

UC Berkeley

UC Berkeley Previously Published Works

Title

Introducing The User Experience Department

Permalink

<https://escholarship.org/uc/item/9889h7xm>

Journal

Computing in Science & Engineering, 27(4)

ISSN

1521-9615

Authors

Paine, Drew

Cohn, Marisa Leavitt

Publication Date

2025-10-01

DOI

10.1109/mcse.2025.3628203

Peer reviewed

Introducing The User Experience Department

Drew Paine, Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, USA

Marisa Leavitt Cohn, IT University of Copenhagen, DK-2300 Copenhagen, Denmark

DOI: [10.1109/MCSE.2025.3628203](https://doi.org/10.1109/MCSE.2025.3628203)

Date of current version 9 December 2025.

ABSTRACT

Whether you are an end-user or developer of systems in your domain, your work depends in some way upon access to usable and reliable software. With this commentary we are excited to introduce the User Experience department in Computing Science & Engineering, to examine the user experience of computing in scientific and engineering research by exploring the fascinating dynamics encountered by end-users, end-user developers, engineers at scientific facilities, and others as they work to create and use advanced software.

INTRODUCTION

Software is an essential and powerful resource for scientific and engineering research. Whether you are an end user or developer of systems in your domain, your work depends in some way upon access to usable and reliable software. While software for research is often designed primarily with function and utility in mind, the usability of these tools is still of great consequence to how work gets done. A poorly designed scientific workflow interface directly hinders the reproducibility of research by causing an end user to incorrectly capture configuration decisions necessary to understand an analysis in the future when trying to re-execute the workflow. A high-performance computing package manager may be difficult to learn due to unclear documentation alongside unfamiliar interfaces hindering a new user's onboarding and consequently overall community adoption, reducing the case for continued project funding. Further muddying or obfuscating human understanding of research software's operation and action is especially fraught in an era where scientific and engineering domains are increasingly incorporating artificial intelligence (AI) and machine learning (ML) tools and methods.

Fundamentally, the way these and other tools shape user experience is of increasing significance. By user experience, we include not only questions of whether user interfaces are pleasant and easy to use, but broader questions about how software

- › shapes the quality of work experiences
- › mediates collaboration and communication across domains of expertise
- › acts as a means to organize scientific and engineering endeavors through its routines, procedures, and representations.

In pursuit of these questions, we are excited to introduce the User Experience Department in *Computing in Science & Engineering (CiSE)*, which will focus on user experience (UX) with the aim to open up these conversations to the magazine's readership.

This department will examine the user experience of computing in scientific and engineering research by exploring the fascinating dynamics encountered by end users, end-user developers, engineers at scientific facilities, and others as they work to create and use advanced software. The dynamics at hand span across the social, organizational, and technical issues that affect the development, stewardship, and ongoing use of software in scientific and engineering contexts. As department editors, we aim to bring commentary to this community, focusing on concerns and experiences that impact a wide range of projects, products, users, and developers. Above all, we are eager to address topics of interest to the community and encourage readers to contact us with topics of concern or curiosity (more details later in this column).

Here, we lay out our perspective and motivation for discussing UX in the context of *CiSE*. We provide a brief characterization of our perspective on UX for research software. UX is important, regardless of whether we focus on software developed in-house to solve a narrow problem or on larger, more complex software systems that address multifaceted concerns stewarded by large, distributed teams across institutions. Users' experiences with research software vary from unobtrusive and simple to overly complex and irritating to unusable in ways that inhibit research inquiry. We all encounter services, systems, or applications whose design leaves us frustrated, confused, or worst of all, unable to accomplish the goal or task we set out to achieve. With this department, we draw upon these experiences within the community to surface insights about the unique character of UX as a set of challenges and concerns within computing for science and engineering. Furthermore, we utilize this department to highlight insights from the broader UX research field in ways that translate to the *CiSE* audience and help readers improve the development and use of research software.

We can all draw upon our own experiences to form ideas of good or bad UX in software. However, UX is a formal discipline, situated within the broader research agenda of user- or human-centered design. User- or human-centered design is an iterative design process taking the actual needs, wants, and limitations of end users into account during each phase of the design process [1], [2]. UX encompasses a specialized form of practitioner expertise, with dedicated teams who conduct work that contributes to many successful commercial software and services. These teams shape everything from a potential user's first impression of a product to their onboarding and ongoing use. As an academic field, UX comprises an applied social and technical discipline focused on designing, building, and evaluating software or services to ensure they are usable, reliable, and even joyful for people [3]. Researchers and engineers with UX training are increasingly part of scientific software teams at universities and other research laboratories. While roles and titles vary, they commonly cover UX research, design, and engineering work as part of a research software engineering role (see "*Some Common User Experience Terminology*"). Overall, individuals in scientific UX roles play a crucial role supporting the development process, making the resulting products simpler to use and more relevant for their intended user bases.

We aim to further coalesce and advance the practice of UX in scientific and engineering software environments. In the remainder of this column, we situate UX in scientific software with an abstract example, briefly review cases from our own qualitative research that investigates scientific software practices and user experience, and underline how we see social sciences perspectives offering valuable insights on research software work to *CiSE* readers. We conclude with driving questions motivating our future columns and our desire to hear from you, the reader, about topics of interest.

Sidebar: SOME COMMON USER EXPERIENCE TERMINOLOGY

For quick reference we synthesize some core terminology and concepts spanning this commentary.

- › User- or human-centered design: This is an iterative design process taking the actual needs, wants, and limitations of end users into account during each phase of the design process [1], [2].
- › User experience (UX): This is an applied social and technical discipline focused on designing, building, and evaluating software or services to ensure they are usable, reliable, and even joyful for people [3].
- › Sociotechnical design: This is a conceptual framing that understands social and technical aspects of organizational work as fundamentally intertwined in ways that are particular to specific contexts where software mediates work [4]. A sociotechnical approach involves paying attention to the mutually social and technical dynamics in a given context as well as how these dynamics vary across place and time and disciplinary perspectives.
- › Scientific UX practitioner: These are three common UX roles in research software environments include UX Researcher, UX Designer, and UX Engineer. Individual practitioners often perform activities associated with more than one role over the course of a project given the limited size of teams.
 - UX researcher: These are individuals conducting foundational or evaluative research by employing qualitative or quantitative methods (e.g., semistructured interviews, participant observation, surveys, and so on). Foundational research is used to identify fundamental needs and challenges that affect product requirements. Evaluative identifies usability issues with a product that require enhancement.
 - UX designer: These are individuals designing solutions to address user needs and challenges. Designers sketch user interface concepts based on requirements identified through research, define the end-to-end experience of a product (often with a journey map), and help conduct usability evaluations to identify issues with products.
 - UX engineer: These are individuals implementing designs as part of a functioning system. Often develop graphical user interfaces, particularly web-based, and coordinate with back-end or full stack engineers to ensure a fully functioning tool or service.
- › Journey map: This is a visual representation of an end user's process for accomplishing a goal with a product [5], [6]. Helpful as an analytical tool with sociotechnical data to understand end user and organizational perspectives and processes that need to be addressed when designing products or systems.

SITUATING UX IN COMPUTATIONAL SCIENCE AND ENGINEERING

The *CiSE* community has long discussed the quality and reliability of computational software and systems, given their direct impacts on knowledge production and reproducibility. Members of this community have already begun to recognize the importance of UX as a broader set of concerns that expand as the complexity of computing and software systems (and the facilities providing them) grows. Increasingly, the usability of software and systems impacts a research team's ability to accomplish their goals. For example, the usability of matplotlib's application programming interface (API) and documentation directly shapes researchers' ability to efficiently and effectively visualize their data when they integrate matplotlib into a complex data analysis pipeline [7].

Researchers and practitioners across disciplines have seen their means of interaction with research software expand from command-line interfaces to multifaceted APIs and graphical user interfaces, accompanied by varied documentation. UX approaches are able to support all research software stakeholders by helping to reduce their frustrations and by improving the software's utility for scientific and engineering research problems. UX knowledge is best applied not only to end-user interfaces (GUIs, APIs, CLIs, and documentation), but also to organizational processes shaping the development, delivery, and support of an essential piece of software. For example, imagine you are a scientist attempting to check whether last night's telescope data are available for processing and analysis. A task as simple as checking the availability of new data and the computing systems you expect to use to process and analyze this data depends on whether a facility can reliably communicate status information. This expectation is common for web services, but not always easily managed by providers of scientific infrastructures, which often span bespoke or long-lived legacy systems as well as commercial resources.

Presumably, it is fairly simple for a facility to communicate its system status via a webpage or command line query tied to a simple database. However, imagine you are a UX designer who is tasked with redesigning a status webpage because a large number of support tickets indicate it is difficult to determine when a machine is available to use. As a UX designer, you begin by examining the current mechanisms for communicating resource statuses (whether via a webpage, command line, or email). You systematically examine what information these mechanisms provide, gather analytics on current usage patterns, review support tickets shared by user engagement staff, and survey or interview a variety of end users to understand how they use the existing interfaces. With your research data in hand, you analyze it and produce a journey map [5], [6] that visualizes the end-to-end experience and touchpoints that end users and staff encounter.

Key to your journey map are highlighted pain points where problems are faced by end users. You end up noticing something curious about identified pain points. The usability of the existing status webpage was not really a major issue. Instead, end users report issues with the information being shared on the webpage. These end users state that the status information is

out of line with their interactions when accessing the latest night's datasets. Interviewees consistently told you that the status website claims last night's dataset, and the computing resources required to process it, are fully available. However, when attempting to run an analysis in a Jupyter Notebook they find that their code is lagging. In particular, it hangs when trying to access the latest data, and they are frequently disconnected from the system. You find this odd and pivot to also interview operations staff to investigate how decisions are made about the communication of a particular system's status. These new interviews make it clear that internal examination of system errors is a varied, sometimes time-consuming process before the facility is ready to communicate publicly. Adding these details to the journey map illustrates how these processes slow down communication and directly lead to end-user pain points. Your UX work reveals that the simple status page is dependent on a complex, varied organizational process that end users are not aware of that requires time to ensure accurate information can be shared.

In this scenario, as a UX designer, you end up not only designing a new status page but also facilitating discussions with other staff about their processes to ensure status information can be updated more rapidly. Leveraging your journey map with pain points, you are able to work with operations staff to redesign their internal approaches and establish updated policies for updating the community when a system is not fully operational. As a UX designer, your task started with a webpage redesign but unearthed the need to revisit internal procedures and policies so that the end-user facing status webpage can help the facility deliver a smooth and informative overall experience for these astronomers' everyday data analysis work.

CASES FROM SOCIOTECHNICAL STUDIES OF SCIENTIFIC SOFTWARE

In the aforementioned abstract example, we show how the end-to-end design and usability of the varied services and systems, along with the underlying internal processes of these providers, fundamentally shapes the UX of a key computing site for scientific work. A particular area of research relevant to the UX Department explores such dynamics through the lens of sociotechnical design of organizational work. Sociotechnical refers to a conceptual framing that understands social and technical aspects of organizational work as fundamentally intertwined in ways that are particular to specific contexts where software mediates work. Taking a sociotechnical approach means attending to the mutually constructed dynamics between social and technical aspects in a given context as well as how these dynamics vary across place and time and disciplinary perspective. In adopting this lens, we consider UX to broadly encompass the ways such sociotechnical dynamics of software impact the experiences of conducting science and engineering research in everyday work settings. We briefly discuss three cases from our own prior research.

In one case, Paine and Lee [8] examine how testing and evaluation—concerns of longstanding interest to CiSE—are shaped by different testing tools, regimes, and schemas in a scientific group. The author's ethnographic research explored a cosmology team's software pipeline development, a primary object of this team's science. Paine and Lee's study presents a distinct

view on scientific software testing practices that demonstrates how this group's scientific method is instantiated in part through their sociotechnical software practices. These software practices are distinct but complementary to canonical software engineering methods. In particular, software developers typically formalize testing by writing and automating execution of unit tests to verify correct behavior of particular pieces of code, using a language's testing framework, along with a full regression test suite to repeatedly evaluate overall application behavior and ensure functionality is stable over time. In contrast, Paine and Lee showed how this cosmology group crafted a practice for systematically testing changes to their software without automated code-based unit or regression testing that would have required a framework from the programming language in use. Instead, members of this group selected a data product representing one night's observation from their radio telescope then took the latest changes made to their complex data analysis software to process that data product. Out of this processing and analysis, this cosmology group generated a set of carefully designed plots that could be compared against a control set. Enabling comparison of version A to B of plots enabled these scientists to evaluate the effects of changing a section of their analysis pipeline's code (e.g., adjustment of an algorithm) on the scientific result. Paine and Lee's study demonstrated that this plot-driven testing approach enabled this cosmology group to overcome a programming language limitation (in this case a lack of strong testing framework support at that time) to establish and sustain the conditions in which they ensure their scientific method encapsulation in software is reliable and reproducible.

In another case, Mazmanian, Cohn, and Dourish [9] explore how, for a space science project, the software tools used by different domain experts (navigation, spacecraft operations, scientific planning) generate divergent visualizations of the same phenomena (how a spacecraft is physically performing millions of miles away). The divergent visualizations lead to tensions between teams when arriving at the shared solutions necessary to conduct scientific observations. Here, software used by engineers during mission planning to flag out-of-bounds solutions to changing the spacecraft's orientation are designed with different interactive features than those built for and used by scientists to assess the same spacecraft as they plan a scientific observation. The two teams used related criteria about the spacecraft, but distinct representations of the problem—rooted in physical engineering concerns—leading to misunderstandings. The design and use of two distinct visualizations, presented in different user interfaces, of the same underlying data created a scientific UX challenge. These UX problems arise not for the individual scientific or engineering user, but in how different user experiences are configured by different needs, scales, and rhythms of work.

As both brief previous cases show, what counts as usable scientific or research software can vary based on perspective and an individual's relationship to a project and its products. Hinsien [10] outlined varied roles of code in the conduct of scientific work, noting how they may be multifaceted and overlapping but importantly evolve over time as a code writer's purpose changes or the needs of a scientific community change. Cohn's [11] ethnographic study of a NASA spacecraft mission explores these varied roles by unpacking the changing temporal characteristics of different pieces of software essential to the success of this longitudinal mission. End users and developers bring together old, legacy code with new in the course of

managing a complex stack necessary for a research mission to be successful decades after core elements were built and launched into space. In this case, UX might refer not only to whether software tools successfully accomplish a given task, but also how the criteria used to assess usability shift over time, depending on changing programming paradigms, management methodologies, and organizational perspectives on what software can do or how it can be updated. This ethnographic study draws attention to the ongoing user experience work required to keep legacy code “present” in the changing paradigm of a team’s work practices.

WHAT CAN RESEARCH SOFTWARE PRACTITIONERS LEARN FROM SOCIAL SCIENCE KNOWLEDGE?

The aforementioned studies highlighted illustrate how sociotechnical design perspectives bring different orientations to system design and use that complement other technical design approaches (e.g., software engineering practices, hardware design). This department will draw from a range of academic research, from computer science-adjacent fields such as human–computer interaction (HCI) or computer-supported cooperative work to social sciences such as psychology, sociology, organizational, and science and technology studies, to bring such perspectives to *CiSE* readers. We strongly believe that the sociotechnical design expertise of scientific UX practitioners, and scholars in associated academic fields, can offer the *CiSE* reader developing and using software different conceptual tools and perspectives to incorporate in their own research and development approaches. Members of the *CiSE* community often implicitly understand many of these concerns from their own research and development experiences, and we find that practitioners of research software stand to benefit from the systematically gathered social sciences knowledge that UX methods translate into design actions.

Design and research are at their core reflective practices where practitioners are trained to ponder and evaluate their own roles in work. The explorations of these fields can provide key contextual insights, and with this department we aim to provide a bridge between the insights from these social science literatures to that of technical practitioners in scientific software. Social science disciplinary perspectives bring an array of approaches through which practitioners and scholars investigate and shine light on the complex work of science and software’s role in it. Importantly sociotechnical research and findings help unearth and discuss the complexities of seemingly disparate or mundane aspects to software work, from what counts as a software as part of careers and research work to offering a critical interrogation of seemingly everyday tasks like testing or maintaining code for long-term, reliable use as noted earlier [8], [11], [12].

Open Questions the Department Will Explore

Overall, we find that the knowledge generated through social sciences inquiries leads to a few key questions that we envision this UX Department exploring over time, including the following:

› What knowledge and skills do teams and organizations need to implement to build usable scientific software?

USER EXPERIENCE

- › How does the usability of research software systems impact the reproducibility of scientific results? How is designing one “simple” element wed up in a nexus of complexities?
- › How do shifting sociotechnical dynamics impact usability and stewardship of software as a core research product?
- › What do these shifts mean in practice for an end user’s experience? A developer-user’s experience? How does using AI/ML tools and techniques shape the user experience of complex research systems?
- › What do researchers and practitioners need to understand to design more usable AI/ML systems?

SUMMARIZING OUR DEPARTMENT AIMS

As coeditors we are fostering commentaries which examine and unpack the larger milieus in which scientists and engineers create and use software. We have seen the impacts of stakeholders with varied backgrounds on the usability, utility, and success of software teams and aim to give those voices a space in which to share their insights, challenges, and resources with the broader *CiSE* community. Whether your interest and focus is on (research) software engineering or scientific programming, AI/ML, high-performance or quantum computing systems, we aim to explore timely and timeless dynamics to the work of making research happen.

Contact Us and Get Involved

Have a topic that you would like us to explore? Want to contribute a case study or commentary based on you or your team’s experiences?

As department editors we are eager to hear from the community and consider how best to surface and foreground a broad range of voices and experiences. Please contact both of us with suggestions, questions, or general comments.

REFERENCES

- [1] B. Shneiderman, C. Plaisant, M. Cohen, S. Jacobs, N. Elmqvist, and N. Diakopoulos, *Designing the User Interface: Strategies for Effective Human-Computer Interaction (6th Edition)*, 6th ed. Pearson, 2016, p. 608.
- [2] Interaction Design Foundation-IxDF, “*What is User Centered Design (UCD)?*,” Interaction Design Foundation - IxDF. Accessed: Oct. 28, 2025. [Online]. Available: <https://www.interaction-design.org/literature/topics/user-centered-design>
- [3] D. Paine, “Framing user experience (UX) across the scientific software lifecycle,” Better Scientific Software (BSSw). Accessed: Nov. 08, 2024. [Online]. Available: https://bssw.io/blog_posts/framing-user-experience-ux-across-the-scientific-software-lifecycle

- [4] M. S. Ackerman, “The intellectual challenge of CSCW: the gap between social requirements and technical feasibility,” *Human-Computer Interaction*, vol. 15, no. 2, pp. 179–203, 2000, doi: 10.1207/s15327051hci1523_5.
- [5] J. Kalbach, *Mapping Experiences A Complete Guide to Customer Alignment Through Journeys, Blueprints & Diagrams*, 2nd edition. Sebastopol, CA: O’Reilly Media, 2021.
- [6] S. Gibbons, “Journey Mapping 101,” Nielsen Norman Group. Accessed: Oct. 28, 2025. [Online]. Available: <https://www.nngroup.com/articles/journey-mapping-101/>
- [7] “Matplotlib — Visualization with Python.” Accessed: Sep. 16, 2025. [Online]. Available: <https://matplotlib.org/>
- [8] D. Paine and C. P. Lee, “‘Who Has Plots?’: Contextualizing Scientific Software, Practice, and Visualizations,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 1, no. CSCW, Nov. 2017, doi: 10.1145/3134720.
- [9] M. Mazmanian, M. Cohn, and P. Dourish, “Dynamic Reconfiguration in Planetary Exploration: A Sociomaterial Ethnography,” *MIS Quarterly*, vol. 38, no. 3, pp. 831–848, 2014, doi: 10.25300/MISQ/2014/38.3.09.
- [10] K. Hinsén, “The Roles of Code in Computational Science,” *Comput. Sci. Eng.*, vol. 19, no. 1, pp. 78–82, 2017, doi: 10.1109/MCSE.2017.18.
- [11] M. L. Cohn, “Keeping Software Present: Software as a Timely Object for STS Studies of the Digital,” in *digitalSTS: A Field Guide for Science & Technology Studies*, J. Vertesi, D. Ribes, C. DiSalvo, Y. Loukissas, L. Forlano, D. K. Rosner, S. J. Jackson, and H. R. Shell, Eds., in *A Field Guide for Science & Technology Studies*, Princeton University Press, 2019, pp. 423–446. [Online]. Available: <http://www.jstor.org/stable/j.ctvc77mp9.31>
- [12] C. R. B. de Souza, J. Froehlich, and P. Dourish, “Seeking the source: software source code as a social and technical artifact,” presented at the Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work, ACM, 2005, pp. 197–206. doi: 10.1145/1099203.1099239.

DREW PAINE is a lead user experience researcher in the Scientific Data Division at the Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, USA. Contact him at pained@lbl.gov.

MARISA LEAVITT COHN is an associate professor at the IT University Copenhagen, DK-2300 Copenhagen, Denmark. Contact her at marisa@itu.dk.

Paine’s work at LBNL is supported by the Alfred P. Sloan Foundation, the U.S. National Science Foundation under Interagency Agreement #A2407-049-089-064206.0, and the U.S. Department of Energy Advanced Scientific Computing Research under contract DE-AC02-05CH11231. The views expressed do not necessarily reflect the views of any funding agency or any other organization.