How to Hackathon: Socio-technical Tradeoffs in Brief, Intensive Collocation

Erik H. Trainer, Arun Kalyanasundaram, Chalalai Chaihirunkarn, James D. Herbsleb
Institute for Software Research
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
{etrainer, arunkaly, cchaihir, jdh}@cs.cmu.edu

ABSTRACT

Hackathons are events where people who are not normally collocated converge for a few days to write code together. Hackathons, it seems, are everywhere. We know that *long-term* collocation helps advance technical work and facilitate enduring interpersonal relationships, but can similar benefits come from brief, hackathon-style collocation? How do participants spend their time preparing, working face-to-face, and following through these brief encounters? Do the activities participants select suggest a tradeoff between the social and technical benefits of collocation? We present results from a multiple-case study that suggest the way that hackathon-style collocation advances technical work varies across technical domain, community structure, and expertise of participants. Building social ties, in contrast, seems relatively constant across hackathons. Results from different hackathon team formation strategies suggest a tradeoff between advancing technical work and building social ties. Our findings have implications for technology support that needs to be in place for hackathons and for understanding the role of brief interludes of collocation in loosely-coupled, geographically distributed work.

Author Keywords

Hackathons; collocation; scientific software; multiple-case study; qualitative methods.

ACM Classification Keywords

H.5.3. [Information interfaces and presentation (e.g., HCI)]: Group and Organization Interfaces – Computer-supported cooperative work.

INTRODUCTION

Hackathons are events where people who are not normally collocated converge for a few days to write code together. Hackathons, seemingly, are everywhere. They are held, as one might expect, by software companies such as Facebook, Yahoo!, Google, and many, many more. But they are also becoming an integral part of the tech educational scene. There were about 40 university-based hackathons in 2014, and 150 are expected in 2015, some with over 1,000 participants [20]. Scientific communities are also jumping on the hackathon bandwagon. Science Hack Day [5] lists 50 hackathons in support of scientific communities around the world. This is only a partial listing, as a web search of *science* and *hackathon* clearly shows.

While collocated teams, distributed teams, and open-source development have all received considerable attention in the CSCW literature, we do not know much about hackathons, these brief bursts of collocated activity that punctuate the software work of diverse groups, organizations, and communities. Research has identified developmental phases teams generally go through before they are able to perform effectively [8,35]. Which elements of these phases do organizers and participants emphasize when preparing for a hackathon? What are the social and technical outcomes of hackathons, and how do they achieve them? How does brief collocation fit into the flow of activities for developers, projects, and users? With a few exceptions (e.g., [7,36]), there is surprisingly little research into what happens during a hackathon, and even less about how it matters to community members' work.

We conducted a multiple-case study [38] of three hackathons. We build on a rich tradition of CSCW research on collocated work, focusing in particular on research showing the benefits of collocation for performing technical work and for fostering coworker familiarity in distributed teams. *Radical collocation* [33], where team members are in nearly continuous close proximity to one another, suggests the focus of hackathons on using the rich affordances of temporary collocation may rapidly advance technical work. In addition, as hackathon participants observe and interact with another, the social aspects of collocation share some elements of a "site visit," which provides fertile ground for developing *situated coworker familiarity* [14], an understanding of their collaborators in relation to themselves and their work, that can result in durable social ties and other positive outcomes.

Hackathons, however, are distinct from the contexts in which radical collocation and situated coworker familiarity have been studied. They are brief, and generally are not

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
CSCW '16, February 27-March 02, 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-3592-8/16/02…$15.00
DOI: http://dx.doi.org/10.1145/2818048.2819946

physically located in a subgroup's regular workspace. We seek to understand the life cycle of a hackathon, from preparation through follow up, and to examine its role in the ongoing work of participants and projects. In the following sections we review related research, describe our study, present our results, and discuss the implications of our findings for technology support and future work.

BACKGROUND

What is the Point of a Hackathon?

Hackathons tend to either aim at software development with specific applications and technology, or at applying technology for a specific purpose, such as social issues and business objectives [3]. For instance, open-source software projects like PyPy, OpenBSD, and Linux put on hackathons to rapidly advance work on specific development issues. National and local government agencies tend to hold hackathons to build technologies that address social issues, such as helping the elderly cope with dementia [11]. Technology companies like Google, Facebook, and Yahoo! put on hackathons to encourage new product innovation, such as the Facebook's *Like* button [16].

Team Formation

There is a temporal flow to the growth and development of a team. Development of trust [37] and "team cognition" [12] are essential for effective teams, and are relatively difficult to develop in online settings, compared to face-to-face. Group development research also shows that over time, groups go through stages of *forming*, *storming*, *norming*, and *performing* [8,35]. In the *forming* stage, individuals attempt to identify the nature of the task and what information is required for it. To get to know one another they exchange personal information. In the *storming* phase, conflict may arise as team members try to establish themselves in relation to other team members and the leader, who may receive challenges from team members. Team members then move to the *norming* phase, where they exchange task information and develop their process and working style. Finally, in the *performing* stage, the team works effectively with minimal emotional interaction, adhering to their established norms.

To make sure that very brief hackathon time can be used efficiently from the start, organizers and participants undergo preparation activities. How much these activities resemble those of *forming*, *storming*, and *norming* is not entirely clear. Because time is compressed, participants may choose to emphasize some preparation activities over others or leave some out entirely, deferring them to the collocated period. Furthermore, it seems very likely that much of the communications related to preparation will happen using information and communication technologies (ICTs), mediums less rich than face-to-face. How and to what extent these activities are completed will likely influence how time at the hackathon is used as well as the outcomes. Accordingly, we ask:

RQ1: What preparatory activities do participants engage in, and how do these preparations influence the hackathon activities and their outcomes?

Benefits of Face-to-Face Interaction

Face-to-face interaction has long been considered an important aspect of collaboration and the richest form of social interaction [28]. Proximity increases perceptions of familiarity [17], which contribute to the development of social ties [24,30]. People are more likely to deceive, be less persuaded by, and initially cooperate less with someone they believe to be far away [2].

Over the past decade, numerous studies of periodic face-to-face meetings in the life of distributed teams suggest that these meetings are a critical part of establishing strong relationships. They help coworkers develop trust and rapport [10,27,29], build social networks [29], and access situated knowledge [32]. In a longitudinal study of three globally distributed teams, Maznevski and Chudoba [25], describe how regular face-to-face meetings create temporal rhythms that enable higher levels of coordination. Eating and drinking together and moments of physical contact (e.g., handshakes, pats on the back) are fundamental ways that distributed workers connect with each other [27].

In contrast, the role of face-to-face meetings for open-source and other online communities has been largely unexplored. The handful of studies that do look at face-to-face interactions (e.g., [7,36]) in open-source software development find that the interactions increase participation in follow-up work and facilitate socialization of new members. To date, research has done little to connect these results back to CSCW theory.

We identify two theoretical frameworks particularly useful in analyzing our data: *radical collocation* and *situated coworker familiarity*. Radical collocation [33] is a strategy where an entire development team is put in one room for the duration of a project in a physical arrangement that resembles many hackathons. The team room is typically outfitted with individual workstations and central worktables where multiple people can sit and work. Nearby the team room are breakout conference rooms where groups can work privately without distractions. Whiteboards run along the walls of team and conference rooms.

Studies of radical collocation find that it speeds up software development work that is normally spread across a building or a campus. The *spatiality* that proximity affords allows team members to easily move between activities, point to visible artifacts, mark them to reflect agreed-upon changes, and observe other participants moment to moment to identify members puzzled or deep in thought. *Overhearing* conversations allows team members to have impromptu meetings and training sessions around the artifacts themselves to address important issues and problems (e.g., [13]). These benefits can lead to significant productivity gains. Teasley et al. [33] found that radically collocated software development teams doubled their productivity compared with the previous company baseline.

The brevity of hackathons poses challenges for taking advantage of the affordances of radical collocation to advance technical work. It takes time for groups to construct shared artifacts and to visibly mark them. There must be some agreement on terminology, awareness of when one's skills are needed, and agreement on norms of interacting in order for groups to adopt the flexible, interactive practices of radical collocation. Hence, we ask:

RQ2: How and under what conditions do the participants use the affordances of working face-to-face to realize the benefits of radical collocation?

While radical collocation focuses on the affordances of the immediate environment and how they facilitate technical work, *situated coworker familiarity* focuses on enduring interpersonal impacts. Situated coworker familiarity is a "multiplex understanding that coworkers have of their counterparts in relation to themselves and their work together" [14:797]. Situated coworker familiarity is established when people visit the sites of their distant colleagues and share a space with their coworkers for an extended period of time, typically several weeks. These site visits evoke two types of activities, *interacting* and *observing*. Interacting involves discussing work and personal information and socializing, which helps people become more familiar with coworker's styles and preferences. Observing others' work and social behavior allows visitors to see the context in which their remote colleagues work, which helps make sense of their behavior. For instance, the authors described how a visitor interpreted a normally remote coworker's abruptness toward him as rude; seeing this coworker interact with all of his local colleagues this way, however, allowed the visitor to realize that the behavior was not personal.

Interacting and observing lead to familiarity with coworkers' work and communication styles, capabilities and interests, personalities, work and social roles, and cultural context [14]. After visitors return home, the familiarity they have built up enables closer relationships, which are characterized by increased responsiveness, more frequent communications, and more personal disclosure and discussion of difficult topics.

With respect to situated coworker familiarity, there are important differences, beyond brevity, between site visits and hackathons. In a site visit, the visitor gets deeper familiarity with hosts as a result of having shared their context. At a hackathon, *everyone* is de-situated from the context of his or her regular day-to-day work, although it seems possible that the participants experience themselves as situated together in a distinct "place" [21]. A "place" comprises people who provide distraction, protect from distraction, and serve as an audience for one's work. The influence of this distinct and possibly unfamiliar environment on interpersonal interactions is not entirely clear. Thus, we ask:

RQ3: How and under what conditions do the participants use the affordances of collocation to realize the benefits of situated coworker familiarity?

Given the relative brevity of hackathon face-to-face encounters, one might expect that much of the technical work will exist only in the form of prototypes or demonstrations at the hackathon's conclusion. Follow-through activities that complete and integrate work products may be particularly important in realizing any lasting benefits. Therefore, we ask:

RQ4: What kinds of follow through work do hackathon participants perform, and how does this work complete or enhance the outcomes?

The activities that facilitate situated familiarity and those that facilitate radically collocated technical work partially overlap. A spontaneous tutorial or help offered on a particular topic, for example, can enhance the work at hand and also build up familiarity. These sets of activities are not identical, however. Familiarity is enhanced by social activities that do not immediately advance the work. Much code is written by individuals, out of the view of others, and not necessarily advancing familiarity. Accordingly, we ask:

RQ5: Does the way activities are selected result in a tradeoff between advancing technical work and building social ties?

METHOD

To answer our research questions, we conducted a multiple-case study [38] of three hackathons applied to scientific software. We chose scientific software as the setting for this study for two reasons. First, we needed to keep the domain constant in order to reduce the variance in experiences that other kinds of hackathons, such as social issue oriented and tech educational oriented, introduce. Second, previous work by others suggests that hackathons may be well suited to address the specialized problems of scientific software, which include learning about available tools outside one's own research lab, understanding the larger scientific community's needs, and providing needed maintenance on software floundering due to the short-term financial support associated with research grants [15,19,26,34].

Several considerations led us to our set of cases. Our first criterion was to pick a hackathon that involved a single community and a hackathon that involved multiple communities to see differences in the ways participants established common ground so that they could hit the ground running during collocation. We expected that when different communities come together, there are likely additional mechanisms needed to learn about each other's expertise, motivations, and ways of thinking about the work in order to establish familiarity on which to build during the hackathon. In addition, it will likely take more time for different communities to construct shared artifacts and understandings of those artifacts, come to agreement on technologies that will be used, and develop competency with these technologies compared to hackathons involving a single community. What are the challenges of linking multiple communities at a hackathon, and how do participants address them in order to realize the benefits of situated coworker familiarity and radical collocation?

We found two hackathons meeting this criterion in the Open Bioinformatics Foundation (OBF)'s Codefest 2014 (referred to hereafter as "OpenBio") and the 2014 NSF DataVis Hackathon for Polar CyberInfrastructure (referred to hereafter as "PolarVis"). Held preceding the OBF's annual Bioinformatics Open Source Conference (BOSC), OpenBio was a single-community, two-day hackathon aimed at giving developers of bioinformatics open-source software libraries such as Biopython [4] and scientific workflow platforms like Galaxy [9] a chance to be fully focused on their projects. Attendance fluctuated, with about 45 participants on the first day, and 35 on the second day. PolarVis was a two-day hackathon aimed at linking polar scientists and data visualization experts to produce novel and high impact prototypes and visualizations. Compared with OpenBio, attendance at PolarVis was steady, with the same 39 participants on the first and second day. Because of the difference in community structure, PolarVis was a *theoretical replication* of OpenBio [38].

We sought a third case that would contrast with our original pair on other dimensions, serving as another theoretical replication. We selected the National Evolutionary Synthesis Center (NESCent) Population Genetics in R Hackathon 2015 (referred to hereafter as "PopGen"), a hackathon that aimed to foster an interoperating ecosystem of tools and resources for population genetics data analysis using the popular R platform. Whereas PolarVis comprised two different communities working on related problems, PopGen comprised a single community. We therefore expected to see fewer mechanisms for developing common ground. Although both OpenBio and PopGen comprised participants from a single community, OpenBio had primarily *developers* while PopGen had *different classes of users*, including end users contributing use cases, end users with some programming experience wanting to learn how to develop reusable *packages* (the unit of code distribution in R), and software developers. We expected this contrast in roles and programming experience to be helpful in understanding how participants divided by expertise rather than community membership might use the affordances of collocation. Would PopGen participants use spatiality primarily to learn about the codebases of the universe of population genetics tools from the lead developers? Would this come at an expense of developing familiarity with other population genetics scientists? Attendance for PopGen was 28 participants, and in contrast to OpenBio and PolarVis, PopGen was five days long.

Data Collection

We collected multiple sources of evidence, including event documentation (e.g., mailing list discussions, agendas, announcements, idea lists, and team progress reports) to understand planning practices, 71 hours of on-site observations (OpenBio=17 hours, PolarVis=17 hours, PopGen=37 hours) to understand event dynamics (e.g., how teams form around tasks), and 23 semi-structured interviews to understand in more detail the interactions we observed and the reasons behind them. We conducted all interviews post-hackathon, except for two PolarVis participants who we interviewed at the event.

At each hackathon we captured photographs of the event space, daily team stand-up reports, work breaks, technical sessions, and team meetings. The organizers of OpenBio and PopGen allowed us to video record participant introductions, stand-up reports, and final demonstrations. Legal and insurance issues associated with the PolarVis event space prohibited all video recording.

In selecting interviewees we aimed for coverage across hackathon teams. For PopGen, we looked across the spectrum of participant roles, aiming to see examples of teaching and learning as well as end user feedback. Our on-site observational notes helped us develop probes around the motivations for concrete interactions, how they happened, and their results. We solicited participants by e-mail and interviewed them using either Skype or Google Hangouts. We interviewed one participant by phone. Interviews typically lasted just under an hour. A professional transcription services firm transcribed all interviews. Fourteen interviews were conducted with developers, four with end users (little to no development experience), four with end user developers (end users with moderate development experience), and one with a manager. Three of the developers were also hackathon organizers. Throughout this paper, we denote quotations from developers with "D," end users with "U," end user developers with "U-D," managers with "M," and organizers with "O," each with a unique number for identification. Table 1 lists ranges of participant IDs for each hackathon.

We created a pre-survey to understand participant expectations ("What would the ideal outcome of this hackathon be to you?"), tasks participants desired to work on ("Please specify one or more tasks you want to accomplish at the hackathon"), and preparation for those tasks ("What preparation did you do for the above tasks? Select all that apply. [list]").

Hackathon	Participant IDs
OpenBio	1-7
PolarVis	8-15
PopGen	16-23

Table 1. The range of participant IDs for each hackathon.

One week before each hackathon, the organizers e-mailed a link to our survey to all registered participants.

We created a post-survey to assess if, how, and why outcomes did or did not match expectations. The survey consisted of open-ended questions and questions on a 5-point Likert scale. It asked about participants' satisfaction with their teams' work ("To what extent were you satisfied or dissatisfied with the work completed in your team?"), reasons for this ("What were the reasons for the extent to which you were satisfied or dissatisfied with the work completed in your team?"), perceived outcomes ("In your opinion, what were your most important outcomes of the event?"), and if outcomes matched expectations ("Think about what your ideal outcome coming into the event was. To what extent was this outcome achieved?"). Some participants completed a paper version of the post-survey on the last day of the hackathon; others chose to complete an online version. Response rates for OpenBio, PolarVis, and PopGen were 68%, 100%, and 75% respectively.

Finally, we obtained work artifacts (e.g., presentation slides, committed source-code changes) throughout each hackathon in order to triangulate on our qualitative data and compare outputs from each hackathon.

Data Analysis

We applied qualitative analysis techniques described by Corbin and Strauss [6] to our interview transcripts, observational notes, and event documentation. We first imported these materials into the Dedoose qualitative data analysis software [31]. Three of the authors independently conducted open coding on the text about activities before, during, and after each hackathon, differences among them, and hackathon outputs.

In the next phase of analysis we wrote, shared, and discussed descriptive memos about emerging themes in the data. We used the video recordings to corroborate and augment our observations. We met weekly to unify, refine, and collapse codes where there was commonality, using themes from our memos as support. We applied the resulting set of codes to the remaining data, adding codes when necessary and continuing until theoretical saturation.

RESULTS

RQ1: What preparatory activities do participants engage in, and how do these preparations influence the hackathon activities and their outcomes?

In some ways, preparation activities clearly resembled those of the stages of group formation, particularly establishing relationships during *forming*. PolarVis and PopGen participants interacted with one another using GitHub's issue tracker, a tool usually used by software developers to track bugs in a software project. At PolarVis, for example, domain scientists created an issue describing a research problem, and a use case describing how technology might be used to solve it, and any relevant data (e.g., U10). Software developers replied to the original post, often asking scientists to clarify aspects of the problem (e.g., how the data is formatted, what other libraries the tool might be used with), and then code up a prototype addressing parts of the problem. D11 summed up the benefit of these activities:

"*So that GitHub pre-meeting activity was helpful to me to orient, learn the problems that are interesting, to learn some of the profiles of the researchers. 'Ah, this is a very visionary person who wants to do this. This is somebody who's providing data specific to this community'" (D11).*

Some hackathon organizers also encouraged participants to introduce themselves on a mailing list created for the event (O16). We found evidence that this helped participants identify others with shared interests (e.g., U-D22, U23). For instance, U-D22 told us: *"What [another participant] had said sounded quite similar to the kind of problems that I work on"* (U-D22). Realizing this, she e-mailed the other participant prior to the hackathon and they ended up working together on the same team throughout the event.

In other ways, hackathon preparation seemed to re-arrange the early stages of group development. Participants collectively created tasks *before* forming teams, rather than the other way around as the *forming* stage suggests. The exception was OpenBio, in which tasks came from existing teams' product roadmaps and issues that users had posted in the issue tracker. PolarVis and PopGen participants, in contrast, generated tasks by creating an issue in GitHub as described previously. Participants from multiple disciplines were involved here asking for elaboration, providing required expertise, suggesting improvements, and pointing to existing solutions that might facilitate completion of the tasks. Because these participants were starting with basically no familiarity with one another, they had to learn about each other's interests and skills in order to define a practical set of tasks to work on. This was a pre-requisite to forming teams around tasks.

With respect to tools and data, hackathon preparation also shifted some aspects of *norming* ahead of the *forming* stage. Brainstorming tasks using GitHub's issue tracker for instance, familiarized participants with appropriate ways to report bugs (albeit indirectly); reporting bugs is an important part of a software development process, and tools supporting it need to be in place before technical work can proceed. Participants also used GitHub's wiki functionality to create a list of software that should be installed prior to the hackathon (O12, D18, U20).

The lack of familiarity before a hackathon may make it hard to form appropriate norms. For example, organizers may assume a base familiarity with the tools that may not hold. Software developers at PolarVis, for instance, suspected that some domain scientists did not propose ideas because they could not figure out how to use GitHub (U9, O12). One domain scientist approached O12 at the beginning of the hackathon and started talking about his goal to find up to date documentation for the locations of shipping vessels in polar regions. O12 was surprised that the participant had not proposed this idea on the issue tracker, and it became apparent to him in their discussion that the participant did not know how. PopGen organizers saw the software development features of GitHub as vital to hackathon work, so much so that they offered a tutorial on GitHub at the beginning of the hackathon.

RQ2: How and under what conditions do the participants use the affordances of working face-to-face to realize the benefits of radical collocation?

The hackathon spaces of our participants very much resembled those of radically collocated teams. Each venue was configured to seat all participants in a single room, and to accommodate multiple teams. According to survey responses, the average size of a team in OpenBio was 4 (min=1, max=8), in PolarVis it was 7 (min=2, max=14), and in PopGen it was 6 (min=4, max=8). There were breakout rooms available to teams who wanted to have conversations without distractions from the main room.

On the one hand, participants used the affordances of radical collocation in ways we would expect from theory [23,33]. For instance, hackathon participants gathered around whiteboards to sketch out ideas, make architectural and design decisions, and reflect on alternatives. Overhearing issues raised by others allowed some participants to hold impromptu training sessions where people could move in and out depending on their interest and expertise in the topic (U-D17, D18, U-D19, U-D22).

On the other hand, because of the variety in roles and levels of expertise of hackathon participants, we saw affordances used for purposes seldom reported in the literature: eliciting end user feedback and education and training. At OpenBio and PopGen, for instance, we observed a "generative" form of requirements gathering. Some developers at OpenBio would implement a series of new features, walk over to users to try them out, and ask those same users approach them throughout the event to address issues and bugs they encountered in use (D3, D5). We observed more of this at PopGen, likely due to differences in how tasks were identified before the hackathon. Because tasks were generally more open-ended, e.g., improving interoperability as opposed to fixing specific bugs, additional clarifications of the use cases and design inputs were needed. The pool of end users on hand was also larger. End users working with developers would periodically approach other teams to clarify use cases or needs (U20, D21). In team discussions about design of the tools, end users and developers realized that certain use cases were unclear (e.g., what format the data is in when the tool reads it). Team members decided that while developers wrote code, end users should initiate these conversations with other teams. Other teams expected end users to approach them with such questions because the teams trying to clarify needs would announce that they needed help during their daily progress reports to everyone.

Spatiality and overhearing were crucial for training sessions directed at scientists who were not familiar with certain tools. One type of training we observed only at PopGen was what participants referred to as a "bootcamp," an interactive tutorial run by developers to get end users up to speed on a particular technology or codebase used during the hackathon. We believe this was related to the multiple classes of participants that were unique to PopGen. Participants sat around the tutorial leader with their laptops and the leader used a projector to project his or her screen. Participants followed along on their computers. Anyone with questions would shout out, and either the leader or someone else with experience would answer the question, sometimes coming around to look at the asker's screen to examine the issue. However, there were very few volunteers compared to the number of participants and hence it was difficult to monitor and address the issues faced by all participants. In addition, some participants were not comfortable asking for help.

Participants' working styles changed during moments requiring high coordination. The learning curve associated with GitHub's workflow prevented many participants from using it early on to share code. As a result, some people e-mailed their code to their team members or used file sharing tools such as Google Drive. Eventually, however, in order to present a working prototype, code from team members needed to be integrated. Some participants would e-mail their code to one team member who would take the responsibility for uploading everything to the shared repository. Others would use the GitHub workflow to push their contributions to the shared repository. However, if and when something went wrong (e.g., incorrect output, failure to compile), there was very little time to fix the issues. At PopGen, some teams used continuous integration tools that frequently integrate people's new or changed code with the team's repository to avoid conflicts and build failures that result due to making changes in isolation over time. However, setting up these tools was time consuming; one participant said he spent more than a day on this task.

RQ3: How and under what conditions do the participants use the affordances of collocation to realize the benefits of situated coworker familiarity?

As we would expect, co-presence facilitated interacting and observing work. During tool demonstrations, for instance, developers were able to learn much about end users' needs. At PolarVis for instance, while demonstrating his metadata search tool for polar science datasets and having polar scientists try it out, O12 learned:

"[Polar scientists] want to be able to search on [a single metadata attribute on a file] and they want to be able to say like give me all the files from this specific data set or this slice of the dataset…it's not like top level, it's not explicit, it's very implicit within the dataset" (O12).

Socializing during coffee breaks, meals, and bus rides to the hackathon venue from the hotel helped participants learn about each other's interests, their approaches to shared intellectual concerns, and reflect on opportunities for collaboration. These discussions led to collaboration plans for writing grant proposals (U9, D13), working on manuscripts (D15, U20, D22) and collaborating on source-code projects outside the scope of the hackathon (D2, D7, D18, U20, D21).

Watching others code allowed participants to gauge each other's expertise and understand the programming conventions and practices of experienced programmers (D7, U10, U-D17, U-D22, U23). To learn how to use particular frameworks and data structures, participants would go over to team members who were using those frameworks to code and watch over their shoulder as they were coding. The more experienced team members would say what they were doing and why they were doing it. This greatly helped watchers, who would then know where to go for help later in the hackathon.

Hackathons regularly facilitate learning about behaviors not often explored in site visits, such as how people work under pressure. In the hours before final demonstrations of the work products, they worked to solve errors together such as missing source-code dependencies or overwriting each other's changes in the code repository. This work allowed participants to understand how their team members reacted to problems along the way, and the shared experience helped them develop stronger connections (D2, D5, D7, D18). Participants told us that this intense collaboration lowered the barrier to future collaboration and helped to enable more communications (D5, D6, U9, D13, D18, U-D22). D5 said she and her new collaborator *"understand each other's personalities and perspectives and what motivates us, and we can drop each other notes"* (D5).

In some ways the hackathons provided a shared "place" [21]. D6 described how OpenBio served as a place to establish and explore common issues in the bioinformatics community. She told us:

"…there was an introduction in the morning where people got to say who they are and what they were interested in and I mentioned that I was interested in learning a little bit about [platform name]…and whether there were any opportunities to cooperate or interoperate. And so a little bit later that morning, [developer name] from [platform name] came over, and introduced himself, and started the conversation about what is [D6's tool]? What is [platform name]? What can we do to reduce duplicated work in the open-source community?...so that conversation continued and we were also joined by [D1]…and also [D5]… and so we had a circle discussion in the kitchen for a couple of hours…So basically that discussion continued all day into something more serious where we decided maybe this is something that we could really work on together" (D6).

RQ4: What kinds of follow through work do hackathon participants perform, and how does this work complete or enhance the outcomes?

Unlike radically collocated teams who spend months together surrounded by physical project artifacts and their colleagues, hackathon participants quickly leave these artifacts behind, left only with what they have captured digitally. Moreover, many tasks are incomplete when the hackathon ends. To the extent that finishing hackathon tasks is a priority, participants generally want to preserve as much context and meaning surrounding the people and artifacts as possible.

Much of the needed work to complete and integrate tasks was recorded during the hackathon but in different locations: software issue trackers, personal notebooks, shared wikis, and file sharing folders. For instance, U23 described his team's process as follows: *"So, some of that kind of stuff was shunted in [GitHub]. And then other general files and things of certain kinds. Some of the datasets we've used—which are larger, don't really share that well on GitHub—those kind of little file-size limitations. So, we shared those via the Google Drive"* (U23). Participants also took pictures of content on flip charts and whiteboards. The ease with which participants could retrieve this work later is an open question. For instance, although having told us that he received valuable feedback on his tool, D15 had noticeable difficulty finding any specific item of advice in his notebook.

Participants recognized the need for continued coordination beyond the hackathon. OpenBio and PopGen participants scheduled follow-up teleconferences with their team members (D1, D5, D6, U-D17, U20, D21), and arranged to meet some of them at other hackathons (U20, D21). Some participants created online discussion groups. Most notably, as of this writing, a working group that emerged from OpenBio is still meeting every two weeks via Google Hangouts (D1, D5, D6). To coordinate work and make decisions they use a mailing list. Several weeks after PopGen, participants used a mailing list to discuss venues for publications describing each team's work products. As of this writing, six months after the hackathon, participants have submitted seven article proposals for publication in a special issue of a journal.

Beyond technical work, we have some evidence suggesting efforts to strengthen interpersonal relationships established at the hackathons. Multiple developers told us how they would likely see users they met at the hackathon at future conferences (D2, D7, O12, D14, D18). After PopGen, participants created a private Facebook group, which more than fifty percent of the participants joined. We noticed participants sharing photos and discussion of personal topics, such as going out for a drink or buying a laptop.

RQ5: Does the way activities are selected result in a tradeoff between advancing technical work and building social ties?

We did find evidence suggesting a tradeoff between advancing technical work and building social ties. The clearest example of this was from the way teams formed during the hackathon. Our observations revealed three distinct team formation strategies. In the *open shepherding* style of OpenBio, most participants came to the hackathon already associated with a project (and therefore a team) since OpenBio's objective was to give these developers focused time on their projects. As a result, most participants by default sat with their usually remote colleagues. There were, however, "free agents," attendees not associated with these projects. During individual introductions, the event organizer suggested matches between free agents and existing teams and teams with each other.

In contrast, PolarVis and PopGen used project pitches, short presentations made by participants to everyone in attendance describing ideas intended for wider adoption. Most were based on ideas discussed in the *preparation* stage. After pitches there was time for participants to ask questions, discuss the projects, and sign up for them. In the *selection by organizer* style of PolarVis, participants indicated their interest in ideas by writing their names on flip charts (one flip chart per idea). The organizers selected a few ideas with high interest to work on first. Other high interest ideas were reserved for later, according to the organizers, in order to balance between ideas that had lower interest. On the second day, participants were encouraged to work on different ideas to disperse participants' enthusiasm and energy across ideas (O8). Periodically the organizers walked around and determined which teams were "complete" and which needed more time. When teams were complete, new teams formed around remaining ideas.

In the *selection by attraction* style of PopGen, ideas that people got behind were de facto selected. Participants wrote down ideas they thought would be interesting, one idea per sheet. Participants then discussed their ideas with others sitting at their table, and each table was asked to pitch the most important idea. The organizer wrote this idea down on a chart and attached the relevant post it-notes. Each table used different color notes. This was repeated in round robin fashion. If ideas from other tables were similar, the post-its were attached to the same chart. Volunteers were then asked to stand next to the flip charts, and everyone else was free to wander around the room, discussing pitches, offering suggestions, and deciding how to fit in. In contrast to the *selection by organizer* style, teams in the *selection by attraction* style stayed together for the whole hackathon.

Table 2 summarizes the expected and actual outputs of each hackathon, as identified in our surveys. One can see, for instance, that OpenBio and PopGen teams were more successful than PolarVis teams in achieving their technical objectives. To determine the level of technical progress we extracted source-code commits made to all repositories represented at each hackathon two weeks before, during, and two weeks after. Figure 1 shows that compared with OpenBio and PopGen teams, PolarVis teams made few commits to software repositories. PolarVis teams were also less satisfied with their technical output. Only 66% (21/32) of PolarVis participants were satisfied or very satisfied with what was achieved in their team. In contrast, 81% (25/31) of OpenBio and 86% (18/21) of PopGen participants were satisfied or very satisfied with technical work achieved in their team. These observed differences are not statistically significant (Fisher Exact Test p=.24) so they should be taken only as suggestive, i.e., as descriptive of our sample.

Interviews and open-ended survey responses revealed that despite the fact that PolarVis participants were able to develop a base familiarity with each other during *preparation*, they had not been clear about how tasks could be able to combine polar science and development:

"[Participants] weren't able to really be able to see what the real contribution from the other side would be. You know might have been better sessions had been cosponsored or something like that… I don't know how long it takes for a developer to do particular tasks just like they probably don't know how long it would take me to do particular things" (U9).

As such polar scientists and data visualization developers were uncertain how they could concretely contribute to the tasks that were selected (U9, D13, D14).

	Expected Outputs	Actual Outputs
OpenBio	• Bug fixes • New tool features • Improved documentation	• Bug fixes • New tool features • Improved documentation • Workflow Platform Working Group
PolarVis	• New prototypes • Visualization designs • New collaborations • Proposals	• Visualization mock-ups • Bug fixes • New collaborations
PopGen	• Community website • Improved documentation • New R packages • Newly interoperating R packages • Publications	• Community website • Improved documentation • New R packages • Newly interoperating R packages • Publications • Facebook group

Table 2. Expected and actual outputs of each hackathon.

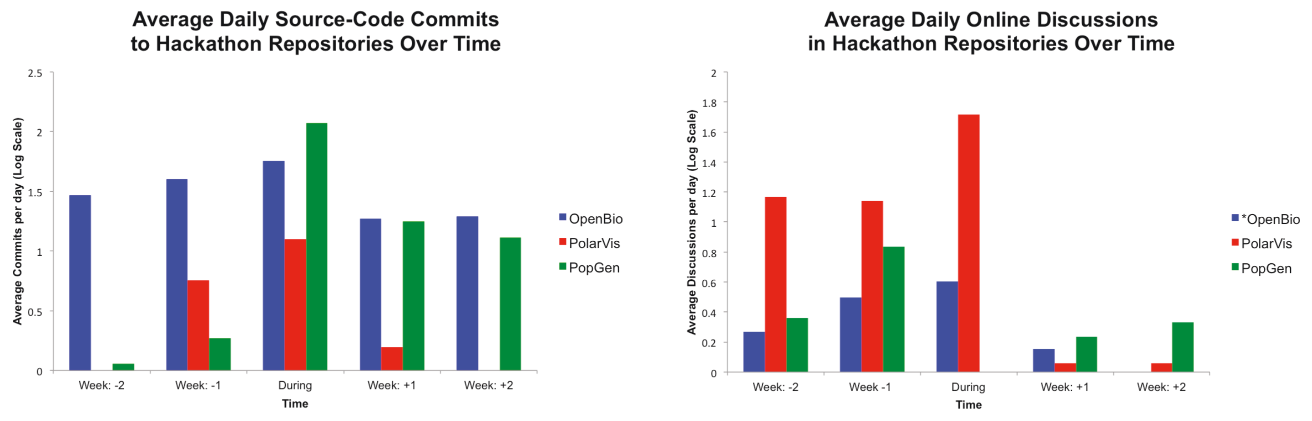

Figure 1. Average number of source-code commits (left) and discussions (right) per day two weeks before, during, and after each hackathon shown on a log scale to heighten the visibility of differences at the bottom of the range. For PolarVis and PopGen we show unique comments posted to the GitHub issue tracker. Because OpenBio participants used a Google Document, we show unique edits to that document extracted from the revision history.

Participants joined teams based primarily on their own interests rather than where their expertise was needed. This led to relatively homogenous teams. Following the *selection by organizer* style, when some teams dissolved mid-day continuing teams had to rehash previous discussions for newcomers (U9), leaving little time to take ideas from concept to realization.

The flip side, however, was that the prolonged discussions that PolarVis teams had during preparation and face-to-face led to collaboration plans, which strengthened existing social ties. High turnover in teams at the event exposed more people to different ideas. Of the 20 participants at PolarVis who had worked together previously, 90% (18/20) described their relationships with others as "much better" because of the hackathon. This was higher than OpenBio, where of the 23 participants who had worked together previously, 57% (13/23) responded "much better." Only nine participants from PopGen had worked together previously, but all nine described their relationships as "much better." We think this may be due to the longer duration of PopGen, which roughly matches the minimum amount of time for the benefits of situated coworker familiarity to accrue [14].

At OpenBio, open-source software teams had mostly identified their tasks ahead of time, writing them down in a shared document but having little public discussion. Because participants had clear goals and expertise, they were able to make rapid progress on their technical work. However, not including domain scientists in brainstorming discussions or at the event mean that there were fewer opportunities to build new social ties. Although there was more training than in OpenBio, PopGen participants were also able to make quite a bit of technical progress. Watching others code resulted in participants learning about programming conventions and practices needed for their work without burdening developers. The longer duration may have also offset losses in productivity due to experts spending time mentoring less experienced programmers.

DISCUSSION

Radical collocation [33] and situated coworker familiarity [14] served as important theoretical background for understanding the potential benefits of hackathons, but there are key differences between hackathons and the contexts in which these theories have been traditionally applied. These differences raise questions about the practices and tool support that hackathons groups need leading up to and winding down from a hackathon in order to effectively start and complete work, and form and cement social ties.

Work

In essence, radical collocation explains how the affordances of the immediate environment facilitate technical work. It takes time, however, for teams to develop trust, shared norms for interacting, and an understanding of work artifacts so that they can use these affordances effectively. Research has shown that teams develop over time in a series of stages.

This paper contributes an understanding of how a very narrow slice of collocation fits into the developmental stages of teams. A hackathon compresses and as such forces careful attention to *forming* and *norming*. *Forming* activities, such as learning about the research interests and goals of other attendees and familiarizing one's self with technologies and datasets happened by way of ICTs. Establishing norms related to tools and datasets critical to the work happens to a degree before teams are formed, but because not everyone will be familiar with them, tutorials may be needed during collocation. Given that attendees' day-to-day responsibilities left very little free time, however, organizers we spoke to did not consider finalizing tasks and teams beforehand to be an option. This shifts aspects of *forming*, *storming*, and *norming* to the collocated period, which reduced the time available to take advantage of the affordances of radical collocation. Future research could investigate and evaluate ways of shifting these activities ahead of collocation, such as scheduling a shared block of time for task brainstorming and prioritization.

Studies of radical collocation [23,33] often take a perspective of the work as a complete "product." But a hackathon ends with the work in multiple, incomplete forms such as mock-up drawings and prototypes. The real benefits of radical collocation come from being able to observe, overhear, and have impromptu training. How then, do participants effectively complete and integrate work that is so dependent on co-presence? Capturing artifacts and context in a form suitable for subsequent distributed work is an important open problem.

Social Ties

Situated coworker familiarity explains how visiting the site of one's coworkers leads to understanding coworkers in relation to one's self and the work together. While our participants acquired familiarity with others' research interests and personalities, we did not find much evidence that they gained insight into an existing context or organizational culture that might explain their collaborators' working styles and behaviors. Rather participants seemed to share a new "place" [21] that provided a forum not only to work on tasks established prior to the coding period of the hackathon, but to define new ones during it.

On the one hand, the value "place" provides is ephemeral; it may or may not boost productivity, and only for so long before it is time to move on [21]. On the other, "site visits" build situated coworker familiarity and can lead to enduring interpersonal relationships [14]. Where do hackathons fit? If one of the benefits of a hackathon is building community, there has to be some non-ephemeral impact on social ties. We found that participants sometimes collaborated with one another beyond the contexts of their hackathon projects. Under what conditions and to what extent these collaborations occur are open questions for future work. As a next step, surveys could be administered to participants weeks or months after a hackathon. Collecting archival data from source-code repositories and mailing lists, such as number of contributions and types of contributions, and linking that data to people would allow researchers to construct social networks representing the social structure of a hackathon. Further data collection could look outside a hackathon at the full set of people who made contributions to projects, revealing connections among hackathon participants, external developers, and end users.

Future work in this area might also focus on practices and technologies for hackathon participants who wish to continue deepening familiarity after a hackathon. For instance, future hackathon participants might benefit by borrowing an idea from participants of PopGen, who created a shared group on a social networking site. These groups permit repeated exposure, by seeing photos and information about members and their recent activities, and self-disclosure, both of which have been shown to strengthen social ties [18:91].

Factors Influencing Usage of Radical Collocation

The stories that our case study tells suggest that the way hackathon-style collocation is used to advance technical work varies across technical domain, community structure, and expertise of participants. While OpenBio participants used collocation to advance work by spending focused time on their projects, PopGen participants filled gaps in expertise among the different roles by gathering requirements and clarifying use cases, as well as providing training needed to advance the technical work. PolarVis participants did not make much technical progress due to not having articulated upfront the contributions needed (e.g., research problems, software development languages, tools) from each community for each task.

Participants at all three hackathons, however, used collocation in similar ways to build familiarity with their fellow attendees. Watching others code enabled participants to familiarize themselves with coding practices. Socializing in a hackathon "place" allowed them to identify common interests. Pressure resulting from deadlines to integrate and complete work products allowed them to learn about each other's personalities. Our findings, for instance, that OpenBio participants created a new working group, PolarVis participants identified potential collaborators on grant proposals, and PopGen participants created a Facebook group and used it to share personal information are evidence that hackathons lead to increased familiarity among participants.

Implications for Design

Brevity of the face-to-face contact in a hackathon places a huge premium on the preparation and follow-through phases. Placing emphasis on the preparation phase helps ensure that the hackathon time can be used efficiently right from the start, and the follow-through phase helps complete work that is started, but very often not finished, at the hackathon. Tools that support preparation and bring the results into the hackathon in a usable form, and tools that capture progress at the hackathon so that incomplete artifacts can be worked on in a distributed way after the hackathon are very helpful.

Social coding environments perform many of the needed functions for preparation, supporting technical work and familiarization, both when the participants are distributed and when they are collocated. For instance, collaborative document editors and wikis in these environments made it easy for participants to create and share lists of software that everyone may not have installed, but that many if not most people will need to use during the hackathon, and understand each other's interests, needs, and skills.

However, they fall short in a couple of ways. First, radical collocation relies on interpersonal relationships where people feel free to ask and offer help, and to work openly in ways others can observe. While this did happen to an extent, we observed that some participants were not comfortable asking for help. The development of trust takes time, but can be facilitated with effective practices even while teams are distributed [1]. These practices and tools could be adapted for the preparation phase of hackathons. A finding of interest is that people tend to trust and want to work with others who share their emotions and ideals. One could imagine, for example, a pre-hackathon exercise where participants make and share short videos talking about their own interests and goals for the hackathon.

The second way in which social coding tools sometimes fell short is that not all participants were familiar with them, nor did they have the incentive to become familiar with them if they did not plan to use them beyond the hackathon. Domain scientists in PolarVis were inadvertently excluded from most of the preparation, which happened on GitHub, a tool with which they were not familiar and did not wish to learn. Less specialized tools seem particularly important when mixing groups that do not share a common tool set.

Tools that support the transition from a face-to-face setting to remote collaboration are useful to continue incomplete work after a hackathon. For instance, participants using whiteboards commonly took pictures of the sketches and later posted them to their team's wiki page. Unfortunately, these pictures cannot be easily evolved after a hackathon. First, whiteboard sketches often do not capture context (e.g., who drew what, their intended meaning). Second, people sketching at whiteboards often do not use standard (e.g., UML) notation that could be used to import the sketches into recognizable digital representations for future use. Digital whiteboards with features that preserve context, like allowing users to tag sketches with descriptions, associate people with what they drew, and create text lists summarizing the work would be very helpful. Such tools could also save the sketches, which do not have to conform to a particular standard, in a digital format, allowing them to be collaboratively edited in real time. Tools like Calico [22] provide features that resemble these.

Such tools often assume, however, that the purpose of sketching is design. We also observed the whiteboard as a teaching tool: developers teaching other developers about the structure of a codebase or explaining how an algorithm works. A feature that could potentially be useful for real time whiteboard sessions would be to allow the people who are not sketching to see the board from the perspective of the sketcher, with synchronous audio or text communication. This could give the viewer insight into how experienced developers think and allow them to ask questions and receive answers in a way that better mimics the flexible ways of interacting at a hackathon.

Study Limitations

We conducted a multiple-case study of hackathons *in situ*, and used interviews and surveys to collect data on hackathon activities and outcomes from participants. We strove to devise a sensible replication strategy, and collected multiple sources of data to triangulate on our observations. Our interviewees were not limited to software developers, but ranged from very technical jobs to end users of the software. They also spanned communities and disciplines. This inclusion is not typical of studies of face-to-face meetings in open-source software development.

All three cases focused on scientific communities, however, which are special in several ways. Scientists likely have a shared view about the importance of science and the value of building up networks of collaborators. Hackathon participants for OpenBio and those for PopGen were from the same scientific domain, meaning they shared scientific vocabulary, intellectual concerns, and awareness of methods. Software plays a secondary, yet necessary, role for many scientists, which may lead to an enhanced willingness to share tools and development practices. Participants at a company-sponsored hackathon, in contrast, may just want to get familiar with a suite of tools. They therefore may not spend so much time learning about each other's objectives and working styles. Hackathons in support of social objectives, on the other hand, may share many advantages of common knowledge and goals that communities of scientists have as hackathon participants.

CONCLUSION

This study provides insight into the hackathon phenomenon, a modern day form of brief collocation that is quite different from most collocation settings studied in existing CSCW research. We extend theories of radical collocation and situated familiarity to apply to this novel setting, and contribute a rich description of hackathon activities from preparation, to the hackathon itself, to the follow through period. Comparing observations across our cases reveals that technical domain, community type, and expertise shape how participants use face-to-face interactions to advance technical work. Building familiarity, however, is relatively constant across different participant compositions and hackathon activities. Our hope is that in addition to informing future empirical studies, our results bring attention to the hackathon model in CSCW and raise the level of discussion about planning and conducting successful engagements.

ACKNOWLEDGMENTS

This research was supported in part by National Science Foundation awards 1064209, 1111750, 0943168, and 1322278, the Alfred P. Sloan Foundation, the National Evolutionary Synthesis Center (NESCent), NESCent's National Science Foundation award EF-0905606, and the Google Open Source Programs Office. We thank Chris Mattmann, Hilmar Lapp, Arlin Stoltzfus, Lewis John Mcgibbney, Saskia van Manen, Brad Chapman, and Michael Heuer, hackathon organizers who provided vital support for and feedback on this study. Thanks also to our participants, and to our anonymous reviewers for their feedback on earlier drafts of this paper.

REFERENCES

1. Ban Al-Ani, Matthew J. Bietz, Yi Wang, et al. 2013. Globally Distributed System Developers: Their Trust Expectations and Processes. *Proceedings of the ACM Conference on Computer-Supported Cooperative Work & Social Computing*, ACM Press, 563–573. http://doi.org/10.1145/2441776.2441840
2. Erin Bradner and Gloria Mark. 2002. Why Distance Matters: Effects on Cooperation, Persuasion and Deception. *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, ACM Press, 226–235. http://doi.org/10.1145/587078.587110
3. Gerard Briscoe and Catherine Mulligan. 2014. *Digital Innovation: The Hackathon Phenomenon*. Retrieved August 4, 2014 from http://www.creativeworkslondon.org.uk/wp-content/uploads/2013/11/Digital-Innovation-The-Hackathon-Phenomenon1.pdf
4. Peter J.A. Cock, Tiago Antao, Jeffrey T. Chang, et al. 2009. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics (Oxford, England)* 25, 11, 1422–3. http://doi.org/10.1093/bioinformatics/btp163
5. Science Hack Day Community. Science Hack Day. Retrieved July 25, 2015 from http://sciencehackday.org
6. Juliet Corbin and Anselm Strauss. 2014. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, Inc., Thousand Oaks, CA.
7. Kevin Crowston, James Howison, Chengetai Masango, and U. Yeliz Eseryel. 2007. The Role of Face-to-Face Meetings in Technology-Supported Self-Organizing Distributed Teams. *IEEE Transactions on Professional Communication* 50, 3, 185–203. http://doi.org/10.1109/TPC.2007.902654
8. Donald B. Egolf and Sondra L. Chester. 2013. *Forming Storming Norming Performing: Successful Communication in Groups and Teams*. iUniverse, Bloomington, IN.
9. Jeremy Goecks, Anton Nekrutenko, and James Taylor. 2010. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology* 11, 8, R86. http://doi.org/10.1186/gb-2010-11-8-r86
10. Rebecca E. Grinter, James D. Herbsleb, and Dewayne E. Perry. 1999. The Geography of Coordination: Dealing with Distance in R&D Work. *Proceedings of the ACM Conference on Supporting Group Work*, ACM Press, 306–315. http://doi.org/10.1145/320297.320333
11. HackerNest. 2014. DementiaHack TORONTO by the British Govt & HackerNest. Retrieved May 11, 2015 from http://www.eventbrite.com/e/dementiahack-toronto-by-the-british-govt-hackernest-tickets-12349265987?aff=estw
12. Jun He, Brian Butler, and William King. 2007. Team Cognition: Development and Evolution in Software Project Teams. *Journal of Management Information Systems* 24, 2, 261–292. http://doi.org/10.2753/MIS0742-1222240210
13. Christian Heath and Paul Luff. 1992. Collaboration and Control: Crisis Management and Multimedia Technology in London Underground Line Control Rooms. *Computer Supported Cooperative Work* 1, 1990, 69–94.
14. Pamela J. Hinds and Catherine Durnell Cramton. 2014. Situated Coworker Familiarity: How Site Visits Transform Relationships Among Distributed Workers. *Organization Science* 25, 3, 794–814.
15. Toshiaki Katayama, Mark D. Wilkinson, Kiyoko F. Aoki-Kinoshita, et al. 2014. BioHackathon series in 2011 and 2012: penetration of ontology and linked data in life science domains. *Journal of Biomedical Semantics* 5, 5, 5. http://doi.org/10.1186/2041-1480-5-5
16. Pedram Keyani. 2012. Stay focused and keep hacking. Retrieved May 11, 2015 from https://www.facebook.com/notes/facebook-engineering/stay-focused-and-keep-hacking/10150842676418920/
17. Sara Kiesler and Jonathon N. Cummings. 2002. What Do We Know about Proximity and Distance in Work Groups? A Legacy of Research on Physical Distance. In *Distributed Work*, Pamela J. Hinds and Sara Kiesler (eds.). MIT Press, Cambridge, MA, 57–80.
18. Robert E. Kraut and Paul Resnick. 2011. *Building Successful Online Communities: Evidence-Based Social Design*. MIT Press, Cambridge, MA.
19. Hilmar Lapp, Sendu Bala, James P. Balhoff, et al. 2007. The 2006 NESCent Phyloinformatics Hackathon: A Field Report. *Evolutionary Bioinformatics* 3, 287–296.
20. Steven Leckart. 2015. The Hackathon Fast Track, From Campus to Silicon Valley. *The New York Times*. Retrieved May 11, 2015 from http://nyti.ms/1CawQxH
21. Michael Liegl. 2014. Nomadicity and the Care of Place-on the Aesthetic and Affective Organization of Space in Freelance Creative Work. *Computer Supported Cooperative Work* 23, 2, 163–183. http://doi.org/10.1007/s10606-014-9198-x

22. Nicolas Mangano, Thomas D. LaToza, Marian Petre, and André van der Hoek. 2014. Supporting informal design with interactive whiteboards. *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 331–340. http://doi.org/10.1145/2556288.2557411

23. Gloria Mark. 2002. Extreme Collaboration. *Communications of the ACM* 45, 6, 89–93. Retrieved June 30, 2014 from http://dl.acm.org/citation.cfm?id=508453

24. Jennifer Marlow and Laura Dabbish. 2012. Designing Interventions to Reduce Psychological Distance in Globally Distributed Teams. *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, ACM Press, 163–166. http://doi.org/10.1145/2141512.2141568

25. Martha L. Maznevski and Katherine M. Chudoba. 2000. Bridging Space Over Time: Global Virtual Team Dynamics and Effectiveness. *Organization Science* 11, 5, 473–492.

26. Steffen Möller, Enis Afgan, Michael Banck, et al. 2014. Community-driven development for computational biology at Sprints, Hackathons and Codefests. *BMC Bioinformatics* 15, Suppl 14, S7. http://doi.org/10.1186/1471-2105-15-S14-S7

27. Bonnie A. Nardi and Steve Whittaker. 2002. The Place of Face-to-Face Communication in Distributed Work. In *Distributed Work*, Pamela J. Hinds and Sara Kiesler (eds.). MIT Press, Cambridge, MA, 83–110.

28. Gary Olson and Judith Olson. 2000. Distance Matters. *Human-Computer Interaction* 15, 2, 139–178. http://doi.org/10.1207/S15327051HCI1523_4

29. Wanda J. Orlikowski. 2002. Knowing in Practice: Enacting a Collective Capability in Distributed Organizing. *Organization Science* 13, 3, 249–273. http://doi.org/10.1287/orsc.13.3.249.2776

30. Patricia M. Sias and Daniel J. Cahill. 1998. From coworkers to friends: The development of peer friendships in the workplace. *Western Journal of Communication* 62, 3, 273–299. http://doi.org/10.1080/10570319809374611

31. LLC SocioCultural Research Consultants. 2014. Dedoose Version 5.0.11, web application for managing, analyzing, and presenting qualitative and mixed method research data. Retrieved from http://www.dedoose.com

32. Deborah Sole and Amy Edmondson. 2002. Situated Knowledge and Learning in Dispersed Teams. *British Journal of Management* 13, 174, 517–534.

33. Stephanie Teasley, Lisa Covi, M.S. Krishnan, and Judith S. Olson. 2000. How Does Radical Collocation Help a Team Succeed? *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, ACM Press, 339–346. http://doi.org/10.1145/358916.359005

34. Erik H. Trainer, Chalalai Chaihirunkarn, Arun Kalyanasundaram, and James D. Herbsleb. 2014. Community Code Engagements: Summer of Code & Hackathons for Community Building in Scientific Software. *Proceedings of the ACM Conference on Supporting Group Work*, ACM Press, 111–121. http://doi.org/10.1145/2660398.2660420

35. Bruce W. Tuckman. 1965. Developmental Sequence in Small Groups. *Psychological Bulletin* 63, 6, 384–399. http://doi.org/10.1037/h0022100

36. Patrick Wagstrom, James D. Herbsleb, Robert E. Kraut, and Audris Mockus. 2010. The impact of commercial organizations on participation in an online community. Presentation at the Annual Academy of Management Meeting, August 10, Academy of Management, Briarcliff Manor, NY.

37. Jeanne M. Wilson, Susan G. Straus, and Bill McEvily. 2006. All in due time: The development of trust in computer-mediated and face-to-face teams. *Organizational Behavior and Human Decision Processes* 99, 16 – 33. http://doi.org/10.1016/j.obhdp.2005.08.001

38. Robert K. Yin. 2014. *Case Study Research*. SAGE Publications, Inc., Thousand Oaks, CA.