenges when mixing methods

Our work started with a notion of data change that was observed in some collaborations. The concept of data change required deeper investigation and refinement over time through our research approaches. Some scientific software projects likely begin with a more constrained and defined problem to tackle. In either situation, scientific software teams conducting user research must determine which methods to incorporate and how to reflexively balance the insights gained with potential biases and other impacts to rigor in execution. We structured our research process with a mix of methods allowing us to take advantage of the rigor of each of the methods while reducing the bias that might occur from using any single method.

Qualitative research's interpretive nature helps us learn from our diverse informants about their ways of working. Our selection of informants, both their disciplinary and project history as well as particular roles in a team, influences the perspectives we hear. Teams employing these methods need to constantly poke at the insights developed, try to identify and keep in mind edge cases, incorporate diverse perspectives from team members with different experiences, and be up front about their motivations for particular decisions.

Challenges from team size. An initial challenge we faced was the size of our research team. One individual was primarily tasked with data collection at a given time, in contrast to commercial environments where there may be a larger number of user researchers assigned to any one product. Scientific software teams should consider how many team members to involve in interviews, tests, and other conversations with scientists depending on the availability and backgrounds of team members as well as the context for the engagement with an informant.

Our first set of interviews was conducted by a principal investigator with qualitative research experience while another member or two of the team listened in and occasionally asked follow up questions. Our second phase of data collection was conducted by a team member trained in qualitative research experience. Having one team member available to conduct interviews, tests, etc. can be helpful since consistency in the questioning can emerge more easily and an informant may be more comfortable when the conversation is one on one rather than many on one. Many to one conversations in contrast can help provide, and gather, more diverse viewpoints through questioning by a range of team members who might focus on different aspects of a problem. Each approach has its ups and downs and teams have to judge what will produce the most useful insights for their work. Piloting a protocol and trying multiple structures is one way to assess this choice.

Similarly, usability tests can be moderated with a single person or with a small team lead by one member. In this case, we had a single individual conducting tests and interviews. This allowed the users to be quite candid and provided a degree of consistency and familiarity

with running this evaluation process. But this one interviewer's ability to attempt to have the user reflect on the experience of using this tool and work practices was limited since they were juggling keeping the session running somewhat on time with probing for more details. Having an additional team member present to observe and at appropriate moments ask questions would help round out the information collected, a strategy we employed in previous projects [11]. This team member could bring a different perspective on the tool being tested from their background (e.g., a developer) and ask different follow-up questions based on their interests/experiences to generate additional insights. In addition, it is also worth considering whether to ask multiple domain scientists to try the tool in a group setting to foster different discussions as Macaulay et al. [6] tried.

We find that being flexible and discussing potential approaches as a team is important to the successful adoption of user research methods and balancing the number of team members to involve in data collection or analysis. The person leading data collection should be trained in these methods, but it is sometimes reasonable to include different team members with other experiences. Science software teams should take advantage of the skills available among their members and determine if training more of the team in qualitative techniques is a good use of resources. Qualitative research is as much art as science and like any activity requires repeated experience for a researcher to improve. If a team cannot include other members in data collection then those individuals might contribute through conversations about findings. These individuals might even gain experience by helping review and analyze data to identify interesting insights (human subjects protocols may impact this opportunity), drawing upon the different backgrounds and perspectives of team members while trying to avoid group thinking.

Pragmatically balancing rigor and bias. Conducting semi-structured interviews, and adopting such a stance to our usability testing, helped us listen to the thoughts and concerns of stakeholders while trying to limit how much we impose our assumptions. The choice and timing of questions asked shapes conversations, and analyzing this data in concert with our project's other research efforts affects our findings. Our exploratory interviews were conducted with four astronomers and earth scientists to identify initial use cases and design considerations. The astronomers had a coherent, computationally general use case where they needed to compare two releases of millions of hierarchically organized files for changes where the first-level of change comparison can be performed by domain-agnostic methods. The earth scientists we engaged with had a more domain-specific issue which needed a direct approach that relied on domain knowledge of data organization and format.

The astronomy case led us to the design of Dan-Man, i.e., a framework that initially uses a black-box approach to comparing data files. This problem was tractable and led to the team focusing on the astronomy style use cases. The inevitable partiality that emerged from this initial data collection was a trade off we had to make given project time pressures, availability of informants, and so on. The earth sciences case informed the Dac-Man plugins that enable data change analyses that rely on data formats, methods, or metrics specific to a particular domain. Our iterative, multi-method process enabled us to prioritize areas for inquiry while also refining our insights and improving the framework. The user research findings helped everyone on the Deduce project better understand and conceptualize data change, noting when statistical metrics of change may or may not fit with scientific practices and identifying needs for tools or frameworks to calculate change as part of workflows.

We could have conducted in-depth ethnographic (the first author's primary approach) to explore work in these domains. This would have required both more time and potentially a larger team to explore the sociology of this work with greater rigor. Instead we labored to keep the potentially limited nature of our use cases in mind and worked to evaluate them through future interviews and tests of the tool built. When conducting six usability tests of the resulting Dac-Man tool we maintained our flexible approach rather than conducting a traditionally rigorous lab-based evaluation. For a more quantitative minded user researcher this decision to sacrifice a focus on micro tasks and the ability to quantitatively measure results may have been a less desirable decision. However, we obtained insights that helped revise the tool, its outputs, and documentation so that a release could be produced that addresses a wide range of use cases, and variable levels of expertise in the scientific community.

- Teams conducting qualitative research should always recognize their biases, as well as biases from participant selection and account for this when interpreting results.
- Balance having a single member or multiple members conduct interviews or test sessions since this impacts dynamics with the participant & the data collected.
- Recognize and pragmatically balance research rigor with the need to build & deliver useful tools by prioritizing actionable use cases while leaving room for expansion with further work.

4 Conclusions

Scientific software teams often face many challenges due to emerging requirements and varying amounts of domain knowledge. Our experiences adopting user research methods allows us to address these challenges. Incorporating multiple user research methods helped our team shape the Dac-Man tool from initial conceptualization to first release to address diverse scientific needs. Our experiences show that being agile and flexible helps science software teams better match the rhythms of scientific collaborations. Teams need to think about the who, what, when, where, and why as they incorporate user research techniques. Balancing who you study and have evaluate outputs from a project greatly influences the insights obtained. Incorporating foundational, generative, and evaluative methods enables a science software team to gather a variety of insights from different perspectives while endeavoring to be sufficiently rigorous. We find in this case, and in some of our past work [10, 11], that teams should employ user research processes in a manner such that they frequently iterate on the project goals. Critical milestones of the project might provide useful guidelines on when and what can be evaluated or when to collect more foundational insights.

With this paper we have offered reflections on our experiences conducting user research in the Deduce project. The successes and challenges that emerged from our work offer scientific software teams an oppor-

tunity to ponder how they might gather foundational and evaluative data to support their project's goals. The three methods we have employed thus far by no means cover the breadth of possible user research approaches. Interested scientific software teams should examine general human-computer interaction introductions [13, 14], resources such as the *Interaction Design Foundation* and its freely available literature[4] or *Code for America's* qualitative research guide [2], or reach out to other teams at their university or workplace who have such a background.

5 Acknowledgements

The authors wish to thank the members of the Deduce team and our study participants for their insights and feedback. This work is supported by the U.S. Department of Energy, Office of Science and Office of Advanced Scientific Computing Research (ASCR) under Contract No. DE-AC02-05CH11231.

References

[1] Aragon, C. R., Poon, S. S., Aldering, G. S., Thomas, R. C., and Quimby, R. Using visual analytics to develop situation awareness in astrophysics. *Information Visualization 8*, 1 (2009), 30–41.

[2] Code for America. Qualitative research at code for america. https://info.codeforamerica.org/qualitative-research, 2020. Accessed 9-July-2020.

[3] De Roure, D., and Goble, C. Software design for empowering scientists. *Software, IEEE 26*, 1 (2009), 88–95.

[4] Ghoshal, D., Ramakrishnan, L., and Agarwal, D. Dac-man: Data change management for scientific datasets on hpc systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis* (Piscataway, NJ, USA, 2018), SC '18, IEEE Press, pp. 72:1–72:13.

[5] Hine, C. *Virtual Ethnography*. SAGE Publications, London, UK, 2000.

[6] Macaulay, C., Sloan, D., Xinyi, J., Forbes, P., Loynton, S., Swedlow, J. R., and Gregor, P. Usability and user-centered design in scientific software development. *Software, IEEE 26*, 1 (2009), 96–102.

[7] Paine, D., Ghoshal, D., and Ramakrishnan, L. Investigating scientific data change with user research methods. Tech. Rep. LBNL-2001347, Lawrence Berkeley National Laboratory, 2020.

[8] Paine, D., and Ramakrishnan, L. Surfacing data change in scientific work. In *Information in Contemporary Society* (2019), N. G. Taylor, C. Christian-Lamb, M. H. Martin, and B. Nardi, Eds., vol. 11420, Springer International Publishing, pp. 15–26.

[9] Poon, S. S., Thomas, R. C., Aragon, C. R., and Lee, B. Context-linked virtual assistants for distributed teams: an astrophysics case study. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work* (1460623, 2008), ACM, pp. 361–370.

[10] Ramakrishnan, L., and Gunter, D. Ten principles for creating usable software for science. In *2017 IEEE 13th International Conference on e-Science (e-Science)* (2017), pp. 210–218.

[11] Ramakrishnan, L., Poon, S., Hendrix, V., Gunter, D., Pastorello, G. Z., and Agarwal, D. Experiences with user-centered design for the tigres workflow api. In *2014 IEEE 10th International Conference on e-Science* (2014), vol. 1, pp. 290–297.

[12] Segal, J. Software development cultures and cooperation problems: A field study of the early stages of development of software for a scientific community. *Computer Supported Cooperative Work (CSCW) 18*, 5 (2009), 581–606.

[4] The *Interaction Design Foundation's* resources are a thorough starting point. https://www.interaction-design.org/literature

[13] Sharp, H., Rogers, Y., and Preece, J. *Interaction design: beyond human-computer interaction*, 2nd ed. John Wiley and Sons Inc., 2007.

[14] Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., and Diakopoulos, N. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (6th Edition)*, 6th ed. Pearson, 2016.

[15] Weiss, R. S. *Learning From Strangers: The Art and Method of Qualitative Interview Studies*. The Free Press, New York, NY, 1995.