Augur: Architecture for Capturing, Reshaping, and Socially Contextualizing Open Source Software Communities

ANONYMOUS AUTHOR(S)

Open source projects are most often evaluated by potential contributors and consumers using metrics that describe a level of activity within the project because those measurements are available. The principle question in the minds of most evaluators, however, is "How healthy and sustainable is this project in the context of its competitors or dependent projects"? Computing, business, and software engineering research have long examined the open source world. Yet measurements and representations of health and sustainability are generally missing, or lack a systematic approach for making sense of the full range of tools and practices that constitute this work. Increasing corporate-communal engagement, the growth in the number of different communication channels and coordination mechanisms used by open source projects further exacerbates the challenge of understanding project health and sustainability. Researchers, in particular, need tools for increasing the feasibility of comprehensive, multi-project health and sustainability studies. From a practice perspective, these same conditions are increasing the difficulty organizations and individuals engaged in open source face when trying to understand the status, condition, and health of a particular project, the project's ecosystem or ecosystems emerging around their specific project context. The research presented here focuses on an architecture for data collection and presentation grounded in engaged field research, participatory design, and fast prototyping of a system called Augur. This study examines the work of a Linux Foundation working group, CHAOSS (Community Health Analytics Open Source Software) during the first three years of the formation through the lens of an architecture for prototyping metrics that evolved directly from the questions and demands of researchers and professional focused on more comprehensive metrics for assessing open source project health and sustainability. We conduct a review of existing literature examining socio-technical phenomena in open source, revealing a collective inconsistency in the manner for collection and presentation of open source software health and sustainability metrics, describe our engagement with the community, and illustrate how Augur's design and implementation as a candidate architecture and methodological approach for examining specific aspects of open source software health emerges from the research.

CCS Concepts: • **Information systems → Open source software**; • **Human-centered computing → Collaborative and social computing systems and tools**; • **Software and its engineering → Designing software**; **Open source model**.

Additional Key Words and Phrases: open source, design, community health, metrics , Augur

ACM Reference Format:
Anonymous Author(s). 2020. Augur: Architecture for Capturing, Reshaping, and Socially Contextualizing Open Source Software Communities. *ACM Trans. Soc. Comput.* 1, 1 (March 2020), 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Open source software is vital to most public computing infrastructure, and an increasingly common component of software development jobs. Following the Heartbleed OpenSSL security bug in 2014, the US Congress sought the aid of the Linux Foundation to help better understand what steps could

2469-7818/2020/3-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

be taken to make open source project health and risk more visible [87]. Open source software health is also a growing concern for highly skilled computing professionals in modern society, [85] as choosing to work on the right project will influence their earning potential. Indirectly, open source software's footprint on society is growing in a number of dimensions. As a point of reference, over five billion dollars[1] of direct labor costs are invested in the Linux Foundation's open source ecosystem. Annual revenue estimates for the Linux Kernel alone are currently 50 billion US dollars, supporting a conservative 500 billion US dollar market valuation.[2] Estimates of market value and labor investment are proxies to help situate the present scale of open source software, but financial measures alone are inadequate when we consider how central open source software is within the collected computing infrastructure of the globe.

Corporate engagement with open source [34, 35] is increasing the demand for coherent, consistent and actionable metrics. This demand is felt most by open source community managers, who play a role in the formal and informal nurturing of developer communities around open source projects and internal open source program offices who manage portfolios of open source projects a firm contributes to [25, 29]. The rapid growth of open source, the large number of ongoing metrics development research studies and the evolving needs of corporations engaged in open source lead to an overwhelming flood of answers to discrete questions about individual projects and ecosystems. Everyone involved in open source software wants a clearer understanding of project and ecosystem health and sustainability. To close the gap between a need for instrumentation of health and sustainability and the overwhelming milieu of available, discrete metrics that may inform health, open source projects need indicators that enable orientation alongside measurement.

Open source project health is defined and operationalized in many ways. Each method aims to make representations of open source project health efficient through the collection, aggregation and analysis of trace data from repositories, issue trackers, and other traces of work and communication [36]. This study adopts the general view that open source project health is a project's ability to continue to produce quality software [54, 61]. Trace data analysis is widely employed to make assessments of open source project health [18, 21, 22], making such mining of software artifacts at once essential for representing open source health, and also at present, incommensurate with the challenge [32, 56]. Research to date on open source project health is incommensurate with the scale and complexity of the problem of representing open source project health for two reasons: 1) the research is spread across several disciplines and is therefore not accumulating shared understanding (and arguably is diffusing disconnected insights), and 2) most research to date focuses on some specific, narrow segment of the expansive world of open source software, thus forestalling more substantial insights.

Specifically, studies of open source health are spread across research communities in software repository mining [4, 7, 22, 45, 55, 67], information systems [18, 23, 35, 60, 89], social computing [9, 12, 13, 56, 86] and software engineering [33, 37]. The aims of open source software health and sustainability metrics (metrics) research differ across disciplines. This opens the second area where current research is incommensurate with the problem of understanding open source health, because most of this research generates narrowly focused metrics on code [22, 44, 47, 59, 65, 75], diversity [5, 24], organizational dynamics [11, 57, 91, 92], risk [46, 48, 55], ecosystem state [45, 63, 64, 91], large scale classification [63, 64, 79], adverse event prediction [1, 55, 67], constructing end user tools [40, 45, 68, 69], and many others.

[1] https://www.linuxfoundation.org/press-release/2015/09/

[2] IBM acquired Red Hat software, an open source services vendor, for more than 10x revenue in 2018, so we use this as a conservative benchmark of market valuation derived from a multiplier of revenue.

Though many of the narrowly focused metrics applied to assess open source projects have utility, they do not get at the question of open source software health and sustainability. Metaphorically they could be thought of as using the results of an individual's annual physical to try and assess public health questions for the region they live in. This public health metaphor maps to the use of discrete, open source metrics to what public health researchers refer to as a type III error; we have the right answers to the wrong questions [70]. To avoid type III error and answer the right questions, open source health and sustainability variation across projects, differences within a project over time, and the presence of specific risk factors must integrate and be weighted against each other based on project context to make a broader claim of evaluating health and sustainability. Open source project context includes things like where a project is in its life-cycle for deciding how to evaluate the value of each metric. For example, a decline in activity could mean either an emergent risk of the community losing contributor engagement, or the stabilization of a fast-growing new technology. Most cases are not so clear cut as those two.

In this paper, we frame a three-year engaged field research and participatory design study within the Linux Foundation's Community Health Analytics in Open Source Software (CHAOSS) project and present a candidate architecture and methodological approach for examining specific aspects of open source software systems - AUGUR. The goal is to bring the CHAOSS metrics to life by designing a system with a UI and API that serves CHAOSS metrics, and to enable comparisons between similar projects. Our findings illustrate a pathway making open source health and sustainability visible in context by emphasizing comparison, transparency, trajectory and visualization through a set of design principles and a specific, implemented technology architecture called Augur. Our contribution is to frame an approach for understanding the overall health and sustainability of open source software projects that puts each project in a context salient for its position in its life-cycle, and within groups of projects with similarities identified through a combination of computational similarity measures and heuristic's defined by individuals and organizations engaged in open source software. This work aims to enable the kind of comparative research necessary for more wide-scale theory construction through the experience of open source participants.

2 LITERATURE REVIEW

2.1 Open source project health

There is a considerable amount of research constructing and presenting indicators of open source project activity, but a bonafide lack of consensus about how indicators derived from trace data might be used to represent a coherent view of open source project health and sustainability. Researchers define project health through the collection of success measures [17]; including metrics related to project output, process, and the outcomes for project members [18]. In this context, success is the result of project activity and the release of code, otherwise the project is abandoned [72]. Further measures of success explore the growth and diversity of projects [5]. Activity measures tell you when a project is finished; health and sustainability measures would identify that trajectory in advance.

Project health can be framed through measures representing separate but related indications of sustainability and survivability [14]. Sustainability is signaled by three factors – community growth, financial resources, and software management [3]. While sustainability evokes shared commitment, survivability aims to surface indications of a project's vigor, resilience, and organization in the face of risk [66]: when bad things happen to good projects, which ones are most likely to overcome? Sustainability and survivability, together, aspire to provide clear signals of a project's likelihood to continue producing quality software [54].

Likelihood of continued software production is a broad way to frame the goal of assessing open source project health. Measurement of health signals is taken up across a wide range of research domains. Aman et al. [2] applied the Pareto principle to open source projects to define a healthy project as a project where the proportion of core developers to non-core ones equates to roughly 80% of the code contributions being produced by 20% of the active developers. However, only a small slice of open source projects fit with this model [90]. Open source project health has also been assessed by examining the bus factor of projects (e.g., the risk associated with losing key project contributors and the knowledge that they possess) [15, 81], and research has shown that many open source projects have low bus factors, implying that they are not likely to survive disruption. In the next section we review a wide range of prior open source software research adjacent to questions of health, drawing out our argument that the present state of research is incommensurate with the problem space.

2.2 Open Source Health Research Solves a Different Problem

Researchers explore open source project health in different contexts and employ many different measures. Additionally, researchers often apply measures from other domains that don't take into account the unique nature of open source projects. What is clear is that activity is a common proxy for understanding project health [54], however, research focused on discrete activities addresses only the presence of activity, offering little insight about the complexities of future sustainability or survivability. Activity metrics illustrate the likelihood that a project will ever get traction, but tells us little about its arc following launch.

Activity is insufficient as a proxy for open source health particularly because many open source health studies are conducted at the smallest unit of analysis, building metrics up from one type of repository activity, such as a software commit [31]. Alternately, large scale summaries of open source health often squash entire repository histories and draw inferences about projects and ecosystems [45] without consideration of the evolution of projects over time. Table 1 illustrates the wide variety of measures used to operationalize open source project health using activity metrics or large scale summaries. Much of the early literature on open source community health focused on success measures [18, 19, 51, 53, 78, 88], however as open source health research has matured, the focus has moved to measures of sustainability [3, 8, 14, 31, 62, 71, 74, 76] and further, to an understanding that open source health must include considerations for social interactions and project diversity when estimating survivability [5, 20, 21, 41, 61, 80, 84]. Sustainability and survivability are difficult to estimate based on the current or historical level of activity alone, especially in an era where corporate engagement is a rapidly increasing aspect of open source ecosystems across the globe [35].

Our analysis of the literature surfaces a number of discrete metrics aimed at different aspects of open source project health, including a) sufficient scale, b) project culture, c) process quality, d) product quality, e) contextualized risk, f) license risk, and g) corporatization and access to resources. Each set of discrete metrics is useful, and reflects a particular aspect of an open source project that can contribute to its likelihood of continuing to produce quality software. This contemporary focus on the development of discrete metrics from open source project activity is necessary for ascertaining project health but does not make project health visible. While open source practitioners yearn for insight, they are often confused by an overwhelming array of dashboards that force together some set of "frankenmetrics" derived from the categories we identify. Open source metrics research is discrete, conducted at a point in time and not focused on the specific needs of metrics consumer communities like open source community managers and open source program offices. Open source health aims at a different set of questions. The "measurement of open source project health is further confounded because health metrics may have different meanings for different

Term Used	Operationalization of Project Health	Published
Success	(a) Number of contributors	2002 [51]
Success	(b) Popularity on the web	2005 [88]
Success	(c) Measures of the Output of Systems Development, Measures of the Process of Systems Development, and Effects on Project Teams.	2006 [18]
Success	(c) Codification of the messages recorded in the bug tracking system	2008 [19]
Success	(c) Software quality, community service quality, and user satisfaction	2009 [53]
Success	(f) Open Source License selection	2009 [78]
Success	(c) Quality model - Size and regeneration adequacy, Interactivity and workload adequacy, and Composition adequacy	2010 [43]
Success	(a) Project popularity and developer activity	2012 [58]
Success	(a) Long-term involvement of users and developers, project continually develops, matures, and evolves over time	2012 [52]
Success	(a) Number of users and developers	2012 [73]
Success	(a) Long-term involvement of users and developers, project continually develops, matures, and evolves over time	2012 [52]
Success	(a) Number of users and developers	2012 [73]
Success	(a) Long-term involvement of users and developers, project continually develops, matures, and evolves over time	2012 [52]
Success	(a) Number of users and developers	2012 [73]
Success	(b) Diversity - separation, variety, and disparity	2013 [20]
Success	(a) Pareto principle - 80% of contributions come from 20% of contributors	2015 [90]
Success	(b) Project growth and social diversity	2016 [5]
Success	(d) Low software coupling and high interactive discussion	2016 [21]
Success	(b) Concentration Index - Openness	2016 [84]
Sustainability	(d) Quality model - product quality, process maturity and sustainability	2009 [76]
Sustainability	(a) Development base (size), project age and the size of niche	2010 [14]
Sustainability	(c) Patch contribution and patch process	2010 [74]
Sustainability	(g) Community growth, financial resources and software management	2011 [3]
Survivability	(b) Viability index - vigor, resilience, and organization	2012 [66]
Sustainability	(b) Relationships among people, technologies, and organizations.	2012 [8]
Sustainability & Success	(b) Virtuous Circle - Feedback loop of open source best practices	2013 [71]
Sustainability	(e) Ability to Fork	2013 [62]
Sustainability	(a) Commits, retention of committers	2014 [31]
Risk	Truck factor - Number of key people that need to leave for a project to stall due to lack of knowledge	2011 [81]
Risk	(a) Bus Factor - Number of key people that need to leave for a project to stall due to lack of knowledge	2015 [15]
Health	(e) Productivity, robustness, and Niche creation	2014 [45]
Health	(b) Positive experience, trust in the leadership of the project leader, the demonstration of reciprocity, marketing the community, enriching knowledge, and face-to-face meetings	2015 [61]
Health	(b) Social Health - Ability to get help	2016 [41]
Health	(a) Pareto principle - 80% of contributions come from 20% of contributors	2017 [2]
Health	(b) Gender Bias - Ratio of female to male contributions accepted	2017 [80]

Table 1. Open Source Health is explored in wide variety of ways. (a) sufficient scale, (b) project culture, (c) process quality, (d) product quality, (e) contextualized risk, (f) license risk, (g) corporatization and access to resources.

projects and there is little consensus about the correct way to calculate metrics across projects" [35, p. 14]. With our work on Augur, we integrate metrics definition work from the CHAOSS project with a set of tools for synthesizing understanding of open source project health from an array of metrics in a more coherent presentation.

3 METHODS

This paper is positioned within an ongoing eight-year research study exploring organizational engagement with open source projects. Our efforts are localized in open source projects that include heavy organizational engagement, mainly projects brokered by the Linux Foundation,[3] a 501(c)(6) trade association and its member companies, which constitute a substantial majority of the technology sector of the economy. The Linux Foundation has helped "establish, build, and sustain some of the most critical open source technologies fostering innovation in every layer of the software stack. The Linux Foundation hosts projects spanning enterprise IT, embedded systems, consumer electronics, cloud, and networking".

Within this context and over the prior eight years, we employed a variety of approaches including participant observation [77], group informatics [36], direct engagement [82] , and critical reflection [28]. Participant observation was used as a field-based approach when members of our research team were directly engaged in the practices we sought to understand [77, 82, 83]. Group informatics was used to reflexively make meaning from the intersection of our fieldwork and the significant amounts of digital trace data that emerges within open source projects; thus ensuring coherence of research constructs in the outcomes produced in our participant observation [9, 36]. Finally, to ground our findings from participant observation [77] and group informatics [36], we used direct engagement and critical reflection as our "process of learning from experience" [28]. Through this field study, data was generated from approximately 200 interviews, 200 survey responses, ten focus groups, 1,000 pages of field notes, constant comparison, content analysis [50], trace ethnography [32], social network analysis [19, 36], and computational linguistic analysis [42].

3.1 Engaged Field Research

This paper focuses on a three-year engagement with the CHAOSS project[4]. Since its inception at the Linux Foundation's Open Source Leadership Summit in Spring 2017, CHAOSS has become a Linux Foundation project, was officially announced at the Open Source Summit North America in Fall 2017[5], and has come to include members from 70+ organizations such as RedHat, Pivotal, Intel, Mozilla, The Eclipse Foundation, Oath, and Bitergia. The mission of the CHAOSS project is to:

(1) Produce integrated, open source software for analyzing software development, and definition of standards and models used in that software in specific use cases
(2) Establish implementation-agnostic open source project health indicators for measuring community activity, contributions, and health.
(3) Optionally, produce standardized open source project health indicators exchange formats, detailed use cases, models, or recommendations to analyze specific issues in the industry/OSS world.

Through this three-year engaged field study, we collected data by observing [77] and writing field notes [26] to capture reflective observations and thoughts throughout the research study. Further, to design and build the Augur project, the research team used cooperative participatory design [39, 49] and group informatics [36] methods. Group informatics focuses on reflexively analyzing electronic

[3]https://www.linuxfoundation.org/projects/linux/
[4]http://chaoss.community
[5]https://events.linuxfoundation.org/events/open-source-summit-north-america-2018/

trace data and triangulating calculated values and ranges with the judgment of people interpreting the results, with the aim of ensuring coherence between what is measured and how it is presented. It is similar to trace ethnography [32] but focused on theoretically coherent scaling of data analysis. Collectively, these methods allow the research team to observe, capture, and describe the shared experiences of open source participants [16, 83] and inform the design of the Augur project.

3.2 Group Informatics

This research further focuses on the work of one project – Augur, a technology implementation of the metrics developed within the CHAOSS project – leveraging an integration of trace data analysis of repositories, issue trackers, and other artifacts to make sense of how the analysis of those traces represent the answers to questions posed by our engaged field research [36]. Throughout our three-year engagement with the CHAOSS project, the research team has become active in numerous open source projects related to project health. This engagement gives us a unique position to understand and report on open source project health research broadly. From this broad understanding, the research team created the Augur project and time was spent designing Augur around the needs expressed by practitioners and researchers in these ecosystems, as well as others. The CHAOSS working group that focuses on the growth, maturity, and decline of open source projects influences the architecture of Augur most directly, while other working groups on diversity & inclusion, common metrics, value, and risk are beginning to emerge but do not have their interests fully defined or reflected in the Augur architecture to date.

3.3 Participatory Design

The Augur project fills an important technology niche not addressed by other analytic tools within CHAOSS. Augur is focused on building human-centered [10] open source project health indicators, defined by collaborations of the CHAOSS project and other open source project stakeholders. Users and designers are represented in project groups and steering committees and take an active part in analysis and design, evaluation of standard systems, and organizational implementation [49].In most research studies, participatory design rarely goes beyond the initial analysis and design stages. This study uses a cooperative experimental systems development approach [39] that includes stakeholders in all stages of the design process. Open source software projects provide a unique outlet for participatory design research, supporting, active user involvement, throughout the design and development process.

The design principles that underlie the Augur architecture for open source project health metrics emerged from discussions in the CHAOSS project over a three-year research engagement. Specifically we engaged in four open source health design sessions that included representatives from dozens of technology firms, piloted Augur for 14 different projects, interviewed consumers and stakeholders periodically on their use and needs, and hosted the Augur project on an open development platform allowing stakeholders to not only provide insight but to collaborate in all aspects of the design and development. As we presented functional designs using live stakeholder data our focus and architecture evolved. Further, as health measures became more visible, specific and nuanced intentions and goals emerged.

4 BUILDING AUGUR

There are 21 active prototypes of Augur currently being used, evaluated, and contributing to making the picture of health and sustainability in open source easier to see. This paper provides an overview of the Augur design principles and general architecture resulting from our engagement with CHAOSS. The field study is in several respects the evolution of our collaborative participatory

design process into the field. Its also the case that the line between field study and participatory design becomes blurred for engaged field research centered design work like Augur.

4.1 Order from CHAOSS

The CHAOSS working group's efforts to develop a set of metrics and the tooling around metrics is complicated by competing perspectives. The standards the working group can develop are limited to the discrete metrics that can be discussed and agreed to, which is not of course the end of the wire for defining health and sustainability in open source. Knowing what is implemented in the Augur Architecture is merely a compass within it, whose intention is to express trust through alignment with a larger set of goals *not* defined by Augur. In this way, and in concert with the participant designers collaborating on Augur, the discrete metrics definitions generated by CHAOSS are integrated into a coherent whole that is defined by the informants in our engaged field research.

4.1.1 Project Stakeholders. Work within the CHAOSS project identified two key stakeholder types - community managers and program managers - interested in community health metrics. Open source community managers oversee communities in the communal side of the corporate communal relationship. The job of the open source community manager is to enable open source project members to achieve personal and project goals [6]. The role does not always have an explicit title nor is the job mutually exclusive. These individuals may take on many roles in a project, including developers, document writers, facilitators and maintainers. The focus for these individuals is building and maintain healthy communities of contributors - towards the successful completion of individual project goals.

Open source program managers oversee corporate interests from corporate side of corporate communal relationships. The key difference between an open source program manager and a community manager is the corporate focus. Open source program managers are primarily concerned with the interests of the corporation and may manage a portfolio of open source projects that are strategically important to the corporation. The responsibilities of an open source program manager are varied but often include - communicating the open source strategy within and outside the company, overseeing the execution of the corporate strategy, facilitating the effective use of open source in within the company, ensuring the quality and frequent release of software, engaging with communities and ensuring that the company contributes to open source projects, license compliance and fostering an open source culture [30].

Community and program managers take a variety of perspectives, depending on where their communities are in the life-cycle of growth, maturity, and decline. This paper is an evolving report of what we are learning from community and program office managers, some of whom we are working with on live experiments with a CHAOSS project prototyping software tool called Augur[6]. At this point, we are paying particular attention to how community managers consume metrics and how the presentation of open source software health and sustainability metrics could make those metrics more and in some cases less useful for evaluating open source project health.

4.1.2 New Goals and Concerns. "Who is going to use these metrics and for what purpose?" is a question that matters a great deal to the people engaged in the CHAOSS project. The development of health and sustainability metrics for open source software surfaced specific concerns among our participants. Making statistics available to people unfamiliar with the state, functioning and current goals of an open source ecosystem creates risk that the numbers will lead to a reflexive, uncontextualized side narrative that could harm the efforts of open source community and project office managers. The risk is that numbers without context will aid in the development of stories

[6]http://www.github.com/CHAOSS/augur

that are not focused on increasing health and sustainability, but instead focused on telling stories that emphasize activity over health. To counteract these concerns, the aims of Augur's development were refined to reflect participant objectives explicitly:

> "Augur empowers open source community managers to tell data-driven stories"

Moreover, development is increasingly focused on key open source community and project manager stories:

- "As a community manager I want to be able to compare the open source projects that I manage"
- "As a community manager I want to identify trends in the data collected about my repositories"
- "As a community manager I need to know which contributors are making the largest impact across projects that I manage"
- "As a community manager I need to know how contributor behavior is changing"
- "As someone who is adopting open source, I need to know what risks my company will face if they use software"
- "As someone who is adopting open source, I need to be able to compare projects to determine which solution will be the least risky to our company"
- "As an open-source contributor, I would like to be able to measure my impact across the projects I work on"
- "As an open-source contributor, I would like to know how I compare to my peers"

4.1.3 From Metrics to Health. Frustrations with the elusiveness of coherent, durable indicators of open source project health pervade the experiences we cataloged in our participatory design process. In practice, there is evidence that community managers and open source program offices occasionally stand up discrete metrics tools, but ultimately fail to find them useful in their search to fully understand open source project health over time. One of the central goals that emerged from our participatory design process was to make it easier for open source stakeholders to "get their bearings" on a project and understand "how things are going". Interactions with stakeholders created insight that this was most easily accomplished when comparisons between internal and external projects over time, are readily available. In both anticipation and response to that central goal, Augur's design made comparisons between projects and the aggregation of related projects a primary aim.

Through our field study and participatory design, four core principles for making sense of data emerged: comparison, transparency, trajectory and visualization. Augur's design focuses on operationalizing those principles through four human centered data science strategies:

(1) *Comparison:* Enabling comparisons, letting people navigate complex unknowns analogically as well as see how their project compares with others they are familiar with.
(2) *Trajectory:* Making time a fundamental dimension in all open source project health indicators as point in time scores are useful. Augur makes useful historical comparisons available to anticipate and reveal activity trajectories.
(3) *Transparency:* Making the provenance from raw data to open source project health indicators visualization transparent.People trust open source project health indicators when they can see the underlying data and providing traceability back to the CHAOSS project open source project health indicators requires transparency.
(4) *Visualization:* Enabling visualizations as downloadable as a .csv, .svg, or other data exchange format from which to build local stories.

Great answers to the wrong questions are more commonplace than we prefer because open source software work is evolving quickly and we do not yet have a list of the right questions for

many specific project situations. To ground how Augur's design strategies are employed in our study with CHAOSS we frame our iterative inquiries around a set of specific, common design tools and contextual questions:

(1) *Goals*: What are metrics going to help you accomplish?
(2) *Use Cases*: When you go to use metrics, what are the use cases you have? A case can be simple, ill-formed and even 'unpretty':
   (a) "My manager wants to know if anyone else is working on this project?"
   (b) "It seems like my community is leveling off? Is it? Or is it just so large now I cannot tell?"

Fundamentally, the community managers and program offices we talked with are clear that they ultimately want to be able to use tools like Augur to construct narrative descriptions of the nuanced status for their projects. The story of open source project health is at its core a story, not a dashboard of discrete metrics. And the story cannot be told without discrete metrics, available in a form that follows the design principles that emerged from our participatory design process.

4.1.4 Data Coverage and Provenance. Data concerning open-source projects are spread across collaboration tools and are not uniform. There are five principal sources of information used to generate metrics (See Table 2). Source code repositories like git, issue trackers (GitHub.com, Bugzilla, etc.), mailing lists, online discussion forums (e.g., Stack Exchange) and a collection of minor sources for code reviewing, contribution queuing, library package management and IRC/Chat. There are some critical issues with each data source and the generation of metrics from them that are addressed in the design of the Augur architecture, implementing the principle of transparency. There is generally a need to ensure you have a sufficient cross-section of key activity, and recognizing how to deal with differences between projects when drawing comparisons.

Data	**Source Examples**	**Augur**
Code	git, GitHub, Gitlab, GHTorrent	Core
Issue Tracking	GitHub, Bugzilla	Core
Coordination	Mailing lists	Experimental
Troubleshooting and learning	Stack Exchange	Emerging
Minor sources	IRC, Libraries.io, code reviews	Experimental

Table 2. Core and non-core data source types with examples from the CHAOSS project and Augur architecture. Core data is included in current deployments of the Augur architecture, emerging data sources are awaiting CHAOSS definitions and experimental data sources are in development branches of Augur.

GitHub.com is the most popular platform for open-source projects to manage **code**, with more than 85 million repositories. Because the development is in the open, most data is available from GitHub, which provides an application program interface (API) to retrieve the data. The GitHub API is the one, canonical data source for Git Repositories stored on GitHub. GHTorrent, which draws its data from the GitHub API, is continuously collecting GitHub data, including both Code and Issue tracking data, through the GitHub API. Anyone can download and perform analysis on the history and provides their history going back to 2010 [38]. These data are freely available to download as a database dump. There are also tools that collect information from git repositories and store the results of each scan for comprehensive analysis later, including Percival [37], Facade and others. Repository data in Augur has 3 specific origins available for consumers of the metrics:

the GitHub API, GHTorrent and a git repository miner called "Facade[7]". Each origin is identified transparently in the user interface as well as the API for any instance of Augur, reducing concerns about comparisons drawn from different data sets with different properties. For example, many long-standing open source projects migrated to GitHub from prior git repositories or other forms of software version control like SVN. Data around those conversions delivered through GHTorrent or the GitHub API is sometimes flawed, reflecting project by project decisions at the time of migration. Git repository statistics, in contrast, tend to be more reliable through those periods.

GitHub and Bugzilla are sources of **issue tracking** within Augur, though these metrics have more diverse sources that are managed by established communities differently. In some cases, open source software companies maintain a community issue tracker alongside internal issue trackers focused on preparing product releases, failed tests and the like. If Issue tracking data is sourced for an implementation, then it is available. Otherwise is not and Augur indicates "no data" for metrics that rely on issue data. For many projects, GHTorrent represents a complete enough set of data about a project. If there is a desire to evaluate the content of issues submitted and their responses then the GitHub API or other issue tracker integration is used, as GHTorrent does not supply issue text.

There is significant interest in how open source projects **coordinate** activities. Of specific interest is identifying mailing list communication that is off-putting for potential project newcomers and diverse populations. There are also several readily available tools for harvesting the contents of mailing lists, and for these data sources, Augur implements a plugin architecture that allows many different forms of integration. Unlike issue trackers, which mostly follow a clear, threaded structure of issue:response, mailing list archives come in forms that to different degrees suppress threading or occlude response structures. As Augur's implementation of these types of data evolves, the limitations and structural characteristics of different archive types are important to place on the surface. Mailing list data is looked to for evidence of different kinds of project coordination.

Many studies of open source software focus on **trouble shooting and learning** systems for understanding how people engage with an open source project, but many different groups in our cohort are actively looking to online discussion systems like Stack Exchange as an early warning signal of project health for consumers and contributors. Newcomers are likely to turn to Stack Exchange when they encounter issues with a project. As such, this type of data source is emerging. Unlike other open source data sources, online discussion forums require compliance with published terms of service which may not always allow collection and redistribution of available data.

There are several open source metrics sources that we evaluate to be **minor sources** at this time because the practices around them vary substantially, their data is by design not archived or there is limited current but potential future interest in their development as metrics. In terms of priorities for looking at open source health and sustainability, these data sources are not likely to prove useful today for population measures. However, they may prove useful to individual ecosystems where specific tools are frequently used and available.

Bridging the cacophony of metrics and dashboards is aided by showing a clear connection between the data served by Augur and the metrics standards being developed in the CHAOSS project. Figure 1 shows how Augur maintains an up to date inventory of where each CHAOSS metric is implemented within its architecture. This cross reference also includes a cross walk of individual metrics with the working group within CHAOSS that is developing them, and an explicit link to the API endpoint (See Figure 2).

4.1.5 Data Fusion. Open source contributions from GitHub are of two general types: Code and discussion about code. Code activity is organized under commits, and commits can be discussed

[7]http://facade-oss.org

Fig. 1. Metrics status information that includes a link to the CHAOSS metric definition, the working group responsible for the metric and the data source. The online interface also contains status on Augur front end and back end implementation, a link to the API endpoint and the type of graph it is (i.e., time series), which is a useful signal to developers adding graphical representations.

Backend Status	Frontend Status	Name	Group
implemented	unimplemented	Closed Issues	growth-maturity-decline
implemented	unimplemented	Lines Of Code Changed	growth-maturity-decline
implemented	implemented	Bus Factor	experimental
implemented	implemented	Major Tags	experimental
implemented	implemented	Tags	experimental
implemented	unimplemented	Bus Factor	experimental
implemented	unimplemented	Major Tags	experimental
implemented	unimplemented	Tags	experimental

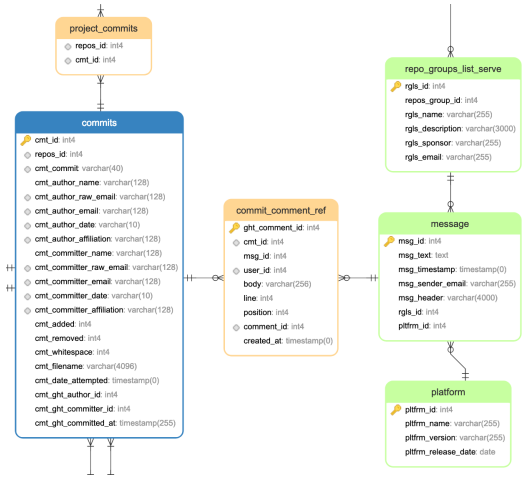
Endpoint	Source	Metric Type
none	githubapi	metric
/api/unstable/<owner>/<repo>/lines_changed	githubapi	metric
/api/unstable/<owner>/<repo>/bus_factor	githubapi	metric
/api/unstable/<owner>/<repo>/timeseries/tags/major	githubapi	timeseries
/api/unstable/<owner>/<repo>/timeseries/tags	githubapi	timeseries
none	githubapi	metric
none	githubapi	metric
none	githubapi	metric

Fig. 2. If an open source project or ecosystem stakeholder only wants to consume curated data for processing, the Augur API Architecture provides important metadata about each data endpoint's origin, classification and connection to the CHAOSS project.

```
{
    "groups": [
        {
            "diversity-inclusion": "Diversity and
            Inclusion",
            "growth-maturity-decline": "Growth, Maturity,
            and Decline",
            "risk": "Risk",
            "value": "Value",
            "activity": "Activity",
            "experimental": "Experimental"
        }
    ],
    "sources": [
        "ghtorrent",
        "ghtorrentplus",
        "githubapi",
        "downloads",
        "facade",
        "publicwww",
        "librariesio",
        "git"
    ],
    "metric_types": [
        "timeseries",
        "metric",
        "git"
    ],
    "tags": {
        "listening": "diversity-inclusion",
        "speaking": "diversity-inclusion",
        ...
    }
}
```

before being merged. In the same way, email is a discussion, but not a discussion tied directly to a code activity. To get to a state where the talk activity around a project can be synthesized in the same way code contributions can, Augur organizes all conversations into one data structure. Messages come from issue trackers, commit comments, mailing lists and other platforms (e.g. Stack Exchange, Bugzilla) where conversation and not code is the principle contribution. This is illustrated in Figure 3 where there is a commits table connected via a reference table (commit_comment_ref) to the messages table.

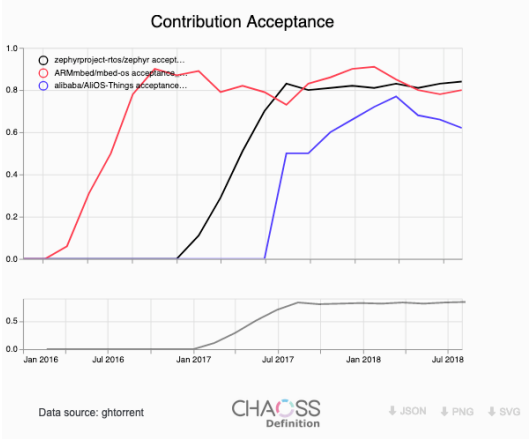
Fig. 3. Augur consumes data from many sources, but stores them in a common data structure that identifies the origin of each piece of data. This allows people exploring open source software health metrics to quickly adapt their view of the data without needing to first restructure the data and then restructure their analysis tools (Augur or otherwise). This portion of the Augur data model shows how communications are stored in a common table. Issues, comments on commits and mailing list archives are all here, structured and annotated.



Consumers of data, of course, still want to know the origins of the data used to assemble their metrics. Augur maintains the origin information for all data, and makes that information available through both its API and user interface. Figure 4 maintains the transparency of data origin by stating it explicitly, linking to the CHAOSS metrics definition and providing direct downloads of the data driving the diagram in JSON format or, the visualization itself in two popular graphics formats. Being able to download the graphic is especially important for Augur's role in helping open source program and community managers tell the stories behind a project's current health and sustainability outlook. Figure 2 shows how metadata is passed from an open source health and sustainability data consumer so that, even if they are not using Augur's front end, data provenance is still transparent. The CHAOSS Group, data source, metric data type and metric tags are passed with each request.

Harvesting data from multiple origins is a recognized problem for anyone who has attempted to perform a connected analysis on more than one data source. This is one of the important ways that the technology architecture for making sense of open source project health is substantially different than the architecture for presenting metrics. There are different degrees of summarization and clarity available in each of sources we have for git repositories, for example. By storing the results of each analysis in a structured format that includes the provenance we are able to quickly shift how information is reported to people interested in open source health. The alternative is to collect unstructured information and shape it into a structure at the time we want to change the analysis. This makes it easier to quickly change which metrics are presented in what way. Instead of needing to refactor data when, for example, we don't want to count white space commits, Augur lets the user make that choice. This has significant advantages for community and program office managers.

Fig. 4. Comparison, transparency, trajectory and storytelling in a single visualization. This visualization compares three nominally competing open source projects in the real time operating system domain through the lens of contribution acceptance rates. The data source is clearly stated and downloadable in a standard JSON file (the 21st century's .csv). You can see the relative changes in acceptance rate over time, and certainly an expert in this domain would interpret those arcs. Finally, since the visualization is downloadable in two common graphics formats it can move from a web page to a presentation or report with ease.



> *Advantages of Integrating multiple sources in a structured data architecture:* For a commit to the twitter/twemoji project in early 2018[8], GHTorrent provides information about the date of the commit, the author of the commit and the organization and repository it was for. Both the GitHub API and our git repository miner extract specific information about the additions and deletions for those commits. Our git repository miner, however, provides information about the white space in the totality of the commit, and each file committed with changes as well as each unchanged file (lots of zeros). The result, with regards to calculating the amount of new code, is that GitHub reports 1,052,157 additions, but our git miner reports only 1,045,584 additions. The difference is the exact number our git miner reports as white space (6,573). To adjust the way commits are counted when accumulating information about project health we only need to make one small change. The data we need is already there.

Several of the observations validate the helpfulness and usefulness of the design principles surfaced through the participatory design process that Augur evolved through. The health metrics API Augur provides is a unique outgrowth of the field study. If we consider the design decision to create structured data as well as the design decision to make this structured data transparent and its provenance delivered through the API, it becomes clear that Augur shows a model for developing health and sustainability information that is distinct from prior research on open source metrics. The API Augur provides is beginning to serve as a data aggregator for open source health and sustainability, and is beginning to provide endpoints that are more than implementations of CHAOSS metrics, but summaries of sets of metrics parameterized in the ways that open source program office and community managers want to consume them in order to advance their work. The field study and participatory design identified seven metric aggregation data endpoints that serve these higher level open source health and sustainability questions.

[8]Commit "sha": 8b6c9d853ef210a166f73177e9b716231fb0269e

(1) For each repository in a collection of repositories being managed, each repository that first appears in the parameterized calendar year, show all commits for that year (total for year by repository). Result ranked from highest number of commits to lowest by default.
(2) For each repository in a collection of repositories being managed, each repository that first appears in the parameterized calendar year, show all lines of code for that year (total for year by repository). Result ranked from highest number of commits to lowest by default.
(3) For each repository in a collection of repositories being managed, each repository's total commits during the current Month, Year or Week. Result ranked from highest number of commits to lowest by default.
(4) For each repository in a collection of repositories being managed, each repository's total commits during the current Month, Year or Week. Result ranked from highest number of LOC to lowest by default.
(5) For a single repository, all the commits and lines of code occurring for the specified year, grouped by the specified interval (week or month)
(6) For a single repository, all the commits and lines of code occurring for the specified year, grouped by the specified interval (week or month) and by the affiliation of individuals and domains that are not mapped as "inside" within the repositories gitdm file. "Unknown" is, in this case, interpreted as "outside".
(7) For each repository in a collection of repositories, all the commits and lines of code occurring for the specified year, grouped by repository and the specified interval (week or month). Results ordered by repository.

5 DISCUSSION

Open source projects are unique engagements. Tooling to measure open source project health needs to take this uniqueness into consideration. That means that dashboards and metrics need to be easily customizable for the specific needs of individual projects and the needs of specific users. Fundamentally, what we demonstrate with Augur's design, evolved from participation in the open source community, is that metrics connected to sustainability and survivability are woven into the particular needs of open source program officers and community managers. When engaged with a standard set of metrics as CHAOSS provides, as well as a coherent set of metrics like those in Augur, individual program officers and community managers are able to add the context of their work to a dashboard-like interface that addresses their core questions.

Augur adds coherence to the discrete metrics CHAOSS specifies, and applied in the context of a particular community manager or program officer's questions, limits the danger of type III errors (answer the wrong questions). The generalization that is important to understand is that open source health and sustainability metrics are effectively constructed for one context and a moment in time; finely tuned for a small market, and difficult to scale or reuse. Health and sustainability is distinct from the generation and dissemination of metrics because of its focus on helping stakeholders construct a meaningful narrative about what's happening in an open source environment. Narratives provide context, and it seems clear that the context in open source software is the difference between a dashboard of metrics that overwhelms its consumer and actionable information. Type III errors exist in health care because populations are compared without considering the differences in those populations. Translating that challenge into open source health and sustainability research might benefit from the lessons in health care simulations that show the difference between life and death is often making sense of the context surrounding a sea of data [27]. It does little good to focus on the blood pressure of somebody while they are choking.

Perhaps the great challenge for any work developing metrics is the overwhelming simplicity of translating the concept of a "metric" to something easily quantified. Where this research does advance health and sustainability is by showing a model for applying human centered design to the measurement of expression of progress for activities conducted by humans. Used carelessly, metrics can be dangerous either because they answer the wrong questions or mislead their recipients regarding the most useful answer. The depth of our engagement with the CHAOSS community, in concert with our development of the Augur architecture, reflects a confluence of engaged field research and active design that evaded our attempts to find comparable research.

5.1 Storytelling

"Who is going to use these metrics and for what purpose?" is a question that matters a great deal to the people engaged in the CHAOSS project. The development of health and sustainability metrics for open source software surfaced specific concerns among our participants. Making statistics available to people unfamiliar with the state, functioning and current goals of an open source ecosystem creates risk that the numbers will lead to a reflexive, uncontextualized side narrative that could harm the efforts of open source community and project office managers. The risk is that numbers without context will aid in the development of stories that are not focused on increasing health and sustainability, but instead focused on telling stories that emphasize activity over health. To counteract these concerns, the aims of Augur's development are being refined to reflect participant objectives explicitly:

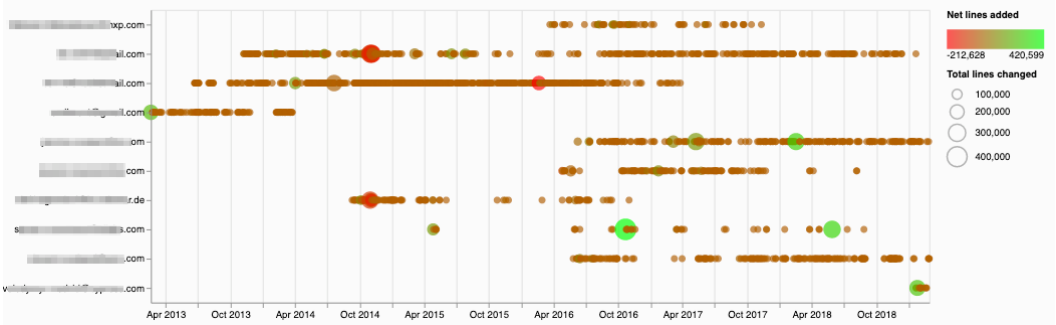
> "Augur empowers open source community managers to tell data-driven stories"

Moreover, development is increasingly focused on key open source community and project manager stories:

- "As a community manager I want to be able to compare the open source projects that I manage"
- "As a community manager I want to identify trends in the data collected about my repositories"
- "As a community manager I need to know which contributors are making the largest impact across projects that I manage"
- "As a community manager I need to know how contributor behavior is changing"
- "As someone who is adopting open source, I need to know what risks my company will face if they use software"
- "As someone who is adopting open source, I need to be able to compare projects to determine which solution will be the least risky to our company"
- "As an open-source contributor, I would like to be able to measure my impact across the projects I work on"
- "As an open-source contributor, I would like to know how I compare to my peers"

Telling a health and sustainability story for an open source project requires more than a collection of the metrics available and defined for a project. Because having tools that are useful for telling stories is of such importance to open source program and community managers the visual design of data in Augur is aimed at providing a synthesis of several related metrics in visualizations that solve specific problems or answer specific questions that emerged during design. Figure5 presents the full life of one open source project in a way that makes key information buried in commit records visible at a glance. Specifically, how stable is leadership on the project? Is anyone who is historically very active suddenly not active? (Look at row 2.) How frequently are the significant spikes in activity? Are there drive-by contributors making significant contributions? Each of these questions emerged as important for telling the stories behind open source health and sustainability trajectories within a project and is served by Figure 5.

Fig. 5. Changing dynamics and leadership: Activity metrics provide an aggregation of actions in a repository and around a project that show people are still interested. Community managers also need awareness of how different core developer activity levels and contributions are changing. In this one graph we see activity level for major contributors over time. The large red and green dots also indicate periods of substantial change in the code base, and identify the individual making those contributions. Regular releases and routine hand offs between different core contributors are signals of health and sustainability.



Health and sustainability is not entirely a matter of processing traces of activity to answer new questions. There are also actions that can be and are prescribed for open source projects to follow in order to demonstrate sufficient, explicit health characteristics defined by a community. For example, figure 6 shows Augur's badge for the Core Infrastructure Initiative's (CII) [9] badging program. CII badging indicates a projects level of compliance with open "source project best practices" as defined by the Linux Foundation's core infrastructure initiative, which focuses on CyberSecurity in open source software. The goal of the program is to encourage projects to produce better more secure software and allow others to determine if a project is following best practices. The 94% compliance level indicated in Figure 6 represents that Ruby on Rails does not meet all of the criteria to receive the best practices badge, but the application is in-progress. Projects receive the passing badge if they meet all of the required criteria [10]. The status of each criterion, for a given project, can be 'Met', 'Unmet', 'N/A' or 'unknown'. Each criterion is in one of four categories: 'MUST', 'SHOULD', 'SUGGESTED', or 'FUTURE'. To obtain a badge, all MUST and MUST NOT criteria must be met, all SHOULD criteria must be met OR the rationale for not implementing the criterion must be documented, and all SUGGESTED criteria have to be rated as met or unmet.

Fig. 6. This is Augur's badging for the Core Infrastructure Initiative, a specific and explicit health and sustainability indicator that is focused on Cybersecurity.

The Augur architecture emerged from a stark recognition that a good deal of open source metrics research to date has not arrived at the central importance of storytelling. While we do not resolve

[9] https://www.coreinfrastructure.org/
[10] urlhttps://github.com/coreinfrastructure/best-practices-badge/blob/master/doc/criteria.md

this gap, we argue that the significance of the architecture and functioning software we developed is its illustration that supporting storytelling is critically important for any effort to measure and monitor open source software health and sustainability. Counting activity measures the heartbeat, not the health & sustainability.

6 CONCLUSION

Integrating data from multiple sources and across multiple projects exceeds the resources of many social computing researchers. The Augur architecture is both a resource for open source scholars to leverage in future studies and, we think a model for considering how we study phenomena that spans contexts. To this end, Augur's architecture may serve social computing research more broadly construed.

Furthermore, theory construction around technologically mediated organizational forms like open source software is constrained by the same opaqueness that frustrates open source practitioners seeking a firmer grasp on the health and sustainability of their corporate communal engagements. The research aims of the Augur architecture are aimed squarely at questions of open source health and sustainability, but the potential is greater. Our participatory design ethic is, like a lot of engaged field research, leading us to consider the development of theories that explain the dynamics we observe. Because the Augur architecture affords us a clearer view into the ways human action in open source influence health and sustainability, its within the realm of at least imagining to consider how and to what extent tools that take a more human centered perspective in the study of sociotechnical phenomena may, in fact, help motivate and inform essential theories about how sociotechnical environments like open source software operate.

REFERENCES

[1] Ibrahim Abunadi and Mamdouh Alenezi. 2015. Towards Cross Project Vulnerability Prediction in Open Source Web Applications. In *Proceedings of the The International Conference on Engineering & MIS 2015 (ICEMIS '15)*. ACM, New York, NY, USA, 42:1–42:5. https://doi.org/10.1145/2832987.2833051

[2] H. Aman, A. E. Burhandenny, S. Amasaki, T. Yokogawa, and M. Kawahara. 2017. A Health Index of Open Source Projects Focusing on Pareto Distribution of Developer's Contribution. In *2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP)*. 29–34. https://doi.org/10.1109/IWESEP.2017.14

[3] Flávia Linhalis Arantes and Fernanda Maria Pereira Freire. 2011. Aspects of an open source software sustainable life cycle. In *IFIP International Conference on Open Source Systems*. Springer, 325–329.

[4] Ross D. Arnold and Jon P. Wade. 2015. A definition of systems thinking: a systems approach. *Procedia Computer Science* 44 (2015), 669–678.

[5] Joop Aué, Michiel Haisma, Kristín Fjóla Tómasdóttir, and Alberto Bacchelli. 2016. Social Diversity and Growth Levels of Open Source Software Projects on GitHub. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '16)*. ACM, New York, NY, USA, 41:1–41:6. https://doi.org/10.1145/2961111.2962633

[6] Jono Bacon. 2012. *The art of community: Building the new age of participation.* " O'Reilly Media, Inc.".

[7] Andrew Begel, James D. Herbsleb, and Margaret-Anne Storey. 2012. The future of collaborative software development. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion*. ACM, 17–18.

[8] Matthew J Bietz, Toni Ferro, and Charlotte P Lee. 2012. Sustaining the development of cyberinfrastructure: an organization adapting to change. (2012), 10.

[9] Kelly Blincoe, Jyoti Sheoran, Sean Goggins, Eva Petakovic, and Daniela Damian. 2016. Understanding the popular users: Following, affiliation influence and leadership on GitHub. *Information and Software Technology* 70 (2016), 30–39.

[10] Richard Buchanan. 2001. Human dignity and human rights: Thoughts on the principles of human-centered design. *Design issues* 17, 3 (2001), 35–39.

[11] Andrea Capiluppi, Alexander Serebrenik, and Ahmmad Youssef. 2012. Developing an H-index for OSS Developers. In *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR '12)*. IEEE Press, Piscataway, NJ, USA, 251–254. http://dl.acm.org/citation.cfm?id=2664446.2664484

[12] Marcelo Cataldo, Matthew Bass, James D. Herbsleb, and Len Bass. 2007. On coordination mechanisms in global software development. In *International Conference on Global Software Engineering (ICGSE 2007)*. IEEE, 71–80.

[13] Marcelo Cataldo and James D. Herbsleb. 2008. Communication networks in geographically distributed software development. ACM Press, 579. https://doi.org/10.1145/1460563.1460654

[14] Indushobha Chengalur-Smith, Anna Sidorova, and Sherae Daniel. 2010. Sustainability of Free/Libre Open Source Projects: A Longitudinal Study. *Journal of the Association for Information Systems* 11, 11 (Nov. 2010). http://aisel.aisnet.org/jais/vol11/iss11/5

[15] V. Cosentino, J. L. C. Izquierdo, and J. Cabot. 2015. Assessing the bus factor of Git repositories. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. 499–503. https://doi.org/10.1109/SANER.2015.7081864

[16] Andy Crabtree. 1998. Ethnography in participatory design. In *Proceedings of the 1998 Participatory design Conference*. Computer Professionals for Social Responsibility Stanford, CA, 93–105.

[17] Kevin Crowston, Hala Annabi, and James Howison. 2003. Defining Open Source Software Project Success. In *in Proceedings of the 24th International Conference on Information Systems (ICIS 2003*. 327–340.

[18] Kevin Crowston, James Howison, and Hala Annabi. 2006. Information systems success in free and open source software development: Theory and measures. *Software Process: Improvement and Practice* 11, 2 (2006), 123–148.

[19] Kevin Crowston and Barbara Scozzi. 2008. Bug Fixing Practices within Free/Libre Open Source Software Development Teams1. *Journal of Database Management* 19, 2 (April 2008), 1–30. bibtex: Crowston2008.

[20] Sherae Daniel, Ritu Agarwal, and Katherine J. Stewart. 2013. The effects of diversity in global, distributed collectives: A study of open source project success. *Information Systems Research* 24, 2 (2013), 312–333.

[21] Sherae Daniel and Katherine Stewart. 2016. Open source project success: Resource access, flow, and integration. *The Journal of Strategic Information Systems* 25, 3 (Oct. 2016), 159–176. https://doi.org/10.1016/j.jsis.2016.02.006

[22] Julius Davies, Daniel M. German, Michael W. Godfrey, and Abram Hindle. 2011. Software Bertillonage: Finding the Provenance of an Entity. In *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR '11)*. ACM, New York, NY, USA, 183–192. https://doi.org/10.1145/1985441.1985468

[23] Angelika Dimoka, Yili Hong, and Paul A. Pavlou. 2012. On product uncertainty in online markets: Theory and evidence. *MIS quarterly* 36 (2012).

[24] Nadia Eghbal. 2016. *Roads and bridges: The unseen labor behind our digital infrastructure*. Technical Report. Ford Foundation.

[25] Margaret S. Elliott and Walt Scacchi. 2003. Free software developers as an occupational community: resolving conflicts and fostering collaboration. In *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*. ACM, 21–30.

[26] Robert M. Emerson, Rachel I. Fretz, and Linda L. Shaw. 2011. *Writing ethnographic fieldnotes*. University of Chicago Press.

[27] Henrik Engström, Magnus Andersson Hagiwara, Per Backlund, Mikael Lebram, Lars Lundberg, Mikael Johannesson, Anders Sterner, and Hanna Maurin Söderholm. 2016. The impact of contextualization on immersion in healthcare simulation. *Advances in Simulation* 1, 1 (Jan. 2016). https://doi.org/10.1186/s41077-016-0009-y

[28] Jan Fook. 2011. Developing critical reflection as a research method. In *Creative spaces for qualitative researching*. Springer, 55–64.

[29] Apache Foundation. 2019. Apache Community Development. https://community.apache.org/

[30] Linux Foundation. 2019. Creating an Open Source Program. https://www.linuxfoundation.org/resources/open-source-guides/creating-an-open-source-program/

[31] Jonas Gamalielsson and Björn Lundell. 2014. Sustainability of open source software communities beyond a fork: How and why has the libreoffice project evolved? *Journal of Systems and Software* 89 (2014), 128–145. http://www.sciencedirect.com/science/article/pii/S0164121213002744

[32] R. Stuart Geiger and David Ribes. 2011. Trace ethnography: Following coordination through documentary practices. In *2011 44th Hawaii International Conference on System Sciences*. IEEE, 1–10.

[33] Daniel M. German, Yuki Manabe, and Katsuro Inoue. 2010. A sentence-matching method for automatic license identification of source code files. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*. ACM, 437–446.

[34] Matt Germonprez, Julie E. Kendall, Kenneth E. Kendall, Lars Mathiassen, Brett Young, and Brian Warner. 2017. A theory of responsive design: A field study of corporate engagement with open source communities. *Information Systems Research* 28, 1 (2017), 64–83. https://doi.org/10.1287/isre.2016.0662

[35] Matt Germonprez, Georg J.P. Link, Kevin Lumbard, and Sean Goggins. 2018. Eight Observations and 24 Research Questions About Open Source Projects: Illuminating New Realities. *Proc. ACM Hum.-Comput. Interact.* 2, CSCW (Nov. 2018), 57:1–57:22. https://doi.org/10.1145/3274326

[36] Sean P. Goggins, Christopher Mascaro, and Giuseppe Valetto. 2013. Group informatics: A methodological approach and ontology for sociotechnical group research. *Journal of the American Society for Information Science and Technology* 64, 3 (2013), 516–539.

[37] Jesus M. Gonzalez-Barahona, Paul Sherwood, Gregorio Robles, and Daniel Izquierdo. 2017. Technical lag in software compilations: Measuring how outdated a software deployment is. In *IFIP International Conference on Open Source Systems*. Springer, Cham, 182–192.

[38] Georgios Gousios. 2013. The GHTorent dataset and tool suite. In *Proceedings of the 10th working conference on mining software repositories*. IEEE Press, 233–236.

[39] K. Gronbaek, Morten Kyng, and Preben Mogensen. 1997. Toward a cooperative experimental system development approach. *Computers and design in context* (1997), 201–238.

[40] Anna Hannemann, Kristjan Liiva, and Ralf Klamma. 2014. Navigation Support in Evolving Open-Source Communities by a Web-Based Dashboard. In *Open Source Software: Mobile Open Source Technologies*, Luis Corral, Alberto Sillitti, Giancarlo Succi, Jelena Vlasenko, and Anthony I. Wasserman (Eds.). Number 427 in IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, 11–20. https://doi.org/10.1007/978-3-642-55128-4_2

[41] Andrew Head. 2016. Social Health Cues Developers Use when Choosing Open Source Packages. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2016)*. ACM, New York, NY, USA, 1133–1135. https://doi.org/10.1145/2950290.2983973

[42] Joshua E. Introne and Marcus Drescher. 2013. Analyzing the flow of knowledge in computer mediated teams. In *Proceedings of the 2013 conference on Computer supported cooperative work (CSCW '13)*. ACM, New York, NY, USA, 341–356. https://doi.org/10.1145/2441776.2441816

[43] D. Izquierdo-Cortazar, J. M. Gonzalez-Barahona, S. Duenas, and G. Robles. 2010. Towards Automated Quality Models for Software Development Communities: The QualOSS and FLOSSMetrics Case. In *2010 Seventh International Conference on the Quality of Information and Communications Technology*. 364–369. https://doi.org/10.1109/QUATIC.2010.66

[44] Daniel Izquierdo-Cortazar, Nelson Sekitoleko, Jesus M. Gonzalez-Barahona, and Lars Kurth. 2017. Using Metrics to Track Code Review Performance. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE'17)*. ACM, New York, NY, USA, 214–223. https://doi.org/10.1145/3084226.3084247

[45] Slinger Jansen. 2014. Measuring the health of open source software ecosystems: Beyond the scope of project health. *Information and Software Technology* 56, 11 (2014), 1508–1519. http://www.sciencedirect.com/science/article/pii/S0950584914000871

[46] Kalpana Johari and Arvinder Kaur. 2011. Effect of Software Evolution on Software Metrics: An Open Source Case Study. *SIGSOFT Softw. Eng. Notes* 36, 5 (Sept. 2011), 1–8. https://doi.org/10.1145/2020976.2020987

[47] Kalpana Johari and Arvinder Kaur. 2012. Validation of Object Oriented Metrics Using Open Source Software System: An Empirical Study. *SIGSOFT Softw. Eng. Notes* 37, 1 (Jan. 2012), 1–4. https://doi.org/10.1145/2088883.2088893

[48] Inderpreet Kaur and Arvinder Kaur. 2012. Empirical Study of Software Quality Estimation. In *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology (CCSEIT '12)*. ACM, New York, NY, USA, 694–700. https://doi.org/10.1145/2393216.2393332

[49] Finn Kensing and Jeanette Blomberg. 1998. Participatory Design: Issues and Concerns. *Computer Supported Cooperative Work (CSCW)* 7, 3 (Sept. 1998), 167–185. https://doi.org/10.1023/A:1008689307411

[50] Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology*. Sage Publications, Thousand Oaks, CA.

[51] Sandeep Krishnamurthy. 2002. Cave or Community? An Empirical Examination of 100 Mature Open Source Projects. *First Monday* 7, 6 (June 2002). http://firstmonday.org/ojs/index.php/fm/article/view/960

[52] Liz Laffan. 2012. A new way of measuring openness: The open governance index. *Technology Innovation Management Review* (2012), 18–24.

[53] Sang-Yong Tom Lee, Hee-Woong Kim, and Sumeet Gupta. 2009. Measuring open source software success. *Omega* 37, 2 (2009), 426–438.

[54] Georg J.P. Link and Matt Germonprez. 2018. Assessing Open Source Project Health. *AMCIS 2018 Proceedings* (Aug. 2018).

[55] Lech Madeyski and Marcin Kawalerowicz. 2017. Continuous Defect Prediction: The Idea and a Related Dataset. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR '17)*. IEEE Press, Piscataway, NJ, USA, 515–518. https://doi.org/10.1109/MSR.2017.46

[56] Nora McDonald, Kelly Blincoe, E. V. A. Petakovic, and Sean Goggins. 2014. Modeling distributed collaboration on Github. *Advances in Complex Systems* 17, 07n08 (2014), 1450024.

[57] Andrew Meneely, Alberto C. Rodriguez Tejeda, Brian Spates, Shannon Trudeau, Danielle Neuberger, Katherine Whitlock, Christopher Ketant, and Kayla Davis. 2014. An Empirical Investigation of Socio-technical Code Review Metrics and Security Vulnerabilities. In *Proceedings of the 6th International Workshop on Social Software Engineering (SSE 2014)*. ACM, New York, NY, USA, 37–44. https://doi.org/10.1145/2661685.2661687

[58] Vishal Midha and Prashant Palvia. 2012. Factors affecting the success of Open Source Software. *Journal of Systems and Software* 85, 4 (2012), 895–905.

[59] Manishankar Mondal, Chanchal K. Roy, and Kevin A. Schneider. 2012. Connectivity of Co-changed Method Groups: A Case Study on Open Source Systems. In *Proceedings of the 2012 Conference of the Center for Advanced Studies on*

Collaborative Research (CASCON '12). IBM Corp., Riverton, NJ, USA, 205–219. http://dl.acm.org/citation.cfm?id=2399776.2399795

[60] Drashko Nakikj and Lena Mamykina. 2018. Lost in Migration: Information Management and Community Building in an Online Health Community. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 146.

[61] Damrongsak Naparat, Patrick Finnegan, and Michael Cahalane. 2015. Healthy Community and Healthy Commons: 'Opensourcing' as a Sustainable Model of Software Production. *Australasian Journal of Information Systems* 19, 0 (2015). https://doi.org/10.3127/ajis.v19i0.1221 bibtex: Naparat2014.

[62] Linus Nyman and Juho Lindman. 2013. Code forking, governance, and sustainability in open source software. *Technology Innovation Management Review* 3, 1 (2013), 7–12. https://timreview.ca/article/644

[63] Marco Ortu, Giuseppe Destefanis, Mohamad Kassab, and Michele Marchesi. 2015. Measuring and Understanding the Effectiveness of JIRA Developers Communities. In *Proceedings of the Sixth International Workshop on Emerging Trends in Software Metrics (WETSoM '15)*. IEEE Press, Piscataway, NJ, USA, 3–10. http://dl.acm.org/citation.cfm?id=2821491.2821495

[64] Konstantinos Plakidas, Srdjan Stevanetic, Daniel Schall, Tudor B. Ionescu, and Uwe Zdun. 2016. How Do Software Ecosystems Evolve? A Quantitative Assessment of the R Ecosystem.. In *Proceedings of the 20th International Systems and Software Product Line Conference (SPLC '16)*. ACM, New York, NY, USA, 89–98. https://doi.org/10.1145/2934466.2934488

[65] Foyzur Rahman and Premkumar Devanbu. 2013. How, and Why, Process Metrics Are Better. In *Proceedings of the 2013 International Conference on Software Engineering (ICSE '13)*. IEEE Press, Piscataway, NJ, USA, 432–441. http://dl.acm.org/citation.cfm?id=2486788.2486846

[66] U. Raja and M. J. Tretter. 2012. Defining and Evaluating a Measure of Open Source Project Survivability. *IEEE Transactions on Software Engineering* 38, 1 (Jan. 2012), 163–174. https://doi.org/10.1109/TSE.2011.39

[67] Thomas Rausch, Waldemar Hummer, Philipp Leitner, and Stefan Schulte. 2017. An Empirical Analysis of Build Failures in the Continuous Integration Workflows of Java-based Open-source Software. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR '17)*. IEEE Press, Piscataway, NJ, USA, 345–355. https://doi.org/10.1109/MSR.2017.54

[68] Christoffer Rosen, Ben Grawi, and Emad Shihab. 2015. Commit Guru: Analytics and Risk Prediction of Software Commits. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)*. ACM, New York, NY, USA, 966–969. https://doi.org/10.1145/2786805.2803183

[69] Daniel Rozenberg, Ivan Beschastnikh, Fabian Kosmale, Valerie Poser, Heiko Becker, Marc Palyart, and Gail C. Murphy. 2016. Comparing Repositories Visually with Repograms. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16)*. ACM, New York, NY, USA, 109–120. https://doi.org/10.1145/2901739.2901768

[70] S Schwartz and K M Carpenter. 1999. The right answer for the wrong question: consequences of type III error for public health research. *American Journal of Public Health* 89, 8 (Aug. 1999), 1175–1180. https://doi.org/10.2105/AJPH.89.8.1175

[71] Charles M. Schweik. 2013. Sustainability in open source software commons: Lessons learned from an empirical study of SourceForge projects. *Technology Innovation Management Review* 3, 1 (2013).

[72] Charles M. Schweik and Robert C. English. 2012. *Internet success: a study of open-source software commons*. MIT Press, Cambridge, Mass. http://linker.worldcat.org/?jHome=http%3A%2F%2Fleo.lib.unomaha.edu%2Flogin%3Furl%3Dhttp%3A%2F%2Fsite.ebrary.com%2Flib%2Funomaha%2FTop%3Fid%3D10571239&linktype=best

[73] Ravi Sen, Siddhartha S. Singh, and Sharad Borle. 2012. Open source software success: Measures and analysis. *Decision Support Systems* 52, 2 (2012), 364–372.

[74] Bhuricha Deen Sethanandha, Bart Massey, and William Jones. 2010. Managing open source contributions for software project sustainability. In *PICMET 2010 TECHNOLOGY MANAGEMENT FOR GLOBAL ECONOMIC GROWTH*. IEEE, 1–9.

[75] Meera Sharma, Madhu Kumari, and V. B. Singh. 2015. Post Release Versions Based Code Change Quality Metrics. In *Proceedings of the Third International Symposium on Women in Computing and Informatics (WCI '15)*. ACM, New York, NY, USA, 235–243. https://doi.org/10.1145/2791405.2791466

[76] M. Soto and M. Ciolkowski. 2009. The QualOSS open source assessment model measuring the performance of open source communities. In *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. 498–501. https://doi.org/10.1109/ESEM.2009.5314237

[77] James P. Spradley. 2016. *Participant Observation*. Waveland Press. Google-Books-ID: q7DlCwAAQBAJ.

[78] Chandrasekar Subramaniam, Ravi Sen, and Matthew L. Nelson. 2009. Determinants of open source software project success: A longitudinal study. *Decision Support Systems* 46, 2 (2009), 576–585.

[79] Fathi Taibi. 2013. Reusability of Open-source Program Code: A Conceptual Model and Empirical Investigation. *SIGSOFT Softw. Eng. Notes* 38, 4 (July 2013), 1–5. https://doi.org/10.1145/2492248.2492276

[80] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. 2017. Gender differences and bias in open source: pull request acceptance of women versus men. *PeerJ Computer Science* 3 (May 2017), e111. https://doi.org/10.7717/peerj-cs.111

[81] Marco Torchiano, Filippo Ricca, and Alessandro Marchetto. 2011. Is My Project's Truck Factor Low?: Theoretical and Empirical Considerations About the Truck Factor Threshold. In *Proceedings of the 2Nd International Workshop on Emerging Trends in Software Metrics (WETSoM '11)*. ACM, New York, NY, USA, 12–18. https://doi.org/10.1145/1985374.1985379

[82] Andrew H. Van de Ven. 2007. *Engaged scholarship: A guide for organizational and social research.* Oxford University Press on Demand.

[83] John van Maanen. 1988. *Tales of the field: on writing ethnography.* The University of Chicago Press. bibtex: Maanen1988.

[84] Ir Robert Viseur. 2016. Open Concentration Index: Measure of Market Concentration in Open Source Industry. In *Proceedings of the 12th International Symposium on Open Collaboration (OpenSym '16)*. ACM, New York, NY, USA, 6:1–6:4. https://doi.org/10.1145/2957792.2957796

[85] John Vivian, Arjun Arkal Rao, Frank Austin Nothaft, Christopher Ketchum, Joel Armstrong, Adam Novak, Jacob Pfeil, Jake Narkizian, Alden D Deran, Audrey Musselman-Brown, Hannes Schmidt, Peter Amstutz, Brian Craft, Mary Goldman, Kate Rosenbloom, Melissa Cline, Brian O'Connor, Megan Hanna, Chet Birger, W James Kent, David A Patterson, Anthony D Joseph, Jingchun Zhu, Sasha Zaranek, Gad Getz, David Haussler, and Benedict Paten. 2017. Toil enables reproducible, open source, big biomedical data analyses. *Nature Biotechnology* 35 (April 2017), 314. https://doi.org/10.1038/nbt.3772

[86] P. Wagstrom, J. D. Herbsleb, R. E. Kraut, and A. Mockus. 2010. The impact of commercial organizations on volunteer participation in an online community. In *Academy of Management Annual Meeting*.

[87] Greg Walden and Gregg Harper. 2018. Letter to Mr. Zemlin from the One Hundred Fifteenth Congress of the United States Committee on Energy and Commerce. https://energycommerce.house.gov/wp-content/uploads/2018/04/040218-Linux-Evaluation-of-OSS-Ecosystem.pdf

[88] Dawid Weiss. 2005. Measuring success of open source projects using web search engines. (2005).

[89] John D. Wells, Joseph S. Valacich, and Traci J. Hess. 2011. What signal are you sending? How website quality influences perceptions of product quality and purchase intentions. *MIS quarterly* (2011), 373–396.

[90] Kazuhiro Yamashita, Shane McIntosh, Yasutaka Kamei, Ahmed E. Hassan, and Naoyasu Ubayashi. 2015. Revisiting the applicability of the pareto principle to core development teams in open source software projects. In *Proceedings of the 14th International Workshop on Principles of Software Evolution*. ACM, 46–55.

[91] Kazuhiro Yamashita, Shane McIntosh, Yasutaka Kamei, and Naoyasu Ubayashi. 2014. Magnet or Sticky? An OSS Project-by-project Typology. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR 2014)*. ACM, New York, NY, USA, 344–347. https://doi.org/10.1145/2597073.2597116

[92] Ahmmad Youssef and Andrea Capiluppi. 2015. The Impact of Developer Team Sizes on the Structural Attributes of Software. In *Proceedings of the 14th International Workshop on Principles of Software Evolution (IWPSE 2015)*. ACM, New York, NY, USA, 38–45. https://doi.org/10.1145/2804360.2804365