# The Bones of the System: A Case Study of Logging and Telemetry at Microsoft

Titus Barik[1], Robert DeLine[2], Steven Drucker[2], Danyel Fisher[2]

[1]North Carolina State University
Raleigh, North Carolina, USA
tbarik@ncsu.edu

[2]Microsoft Research
Redmond, Washington, USA
{rdeline, sdrucker, danyelf}@microsoft.com

## ABSTRACT

Large software organizations are transitioning to event data platforms as they culturally shift to better support data-driven decision making. This paper offers a case study at Microsoft during such a transition. Through qualitative interviews of 28 participants, and a quantitative survey of 1,823 respondents, we catalog a diverse set of activities that leverage event data sources, identify challenges in conducting these activities, and describe tensions that emerge in data-driven cultures as event data flow through these activities within the organization. We find that the use of event data span every job role in our interviews and survey, that different perspectives on event data create tensions between roles or teams, and that professionals report social and technical challenges across activities.

## Categories and Subject Descriptors

D.2.0 [**Software Engineering**]: General—*Standards*; D.2.9 [**Software Engineering**]: Management—*Programming teams*

## Keywords

boundary object, collaboration, logging, telemetry, developer tools, practices

> *Let's log that... It's like in the bones of the system. (Ryan$_{dev}$)*

## 1. INTRODUCTION

When software engineers talk about "log data," they often refer to the output from trace statements in source code, which are intended to help with troubleshooting and debugging [6, 23, 24, 22]. However, as software companies have moved to software as a service [3], the concept of "log data" has dramatically broadened in scope.

Cloud-hosted services centralize the collection of data about software, service, and customer behavior, which allows the data to be collected continuously at scale [13]. To cope with scale, software teams have adopted big data platforms, like Hadoop. Large software companies, like Google and Microsoft, both use these data platforms internally and also provide them to third parties through services like Google Web Analytics and Microsoft Application Insights. To allow continuous monitoring, teams have adopted streaming data platforms, like Storm. Teams often consume this streaming data, called *telemetry*, through live dashboards and periodic reports. Both the data at rest (logs) and the data in motion (telemetry) are conceptually the same: timestamped, customizable records that capture system and user events. Because of this similarity, we use the term *event data* to cover both cases.

The motivation for adopting event data platforms is to support the shift to data-driven decision making [14], also called software analytics [12, 25]. Understanding this shift is important for two reasons. First, the new data-driven team practices provide the context for future innovations in software engineering tools and processes. Second, the shift itself provides the opportunity for software engineering research to smooth the transition. The most salient feature in the shift from traditional logging to event data is the widening of scope: while traditional logging was primarily focused on one team role (developers) and one work activity (troubleshooting), the goal of adopting these new platforms is for the entire team to benefit from the availability of event data. This leads to the following research questions:

**RQ1** How are different team roles using event data and for what work activities?
**RQ2** Does the use of event data cause organizational issues, or tensions, between teams or team roles?
**RQ3** What challenges do software engineers face when working with event data?

To address these questions, we conducted a mixed-methods study of how teams at Microsoft use event data. First, we conducted 28 semi-structured interviews with software engineers in a variety of roles, namely, software developers, program managers, operations engineers, content creators, data scientists, and managers. We coded the interview transcripts to form a grounded theory of how teams use event data, which we then confirmed with a survey with 1,823 respondents. Our findings are:

1. The use of event data spans every job role in our interviews and survey. Software engineers work with event data in eight activities: (1) *engineering the data pipeline* (implementing and maintaining infrastructure for event data); (2) *instrumenting for event data* (adding logs to the code base);

*ICSE '16 Companion, May 14 - 22, 2016, Austin, TX, USA*
© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-4205-6/16/05...$15.00
DOI: http://dx.doi.org/10.1145/2889160.2889231

(3) *"doing data science"* (performing experiments to make recommendations); (4) *troubleshooting problems* (debugging the cause of an issue); (5) *monitoring system health* (inspecting event data for performance and uptime); (6) *improving the user experience* (understanding user behavior); (7) *triaging work items* (prioritizing and assigning bugs); and (8) *making business decisions* (planning about products and technology investments). Our expectation was that job roles would mostly focus on subsets of these work activities, for example, developers doing troubleshooting, program managers doing triaging, and operations doing monitoring. Surprisingly, our study shows that these activities crosscut job roles.

2. The event data act as a *boundary object* between roles, that is, the data are shared among roles and are sufficiently flexible to be used for different purposes. For instance, the same event data can allow one team member to find the root cause of a crash, another to rank the most frequent features that customers use, and another to decide whether to open a new data center in China. Occasionally, these different perspectives create conflict between roles or teams. For example, the desire to scrub personal information from the data can lose details that are useful to find the root cause of a problem.

3. Software engineers report a number of challenges across all work activities. The worst-rated problem for all but one activity was combining data from multiple sources (for data science, it was the second-worst problem). Other problems that rate highly include the ease of use of the tools, the amount of clerical work required, the difficulty of getting relevant insights, and the amount of time that the activity requires. These challenges suggest open problems for researchers to address.

## 2. BACKGROUND

Previous research on logging has focused on its use and production by developers for troubleshooting [15, 19, 13]. Two previous empirical studies document logging in practice. Yuan *et al.* mined the revision history of four large, long-term open-source projects and found that logging statements are pervasive and that project developers revise those statements a third of the time to get them right. Fu *et al.* analyzed millions of lines of Microsoft source code to build an empirical model of where developers choose to log, which they validated with a survey of 54 developers [6]. Other research makes the use of logs for troubleshooting more productive, for example, by automating the process of finding likely executions [22] and by enhancing the program's trace statements [24].

The use of log data for software analytics is part of a larger trend of software companies adopting data science. Kim *et al.* interviewed 16 data scientists at Microsoft and found five distinct working styles: insight providers, who work with engineers to provide answers to managers' questions; modeling specialists, who consult with teams about building predictive models; platform builders, who engineer the infrastructure to collect, store and analyze data; polymaths, who did a little of everything on their own; and team leaders, who run data science teams [9]. While this previous study focused on the emerging role of the data scientist, the study in this paper widens the lens to look at how data is affecting all roles across the entire company, including the boundaries between roles. Other studies have documented the work practices and pain points of data scientists, outside of software engineering. Fisher *et al.* describe data science workflows with long waits on batch-style data query systems and frequent tool switching [5]. Kandel *et al.* organized 35 interviewed data analysts into three personas based on tool usage—hackers, scripters, and application users—and describe the dominance of data preparation ("data wrangling") in the work practice [8].

**Table 1: Interview Participants by Role, with Their Years at Microsoft and Current Team**

**Developers (DEV)** ($n = 10, \mu = 9.2$ yrs)
Albert (16, productivity tools), Austin (5, games), Chad (9, operating systems), Dylan (11, development tools), Gary (8, productivity tools), Ida (5, productivity tools) Keith (7, games), Mark (4, databases), Ryan (15, productivity tools), Sean (12, operating systems)

**Program Managers (PM)** ($n = 9, \mu = 10.7$ yrs)
Amanda (19, R&D), Amber (15, databases), Anna (5, operating systems), Blake (9, games), Desmond (15, operating systems), Henry (12, operating systems), Jon (3, operating systems), Steve (9, cloud platform), Ulysses (8, mobile device)

**Operations Engineers (OPS)** ($n = 2, \mu = 10.5$ yrs)
Caleb (7, sales and services), Ronald (14, cloud platform)

**Content Developers (CD)** ($n = 2, \mu = 12.5$ yrs)
Dale (15, productivity tools), Marvin (10, development tools)

**Data Scientists (DS)** ($n = 4, \mu = 13$ yrs)
Brian (1, productivity tools), Jerry (7, operating systems), Paul (18, operating systems), Rudy (26, data pipeline)

**Service Engineer (SRV)** ($n = 1, \mu = 9$ yrs)
Arthur (9, games)

## 3. STUDY

Our study methodology consisted of two phases. In the first phase, we conducted qualitative, semi-structured interviews across a variety of roles at Microsoft. In the second phase, we used the transcribed interviews to generate a quantitative, closed-end survey to verify and generalize the results.[1]

### 3.1 Semi-structured Interviews

**Initial recruitment survey.** To identify potential interview participants, we randomly deployed an initial recruitment survey to 1,897 employees at Microsoft's corporate headquarters in Redmond, Washington, USA. This short survey provided a definition of event data, asked participants if they used these types of sources, and requested permission for us to follow-up with them. We received 211 responses (response rate of 11%), of which 166 (79%) consented to follow-up. We filtered out participants who had less than three years of experience, since employees with less than that experience are often still learning their roles [18].

**Participant selection.** We scheduled one-hour interviews over a five-week period in June and July of 2015. Consistent with grounded theory, we used an iterative theoretical sampling approach that attempts to maximize new

[1]The initial recruitment survey, semi-structured interview guide, and quantitative survey materials are available online at http://aka.ms/ckel58.

information gained and obtain diverse coverage about event data perspectives from each additional participant [21]. We obtained *theoretical saturation*, that is, the point at which we were obtaining interchangeable stories, at 28 participants (see Table 1).

**Interview protocol.** We conducted semi-structured interviews that focused on the informants' recent experiences with event data. The use of "storytelling" from actual projects, rather than general discussion, has been shown to be more accurate and offer rich, contextualized detail over standard interviews [11]. Using guidelines from Hova and Anda, we conducted interviews in pairs, with one interviewer taking the lead while the other takes notes and asks additional questions [7]. In the style of contextual inquiry, we interviewed participants in their own offices, where they often showed us relevant artifacts, like log files and telemetry reports. We recorded all interviews for later transcription, and interviews typically lasted just under an hour.

**Analysis.** We used a grounded theory approach to analyze the interview data [21]. After transcribing the interviews, we used the ATLAS.ti[2] data analysis software for qualitatively coding the data [16]. We performed coding over multiple iterations. In the first cycle, we used descriptive coding, that is, assigning short phases or labels to data, and process coding, using gerunds to classify actions in the transcripts. From first-cycle coding, we identified eight activities performed by individuals using event data sources. In the second cycle, we performed axial coding within each activity to ground and characterize the description of the activity. We stratify these activities through the analytical framework of *boundary objects*, which we elaborate in Section 4.

**Member checking.** We used *member checking* to raise the trustworthiness of our interpretation of the participants' interview remarks. In this strategy, participants are offered an opportunity to approve our interpretation to determine "whether the data analysis is congruent with participants' experiences" [4]. Following Carlson's guidelines [2], we conducted a single-event member check using a preprint version of this paper, which gave the participants an opportunity to review their quoted remarks in the context of the larger work. We received five replies from the participants, of which only one required minor changes to their quotation.

## 3.2 Follow-up Survey

We conducted a quantitative survey to help triangulate, generalize, and increase the credibility of our qualitative findings.

**Protocol.** We asked participants basic demographic questions about their profession and experience at Microsoft. For each of the eight activities, we asked the participant the frequency of performing the activity on a standard 6-point frequency scale. We also provided participants an opportunity specify an "Other" activity, in order to help assess whether we had reached theoretical saturation.

For each activity performed once or more per week, the participant received an additional page about the challenges and team dependencies of the activity. We asked participants about the most recent time in which they performed the activity. The challenges were drawn from the interviews and presented as statements on a 5-point agreement Likert scale. To understand how event data serve as a boundary object, we asked participants to identify which teammate's activity prompted them to perform their current activity.

[2]http://atlasti.com/

**Participant selection.** We randomly sampled 6,451 participants from the corporate address book, from all locations worldwide. To increase the role diversity of our responses, we weighted the sampling based on the overall distribution of professions. We received 1,823 responses (28% response rate).

## 4. OVERVIEW OF FINDINGS

In this section, we use the analytic framework of *boundary objects* as a method of understanding how event data flow through organizations:

> Boundary objects are objects which are both plastic enough to adapt to local needs and the constraints for the several parties employing them, yet robust enough to maintain a common identity across sites. They are weakly structured in common use, and become strongly structured in individual-site use. [20]

As we argue in this paper, event data act as boundary objects. In our grounded theory analysis of activities, we find that event data not only serve to "iteratively coordinate perspectives and to bring disparate communities of practice into alignment" [10], but are also "woven into the fabric of organizational life" [17] and "do not work as stand-alone deliverables; they need to be explained and elaborated" [1].

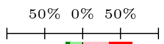
**Event data activities.** Our open and axial coding of the interview data revealed eight distinct activities, which we organized in terms of distance from the data. The activities are: *Data Engineering*: Engineering the Data Pipeline (PIP), Instrumenting for Logs or Telemetry (INS); *Data Analysis*: "Doing Data Science" (DS); *Software Maintenance*: Monitoring Services (MON), Troubleshooting Problems (TS); *Decision Making*: Improving the User Experience (UX), Triaging Work Items (TRI), Making Business Decisions (BUS).

**Frequency of activities.** Most respondents do these activities weekly: 66% do troubleshooting once or more per week; 63%, monitoring; 56% triaging; 54%, data pipeline; 50%, instrumentation; 47% for business decisions; and 44% for data science. Between 91-94% of respondents report using event data in these eight activities. Respondents rarely selected the OTHER option in the activity survey, which increases our confidence that we reached theoretical saturation for activities in our interviews. In all cases, the OTHER responses were specialized versions of the eight activities, for example, "debugging issues" for Troubleshooting Problems and "tracking outages" for Monitoring. We discuss these activities in in detail in Section 5.

**Activity challenges.** In our interviews, we heard many stories about challenges faced while performing activities. To generalize these challenges, we aggregated the survey data for each activity and present challenges for which over 50% of respondents agree or strongly agree with the challenge statement (Table 2). These responses show that combining multiple sources of data is the biggest challenge for almost all activities, that those who make business decisions face a significant number of challenges, and that instrumenting event data is the only activity free of significant challenges.

**Activity distribution among professions.** Initially we expected event data activities to cluster around professional disciplines, e.g., program managers triage work items

Table 2: **Challenges Reported in Follow-up Survey Responses**

| | | Likert Resp. Counts[1] | | | | | |
|---|---|---|---|---|---|---|---|
| | **% Agree** | **SD** | **D** | **N** | **A** | **SA** | **Distribution[2]** |
| **Troubleshooting Problems** (4) | | | | | | |  |
| Combining multiple sources of data is difficult. | 65% | 15 | 39 | 32 | 87 | 73 | |
| The tools for this activity are too slow. | 52% | 13 | 59 | 49 | 69 | 62 | |
| The tools for this activity aren't easy to use. | 51% | 12 | 61 | 49 | 79 | 50 | |
| The activity involves too much clerical effort. | 50% | 5 | 12 | 13 | 13 | 17 | |
| **Monitoring System Health** (1) | | | | | | | |
| Combining multiple sources of data is difficult. | 58% | 6 | 37 | 23 | 45 | 46 | |
| **Doing Data Science** (5) | | | | | | | |
| The tools for this activity aren't easy to use. | 69% | 3 | 7 | 5 | 22 | 11 | |
| The tools for this activity are too slow. | 60% | 2 | 3 | 14 | 17 | 12 | |
| The activity requires more effort than I have time for. | 59% | 3 | 9 | 6 | 15 | 11 | |
| I have to do a lot of coordination with other people. | 58% | 3 | 9 | 7 | 15 | 11 | |
| I have to wait on other people. | 51% | 3 | 11 | 9 | 17 | 7 | |
| **Making Business Decisions** (11) | | | | | | | |
| Combining multiple sources of data is difficult. | 73% | 2 | 8 | 4 | 22 | 16 | |
| The activity requires more effort than I have time for. | 69% | 2 | 7 | 7 | 25 | 11 | |
| The tools for this activity aren't easy to use. | 68% | 0 | 10 | 7 | 21 | 15 | |
| The activity involves too much clerical effort. | 67% | 0 | 10 | 7 | 21 | 14 | |
| The tools for this activity are too slow. | 67% | 0 | 9 | 8 | 22 | 12 | |
| The tools for this activity are flaky or unreliable. | 58% | 0 | 12 | 10 | 19 | 12 | |
| I have to do a lot of coordination with other people. | 57% | 5 | 9 | 9 | 17 | 13 | |
| The tools make it difficult for me to get the insights I want. | 56% | 1 | 15 | 7 | 18 | 11 | |
| I have to wait on other people. | 55% | 5 | 10 | 9 | 19 | 10 | |
| The activity involves too much mental effort. | 51% | 3 | 14 | 8 | 17 | 9 | |
| The data doesn't contain the information I want. | 50% | 2 | 18 | 6 | 17 | 9 | |
| **Improving the User Experience** (5) | | | | | | | |
| The activity requires more effort than I have time for. | 52% | 4 | 12 | 13 | 22 | 10 | |
| The tools for this activity aren't easy to use. | 52% | 3 | 17 | 9 | 19 | 13 | |
| The activity involves too much clerical effort. | 51% | 4 | 14 | 12 | 17 | 14 | |
| The tools for this activity are too slow. | 51% | 3 | 14 | 13 | 16 | 15 | |
| Combining multiple sources of data is difficult. | 50% | 5 | 12 | 13 | 13 | 17 | |
| **Engineering the Data Pipeline** (1) | | | | | | | |
| Combining multiple sources of data is difficult. | 53% | 5 | 13 | 11 | 18 | 15 | |
| **Instrumenting for Logs or Telemetry** (0) | | | | | | | |
| **Triaging Work Items** (1) | | | | | | | |
| The activity requires more effort than I have time for. | 52% | 9 | 13 | 7 | 18 | 14 | |

[1] Likert responses: Strongly Disagree (SD), Disagree (D), Neutral (N), Agree (A), Strongly Agree (SA).
[2] Net stacked distribution removes the Neutral option and shows the skew between positive (more challenging) and negative (less challenging) responses. 🟩 Strongly Disagree, 🟩 Disagree, 🟥 Agree; 🟥 Strongly Agree.

**Table 3: Top Activities by Profession**

| Profession | Primary | | Secondary | |
|---|---|---|---|---|
| **Business Operations** (42) | BUS | 20% | UX | 19% |
| **Engineering** | | | | |
| Software Engineering (393) | TS | 20% | MON | 16% |
| Software Development (241) | TS | 20% | MON | 17% |
| Program Management (195) | TRI | 16% | TS | 15% |
| Service Engineering (71) | MON | 20% | TS | 19% |
| Software Testing (23) | TS | 21% | TRI | 15% |
| Applied Sciences (18) | DS | 20% | PIP | 17% |
| Content Publishing (17) | UX | 30% | TRI | 19% |
| **IT Operations** | | | | |
| Service Engineering (36) | TS | 21% | MON | 21% |
| Program Management (31) | TRI | 19% | BUS | 17% |
| Service Operations (28) | TS | 24% | TRI | 19% |
| Software Development (9) | TS | 18% | MON | 18% |
| Solution Management (9) | BUS | 25% | TRI | 19% |
| **Research** (19) | DS | 24% | BUS | 18% |
| **Sales** (55) | BUS | 29% | UX | 25% |
| **Services** | | | | |
| Support Engineering (127) | TS | 28% | MON | 19% |
| Delivery Management (35) | MON | 18% | BUS | 16% |
| Technical Delivery (34) | TS | 21% | MON | 19% |
| Support Delivery (29) | TS | 23% | MON | 22% |
| Services Leadership (7) | BUS | 23% | TS | 15% |

and developers troubleshoot. Instead, we find that activities crosscut professions and that a single profession conducts multiple activities (Table 3).

**Co-occurring activities.** Individual respondents reported doing a mix of activities on a weekly basis (▮▮▮▮▪▪▪, 1-8 activities). The most frequently reported pair of activities was Monitoring and Troubleshooting (46% of respondents). All of the other pairs of activities were reported by 11–26% of respondents, so no pair of activities can be considered rare. The lack of clustering of activities among individuals and professions may be due to the emerging nature of data science. These activities have not yet reached a stage of professionalization.

**Collaboration with event data.** In the interviews, the informants often reported that their activities with event data were prompted by colleagues working with the data. For example, a program manager improving the user experience might prompt a developer to instrument the software to see how often a feature is used. In the survey, for each reported activity, we asked whether it was prompted by a colleague's activity and, if so, which one (Figure 1). The clearest trend is task delegation, where the same activity is passed from colleague to colleague (diagonal line). Further, troubleshooting is a frequent prompt for all activities, and many activities lead to work on the data pipeline.

## 5. ACTIVITY DESCRIPTIONS

We detail the eight activities, and present them in terms of distance from the data. Through our qualitative interviews, we characterize the nature of each activity. Through the informants' experiences, we offer additional depth to the challenges identified from our follow-up survey (Table 2).

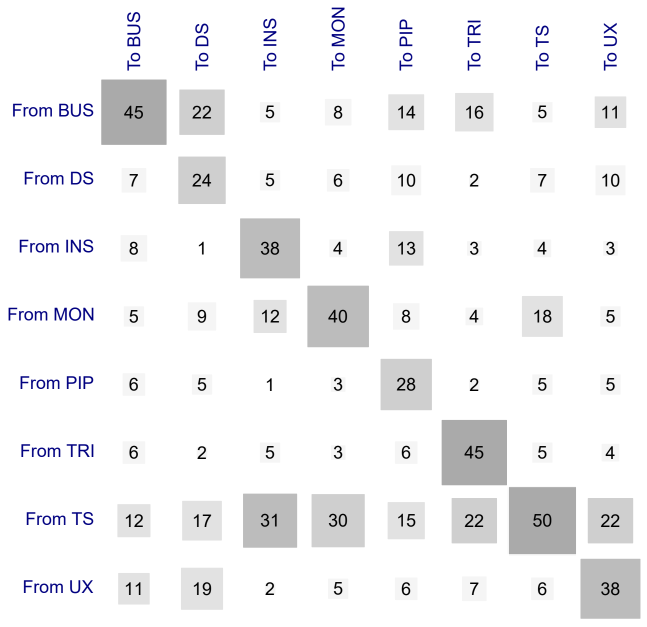


**Figure 1: Collaboration diagram of how activities flow from one activity to another.**

### 5.1 Data Engineering

#### *5.1.1 Engineering the Data Pipeline (PIP)*

This activity involves designing, implementing, and maintaining the infrastructure to collect, store, and query data. This infrastructure acts as a bridge between a product team's customers and its engineering team. It involves both data collection software running on the customers' devices, as well as APIs and tools by which a team's engineers can clean, filter, and aggregate the data. As the scale and velocity of the data increases, assigning this activity to a centralized, expert team has many advantages. As Austin$_{\text{dev}}$ explains, "Whereas in the past, you could maintain your own system, the volume of traffic coming through, the volume of storage required, and then the amount of processing needed makes it difficult to really make sense of any of that data. It's becoming more than individual teams are capable of handling."

Shared pipelines also improve team coordination: "I think having each team own individual telemetry when not looking at the whole, holistic picture can be a little dangerous" (Ulysses$_{\text{pm}}$). One approach to coordination is to standardize data collection. Jon$_{\text{pm}}$ argues, "The [pipeline teams] are trying to get everyone to agree on what goes into a standard log, so at least it's uniform."

Centralizing this activity also means that improvements to the data pipeline benefit many teams, for example, work to improve data quality and curation. Rudy$_{\text{ds}}$ says, "I would say that our number one amount of work effort goes into data quality. That's quality of the data that's being sent up, that's quality of the data that's coming through the system, quality of the data that's cooked, quality of the code to pull the data." Centralizing tool efforts can also solve common problems across many teams. For example, Gary$_{\text{dev}}$'s team is working on the common challenge of combining data from multiple sources: "We have tools that consume these logs and combines them into a single view. You can do things like mesh timelines together, like if two logs are both running at the same time, it can combine the timelines so you can see all the events that happen in order" (Gary$_{\text{dev}}$).

This coordination is a two-way street, and teams often have to deal with limits that centralized data pipelines impose, like restricted sampling rates. Rudy$_{\text{ds}}$ says, "Right now one of my big projects is really about sampling of events. . .We're looking at all events coming from PCs and phones, and [game consoles] — they have already had their own sample rates in their work. It [now] flows through the same telemetry system." Data pipelines similarly impose retention policies. Rudy$_{\text{ds}}$ adds, "We have some retention policies in place that kind of spans all of our data. We haven't updated that based on what we're actually seeing for event flow coming in. So if the event flow is massive, way beyond what we can handle long term, we might change our retention policy."

### *5.1.2 Instrumenting for Logs or Telemetry (INS)*

The activity of instrumenting is "basically print statements which are manually added to the code base" (Ida$_{\text{dev}}$). As Ulysses$_{\text{pm}}$ says, "They essentially will put something in like `Log.AddLine`. That's the most common form of logging that we use for our testing." The format of instrumentation varies wildly in its formality — from unstructured, "manually-crafted" (Ida$_{\text{dev}}$) strings, to semi-structured log events (e.g, having "tags" (Gary$_{\text{dev}}$) in conjunction with custom strings), to structured schemas containing a core set of values, like correlation identifiers (Keith$_{\text{dev}}$).

Instrumenting is the genesis of the event data. The requests for events funnel in from a variety of other activities: from collaborators who improve the user experience, make business decisions, and do data science. "It's generally the developers who are enriching the code" (Brian$_{\text{ds}}$). These developers are either in quality teams ("Given that I'm a PM, I work with folks on the quality to make sure that all our data points are logged." (Anna$_{\text{pm}}$)), or in product groups.

Often, full-time quality teams will take their own initiative and preemptively instrument relevant metrics (Anna$_{\text{pm}}$ and Ida$_{\text{dev}}$). In data-driven environments, such instrumentation is moving towards rule-based approaches, where instrumentation can be added once and then toggled without having to change the code itself. This functionality has enabled data-driven organizations to collect data not just during testing, but long after the product is deployed into retail. As Austin$_{\text{dev}}$ remarks, "What's really great for us is to be able to real-time turn on and off what log stuff you're collecting at a pretty granular level. And to be able to get the performance back when it is turned off is a big thing for us."

Our survey responses indicated that individuals who instrument don't experience significant challenges (Table 2). Certainly, at times adding instrumentation is rather mundane. Jon$_{\text{pm}}$ comments "It's like: I know I have to do this, I see the value in it, but I'd rather go build something shiny and not do that. We were never big into logging until they said [after reorganization], we need to be better with our data, we need to be data-driven." However, the effects of instrumentation are absorbed and felt by other activities, as those collaborators must combine multiple data sources, or spend effort parsing unstructured event data records. Chad$_{\text{dev}}$ explains, "I think the telemetry is being done too piecemeal in the way the data is organized. There's a lot of duplication."

## 5.2 Data Analysis

### *5.2.1 "Doing Data Science" (DS)*

Our name for this activity, "doing data science," is taken directly from the interviews of Jerry$_{\text{ds}}$ and Brian$_{\text{ds}}$. Our interviews revealed two organizational models for data scientists. The first model is a "hub-and-spoke" (Rudy$_{\text{ds}}$) model, in which the scientists act as consultants or partners (the *hub*) for other teams (the *spokes*) who need analysis services. The second model is embedded, in that "each team has their own [data science] group for giving these insights" (Marvin$_{\text{cd}}$). In both organizational models, data scientists help to take abstract questions and turn them into a concrete, actionable process or recommendation. Jerry$_{\text{ds}}$ says, "If I can answer your question in an organized way, then actually I probably can solve the problem. Because I can just run a script. I think this kind of problem that we face every day is really case-by-case.".

To make these recommendations, data scientists perform experiments, such as A/B testing or flighting, or exploratory data analysis (Brian$_{\text{ds}}$). Desmond$_{\text{pm}}$ tells us that "these recommendations can be quite precise and narrow, or they can be quite broad, impacting core scenarios, either at the marketing level or the engineering level. They are always backed with data. Often backed with analysis. And they are also coupled with business impact associated with following the recommendation."

For a data scientist, coordination with other teams or roles is a necessity and a challenge (Figure 2). Jerry$_{\text{ds}}$ argues, "if you get more data then you are more powerful in terms of doing your machine learning. If you get more friends, like on teams, you have more tools to take actions on users." For collaborating teams, their analysis services are essential. As Henry$_{\text{pm}}$ remarks, "data scientists are really domain experts."

Unfortunately, as Brian$_{\text{ds}}$ describes, centralizing the data science team means that "you have customers that greatly outnumber you, who want to ask questions of the data" (Brian$_{\text{ds}}$) and data scientists "get asked to consult on a lot of projects" (Rudy$_{\text{ds}}$). Because of the high number of requests, many data scientist teams have explicit processes to prioritize these demands, in which they might "have a formal form, called an investigation, in which I pose it to them [a question], and then they go out and analyze all these different data that they might have and report back" (Marvin$_{\text{cd}}$).

Data science teams are attempting to offload some of these time sinks by encouraging more "informal", self-service data scientist roles. Brian$_{\text{ds}}$ explains, "Any time that we have a new data set, we identify very quickly when these become interrupters. We rapidly try to move them into the self-service mode. We provide them the queries we're using to get the data. We let them to play with the report and change the parameters they want." An example of this role is the Content Analyst, who are "junior data scientists in some ways, [having] some basic analysis skills. Because they're former editors, they're able to find a lot of insights into what we can change" (Dale$_{\text{cd}}$).

## 5.3 Software Maintenance

### *5.3.1 Troubleshooting Problems (TS)*

This activity involves debugging to diagnose the root cause of an issue, reproducing bugs, and looking for patterns in log data or telemetry to explain execution behavior. Typically, individuals who are troubleshooting have access to the source code, although in some situations, such as operations, they only have the telemetry data. This activity's goal is either solving the problem or re-assigning it to a more appropriate person or team. Ida$_{\text{dev}}$ elaborates, "The way I personally use [logs] is usually for debugging. I get a report of a bug, and now I'm trying to reproduce it on my machine. ...when there is lots of communication with external components and callbacks are involved, that's ...when I enable a whole set of logs on my machine."

The informants were remarkably similar in their clerical approach to working with log data, often using basic text editors ("You just open it in Notepad and browse through it." (Gary$_{\text{dev}}$)). Troubleshooters performed a common set of activities using their tools. (1) *Filtering data.* "Because as you saw, our logging is very verbose, when you go to the feature-specific level, and when you enable it all, it just becomes unbearable" (Ida$_{\text{dev}}$). (2) *Searching data.* "In those cases, you can easily open the log, `CTRL+F`, look for the word unexpected, and that takes you pretty much to where some of the errors have happened" (Gary$_{\text{dev}}$). (3) *Identifying patterns.* "There's so many different data points and right now, figuring things out or seeing patterns is all in people's heads" (Ulysses$_{\text{pm}}$). (4) *Correlating data.* "And this is where the correlation becomes very important. It's not that they are saying 'select this particular character position and place it,' they are saying 'we have this text range, with that ID, with that memory address, and we want to select that text range"' (Ida$_{\text{dev}}$). (5) *Combining data.* "I think being able to take multiple log files and combine them together into a single timeline is very helpful, because it can get very difficult to try and go back and forth between multiple windows" (Gary$_{\text{dev}}$). (6) *Comparing data.* "The other thing that's tough is if you have something that's succeeding and failing, trying to compare two similar runs can be tough" (Chad$_{\text{dev}}$).

### 5.3.2 Monitoring Services (MON)

On the operations side, monitoring is a service-oriented activity to inspect telemetry and logs to determine whether a "service is healthy or not" (Jon$_{\text{pm}}$), or examine "uptime, performance, and reliability" (Caleb$_{\text{ops}}$) and "blips and outages" (Caleb$_{\text{ops}}$). Henry$_{\text{pm}}$ uses a plumbing analogy, "If you look at me or our team, we own pipelines where the water flows. We try to see: Is the water flowing in all places smoothly?" For our informants, monitoring and troubleshooting are distinct activities, though one activity often feeds into the other. A key difference is that monitoring is about the status of services, whereas troubleshooting is about root cause. As Ulysses$_{\text{pm}}$ comments, "A lot of that logging would flow in front of you in real-time. One of the common things that happened there was that you always look for patterns. It wasn't really a root cause of what was going on, but it was essentially: hey things are not going bad yet, but what's the status of the run as it is ongoing? ...it was a lot of mental pattern recognition to say: things are looking good, things are not looking good."

Monitoring is proactive. Prior to the data-driven culture, remarks Ulysses$_{\text{pm}}$, "logging was never considered to be an important thing ...people were reactive." Logging was primarily used to "try and get to the root-cause. Things happened, postmortem, and then we would slowly react to them" (Ulysses$_{\text{pm}}$). Jon$_{\text{pm}}$ provides further insights on this proactive shift in thinking: "It's a shift in a lot of ways. I believe in 'services thinking' for a services company ...You don't have to test as much as you think you do, if you can deploy fast, fix fast."

Telemetry is presented through real-time, interactive interfaces, called dashboards. Unlike reports, which are "typically something pre-canned" (Keith$_{\text{dev}}$), dashboards enable the user to actively explore the data. One essential capability of these dashboards are visualizations that display specific indicators, such as CPU usage. Visualizing data over time, or trending, is particularly important (Austin$_{\text{dev}}$, Jon$_{\text{pm}}$). "Being able to visualize what's happening with performance and what's going on with the system has been valuable" (Austin$_{\text{dev}}$).

One of the challenges in monitoring is combining data (Table 2), which arises because information necessary for monitoring must be aggregated from many different sources. Keith$_{\text{dev}}$ has observed that "metrics are hard to correlate with other things" and Caleb$_{\text{ops}}$ expands on what makes combining data so challenging: "And we have so many dependencies. In the environment we support, there are 16 different dependencies, 16 different breakpoints. All these different stacks, and it can break anywhere. That's the hard part — is getting them a dashboard that looks across that entire ecosystem."

When "blips and outages" (Caleb$_{\text{ops}}$) do happen, key individuals are alerted to a potential anomaly. When an actual alert occurs, this is a natural boundary point for transitioning to the troubleshooting activity. These alerts trigger as they exceed the tolerance of defined thresholds. But before doing so, the individual must "identify with fairly high confidence" (Brian$_{\text{ds}}$) that the event is actually occurring.

Caleb$_{\text{ops}}$ suggests that with the scale of monitoring today, more sophisticated machine learning approaches are essential. Otherwise, we're simply throwing "more bodies at the alerts" (Caleb$_{\text{ops}}$). Keith$_{\text{dev}}$ agrees: "We could be missing other indicators, we just don't know about it. Who has time to dig into it all the time? I'm sure a computer could figure it out."

Monitoring also feeds into other activities besides troubleshooting. From informants from service engineering (a bridge between engineering and operations), we heard that monitoring plays a role in business decisions, such as for capacity planning — "to build up our servers" (Caleb$_{\text{ops}}$). Other professions, like project managers, have become more involved in monitoring for improving the user experience. Ulysses$_{\text{pm}}$ says, "PMs will essentially say, here's the trending in the system. And then they'll surface that to the dev leads, ...in a formal format like a weekly meeting." Jon$_{\text{pm}}$ indicated similar thoughts: "We're relying more on the logging that generates the passive monitoring so that if things break in production we can respond quickly" (Jon$_{\text{pm}}$).

## 5.4 Decision Making

### 5.4.1 Improving the User Experience (UX)

During this activity, event data is consumed in order to understand user behavior, such as learning how users interact with features. For example, Ulysses$_{\text{pm}}$ uses event data to understand "if people are going from my tool and abandoning my tool and going to someone elses' tool" (Ulysses$_{\text{pm}}$), while project managers "get involved [with improving user experience] when they define for a new feature what kind of data they would like to collect for that feature" (Ida$_{\text{dev}}$).

Typically, individuals in this activity do not look at event data directly. Instead, collaborating quality teams produce reports. These reports play a significant role in how individuals in this activity interpret and make decisions about improving the user experience. As Dale$_{\mathrm{cd}}$ indicates, “I only saw the reports, I didn’t actually do the queries.” Gary$_{\mathrm{dev}}$ adds, “As we share out those reports, anyone who can read a table with percentages in it can get an idea as to how the product is working. The audience is all over the place. Usually the raw data logs don’t get looked at much beyond the engineering side of things.”

While they often have to wait on other people (e.g., quality teams and “Sherlockers” (Marvin$_{\mathrm{cd}}$)) for their UX questions, we found that individuals in this activity accept this waiting time, which, “depending on what their queue looks like, takes between a few days to a couple of weeks. I’m okay with that ...they have specialized knowledge, they’re spending the time trying to understand the numbers, and they have expertise in the content field, so they can actually provide me feedback that is very actionable on my part” (Marvin$_{\mathrm{cd}}$).

One challenge we identified in the interviews is that certain UX roles, like designers, don’t have extensive technical experience. This makes using tools difficult (Table 2). Dale$_{\mathrm{cd}}$ says, “There’s an OLAP cube, and I have to re-teach myself how to use it. It’s kinda a pain. [Though] once it’s in Excel, we can do whatever we want.” Austin$_{\mathrm{dev}}$ explains, “We have some more technical designers that are able to get in and play with scripts a little and understand some of that. It would be great if there was a way for them to be able to go in and be able to go in and create those queries themselves. It’s just as simple as writing a SQL query for example, but if you don’t understand SQL you can’t do that. I think that’s where the tools in general are weak.”

A second challenge from our interviews is that self-service is an additional time and resource commitment over their existing responsibilities. As Marvin$_{\mathrm{cd}}$ tells us, “If I feel like I have a space of open time and I’m really inspired, then I’ll go learn it and I’ll surprise my boss, but if I fail then it’s a waste of time. There are so many data sources and so many roles, you have to invest the time or have the resources allocated that specialize in these areas.”

In this transition, others, however, are more optimistic. Dale$_{\mathrm{cd}}$ says “I love playing around with stuff like that. It may not lead to anything, but sometimes you get these interesting insights. I love getting the data because I’m always learning something, and it’s always something surprising.”

### 5.4.2 Triaging Work Items (TRI)

Triaging prioritizes bugs or features and assigns activities to team members. Anna$_{\mathrm{pm}}$ notes how event data sources are used: “I triage a lot of bugs, so it’s important for me to have a general understanding of what to look for in the log files to help route bugs the right way.” Triage can be as much about what work is rejected as accepted: “I tend to be what we call the shield for some of these escalations” (Gary$_{\mathrm{dev}}$).

Reports are also used extensively to triage work items, but our interviewers indicate that for triaging, using and making reports to inform triage decisions is time consuming (Table 2). Henry$_{\mathrm{pm}}$ says, “Defining reports and visualization is extremely difficult. It takes about a week’s time for one report.” He continues, “If I had this report it would have been great. But there’s no tool to easily generate that. It’s just raw data, it’s like looking at a water molecule.” Anna$_{\mathrm{pm}}$ concurs. “We want to use data to empower decision making, but we need the training to make the right decisions and to ask the right questions. People love data, but then nobody knows exactly how to use it, or we don’t trust people to use it properly in addition to the reporting being clear, updated, explaining things in the way that they should be explained so it’s easy to consume.”

Even though some triagers are technical, they tend, as a matter of time constraints, to look at the surface-level details to triage a particular issue. After that, they delegate the item to a more specific person. Gary$_{\mathrm{dev}}$ says: “A lot depends on how busy I am myself versus how many escalations there are going on in parallel. If it’s a slow period I might actually, even though it’s not necessarily my feature area, go through and figure it out myself.”

Another challenge is trusting reports. Anna$_{\mathrm{pm}}$ comments, “One of my greatest challenges, though, is having confidence in the data that’s being collected as well as confidence in my ability as well as my peers’ ability to analyze it. To identify where oh are there weird things that we’re seeing in the data, from the data quality perspective, or are these legitimate events that were captured from the user.”

In turn, lack of confidence and trust in the data results in uncertainly in the triaging process. At times, it’s difficult to be proactive because of this. “When we’re planning, so far, I have found that we’re in super scary reactive mode. So it’s like we found something through problems in logs, and we’re fixing it right now, so it’s not a plan. Because someone has already hit a problem” (Blake$_{\mathrm{pm}}$).

Another significant time effort results from having to make triage decisions based on multiple sources of information. Participants identified correlation as one of the time-consuming challenges, particularly when information is incomplete. Blake$_{\mathrm{pm}}$ comments, “It’s hard because, at least at this point so far, it’s not something that I could glean from processing a log. It really is a thing where I look at the number of bug reports we’ve gotten and make a subjective call of: how much more investigation do we want to do on this particular report?”

### 5.4.3 Making Business Decisions (BUS)

This activity involves making business decisions regarding product planning, marketing strategies, technology investments, customer retention, and product release planning.

Our informants described the iterative nature of business decision making, in particular, how business questions are answered collaboratively. The questions are answered using event data sources by product-level teams or partner data science teams, and these answers are funneled back to leadership as reports, and discussed at meetings (e.g., “ship rooms” (Gary$_{\mathrm{dev}}$)). Desmond$_{\mathrm{pm}}$, explains how this process works, saying, “We developed the questions first. We said: what are the things that we want to know? What hooks do we need to put into the system and what kind of reports do we need to get back out in order to answer those questions? We didn’t just take raw dumps and then try and figure out what we could learn, we were targeting specific learnings.”

Because questions are higher-level, they require combining multiple data sources to provide a recommendation or decision (Table 2). Often, Henry$_{\mathrm{pm}}$ indicates, “Correlation is manual and there are two separate organizations who are looking and working on it, and then nobody knows. And there is a lot of effort.” Desmond$_{\mathrm{pm}}$ says, “There’s no clear way to tie multiple data sources together to drive business decisions.” Ulysses$_{\text{pm}}$ adds, “When you start joining it against multiple data sources, that’s when you get key insights.”

The collaborative nature of answering business questions often require waiting on other people. Desmond$_{\text{pm}}$ describes why multiple rounds are needed to get the necessary information, noting, “An engineer who is trying to build something for a business person who cares about it understands the code but they don’t truly understand the business. The business person truly understands the business need, but they don’t know how to actually go do the implementation. So you always lose something in translation. You also lose something in efficiency, because now you’ve got two people involved and the latency that occurs in that relationship.”

To avoid this coordination, teams are looking towards a self-service model for answering business questions. However, the barrier to self-service is the technical nature of data tools. Blake$_{\text{pm}}$ discusses this friction: “I feel like it’s a pretty heavy weight process to go write the scripts and [batch] jobs that distill the telemetry data that we have and then put it into a dashboard. And if you ask the wrong question, suddenly it was really expensive to go get that indicator and then try and tweak that question a little bit.” Desmond$_{\text{pm}}$ describes similar difficulties with these tools: “If you don’t have enough technical depth, you’re creating an ask for somebody with adequate technical depth, and then you lose context.”

## 6. LIMITATIONS

This case study was conducted at a single software company, hence the results may not generalize to other institutions. One focus on the study is to understand the organizational implications of adopting data-driven decision making, like coordination and the division of tasks across roles. Recruiting participants from a single company allowed us to hear about these issues from diverse perspectives, in a way that recruiting unrelated participants from different companies would not. To mitigate the threat to generalizability, we recruited participants across all the company’s major job roles and its diverse businesses, including devices, games, online services, desktop software, and operating systems.

To mitigate threats to internal validity, particularly researcher bias, we chose participants at random and used grounded theory techniques. We recruited our participants based on random samples of the corporate address book to avoid presuming which roles or teams would have relevant experiences. The invited employees who chose to participate are likely more interested in event data than the general population (self-selection bias). We defined “event data” quite broadly to encourage a wide-range of participation and consistently began our interviews with open-ended questions (e.g. “Tell us about a recent experience in which you worked with event data”) to avoid biasing what the participant shared.

## 7. DISCUSSION

### 7.1 Activity Tensions

In this section, we describe tensions that materialize as a result of the interaction of event data between activities.

**Privacy is cross-cutting.** Privacy encompasses not only customer privacy, but also security and compliance. Privacy policies have a ripple effect throughout all of the activities. For example, Amber$_{\text{pm}}$ comments that “in troubleshooting, we’re not even allowed, as per the terms, to move it from one data center to another, because it’s a different geographical boundary.” She adds that these processes can add additional complexity to activities: “This obfuscating, encrypting, hashing, and then bringing it back and then finish the job, that’s the worst, hated, the most hated thing.”

Compliance has effects on other activities and their ability to work with event data, one of which is the ability to share data between partners. Austin$_{\text{dev}}$ comments, “For a lot of the internal stuff, it proved invaluable at times to look at but our partners absolutely did not access to it.”

**Balancing instrumentation and retention.** Storing data has a cost, although this cost is not always made explicit to other activities. This creates tension between Engineering the Data Pipeline, and activities such as Troubleshooting Problems and Monitoring System Health, which give rise to event data. Jon$_{\text{pm}}$ remarks, “everyone said, just log this and it’s in Cosmos forever, but it’s actually not true.” What’s difficult about the trade-off between instrumentation and the data pipeline is that in order to proactively monitor, one has to be able to speculate about future data that they don’t immediately require in the present.

Furthermore, in organizations that have a centralized data-driven approach, individuals performing instrumentation can unintentionally impact other parts of the organization. Jon$_{\text{pm}}$ comments, “[Instrumentation] is one of those things where you’re getting penalized for everyone else.”

**Making business decisions requires investments in telemetry.** Prior to the data-driven culture, Desmond$_{\text{pm}}$ remarks, “It’s almost always true, in our past, that logging and telemetry took second fiddle to new feature work. If I had limited developer resources, and I could invest in understanding better how the existing code was working, or adding new code that lit up new features, I’m always going to choose new features. Getting telemetry running inside a code base is often neglected for those reasons.” In turn, there’s a cost-benefit tension in the cost of instrumenting the necessary telemetry, and the benefit of the answer for making business decisions.

**Identifying consumers and producers of the event data.** Another tension that occurs between the consumers of event data and the event producers (that is, Instrumenting) is a lack of knowledge of what event data is added to the system, and who is consuming it. Austin$_{\text{dev}}$ notes, “I don’t know it’s that we can’t share. You don’t know who to ask to find out if you can. And it’s kind of hard to find somebody who is authoritative who can say yes.” Because event data is often generated as an independent event, such as from proactive instrumentation, professionals aren’t always aware that the data exists. Ulysses$_{\text{pm}}$ observes, “So a lot of it is just a ton of time wastage. People have to get to what they are looking for, and we don’t show it to them. And that’s a key failure, at least in my opinion.”

### 7.2 Implications for Tool Designers

In this study, all roles across the company, including notably non-engineering roles, report seeking insights from event data. While professionals in non-engineering roles often have domain knowledge and role-specific questions, they may lack the technical skills currently required to analyze event data to answer those questions. Participants reported a variety of coping strategies for these skill gaps, which amount to different divisions of labor. For example, some of our interviewees had a centralized data science team, which take formal requests and issue reports. Other teams had engineers set up dashboards or periodic reports that provide the whole team with pre-determined measures. Other teams relied on form-based, interactive tools that provide answers to a fixed set of queries. The downsides to this approach are the lack of flexibility for dealing with new information needs and the bottleneck of having too few experts. An open research question is the degree to which we can "democratize" data science, that is, allow non-technical team members to analyze data to get high-confidence answers to unanticipated questions.

## 8. CONCLUSIONS

Event data in large organizations, such as Microsoft, act as boundary objects that flow through activities within the organization. Each activity adapts the event data to their requirements, yet the identity of the event data is maintained. In our interviews and surveys, we found that all activities leverage event data.

Several challenges have emerged in Microsoft's transition to a data-driven culture. First, the non-specialization of activities suggests that responsibilities within the organization are fluid, and that these fluid boundaries create tensions between activities. Second, we found that professionals within each activity experience several challenges in performing the activity, for example, coordinating with others to obtain insights, the amount of clerical effort required to work with the event data, and limited or difficult to use tools. As more companies transition to a data-driven culture, we expect that researchers will need to develop novel processes and tools to meet these emerging challenges.

## 9. REFERENCES

[1] J. K. Blomkvist, J. Persson, and J. Åberg. Communication through boundary objects in distributed agile teams. In *CHI '15*, pages 1875–1884, Apr. 2015.

[2] J. A. Carlson. Avoiding traps in member checking. *The Qualitative Report*, 15(5):1102–1113, Sept. 2010.

[3] J. Cito, P. Leitner, T. Fritz, and H. C. Gall. The making of cloud applications: An empirical study on software development for the cloud. In *ESEC/FSE '15*, pages 393–403, Aug. 2015.

[4] M. Curtin and E. Fossey. Appraising the trustworthiness of qualitative studies: Guidelines for occupational therapists. *Australian Occupational Therapy Journal*, 54(2):88–94, June 2007.

[5] D. Fisher, R. DeLine, M. Czerwinski, and S. Drucker. Interactions with big data analytics. *Interactions*, 19(3):50–59, 2012.

[6] Q. Fu, J. Zhu, W. Hu, J.-G. Lou, R. Ding, Q. Lin, D. Zhang, and T. Xie. Where do developers log? An empirical study on logging practices in industry. In *ICSE '14*, pages 24–33, May 2014.

[7] S. Hove and B. Anda. Experiences from conducting semi-structured interviews in empirical software engineering research. In *METRICS '05*, pages 23–32, 2005.

[8] S. Kandel, A. Paepcke, J. M. Hellerstein, and J. Heer. Enterprise data analysis and visualization: An interview study. *Visualization and Computer Graphics*, 18(12):2917–2926, 2012.

[9] M. Kim, T. Zimmermann, R. DeLine, and A. Begel. The emerging role of data scientists on software development teams. Available as Microsoft Tech Report MSR-TR-2015-30, 2015.

[10] C. P. Lee. Boundary negotiating artifacts: Unbinding the routine of boundary objects and embracing chaos in collaborative work. *CSCW '07*, 16(3):307–339, Apr. 2007.

[11] W. G. Lutters and C. B. Seaman. Revealing actual documentation usage in software maintenance through war stories. *Information and Software Technology*, 49(6):576–587, June 2007.

[12] T. Menzies and T. Zimmermann. Software analytics: so what? *IEEE Software*, 30(4):31–37, 2013.

[13] R. Musson, J. Richards, D. Fisher, C. Bird, B. Bussone, and S. Ganguly. Leveraging the crowd: How 48,000 users helped improve Lync performance. *IEEE Software*, 30(4):38–45, July 2013.

[14] S. Nadella. A data culture for everyone. http://blogs.microsoft.com/blog/2014/04/15/a-data-culture-for-everyone/, 2014.

[15] A. Pecchia, M. Cinque, G. Carrozza, and D. Cotroneo. Industry practices and event logging: Assessment of a critical software development process. In *ICSE '15*, pages 169–178, May 2015.

[16] J. Saldaña. *The Coding Manual for Qualitative Researchers*. SAGE Publications, 2009.

[17] J. Sapsed. Postcards from the edge: Local communities, global programs and boundary objects. *Organization Studies*, 25(9):1515–1534, Nov. 2004.

[18] F. L. Schmidt, J. E. Hunter, and A. N. Outerbridge. Impact of job experience and ability on job knowledge, work sample performance, and supervisory ratings of job performance. *Journal of Applied Psychology*, 71(3):432–439, 1986.

[19] W. Shang, M. Nagappan, A. E. Hassan, and Z. M. Jiang. Understanding log lines using development knowledge. In *ICSME '14*, pages 21–30, Sept. 2014.

[20] S. L. Star and J. R. Griesemer. Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science*, 19(3):387–420, Aug. 1989.

[21] A. L. Strauss. *Qualitative Analysis for Social Scientists*. Cambridge University Press, 1987.

[22] D. Yuan, H. Mai, W. Xiong, L. Tan, Y. Zhou, and S. Pasupathy. SherLog: Error diagnosis by connecting clues from run-time logs. In *ACM SIGARCH News*, volume 38, pages 143–154, 2010.

[23] D. Yuan, S. Park, and Y. Zhou. Characterizing logging practices in open-source software. In *ICSE '12*, pages 102–112, 2012.

[24] D. Yuan, J. Zheng, S. Park, Y. Zhou, and S. Savage. Improving software diagnosability via log enhancement. *ACM Transactions on Computer Systems (TOCS)*, 30(1):4, 2012.

[25] D. Zhang, S. Han, Y. Dang, J.-G. Lou, H. Zhang, and T. Xie. Software analytics in practice. *IEEE Software*, 30(5):30–37, 2013.