Understanding Marine Scientist Software Tool Use

Matthew Lakier
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
matthew.lakier@uwaterloo.ca

Andrew Irwin
Department of Mathematics and Statistics
Dalhousie University
Halifax, Nova Scotia, Canada
a.irwin@dal.ca

Daniel Vogel
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
dvogel@uwaterloo.ca

Abstract

Marine science researchers are heavy users of software tools and systems such as statistics packages, visualization tools, and online data catalogues. Following a constructivist grounded theory approach, we conduct a semi-structured interview study of 23 marine science researchers and research supports within a North American university, to understand their perceptions of and approaches towards using both graphical and code-based software tools and systems. We propose the concept of fragmentation to represent how various factors lead to isolated pockets of views and practices concerning software tool use during the research process. These factors include informal learning of tools, preferences towards doing things from scratch, and a push towards more code-based tools. Based on our findings, we suggest design priorities for user interfaces that could more effectively help support marine scientists make and use software tools and systems.

CCS Concepts

• **Human-centered computing** → **Empirical studies in collaborative and social computing**.

Keywords

data work, data-centric science, oceanography

ACM Reference Format:
Matthew Lakier, Andrew Irwin, and Daniel Vogel. 2025. Understanding Marine Scientist Software Tool Use. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26–May 01, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3706598.3713621

1 Introduction

Marine science research involves a diverse set of specializations such as oceanographers, biologists, geneticists, and statisticians. They are heavy users of software tools and systems, including statistics packages like R and SPSS, visualization tools like Ocean Data View (ODV)[1], online data catalogues like GBIF[2] and OBIS[3], as well as custom tools created by labs or research support groups. Because of the interdisciplinary nature of the field, marine science researchers have varying preferences for and levels of expertise with software tools, and varying ways of understanding and doing computation-related aspects of their work. Better understanding the way these variances affect the work of marine science researchers could enable the design of user interfaces that better support their needs.

Past work has investigated ways that scientists write code (e.g., [27, 43]), ways that scientific software is developed (e.g. [10, 24, 34]), and specific marine scientist practices, such as their practices around data integration [33]. There are also works that have conducted studies in support of developing systems to address particular problems that scientists face, such as code versioning [25] and sensor data exploration [41]. Although marine scientists' work is largely and increasingly code-based, GUI tools are still used extensively. Our work focuses on marine scientists' overall practices around interactive software tool use, including both graphical and code-based tools, and addressing both tool creation and tool use.

To understand the role of software tools in marine science research, we conducted a semi-structured interview study of 23 marine science researchers and research supports within a North American university. We used a constructivist grounded theory approach [6]. Based on our analysis, we propose and define the concept of *fragmentation* as the formation and perpetuation of distinct, isolated pockets of views and practices concerning software tool creation and use. We propose fragmentation not as a problem to be solved, but as a way of understanding marine scientists' current views and practices, with both positive and negative aspects. We characterize forms of fragmentation, including *developing individualized views of practices in the field*, *doing ad hoc software engineering in scientific computing*, *resisting unfamiliar tools*, *doing things from scratch*, and *tools having limited reach*. Based on these and other conceptual categories, we propose three design priorities for user interfaces to integrate with fragmentation and mitigate its negative aspects, by exploring the potential for graphical interfaces, emphasizing collaborative design efforts, and supporting marine scientists with discovery and connection.

The primary contributions of this work are an understanding of marine scientist software tool use, presented in reference to the concept of fragmentation, and design priorities for how user interfaces could be designed to support marine scientists with their software tools.

[1] *Ocean Data View* [42] (https://odv.awi.de/)
[2] *Global Biodiversity Information Facility* (https://www.gbif.org/)
[3] *Ocean Biodiversity Information System* (https://obis.org/)

CHI '25, Yokohama, Japan

ACM ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3713621

2 Background and Related Work

We highlight the relation of this research to "data science", and we describe related past studies of data analysis processes among natural scientists and broader groups, and work investigating scientific software development.

2.1 Studies of data analysis processes

Data analysis is a core aspect of marine scientists' work, which may lead to the generalization that marine scientists are "data scientists". "Data science" has been used to refer to a variety of different practices involving obtaining information from data [39], sometimes at a broad level (e.g. [49]), or sometimes in corporate machine learning or artificial intelligence contexts [37]. Because of the ambiguity, some researchers have started using the term "data workers" to refer to "non-professional data scientists" [49] who "perform data analysis as part of their daily work but would not call themselves data scientists" [28]. Our marine scientist participants generally did not identify with the term "data scientist". Because data is only one aspect of their work, we refer to them more broadly as scientists. We describe some key types of work in this space to differentiate the focus of our analysis.

Several works describe the data analysis process of data scientists or workers. Based on an interview study in the context of ecological sensing data, Wallis et al. [52] describe a nine-stage project data life cycle including stages such as "capture" and "cleaning". Also based on an interview study, Muller et al. [32] characterize the process of how data science workers at IBM analyze data, with similar steps, such as "capture" and "curation". Jung et al. [22] conduct a field study of craft brewers in Korea to understand "how domain experts themselves may engage with data science tools as a type of end-user". They employ craftwork as a sensitizing concept, and outline four "tasks" for working with data (e.g., "monitoring variables", "hypothesizing relationships"). They suggest recommendations for data science tools to help support these tasks. Pedersen and Bossen [38] investigate data work in the context of healthcare business intelligence, describing aspects such as developing reports, integrating data from different sources, and deciding on data registration approaches, all serving as "an effort to manage tensions between the local and the global".

Several works propose categories or personas representative of data science skills and approaches. In their well-known work "Enterprise Data Analysis and Visualization", Kandel et al. [23] conduct interviews with analysts mostly outside of the natural sciences, across multiple organizations. They develop "analyst archetypes", summarize challenges associated with the "tasks" as part of the data analysis process (e.g., "wrangle", "model"), and propose implications for visual analytics tools. Crisan et al. [8] define "roles" of data scientists based on a literature review, and describe their process, including steps like "analysis" and "communication". The participants in our study mostly fall under Kandel et al.'s "scripter" and "application user" archetypes, and Crisan et al.'s "Technical Analyst" or "Moonlighter" roles, due to lack of formal tool training.

Several works describe how data scientists or workers do exploratory programming. Based on an interview study of data scientists mostly outside of natural science domains, Kery et al. [26] describe how they use computational notebooks. Similarly, based on interviews with and surveys of data workers, Subramanian et al. [49] compare the merits of scripts and computational notebooks for exploratory programming. Liu et al. [28] explore why and how data workers make use of "alternatives" for data sources, methods, and so on. They describe reasons for and barriers to exploring alternatives, and propose a taxonomy to categorize different kinds of alternatives.

In contrast with these works, we are not interested in describing stages of an analysis process or characterizing personas of different analyst types. We explore interactive tools beyond those associated with data analysis and beyond just programming tools.

2.2 Differentiating scientific software development from conventional software engineering

Past work has studied how scientists develop software, to understand differences from conventional software engineering approaches and propose ways to improve scientific software development.

A number of works conducted qualitative studies comparing conventional software engineering to scientific software engineering. This has been done in the context of climate modelling software developers [10], mathematicians and scientists [17, 43], and high-performance computing researchers [3]. Common findings include how scientific software has different requirements (e.g., "precisely repeatable results" [10]), involves iterative development, follows different debugging and testing processes, and that the software is considered secondary to the science. Segal [43] characterizes "professional end user developers" as "people [...] who work in highly technical knowledge-rich domains and who develop software to further their own professional goals", noting "they do not think of themselves as software developers". Based on a series of field studies with mathematicians and scientists, she compares professional end-user development with conventional software development. Kelly [24] pushes back against this characterization of scientific software development as a form of "end-user programming", based on a synthesis of her past studies of scientific software development. She proposes the term "knowledge acquisition" as an alternative. This is because in scientific domains, software is never "considered the end product", but it "increases knowledge in one or more knowledge domains".

A number of works in the space of empirical software engineering use quantitative surveys to characterize how scientists or data scientists develop software. These projects provide statistics for aspects such as which software tools are used at which project stage [53], which programming languages are used [35], and the value of software engineering training [15].

Several works study software developers or software engineers involved in developing scientific software, addressing aspects such as the extra forms of effort needed beyond development [51] (e.g., documentation, training), or describing unique characteristics of scientific software [5] (e.g., prioritizing correctness of science over software quality).

In contrast with these works, we do not investigate marine science software use from a software engineering perspective or describe how software engineering concepts could be imported into scientific software development. We conduct a qualitative analysis, focusing on a broader context beyond just software development, including both coding-based and graphical software.

2.3 Studies of natural scientists' processes

Past work has studied natural scientists' processes of developing infrastructure, standardization, collaborations, reproducibility and data sharing, and data production and management.

A number of works focus specifically on the effects or development of technological infrastructure in the natural sciences. Cutcher-Gershenfeld et al. [9] conduct a quantitative survey concerning engagement with a particular geoscience data sharing platform, highlighting potential barriers for engagement. Based on ethnographic fieldwork at ecological field research sites, Jackson and Barbrow [20] describe how computational infrastructural change impacts traditional fieldwork in ecology. While fieldwork came up in our interviews, our research does not focus on the stages of doing fieldwork. Steinhardt [46] provides a detailed description of the "breaking down" of an ocean observatory initiative, and challenges the notion of "failing fast". Neang et al. [34] investigate how members of a particular oceanography research lab work to make a specific software pipeline they are developing repurposable. Based on interviews, they present vignettes describing the different stages of the project as it moved from a "personal tool" to a "resource that is responsive to different stakeholder groups". We briefly touch on some of these ideas in our conceptual categories about the reach of tools and doing open science; however, the process of infrastructure development is not a main focus of our analysis.

In an ecology context, Jackson and Barbrow [21] describe practices around coordination efforts to increase standardization at a local lab and at a national observatory system, highlighting how both experience issues with standardization. At the local level, they describe how a protocol book is used for describing standards within the lab. At the global level, they describe challenges with coordination and enforcing standards in the observatory system. Unlike this work, we focus on a "local" context, and on aspects beyond lab protocol standards. Further, we explore tool use whether the tools are standardized or "radically non-standardized" [21], rather than focusing on standardization efforts.

In a biomedical context, Mao et al. [30] investigate challenges when data scientists collaborate with domain experts. They conduct a deductive qualitative analysis based on a collaboration framework proposed by Olson and Olson [36], mapping findings to categories such as "Common Ground" and "Technology Readiness". They do not distinguish "data scientists" from "data workers". Notably, they briefly mention "fragmented information", referring to information stored across many different systems, but this is a much narrower concept than our broader fragmentation concept.

Research drawing from workshops with domain scientists [11], field studies of bioinformaticians [18], and quantitative surveys of scientists [50] have investigated understandings of reproducibility and data sharing practices, including ways of sharing and reasons for sharing. Lowndes et al. [29] describe their own process as environmental scientists moving to more reproducible tools (e.g., from Excel to R). They note that scientists "tend to develop their own bespoke workarounds" for handling data due to lack of training. Our work highlights how bespoke software tool approaches are associated with many more factors than lack of training.

Other research has investigated scientific data production and management processes. Based on fieldwork at a biological station conducting agricultural work, Burton and Jackson [4] propose the concepts of "coherence" (concerning the "immediate situation") and "integrity" (concerning the "long-term"), and describe data production in relation to these two concepts. Specht et al. [44] investigate data management challenges in an Australian "science synthesis centre", reporting on quantitative effects of involvement with the centre, and challenges expected at different data workflow stages. Neang et al. [33] investigate how ocean scientists coordinate data integration, using the concept of articulation work as a lens. They characterize six "scenarios" representing types of collaboration between observationalists, and between observationalists and modellers. These scenarios address ideas including coming to shared understandings and communication and collaboration efforts.

Unlike these works, our analysis focuses in depth on practices around software tool use, and unlike most of these works, our study is in the context of marine science.

2.4 Studies about change in natural science research

Steinhardt and Jackson [47] provide a theory of what they call "anticipation work", which refers to the "persistent and permanent" practices that people engage in to "move toward some imagined future". Based on interviews and observational work, they provide three vignettes across projects in oceanography and ecology. Notably, they briefly point out how ecology is "a radically *un*standardized field", in that it "has struggled to converge on method and technique above the level of the individual site or PI research program". Our work investigates this lack of convergence in detail, in relation to tool use and in the context of marine science.

Continuing in the vein of Steinhardt and Jackson, Kuksenok et al. [27] use observations and interviews to investigate code work by oceanographers, specifically focusing on "scientists who code" and who have an "attitude of willingness toward uptake of new technologies". They propose a framework to describe the "process of uptake" of new tools or protocols as scientists move towards an "imagined perfect world" where "tools are (1) understandable, (2) perform the necessary task, and (3) persist over time".

Although these two works are both in the context of marine science, they focus specifically on change or orientation towards change. In contrast, our work focuses on software tool use in the present, and on tools broader than just code. We address resistance to change within one of our conceptual categories.

Table 1: Research stage and area of participants.

ID	Stage	Area	ID	Stage	Area
P1	PhD	Applied math & statistics	P13	Master's	Microbial genomics
P2	Postdoc	Marine biology	P14	PhD	Physical & chemical oceanography
P3	Master's	Biological oceanography	P15	PhD	Applied math & statistics
P4	PhD	Biological oceanography	P16	Postdoc	Microbiology
P5	Undergraduate	Biological oceanography	P17	PhD	Applied math & statistics
P6	Postdoc	Marine biology	P18	Professor	Physical oceanography
P7	Undergraduate	Biological oceanography	P19	Adjunct professor	Physical oceanography
P8	Postdoc	Biological oceanography	P20	PhD	Microbial genomics
P9	Postdoc	Immunology (Genomics)	P21	Staff	Biological oceanography
P10	Postdoc	Applied math & statistics	P22	Industry collaborator	Marine biology
P11	Professor	Applied math & statistics	P23	Staff	Earth science
P12	Professor	Bioinformatics			

3 Participants, Interviews, and Analysis

We used constructivist grounded theory [6], following an iterative approach of data collection and analysis. We understand our conceptual categories as constructed interpretations dependent on our own experiences and values, and situated in the specific context of study. The first and third author have backgrounds in HCI. The second author is a senior professor in applied math and statistics working within an oceanography lab at the university of study. Many past studies of scientific work practices have been conducted by those with software engineering backgrounds, and our views may differ from these perspectives.

We recruited 23 participants associated with the university of study (15 men, 8 women). There were 3 participants aged 18–24, 11 participants aged 25–34, 5 aged 35–44, 1 each aged 45-54, 55–64, and 65 or older, and 1 with unspecified age. Participant research stages and areas are listed in Table 1. We primarily focused on researchers, such as professors and graduate students, in the general area of marine science. We initially recruited from within a single oceanography lab (P1–P8), before expanding out to other marine science labs (8 total). We recruited two members of research support teams within marine science (P21 and P23), to gain additional insight on tools from the perspective of those who develop them. We also recruited one data science postdoc (P9) currently working in the space of immunology, because he was part of a shared genomics institute at the university with a strong representation of ocean microbe genomics researchers. Because marine science is highly interdisciplinary, our participants perform research in a range of subareas, including oceanography, marine biology, applied math and statistics, and ocean microbial genomics. As the study progressed, we used theoretical sampling to seek participants with particular knowledge based on theoretical concepts of interest, and to understand variation within our theoretical concepts.

We performed intensive interviews with participants, but early interviews included more informational questions to gain a descriptive understanding of the context. We extended our interview questions over the course of our study, as we became more sensitized towards topics such as transitioning between tools, areas of education, research priorities, and what it means for a tool to be flexible. Interviews were 50 minutes on average, ranging from 14 to 97 minutes. 21 interviews were recorded and transcribed for analysis. Two participants (P11 and P18) requested not to be recorded, for whom we wrote detailed notes for analysis. 19 interviews were conducted in-person, and 4 were conducted via videoconferencing software.

We used initial coding to characterize participants' thoughts, feelings, and actions regarding software tool use. We refined our codes through focused coding and constant comparison between statements made in different interviews. Throughout the process of interviewing and analysis, we used memoing as a way of comparing statements, codes, and other memos, defining concepts, identifying variations of ideas, and reasoning about theoretical categories. We also made extensive use of diagramming to better understand relationships between concepts and make more meaningful comparisons between participant statements.

4 Findings

Based on our analysis, we propose five highly-interconnected conceptual categories, F1–F5, that represent forms of fragmentation within marine science (Figure 1). In this section, we first give a high-level description of how our participants used software tools, followed by brief descriptions of five peripheral process categories, C1–C5, to provide important context for the core fragmentation concepts that follow. The italicized sentences at the beginning of each fragmentation concept describe how it acts as a form of fragmentation. Certain aspects of the conceptual categories we describe have been investigated by past work in related domains. Where relevant, we briefly describe how our findings confirm or contrast with such past work.

4.1 High-level Description of Tool Use

Our participants investigate a wide range of marine science problems, such as taxonomic classification (P16), ocean mixing processes (P19), microorganism genomics (P12), and animal tracking (P10).

Some of our participants engaged in field work and wet lab work, which involved collecting samples on research cruises, or making measurements of previously collected cultures in the lab. For in-lab equipment, data was extracted from equipment by manually reading values from displays, using connected portable media (e.g., USB keys), or through cloud storage-connected computers that interface with the equipment. Specialized analyses, such as elemental analysis

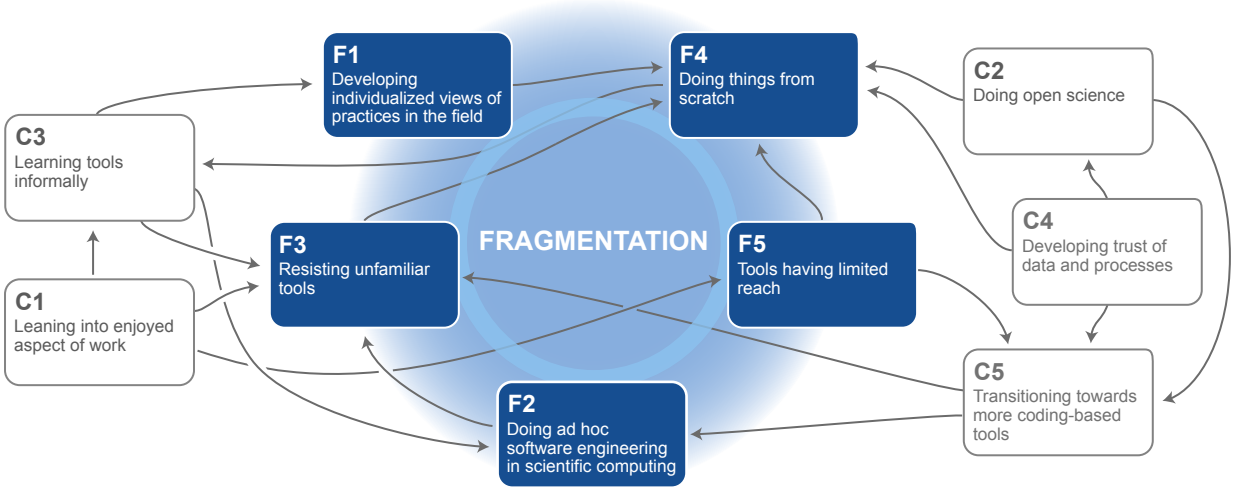

Figure 1: High-level concept map of the contextual (C) and fragmentation (F) concepts, illustrating interconnections and relationship to main concept of fragmentation. Arrows represent influences between concepts.

and gene sequencing, were delegated to other labs or facilities. Data was also collected from equipment such as gliders and floats.[4]

Preparing and analyzing data is important for the scientific work done by our participants. Participants worked with a variety of data types, such as sensor data (P11), acoustic data (P11), genome data (P12), image data (P2), tabular data like CSV and Excel files (P3), NetCDF files[5] (P15), and shapefiles[6] (P10). Data preparation involved quality checking (e.g., finding errors with online data or duplicate records), combining data in different formats from multiple sources (e.g., books, paper supplemental materials, online databases), taking subsets of data from larger data repositories, and cleaning data by removing outliers and modifying column formats. Visualizations were used for tasks such as identifying outliers, producing phylogenetic trees, and understanding statistical model output. Some participants dealt with large amounts of data (e.g., P16, hundreds of terabytes), necessitating use of high-performance computing infrastructure, whereas other participants dealt with small amounts of data using software like Excel.

Software tools are central to a range of marine scientists' research tasks, including for analysis, writing, note-taking, and sharing findings. Excel was used for both data collection and simple analysis. There was also some use of dedicated statistics and visualization tools like SPSS for statistics and ODV (Figure 2) for ocean map visualization. Coding-based tools, like R (e.g., in RStudio), Python, Matlab, were commonly used. Computational notebooks, such as Jupyter, were often used in communication and teaching contexts, but less common for research use. Scripts were often preferred for research due to modularity, ease of running in background, and lower perceived effort than notebooks. While GIS (Geographic Information System) software was used historically, there has been a transition away from these tools to alternative approaches like using R to generate maps. Participants used a variety of different approaches for troubleshooting software tool problems, including online help resources like StackOverflow and ChatGPT, online community support channels for specific tools (e.g., Discord or GitHub communities, paid customer support), documentation, and seeking help from colleagues.

Most participants did not identify with the term "data science", preferring alternatives like a *"scientist that works with data"* (P19). This is because they saw data science as having a different scope (e.g., machine learning-focused), or as more about problem solving for the sake of problem solving, rather than for science. Also, data work was only a part of their overall work; for example, P18 described data work as only an *"auxiliary"* component of oceanography work.

4.2 Contextually-relevant Concepts

The following five concepts are important to understanding the context of our core fragmentation concepts and situate them within in the priorities of marine scientists.

C1: Leaning into enjoyed aspect of work. Participants described preferring to focus on either the 'science side' or the 'tooling side' of their work. Among our participants in researcher roles, these were individual-level preferences towards certain types of work, not a division that could be wholly attributed to subarea. On the science side, participants enjoyed investigating their main scientific questions. Participants on the science side actively tried to make more time for scientific or analysis work, for example, by avoiding learning new tools until they are needed (P3), and using existing domain-specific tools to simplify technical aspects of work (P12). In contrast, participants on the tooling side enjoyed making software work, improving code efficiency, or thinking about data representations. These participants addressed research questions posed by collaborators (e.g., P9), developed their own software (e.g., P10), or made data analysis tools for researchers (e.g., P23).

C2: Doing open science. The push towards open science is affecting how marine scientists work. There is an increasing expectation

[4]Glider: Type of autonomous underwater vehicle. Float: Tool deployed in bodies of water to make measurements. Both carry sensing instrumentation.
[5]https://www.unidata.ucar.edu/software/netcdf/
[6]A vector format for geographic information.

to share code and data publicly. The effects of open science have been detailed in past work in related fields such as ecology [29, 31, 40]. When access to code or data is withheld, for example, due to sensitivity (P10) or to avoid being scooped (P4), reproducibility is hampered. For example, P15 mentioned avoiding sharing code publicly because she feels pressured to make demos to go with it. Past work provides more details on specific barriers to sharing software and data [16, 31, 50]. Making an effort to make research artifacts, code, and data more presentable for sharing was a common task among participants, who felt a sense of responsibility to make these artifacts useful. For example, ODV was used to make more *"beautiful"* map figures (P8).

C3: Learning tools informally. Participants generally learned tools in informal ways. This means they have varied educational backgrounds that often do not involve formal training in tools, programming, or computational thinking (e.g., P11). This finding provides further confirmation that, at least historically, scientists generally lack formal training in programming or software engineering [3, 11, 15].

Rather than seeking more formal training options, our participants generally learned computing skills and tools indirectly through doing their research, which is unsurprising in light of C1. One way this occurred was through trial-and-error. For example, P15 learned how to parameterize code so that she could more easily make tweaks suggested by her supervisor. Some tools have such a small user base that formal training options are not available. For example, P17 needed to learn how to use a package that he only knew about because his *"supervisor had collaborated with the people [who developed the package]"*.

C4: Developing trust of data and processes. Developing trust of data and processes was a key concern for participants. This aligns with past descriptions of scientists desiring confidence in the output of software [24, 34]. For example, in oceanography, deciding on the choice of gridding methods (interpolating data based on samples) is an important process that strongly affects whether resulting visualizations are trustworthy (e.g., P2, P15). Blind application of default gridding values in software like ODV can lead to untrustworthy output (P18). Our participants followed a number of approaches for developing trust, such as scrutinizing others' code, tools, and data based on presence in online repositories like CRAN (P18) or level of trust in the community (P15).

In marine science, a key context for developing trust is when converting raw signals from sensing equipment. These are in units that serve as proxies for desired units, such as how conductivity is a proxy for salinity. This need for trust is because the relationships between raw and desired units are *"subtle"* (P19), and proprietary software from equipment manufacturers obscures any assumptions going into the processing. Participants like P19 were able to develop higher trust in the process by implementing their own unit conversion code.

C5: Transitioning towards more coding-based tools. There is a shift towards more coding-centric tools because of advantages they provide, which parallel disadvantages of traditionally-used GUI tools. This shift highlights a number of criteria that participants consider important for their software tools, such as flexibility, power, and reproducibility. Also, many modern statistical methods are inherently computational, making code a requirement for their effective implementation (P10).

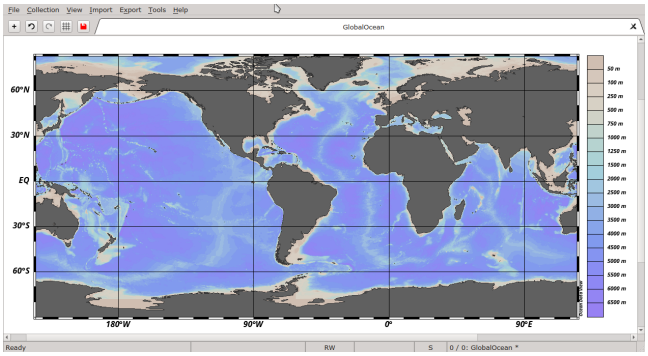

Figure 2: Screenshot of Ocean Data View software tool. Screenshot by author, courtesy of Schlitzer, Reiner (Ocean Data View).

GUI software can hamper flexibility by imposing ways of doing things, such as imposing specific algorithms or visualization types (P2, P20), or assuming particular experiment designs (P16). It can also hamper automation; for example, by not providing a way to redo repetitive tasks (P6), or necessitating manual data import and export (P4). In contrast, coding-based tools support flexibility by providing a wide range of functionality such as statistical analysis, and making figures and maps (P10), enabling reuse of old code for new projects (P2), and enabling functionality to be automated (P19). Code also has greater ability for interoperation of different systems. For example, it can be used to programmatically read formats meant for GUI tools, such as manufacturer-specific sensor data (P19), and different programming languages can call each other.

GUI software can lack power, meaning scope and scale of functionality. For example, statistics software SAS has limitations for making figures (P8), online databases may lack filter options when searching (P6), and GUI software may be inefficient or incompatible with handling large amounts of data (P14). At the same time, many GUI software tools are also highly complex, which impacts the extent of their usefulness. For example, P11 said that ArcGIS *"basically needs a developer"* to do advanced tasks, and P21 described a specific graphical data server system as *"very powerful"* only for *"hardcore"* users.

GUI software can hamper reproducibility, which is counterproductive towards open science (C2). For example, GUI software often involves many manual processing steps that are not recorded (P19). Proprietary GUI software also can require licensing, for which associated fees (P18) and the number of license seats (P20) can be limiting. In contrast, code can serve as documentation for reproducibility purposes. This because it is a *"full explanation of process"* (P18), in line with past work by Mislan et al. in ecology [31].

4.3 F1: Developing individualized views of practices in the field

Having missing or incomplete knowledge of what others in the field are doing is a form of fragmentation of views. Many of our participants had developed individualized views of practices in the field.

This means that they formed preconceptions about how others are using tools, were unfamiliar with tools used by others, did not know the range of capabilities of systems that they or others are using, or were missing ways to learn about what others are doing. Missing the broader context when learning tools informally (C3) may contribute to this. For example, P14 described his concerns with using online resources to learn about package managers: *"I don't really understand why there's so much emphasis on the fact that [using the Conda package manager]* is *the way to go. [...] I think that's because I'm lacking the experience and the skills to really understand all the benefits of it."*

Participants formed preconceptions about other scientists' tool use in the form of generalizations. For example, P3 claimed that *"everyone"* is using R (despite wide adoption of Python and other languages), and P21 claimed that computational notebooks are *"not widely used"* (despite their prevalence in oceanography). It also appeared that more senior researchers were overoptimistic about the extent to which junior researchers adopted code-based approaches, making claims like, *"[oceanographers] make their own stuff"* (P11), and oceanography students are *"highly motivated"* to learn coding (P18).

Our participants' different views of the NetCDF format, a common format for gridded data, serve as another salient example. P11 mostly works with text-based data files (e.g., CSV). He described NetCDF files as just *"for people to serve off spacetime data"*, seeing their use as a unidirectional process of extracting data for use in other formats. In stark contrast, P14 uses NetCDF files as his primary data format, including for output and intermediate data files. He does this because NetCDF files are more storage efficient for sparse, multi-dimensional gridded data: *"NetCDFs are usually very efficient for gridded projects. [...] I have an ocean model that simulates my ocean by dividing it in three dimensions plus time. And those three dimensions are in the grid, right?"* He also appreciates the built-in support for metadata and strong library support: *"I find that Python is so efficient in reading NetCDF. And so, even when I just write out a few columns of data, I still would prefer to create a NetCDF just for the metadata, because you can really have convenient metadata associated with your files. [...] I don't really see the benefits of saving in a CSV file when you can save a NetCDF."*

Beyond preconceptions, another form of individualized views was being unfamiliar with others' tools. Examples include researchers who were unaware of common tools or systems such as ODV (e.g., P11) or OBIS (e.g., P6), and a technical team member who did not know of an alternative data server system highly related to the one they are using (P21). P18 expressed how it is difficult to find out what systems other people are using. This relates to work from Subramanian et al. [49], which found that some data workers do not use computational notebooks because they are "not [...] aware of their existence". In the context of our participants working in bioinformatics, participants often used similar pipeline tools (e.g., QIIME2[7]). However, other participants we asked about workflow tools (e.g., Kepler [1]) were unaware of such tools or perceived them as too *"generic"* for their purposes (e.g., P11). Some participants were also unaware of the range of capabilities of the software they were using. For example, P6 did not realize that R could be used to do image processing. P20 described needing to take a long time to figure out visualization options offered by a tool: *"knowing what is available for visualizing the data [in a graphical tool...] isn't necessarily something that you know off the bat. That takes years to figure out, and sort of teach yourself what graphical visualizations these programs can offer".*

[7] A bioinformatics pipeline for analyzing genome sequencing data (https://qiime2.org/).

4.4 F2: Doing ad hoc software engineering in scientific computing

Code artifacts resulting from individual coding approaches lead to different, isolated practices centred around them. Marine scientists have started to use approaches associated with software engineering (C5), such as versioning files, and implementing and running code. However, because this is in service of scientific needs, such as experimenting in real-time and documenting the process, uses of these approaches are more individual, and may vary from those of software engineers. Participants felt they did software engineering-related tasks in ways that were unusual, or made *"weird"* (P17) use of tools. As with F1, the prevalence of informal learning approaches for computing skills (C3) may also contribute to this tendency.

One ad hoc software engineering approach is modifying and running code interactively in real-time. This includes using packages like Shiny to do interactive data exploration (P10), and using computational notebook cells to tweak visualizations (P14). Many of our participants preferred scripts over notebooks due to familiarity and the ability to do select-and-run execution of code. This means that individual lines would repeatedly be modified, highlighted, and re-run to achieve a desired outcome. For example, P8 described doing multiple statistical tests with the code-based features of SAS as: *"I can just select it and run it, [...] especially when I will do ANOVA for different variables. So I don't like to directly write down the coding, so [much] coding, just for different variables."* This finding contrasts with Subramanian et al.'s [49] study of data workers outside the context of the natural sciences, for which they argue that "scripts are not well suited to exploratory work".

Another ad hoc software engineering approach is using code-based artifacts in ways beyond just instructions to be executed. For example, code serves as documentation of the scientific process (C5), which supports reproducibility (C2). Participants also had individual approaches for using scripts to conceptually group parts of their work, such as grouping based on functionality (P17) or paper sections (P2). Older code was used as a memory aid. Some of our participants returned to their old code to have a starting point for similar new projects (P16) or to help colleagues working on similar problems (P15). Computational notebooks were used to enable embedding of mathematical equations and visualizations (P14), and extra documentation, as also discussed by Kery et al. [26] in the context of data scientists. For example, P16 adds Bash chunks to Jupyter Notebook as documentation of commands used in other parts of her pipeline: *"I also tend to use [R Markdown] to save the commands that I've run in any other part of the pipeline. [...] You can put them in Bash chunks as well. [...] I just use it to document what I've done."*

A third common ad hoc software engineering approach is doing versioning in an informal way. Participants had a variety of approaches for versioning, such as making manual copies of files (P4),

using cloud storage versioning features (P6), and using slide decks to keep records of code outputs (P14). Our findings concerning informal versioning approaches align with Kery et al.'s [25] study of exploratory programming in the context of data scientists.

In line with participants' ad hoc software engineering practices, participants on the 'science side' (C1) also did not view their own code work to be *"programming"* or *"coding"*. Our participants characterized their work like this because they perceived their code to be lacking in some way, such as being messy (P2), or using an easier programming language (P8).

4.5 F3: Resisting unfamiliar tools

Maintaining individual tool use practices by resisting uptake of unfamiliar tools entrenches these differences in common practice, acting as a form of fragmentation. Most participants expressed some level of anxiety or resistance towards learning or using unfamiliar tools. Our findings concerning reasons for resisting unfamiliar tools relate to Liu et al.'s [28] "barriers" to exploring alternatives in the context of data workers in industry; however, we also highlight that alternatives are often not explored and describe how unfamiliar alternative tools are avoided.

In a coding context, resistance is created because programming and programming tools are seen as intimidating, for reasons such as lack of confidence (P6) or the perception that they are too *"advanced"* (P4). Code-based approaches may necessitate writing code in ways that deviate from scientific mental models. For example, P15 said, *"the way our methods work, it's on an observation-by-observation basis, not like a matrix basis"*, which was the format expected by the code. She found this difference between her mental model and the data representation in code to be *"annoying and frustrating"*. Further, a combination of different pressures, such as the expectation for students to figure things out themselves (P12), dogmatic learning resources (P14), and punitive courses (P16), may lead marine scientists to become discouraged when learning coding or other software engineering concepts. For example, P4 described using a trial-and-error approach to learning R for data processing:

> *"I tried myself. I kept trying and trying, because I have seen that kind of data, how R files open, how to run them. So I tried it, [but] it's not happening. I was like, 'Oh my god!' Yeah, then only I decide, 'Oh, it's not worth it to spend my time on something I cannot figure out.' Maybe let it be there, focus on something [else], take some of [a colleague's] time and come to him and get his help. Yeah, for him, it was easy."*

As another example, P16 described how an introductory programming course during her PhD discouraged creative problem solving:

> *"When we were in the class, [the TA] seemed to want people to just do things in one specific way. Whereas, usually in programming, there's a lot of different ways you can achieve a single thing. And if you're learning, if you can make it work in any of those ways, it doesn't matter if it's the most efficient or not. You can work on making it be the quickest way possible later, once you've got the basics down."*

She continued to describe how this kind of treatment came off as punitive, even leading some students to cry: *"He had a bit of a tendency to come around to me like, 'What are you doing! Why have you got that?' And, you're like, 'I don't know. I thought I was working on it.' "* This intimidation factor is highly interconnected with the other concepts, in that lacking formal computing training (C3), increased pressure to use code as a means of advancing open science (C2), and the feeling of not being a programmer (F2) may all contribute. In a GUI context, interfaces such as ArcGIS can also be seen as intimidating (P3) due to their high level of complexity (C5).

Participants on the 'science side' (C1) tended not to prioritize learning unfamiliar tools, instead resisting them by sticking with familiar, *"good enough"* tools. For example, P17 described how he has become used to coding in C++ with RStudio, despite its inconveniences: *"[If] I screwed up the code within the C++, like if I screwed up my indexing somewhere, it makes [RStudio] crash and then [...] I have to terminate it and start it over, and that's aggravating. So I would say it's good enough, but there's definitely better ways, but it's the point I've gotten used to it."* Despite this, participants were curious about and interested in learning new tools that were *"cool"* (P3) or *"popular"* (P15). Participants would not make time to learn new tools for reasons such as struggling to keep up with changes to tools (P2). Participants would also offload or delegate tool-related tasks to others such as technical staff (P11) or collaborators (P22), like how P4 ended up delegating coding work to her colleague. Participants sometimes fell back on familiar tools after unsuccessfully trying to use an unfamiliar tool; for example, using Excel after being unable to figure out how to export figures using R (P6). In contrast, marine scientists more on the 'tooling side' (C1) may experience increased motivation to learn new programming tools. For example, P10 was keen to code and learn new programming tools as a way of *"procrastinating"* from other work.

4.6 F4: Doing things from scratch

Duplicated efforts when creating tools and systems form new isolated instances of associated practices. Our participants generally had a *"bias of wanting to do things from scratch"* (P11), as opposed to reusing existing tools from others or automating tasks. There are cases where existing tools do not exist to achieve the required functionality (P10), necessitating creation of new tools. However, there are also many cases of *"reinventing the wheel"* (P14), despite aspirations to avoid duplication of work. This may be exacerbated when marine scientists are unaware of what tools others are using (F1). These findings expand on several past works that identified a subset of reasons for why scientists may avoid using others' tools, namely the need for "precisely repeatable results" [10] and avoiding the need to "force" work into "the interface supported by the framework" [3, 16].

There may be external social or infrastructural limitations. For example, P23's team had acquired satellite data that they wanted to make publicly available. They asked a local NGO (non-governmental organization) that was hosting similar data if they would be interested in including this satellite data. The NGO needed to refuse because their infrastructure could not support so much data, which resulted in P23's team needing to make their own online database. As another example, P19 highlighted that large-scale system development efforts may lack sufficient funding. Similarly, it may not

be possible to use existing tools because of issues concerning open science (C2). For example, P21 described how his team was unsuccessful at requesting access to a sister institution's glider metadata software:

> *"They have a fantastic metadata software, because they've been doing this for a long time. [...] And so, we asked them many times if they would share it with us. [...] I feel very strongly that, if it's taxpayers dollars, we don't charge for things, because they already paid for it. They did not feel that way. So we ended up writing our own package."*

There may be a desire to do things from scratch to develop trust that a system has been correctly implemented; for example, P19's implementation of signal conversion code described earlier (C4). P19 also makes use of packages where he *"trust[s] the people doing it"* to avoid needing to *"reinvent the wheel"*. Similarly, there is educational value in doing things manually. For example, P17 described doing data preparation manually rather than automating it as a way to become familiar with the data: *"I think there's inherent value [in manually doing data formatting], because I have to look at [the data] a lot. Like, it's pre-visualization to me, [...] it's very useful for me to spend time playing with it just to get it in a shape that makes sense. But to get it in that shape, I have to look at it."* As another example, P7 described doing manual calculations in Excel to learn how lab protocols work, as opposed to automating these calculations. This serves as a form of informal learning (C3). This practice was also observed by Huang et al. [18] in the context of bioinformatics, in which they found that more junior researchers would create scripts as practice.

Existing packages and tools can also be hard to learn, for reasons such as incomplete or out-of-date documentation (P10, P15), low user-friendliness (P1), or lack of flexibility or interoperability (P6, P14). P17 explained this sentiment, elaborating on how it is challenging to find existing tools for niche contexts:

> *"Honestly, it's all about making it easy on myself. And I'm like, 'If I don't see a lot of people needing to do this, then there might be a package, but it might be a hassle to find, might not be on CRAN'. And like, am I really going to be saving time by finding it? Because then I'll [also] have to learn their own specific way of doing things because everyone codes slightly differently and expects differently. And if I just do it myself, it's probably faster."*

Anxiety towards learning new tools (F3), and feeling *"at the mercy of whoever designed [a tool]"* (P6) may also contribute to the tendency to do things from scratch rather than try to learn existing tools. Experienced researchers may create things from scratch because they feel they can do better than what already exists; for example, by providing more flexibility in what a tool can do (P11) or a better choice of algorithms (e.g., P18, using an algorithm that reduces rounding error).

4.7 F5: Tools having limited reach

The limited reach of tools, caused by factors such as the preconception that tool use will be niche and the "cultural gulf" between technical teams and researchers, propagates isolated pockets of practices, acting as a form of fragmentation. Tools made by or for marine scientists may reach a narrower audience than would theoretically find them useful.

There appears to be a somewhat self-fulfilling cycle of expecting tool use to be niche, resulting in the creation of especially niche tools. This also potentially leads to duplication of effort (F4). For example, some participants held the preconception that the number of users of their packages would be small because it is in a niche context (P17, P23). Packages may not be added to trusted online repositories such as CRAN because they are perceived as too niche (P17), preventing others who rely on these repositories, such as P18, from discovering them. Tool creators may treat smaller contributions as categorically different from larger software. For example, P21 said, *"[Within] my group, we've only built completely from scratch one bit of software"*, referring to full systems with maintenance and training. He elaborated, *"We've written a lot of data processing scripts, but I wouldn't call that software per se, right? Lots of times it's one-offs, you know, you're processing from a particular instrument, [and so on]."* These smaller contributions may not be shared with the community if they do not meet the bar of being *"software"*, despite their potential benefits. Tools may also hard-code assumptions specific to particular subareas. For example, P16 described a series of tools all made by one group that are designed to interconnect. However, one of these tools hardcoded the assumption that human samples will be provided, even though the algorithms could theoretically work for a broader range of samples like ocean environment samples. This meant that P16 needed to use a mix of tools from different sources.

There is also a *"cultural gulf"* (P11) between computer scientists (e.g., technical teams) and science researchers, in part due to differing priorities (C1). From a standpoint of software development practices, this has been explored in depth by past work (e.g., [3, 10, 24, 43]). In the context of our participants, this includes a feeling that the generic tools made by *"higher-ups"* are *"out of touch with what [scientists] need"* (P18). For example, P18 described a time when the university's library created a repository for storing scientific data, but it could not even hold one terabyte of data, ultimately rendering it unusable for the labs. This finding is mirrored in ecology work by Aubin et al. [2], which argued that "Top-down initiatives [to encourage data sharing and integration] have found limited success in ecology because they are rarely customized to take into account the specific sociocultural challenges around sharing ecological data".

Misaligned goals between technical and research teams may lead to development of graphical tools with problematic designs (C5). For example, some of our participants mentioned how tools have complex features that may go unused, such as a *"fake cruise"* transect search feature offered by a particular online database, which P2 did not have a use for. P11 also felt that *"lots of software is never taken up by the community"* because it lacked *"an integrated development effort"* across computer scientists and science researchers. Our two participants associated with technical teams described a more hands-off approach to tool interface design. For example, P23 described her team's approach to graphical design as *"mostly within the team"*, and P21 said that his team wants to stay *"away from having public-facing websites"*, in part due to complexity around making user interface design decisions that address *"what's important"* for

users. One consideration is that technical teams who implement new tools focus on *"client-driven"* (P21) work, which may bias the focus of tool implementation towards systems with narrower scopes. For example, even though P21's team is *"the"* technical group for the oceanography department, many of our participants' groups associated with oceanography were not using their services for their main work, instead doing technical work themselves, or relying on their own in-lab technicians or neighbouring labs.

P18 felt that researchers are increasingly working on *"fiddly little things"* to increase publication counts and satisfy *"bean counters"*, rather than doing work that potentially provides more value to the community. P12 elaborated on how this is affected by funding: *"[The] academic funding model is really centred on graduate students, largely, and to a lesser extent, postdocs. And sure, you can have a graduate student who contributes code to a project, but they have a research project to do. They need a thesis, right?"* P18 emphasized the desire for scientists to receive credit for community work such as making software packages. This sentiment is reflected in past work suggesting that recognition for software contributions is lacking in the sciences [12, 43, 45, 47, 51], with exceptions such as in bioinformatics [18].

5 Discussion and User Interface Design Priorities

We discuss our participants' use of code in relation to past work, and propose design priorities for future user interfaces for marine scientists.

5.1 Reflecting on Marine Scientists' Use of Code in Relation to Past Work

The notions of unconventional coding practices in science and software work as secondary in science are recurring ideas in our fragmentation concepts and in past work. We briefly reflect on our analysis and past work in reference to these ideas.

In line with how our participants felt they did software engineering-related tasks in unusual ways (F2), past work has described scientific development and analysis as "iterative" [43], "non-linear" [23], and lacking "formal [...] methodology" [16]. Similarly, our marine scientist participants also generally did not view their own code work as "programming" or "coding" (F2). This similar to past work by Segal [43], which argues that scientists "do not think of themselves as software developers", and work by Kelly [24], which found that "scientists do not see themselves as software developers [...] but as a professional who uses software as part of their exploration".

Past work has often framed this unconventionality as problematic. For example, Lowndes et al. [29] argue that "[w]ithout training, scientists tend to develop their own bespoke workarounds to keep pace, but with this comes wasted time struggling to create their own conventions for managing, wrangling and versioning data". In contrast, our analysis highlights that this time is not inherently "wasted", because there is potential to learn about the algorithms or data (e.g., P17's comment about *"pre-visualization"* in F4), and ad hoc approaches (F2) can support interactive data exploration and avoid struggles (e.g., P15's concern about data representations in F3) by working with data according to a familiar mental model. Kuksenok et al. [27] also share our observation that "Lack of reuse is problematized in software engineering accounts of scientific programming practice, as well as in internal critique that adapts software engineering rhetoric", and they push back against "lack of interest, willingness, or awareness" as reasons for not reusing code. However, based on their specific participant pool of oceanographers with "an attitude of willingness toward uptake of new technologies", they argue that "motivated scientists enthusiastic about best practices choose to adapt them to safeguard the scientific usefulness of the code they work on". In contrast, based on our broader scope of marine scientists that had mixed interest in software engineering concepts, we argue that lack of interest can be a reason for not reusing code. Many of our participants were not "motivated" or "enthusiastic" about "best practices", because they had no reason to be enthusiastic about software engineering-related concepts (C1) unless they offered alluring functionality (F3). Lack of willingness is also a reason because our participants often did not want to spend time learning software (C1), and awareness is also a reason because of informal training (C3) and preconceptions about software use (F1). This is in line with past work characterizing software-related work like coding as "secondary" [17, 43] compared to the main scientific work. Marine scientists may benefit from the development of tools that differ from the practices from conventional software engineering, and they generally do not want to spend their time with software engineering work when they could be working on scientific questions.

5.2 User Interface Design Priorities

Based on our analysis, we propose three design priorities for user interfaces that integrate within current fragmented practices, and help work towards mitigating some of the negative aspects of fragmentation. As a whole, we do not see fragmentation as something to be solved or removed. It is inherent to the unique and valuable multidisciplinary nature of the field, with both positive and negative aspects. For example, concerning F2, marine scientists benefit from a level of flexibility with their tool use practices when adapting software engineering concepts to scientific contexts, yet the individuality of tool use practices may hamper collaboration. This has implications for the design of tools that integrate with fragmentation.

5.2.1 Powerful and simple graphical user interfaces. We should not assume that all marine scientists will shift towards coding-based tools, because the field is multidisciplinary, and because of differences in interest towards computational concepts (C1), and differences in training (C3). At the same time, some negative aspects of fragmentation are intensified as the community tries to establish practices around tools that support concerns like flexibility, power, and reproducibility (C4, C5). For example, the expectation to learn coding can lead to incompatible tool use practices (F2), and resistance to unfamiliar coding-based tools (F3). Because software engineering concepts can be seen as relevant for addressing issues of open science and reproducibility in the natural sciences (e.g. [29]), the burden of implementing these software engineering practices is being put on the scientists, and design solutions are often framed in software engineering terms. Instead, as HCI researchers and designers, we should question this trend of depending on software

engineering concepts for scientific tools. We suggest that creating GUI-based tools to address marine scientists' concerns can help to mitigate the downsides of fragmentation brought on by the push towards coding-based approaches. This suggestion may lead to more fragmentation in that some people are using GUI-based tools whereas others are using code, but if interoperability is designed for, as we describe below, then this serves a beneficial form of fragmentation that enables people can focus on the types of work they are most interested in.

To better suit marine scientists' needs, graphical tools should be designed to address concerns like reproducibility, trust, and power (C4, C5). For reproducibility, graphical interfaces should support the ability to save interaction steps as a series of instructions that can be imported and replayed by other users of the tool. Such solutions could draw from the literature on version history systems for non-code artifacts (e.g., [14, 48]). Interfaces should ensure that steps can be explored and annotated directly, enabling these steps to serve as documentation of process. For trust, graphical interfaces should provide more transparent representations of internal processes as needed, such as which algorithms are used and what assumptions are made when processing data. Interfaces could also be designed to increase the ease of containerizing research software, in the vein of systems like GUIdock [19]. For power, interfaces for marine scientists should support technical decisions and instill a feeling of competence. To avoid intimidation (F3) and support marine scientists with learning tools rather than duplicating efforts (F4), tools should not present themselves as excessively complex. A marine scientist may have a scientific goal in mind, but not have the ability to translate this into computational terms. Negative experiences, like P15's frustration about differences between mental models and computer data representations, and P22's impression that she needs to offload technical work to collaborators, should not stand in the way of scientific work.

Interoperation and integration within existing tools should be prioritized, rather than creating new standalone tools that would require marine scientists to make significant changes to their processes. Creating tools that interoperate with existing tools may make them easier for marine scientists to adopt, reducing the need for marine scientists to contribute to fragmentation by making their own tools from scratch to suit their needs (F4). There may also be reluctance to adopt new tools when existing tools are *"good enough"* (P17, F3). Creating new solutions that integrate into existing tools may increase the chance of adoption. We discuss this idea further in the third priority.

To support open science and facilitate collaborative work between users of different tools, graphical tools should also be designed to interoperate with code-based tools. Ideally, this means using widely-used data formats as the main data format (e.g., CSV, NetCDF). Where this is not possible, packages should be developed alongside graphical applications that allow their formats to created, read, and written from code. Also, code and GUI-based ways of working are not mutually exclusive. Our analysis provides examples of how code is often used within other GUI contexts, such as within computational notebooks (P14) or statistics tools (P8). We did not encounter instances where code was used to extend GUI functionality using macros, plugin systems, or similar. This is unsurprising because marine scientists who find such functionality useful may prefer to just switch to code-based tools for the added flexibility. However, allowing code extensions in GUI-based tools could be beneficial to support experienced coders when collaborating with scientists who prefer GUIs. If tool users are expected to be familiar with code, a more flexible interface (e.g., CLI-based tool for interfacing with the GUIs) would likely be preferred.

5.2.2 Effecting an "an integrated development effort". It is precisely because tools have a limited reach (F5) that they need to have an integrated development effort to help expand their reach. This means that system designers and scientists need to work together (e.g., via participatory design with scientists [7]) to help ensure that new tools meet scientists' needs, and that excessive functionality or complexity is avoided. Practical usefulness and research novelty from a systems point-of-view may be competing objectives. Based on our F5 findings, some recommendations for system design include: having more frequent iteration cycles that engage with the users (e.g., concerning P23 iterating *"mostly within the team"*), considering the assumptions being encoded in tools and whether relaxing them could serve a broader audience (e.g., for P17's tool expecting human samples), determining user needs before (and during) development (e.g., for P18's library anecdote), and assessing which features are used in practice to find avenues for reducing the complexity of interfaces (e.g., for *"fake cruise"* search feature mentioned by P2).

When designing tools, it is important to not assume that what works for other data workers will also work for marine scientists. Our findings highlighted that past work that lumps people from heterogeneous fields under the umbrellas of "data science" or "data work" can obscure nuanced and unique aspects of specific fields like marine science. For example, our participants would often themselves take data off equipment in non-standard proprietary formats, or take manual readings from equipment and measurements in the field. These practices may necessitate unique system designs for interfacing with equipment, transparently converting raw measurements, and converting data representations. As another example, our participants often collaborated with colleagues with varying levels of coding ability, which could be supported by designs that strengthen communication of technical concepts between collaborators.

Following from the earlier interoperability point, interactive system designers should not assume their systems are so niche that they can ignore the potential to make use of or integrate into existing systems. Because duplication of effort when creating new systems may be unavoidable due to external constraints (F4), it is also valuable to consider how interactive systems could be designed in the form of building blocks (e.g., common visualizations, data processing functionality, communication functionality) that can be reused within different marine science tools and contexts.

5.2.3 Supporting discovery and connection. To address concerns around individualized views (F1) and duplication of effort when creating tools (F4), systems should be designed to increase awareness of alternative tools and unused capabilities of tools in use.

One approach is to develop systems that integrate into existing tools or help interfaces, and provide insight into what tools one's colleagues are using. As suggested by P18, it would be beneficial to know what tools one's close colleagues trust, and new interface

designs to support this could draw from trust-based recommender systems (e.g. [54]). Another approach could be to design interfaces to teach or assist users with combining disparate packages or tools.

Scientific workflow systems, including those with graphical interfaces (e.g., [1]), have been proposed in the literature in part to enable sharing of scientific processes. Although our participants working in bioinformatics were familiar with the concept of scientific workflow systems (in the form of bioinformatics pipelines), such systems were mostly unfamiliar or perceived as too generic to our other participants. This suggests that enabling discovery of research artifacts at a more granular level than workflows could provide benefits to the community. Rather than sharing whole workflows, scientists could share smaller pieces of tasks, such as individual scripts, lines of code, cells from computational notebooks, or conceptual approaches. New system designs could draw from systems like "myExperiment" [13], a web service that supports social sharing of scientific workflows. For example, systems could allow scientists to upload and search for pieces of scientific tasks. A design challenge would be enabling efficient discovery of these pieces in a way that integrates into marine scientists' existing processes and tools. Further, as discussed in past work [5], scientists may be reluctant to take up new technology quickly or in a way that does not integrate with their existing tools or processes. Thinking about system design at the level of "workflows" may lead to designs that expect a large degree of change from users, hampering adoption.

6 Conclusion and Future Work

Based on an interview study of marine scientists and research supports, we constructed five conceptual categories that characterize a broader concept we call *fragmentation*, which refers to isolated pockets of views and practices concerning software tools.

In line with our constructivist approach, we see our findings as situated to the context of the university of study and the domain of marine science. Investigating the practices of specific domains is important, because it reveals the nuanced differences from results suggested from more generic terms representing heterogeneous groups (e.g., "data scientist"). We saw these differences in our findings concerning tool documentation production, returning to old code, use of modular scripts, interest in but also resistance of unfamiliar tools, and reasons for doing things from scratch. However, for our concepts that have aspects relating to past work, we found considerable levels of agreement (e.g., the iterative nature of scientific analysis, that scientists do not see themselves as software developers, that scientists may avoid others' tools). This suggests that our fragmentation concept may generalize to broader scientific domains. A potential exception is disciplines in which computer science or software engineering training is more central, such as bioinformatics.

As a whole, our concept of fragmentation characterizes how marine scientists are using software in practice, which includes positive aspects, such as how marine scientists successfully make use of software tools and have learned to adapt them for their needs, as well as aspects that could be improved through future work, such as concerns around awareness of what tools others are using, and duplication of effort when creating tools for others to use. Future work should not assume that all aspects of fragmentation are inherently in need of change. We proposed three priorities for user interface design: (1) exploring how graphical interfaces can be designed to address the software challenges that often result in turning to code-based solutions, (2) collaborating with scientists when creating tools, and (3) supporting scientists discover features within their tools and discover what other tools are being used. There is strong potential for HCI to contribute to the design of interactive systems that support marine scientists at a broad scale.

Acknowledgments

This work made possible by NSERC Discovery Grant 2024-03827. We would also like to extend a big thank you to our participants for taking time out of their busy schedules to meet with us.

References

[1] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. 2004. Kepler: an extensible system for design and execution of scientific workflows. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.* 423–424. doi:10.1109/SSDM.2004.1311241

[2] Isabelle Aubin, Françoise Cardou, Laura Boisvert-Marsh, Eric Garnier, Manuella Strukelj, and Alison D. Munson. 2020. Managing data locally to answer questions globally: The role of collaborative science in ecology. *Journal of Vegetation Science* 31, 3 (2020), 509–517. doi:10.1111/jvs.12864

[3] Victor R. Basili, Jeffrey C. Carver, Daniela Cruzes, Lorin M. Hochstein, Jeffrey K. Hollingsworth, Forrest Shull, and Marvin V. Zelkowitz. 2008. Understanding the High-Performance-Computing Community: A Software Engineer's Perspective. *IEEE Software* 25, 4 (2008), 29–36. doi:10.1109/MS.2008.103

[4] Matt Burton and Steven J. Jackson. 2012. Constancy and Change in Scientific Collaboration: Coherence and Integrity in Long-Term Ecological Data Production. In *2012 45th Hawaii International Conference on System Sciences.* 353–362. doi:10.1109/HICSS.2012.178

[5] Jeffrey C. Carver, Richard P. Kendall, Susan E. Squires, and Douglass E. Post. 2007. Software Development Environments for Scientific and Engineering Software: A Series of Case Studies. In *29th International Conference on Software Engineering (ICSE'07).* 550–559. doi:10.1109/ICSE.2007.77

[6] Kathy Charmaz. 2014. *Constructing Grounded Theory, 2nd Edition.* SAGE Publications Ltd.

[7] Adam Coscia, Haley M. Sapers, Noah Deutsch, Malika Khurana, John S. Magyar, Sergio A. Parra, Daniel R. Utter, Rebecca L. Wipfler, David W. Caress, Eric J. Martin, Jennifer B. Paduan, Maggie Hendrie, Santiago Lombeyda, Hillary Mushkin, Alex Endert, Scott Davidoff, and Victoria J. Orphan. 2024. DeepSee: Multidimensional Visualizations of Seabed Ecosystems. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24).* Association for Computing Machinery, New York, NY, USA, Article 210, 16 pages. doi:10.1145/3613904.3642001

[8] Anamaria Crisan, Brittany Fiore-Gartland, and Melanie Tory. 2021. Passing the Data Baton: A Retrospective Analysis on Data Science Work and Workers. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 1860–1870. doi:10.1109/TVCG.2020.3030340

[9] Joel Cutcher-Gershenfeld, Karen S. Baker, Nicholas Berente, Dorothy R. Carter, Leslie A. DeChurch, Courtney C. Flint, Gabriel Gershenfeld, Michael Haberman, John Leslie King, Christine Kirkpatrick, Eric Knight, Barbara Lawrence, Spenser Lewis, W. Christopher Lenhardt, Pablo Lopez, Matthew S. Mayernik, Charles McElroy, Barbara Mittleman, Victor Nichol, Mark Nolan, Namchul Shin, Cheryl A. Thompson, Susan Winter, and Ilya Zaslavsky. 2016. Build It, But Will They Come? A Geoscience Cyberinfrastructure Baseline Analysis. *Data Science Journal* (July 2016). doi:10.5334/dsj-2016-008

[10] Steve M. Easterbrook and Timothy C. Johns. 2009. Engineering the Software for Understanding Climate Change. *Computing in Science & Engineering* 11, 6 (2009), 65–74. doi:10.1109/MCSE.2009.193

[11] Melanie Feinberg, Will Sutherland, Sarah Beth Nelson, Mohammad Hossein Jarrahi, and Arcot Rajasekar. 2020. The New Reality of Reproducibility: The Role of Data Work in Scientific Research. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW1, Article 35 (May 2020), 22 pages. doi:10.1145/3392840

[12] Carole Goble. 2014. Better Software, Better Research. *IEEE Internet Computing* 18, 5 (2014), 4–8. doi:10.1109/MIC.2014.88

[13] Carole Anne Goble and David Charles De Roure. 2007. myExperiment: social networking for workflow-using e-scientists. In *Proceedings of the 2nd Workshop on Workflows in Support of Large-Scale Science* (Monterey, California, USA) *(WORKS '07).* Association for Computing Machinery, New York, NY, USA, 1–2. doi:10.

1145/1273360.1273361
[14] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology* (New York, New York, USA) *(UIST '10)*. Association for Computing Machinery, New York, NY, USA, 143–152. doi:10.1145/1866029.1866054
[15] Jo Erskine Hannay, Carolyn MacLeod, Janice Singer, Hans Petter Langtangen, Dietmar Pfahl, and Greg Wilson. 2009. How do scientists develop and use scientific software?. In *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*. 1–8. doi:10.1109/SECSE.2009.5069155
[16] Dustin Heaton and Jeffrey C. Carver. 2015. Claims about the use of software engineering practices in science: A systematic literature review. *Information and Software Technology* 67 (2015), 207–219. doi:10.1016/j.infsof.2015.07.011
[17] James Howison and James D. Herbsleb. 2011. Scientific software production: incentives and collaboration. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work* (Hangzhou, China) *(CSCW '11)*. Association for Computing Machinery, New York, NY, USA, 513–522. doi:10.1145/1958824.1958904
[18] Xing Huang, Xianghua Ding, Charlotte P. Lee, Tun Lu, and Ning Gu. 2013. Meanings and boundaries of scientific software sharing. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work* (San Antonio, Texas, USA) *(CSCW '13)*. Association for Computing Machinery, New York, NY, USA, 423–434. doi:10.1145/2441776.2441825
[19] Ling-Hong Hung, Daniel Kristiyanto, Sung Bong Lee, and Ka Yee Yeung. 2016. GUIdock: Using Docker Containers with a Common Graphics User Interface to Address the Reproducibility of Research. *PLOS ONE* 11, 4 (June 2016), 1–14. doi:10.1371/journal.pone.0152686
[20] Steven J. Jackson and Sarah Barbrow. 2013. Infrastructure and vocation: field, calling and computation in ecology. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2873–2882. doi:10.1145/2470654.2481397
[21] Steven J. Jackson and Sarah Barbrow. 2015. Standards and/as Innovation: Protocols, Creativity, and Interactive Systems Development in Ecology. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 1769–1778. doi:10.1145/2702123.2702564
[22] Ju Yeon Jung, Tom Steinberger, John L. King, and Mark S. Ackerman. 2022. How Domain Experts Work with Data: Situating Data Science in the Practices and Settings of Craftwork. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW1, Article 58 (April 2022), 29 pages. doi:10.1145/3512905
[23] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2012. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2917–2926. doi:10.1109/TVCG.2012.219
[24] Diane Kelly. 2015. Scientific software development viewed as knowledge acquisition: Towards understanding the development of risk-averse scientific software. *Journal of Systems and Software* 109 (2015), 50–61. doi:10.1016/j.jss.2015.07.027
[25] Mary Beth Kery, Amber Horvath, and Brad Myers. 2017. Variolite: Supporting Exploratory Programming by Data Scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1265–1276. doi:10.1145/3025453.3025626
[26] Mary Beth Kery, Marissa Radensky, Mahima Arya, Bonnie E. John, and Brad A. Myers. 2018. The Story in the Notebook: Exploratory Data Science using a Literate Programming Tool. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–11. doi:10.1145/3173574.3173748
[27] Kateryna Kuksenok, Cecilia Aragon, James Fogarty, Charlotte P. Lee, and Gina Neff. 2017. Deliberate Individual Change Framework for Understanding Programming Practices in four Oceanography Groups. *Computer Supported Cooperative Work (CSCW)* 26, 4 (Dec. 2017), 663–691. doi:10.1007/s10606-017-9285-x
[28] Jiali Liu, Nadia Boukhelifa, and James R. Eagan. 2020. Understanding the Role of Alternatives in Data Analysis Practices. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 66–76. doi:10.1109/TVCG.2019.2934593
[29] Julia S. Stewart Lowndes, Benjamin D. Best, Courtney Scarborough, Jamie C. Afflerbach, Melanie R. Frazier, Casey C. O'Hara, Ning Jiang, and Benjamin S. Halpern. 2017. Our path to better science in less time using open data science tools. *Nature Ecology & Evolution* 1, 6, Article 160 (May 2017). doi:10.1038/s41559-017-0160
[30] Yaoli Mao, Dakuo Wang, Michael Muller, Kush R. Varshney, Ioana Baldini, Casey Dugan, and Aleksandra Mojsilović. 2019. How Data ScientistsWork Together With Domain Experts in Scientific Collaborations: To Find The Right Answer Or To Ask The Right Question? *Proc. ACM Hum.-Comput. Interact.* 3, GROUP, Article 237 (Dec. 2019), 23 pages. doi:10.1145/3361118
[31] K.A.S. Mislan, Jeffrey M. Heer, and Ethan P. White. 2016. Elevating The Status of Code in Ecology. *Trends in Ecology & Evolution* 31 (2016), 4–7. Issue 1. doi:10.1016/j.tree.2015.11.006
[32] Michael Muller, Ingrid Lange, Dakuo Wang, David Piorkowski, Jason Tsay, Q. Vera Liao, Casey Dugan, and Thomas Erickson. 2019. How Data Science Workers Work with Data: Discovery, Capture, Curation, Design, Creation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3290605.3300356
[33] Andrew B. Neang, Will Sutherland, Michael W. Beach, and Charlotte P. Lee. 2021. Data Integration as Coordination: The Articulation of Data Work in an Ocean Science Collaboration. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW3, Article 256 (Jan. 2021), 25 pages. doi:10.1145/3432955
[34] Andrew B. Neang, Will Sutherland, David Ribes, and Charlotte P. Lee. 2023. Organizing Oceanographic Infrastructure: The Work of Making a Software Pipeline Repurposable. *Proc. ACM Hum.-Comput. Interact.* 7, CSCW1, Article 79 (April 2023), 18 pages. doi:10.1145/3579512
[35] Luke Nguyen-Hoan, Shayne Flint, and Ramesh Sankaranarayana. 2010. A survey of scientific software development. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (Bolzano-Bozen, Italy) *(ESEM '10)*. Association for Computing Machinery, New York, NY, USA, Article 12, 10 pages. doi:10.1145/1852786.1852802
[36] Gary M. Olson and Judith S. Olson. 2000. Distance Matters. *Human-Computer Interaction* 15, 2-3 (2000), 139–178. doi:10.1207/S15327051HCI1523_4
[37] Samir Passi and Steven J. Jackson. 2018. Trust in Data Science: Collaboration, Translation, and Accountability in Corporate Data Science Projects. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW, Article 136 (Nov. 2018), 28 pages. doi:10.1145/3274405
[38] Asbjørn Malte Pedersen and Claus Bossen. 2024. Data Work Between the Local and the Global: An Ethnography of a Healthcare Business Intelligence Unit. *Proc. ACM Hum.-Comput. Interact.* 8, CSCW1, Article 40 (April 2024), 28 pages. doi:10.1145/3637317
[39] Foster Provost and Tom Fawcett. 2013. Data Science and its Relationship to Big Data and Data-Driven Decision Making. *Big Data* 1, 1 (2013), 51–59. doi:10.1089/big.2013.1508
[40] Janine Rüegg, Corinna Gries, Ben Bond-Lamberty, Gabriel J. Bowen, Benjamin S. Felzer, Nancy E. McIntyre, Patricia A. Soranno, Kristin L. Vanderbilt, and Kathleen C. Weathers. 2014. Completing the data life cycle: using information management in macrosystems ecology research. *Frontiers in Ecology and the Environment* 12, 1 (2014), 24–30. doi:10.1890/120375
[41] Thomas Ruth, Stefan Audersch, Linda Huber, Uwe Freiherr von Lukas, and Jakob Zabel. 2015. Using direct-touch interaction for the visual exploration of profiling sensor data. In *OCEANS 2015 - Genova*. 1–8. doi:10.1109/OCEANS-Genova.2015.7271583
[42] Reiner Schlitzer. 2023. Ocean Data View. https://odv.awi.de.
[43] Judith Segal. 2007. Some Problems of Professional End User Developers. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007)*. 111–118. doi:10.1109/VLHCC.2007.17
[44] A. Specht, S. Guru, L. Houghton, L. Keniger, P. Driver, E.G. Ritchie, K. Lai, and A. Treloar. 2015. Data Management Challenges in Analysis and Synthesis in the Ecosystem Sciences. *Science of The Total Environment* 534 (Nov. 2015), 144–158. doi:10.1016/j.scitotenv.2015.03.092
[45] Susan Leigh Star and Karen Ruhleder. 2016. Steps toward an Ecology of Infrastructure: Design and Access for Large Information Spaces. In *Boundary Objects and Beyond*, Geoffrey C. Bowker, Stefan Timmermans, Adele E. Clarke, and Ellen Balka (Eds.). The MIT Press, 377–416. doi:10.7551/mitpress/10113.003.0026
[46] Stephanie B. Steinhardt. 2016. Breaking Down While Building Up: Design and Decline in Emerging Infrastructures. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 2198–2208. doi:10.1145/2858036.2858420
[47] Stephanie B. Steinhardt and Steven J. Jackson. 2015. Anticipation Work: Cultivating Vision in Collective Practice. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW '15)*. Association for Computing Machinery, New York, NY, USA, 443–453. doi:10.1145/2675133.2675298
[48] Sarah Sterman, Molly Jane Nicholas, and Eric Paulos. 2022. Towards Creative Version Control. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW2, Article 336 (Nov. 2022), 25 pages. doi:10.1145/3555756
[49] Krishna Subramanian, Nur Hamdan, and Jan Borchers. 2020. Casual Notebooks and Rigid Scripts: Understanding Data Science Programming. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–5. doi:10.1109/VL/HCC50065.2020.9127207
[50] Carol Tenopir, Suzie Allard, Kimberly Douglass, Arsev Umur Aydinoglu, Lei Wu, Eleanor Read, Maribeth Manoff, and Mike Frame. 2011. Data Sharing by Scientists: Practices and Perceptions. *PLOS ONE* 6, 6 (June 2011), 1–21. doi:10.1371/journal.pone.0021101
[51] Erik H. Trainer, Chalalai Chaihirunkarn, Arun Kalyanasundaram, and James D. Herbsleb. 2015. From Personal Tool to Community Resource: What's the Extra Work and Who Will Do It?. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (Vancouver, BC, Canada) *(CSCW*

'15). Association for Computing Machinery, New York, NY, USA, 417–430. doi:10.1145/2675133.2675172

[52] Jullian C. Wallis, Christine L. Borgman, Matthew S. Mayernik, and Alberto Pepe. 2008. Moving Archival Practices Upstream: An Exploration of the Life Cycle of Ecological Sensing Data in Collaborative Field Research. *International Journal of Digital Curation* 3, 1 (Dec. 2008), 114–126. doi:10.2218/ijdc.v3i1.46

[53] Amy X. Zhang, Michael Muller, and Dakuo Wang. 2020. How do Data Science Workers Collaborate? Roles, Workflows, and Tools. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW1, Article 22 (May 2020), 23 pages. doi:10.1145/3392826

[54] Jia Zhang, Petr Votava, Tsengdar J. Lee, Shrikant Adhikarla, Isaraporn Cherry Kulkumjon, Matthew Schlau, Divya Natesan, and Ramakrishna Nemani. 2013. A Technique of Analyzing Trust Relationships to Facilitate Scientific Service Discovery and Recommendation. In *2013 IEEE International Conference on Services Computing*. 57–64. doi:10.1109/SCC.2013.104