# Flash Organizations: Crowdsourcing Complex Work By Structuring Crowds As Organizations

**Melissa A. Valentine**[1], **Daniela Retelny**[1],
**Alexandra To**[1,2], **Negar Rahmati**[1], **Tulsee Doshi**[1], **Michael S. Bernstein**[1]
Stanford University[1], Carnegie Mellon University[2]
flashorgs@cs.stanford.edu

## ABSTRACT

This paper introduces *flash organizations:* crowds structured like organizations to achieve complex and open-ended goals. Microtask workflows, the dominant crowdsourcing structures today, only enable goals that are so simple and modular that their path can be entirely pre-defined. We present a system that organizes crowd workers into computationally-represented structures inspired by those used in organizations — roles, teams, and hierarchies — which support emergent and adaptive coordination toward open-ended goals. Our system introduces two technical contributions: 1) encoding the crowd's division of labor into de-individualized roles, much as movie crews or disaster response teams use roles to support coordination between on-demand workers who have not worked together before; and 2) reconfiguring these structures through a model inspired by version control, enabling continuous adaptation of the work and the division of labor. We report a deployment in which flash organizations successfully carried out open-ended and complex goals previously out of reach for crowdsourcing, including product design, software development, and game production. This research demonstrates digitally networked organizations that flexibly assemble and reassemble themselves from a globally distributed online workforce to accomplish complex work.

**ACM Classification Keywords**
H.5.3. Information Interfaces and Presentation (e.g. HCI): Group and Organization Interfaces

**Author Keywords**
Crowdsourcing; expert crowd work; flash organizations

## INTRODUCTION

Crowdsourcing mobilizes a massive online workforce into collectives of unprecedented scale. The dominant approach for crowdsourcing is the microtask workflow, which enables contributions at scale by modularizing and pre-specifying all actions [7, 55]. By drawing together experts [70] or amateurs [6], microtask workflows have produced remarkable success in robotic control [48], data clustering [12], galaxy labeling [54], and other goals that can be similarly pre-specified. However, goals that are open-ended and complex, for example invention, production, and engineering [42], remain largely out of reach. Open-ended and complex goals are not easily adapted to microtask workflows because it is difficult to articulate, modularize, and pre-specify all possible actions needed to achieve them [71, 80]. If crowdsourcing remains confined to only the goals so predictable that they can be entirely pre-defined using workflows, crowdsourcing's long-term applicability, scope and value will be severely limited.

*CHI 2017*, May 06 - 11, 2017, Denver, CO, USA
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-4655-9/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025811

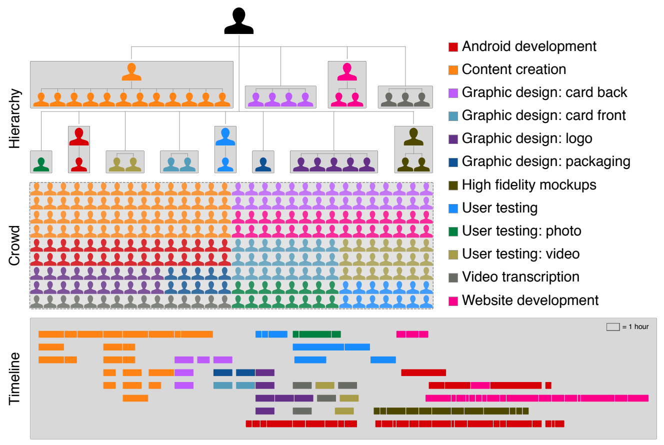


**Figure 1: Flash organizations are crowds that are computationally structured like organizations. They enable automated hiring of expert crowd workers into role structures and continuous reconfiguration of those structures to direct the crowd's activities toward complex goals.**

In this paper, we explore an alternative crowdsourcing approach that can achieve far more open-ended and complex goals: crowds structured like *organizations*. We take inspiration from modern organizations because they regularly orchestrate large groups in pursuit of complex and open-ended goals, whether short-term like disaster response or long-term like spaceflight [8, 9, 63]. Organizations achieve this complexity through a set of formal structures — roles, teams, and hierarchies — that encode responsibilities, interdependencies and information flow without necessarily pre-specifying all actions [15, 83].

We combine organizational structures with computational crowdsourcing techniques to create *flash organizations*: rapidly assembled and reconfigurable organizations composed of online crowd workers (Figure 1). We instantiated this approach in a crowdsourcing platform that computationally convenes large groups of expert crowd workers and directs their efforts to achieve complex goals such as product design, software development and game production.

We introduce two technical contributions that address the central challenges in structuring crowds like organizations. The first problem: organizations typically assume *asset specificity*, the ability for organization members to develop effective collaboration patterns by working together over time [83]. Clearly crowds, with workers rapidly assembled on-demand from platforms such as Upwork (`www.upwork.com`), do not offer asset specificity. So, our system encodes the division of labor into a de-individualized role hierarchy, inspired by movie crews [2] and disaster response teams [8], enabling workers to coordinate using their knowledge of the roles rather than their knowledge of each other.

The second problem: organizational structures need to be continuously reconfigured so that the organization can adapt as work progresses, for example by changing roles or adding teams [9, 63, 83]. Coordinating many workers' reconfigurations in parallel, however, can be challenging. So, our system enables reconfiguration through a model inspired by version control: workers replicate (branch) the current organizational structure and then propose changes (pull requests) for those up the hierarchy chain to review, including the addition of new tasks or roles, changes to task requirements, and revisions of the organizational hierarchy itself.

Enabling new forms of organization could have dramatic impact: organizations have become so influential as the backbone of modern economies that Weber argued them to be the most important social phenomenon of the twentieth century [82]. Flash organizations advance a future where organizations are no longer anchored in traditional Industrial Revolution-era labor models, but are instead fluidly assembled and re-assembled from globally networked labor markets. These properties could eventually enable organizations to adapt at greater speed than today and prototype new ideas far more quickly.

In the rest of the paper, we survey the foundations for this work and describe flash organizations and their system infrastructure. Following this review, we present an evaluation of three flash organizations and demonstrate that our system allows crowds, for the first time, to work iteratively and adaptively to achieve complex and open-ended goals. The three organizations used our system to engage in complex collective behaviors such as spinning up new teams quickly when unplanned changes arose, training experts on-demand in areas such as medical privacy policy when the crowd marketplace could not provide the expertise, and enabling workers to suggest bottom-up changes to the work and the organization.

## RELATED WORK

In this section, we motivate flash organizations through an integration of the crowdsourcing and organizational design research literature, and connect their design to lessons from distributed work and peer production (Table 1).

### Crowdsourcing workflows

Crowdsourcing is the process of making an open call for contributions to a large group of people online [7, 37]. In this paper, we focus especially on *crowd work* [42] (e.g., Amazon Mechanical Turk, Upwork), in which contributors are paid for their efforts. Current crowd work techniques are designed for decomposable tasks that are coordinated by workflows and algorithms [55]. These techniques allow for open-call recruitment at massive scale [67] and have achieved success in modularizable goals such as copyediting [6], real-time transcription [47], and robotics [48]. The workflows can be optimized at runtime among a predefined set of activities [16]. Some even enable collaborative, decentralized coordination instead of step-by-step instructions [46, 86]. As the area advanced, it began to make progress in achieving significantly more complex and interdependent goals [43], such as knowledge aggregation [30], writing [43, 61, 78], ideation [84, 85], clustering [12], and programming [11, 50].

One major challenge to achieving complex goals has been that microtask workflows struggle when the crowd must define new behaviors as work progresses [43, 44]. If crowd workers cannot be given plans in advance, they must form such action plans themselves [51]. However, workers do not always have the context needed to author correct new behaviors [12, 81], resulting in inconsistent or illogical changes that fall short of the intended outcome [44].

Recent work instead sought to achieve complex goals by moving from microtask workers to expert workers. Such systems now support user interface prototyping [70], question-answering and debugging for software engineers [11, 22, 50], worker management [28, 45], remote writing tasks [61], and skill training [77]. For example, flash teams demonstrated that expert workflows can achieve far more complex goals than can be accomplished using microtask workflows [70]. We in fact piloted the current study using the flash teams approach, but the flash teams kept failing at complex and open-ended goals because these goals could not be fully decomposed a priori. We realized that flash teams, like other crowdsourcing approaches, still relied on immutable workflows akin to an assembly line. They always used the same pre-specified sequence of tasks, roles, and dependencies.

Rather than structuring crowds like assembly lines, flash organizations structure crowds like organizations. This perspective implies major design differences from flash teams. First, workers no longer rely on a workflow to know what to do; instead, a centralized hierarchy enables more flexible, de-individuated coordination without pre-specifying all workers' behaviors. Second, flash teams are restricted to fixed tasks, roles, and dependencies, whereas flash organizations introduce a pull request model that enables them to fully reconfigure any organizational structure enabling open-ended adaptation that flash teams cannot achieve. Third, whereas flash teams hire the entire team at once in the beginning, flash organizations' adaptation means the role structure changes throughout the project, requiring on-demand hiring and onboarding. Taken

| | Coordination Structures | Source of labor | Example outcomes |
|---|---|---|---|
| Peer production | Shared repositories (wikis, code) | Open call to volunteers | Wikipedia, Debian Linux [3] |
| Crowdsourcing | Computational microtask workflows | Open call to paid crowdsourcing marketplaces or volunteers | ImageNet [19], GalaxyZoo [54] |
| Traditional organizations | Roles, teams, hierarchy | Employees | Incident Command [8], Xerox [15] |
| Flash organizations | Computationally-reconfigurable roles, teams, hierarchy | Open call to paid crowdsourcing marketplaces | EMS Trauma Report, True Story, Enterprise Workshop Planning Portal |

**Table 1: Coordination infrastructure and labor source in peer production, crowdsourcing, and traditional and flash organizations.**

together, these affordances enable flash organizations to scale to much larger sizes than flash teams, and to accomplish more complex and open-ended goals. So, while flash teams' predefined workflows enable automation and optimization, flash organizations enable open-ended adaptation.

## Organizational design and distributed work

Flash organizations draw on and extend principles from organizational theory. Organizational design research theorizes how a set of customized organizational structures enable coordination [52]. These structures establish (1) roles that encode the work responsibilities of individual actors [41], (2) groupings of individuals (such as teams) that support local problem-solving and interdependent work [13, 29], and (3) hierarchies that support the aggregation of information and broad communication of centralized decisions [15, 87]. Flash organizations computationally represent these structures, which allows them to be visualized and edited, and uses them to guide work and hire workers. Some organizational designs (e.g., holacracy) are beginning to computationally embed organizational structures, but flash organizations are the first centralized organizations that exist entirely online, with no offline complement. Organizational theory also describes how employees and employers are typically matched through the employee's network [23], taking on average three weeks for an organization to hire [17]. Flash organizations use open-calls to online labor markets to recruit interested workers on-demand, which differs dramatically from traditional organizations and requires different design choices and coordination mechanisms.

Organizational design research also provides important insight into virtual and distributed teams. Many of the features afforded by collocated work, such as information exchange [64] and shared context [14], are difficult to replicate in distributed and online environments. Challenges arise due to language and cultural barriers [62, 34], incompatible time zones [65, 68], and misaligned incentives [26, 66]. Flash organizations must design for these issues, especially because the workers will not have met before. We designed our system using best practices for virtual coordination, such as loosely coupled work structures [35, 64], situational awareness [20, 27], current state visualization [10, 57], and rich communication tools [64].

## Peer production

Flash organizations also relate to peer production [3]. Peer production has produced notable successes in Wikipedia and in free and open source software. One of the main differences between flash organizations and peer production is whether idea conception, decision rights, and task execution are centralized or decentralized. Centralization, for example through a leadership hierarchy, supports tightly integrated work [15, 87]; decentralization, as in wiki software, supports more loosely coupled work. Peer production tends to be decentralized, which offers many benefits, but does not easily support integration across modules [4, 33], limiting the complexity of the resulting work [3]. Flash organizations, in contrast, use centralized structures to achieve integrated planning and coordination, even across diverse disciplines. In addition, by tapping into online labor markets, flash organizations overcome peer production's struggles to attract volunteer contributors [31, 32, 56, 75]. In exchange for these benefits, flash organizations face the additional costs of negotiating contracts, motivation crowding [18], and paying for labor.

## FLASH ORGANIZATIONS

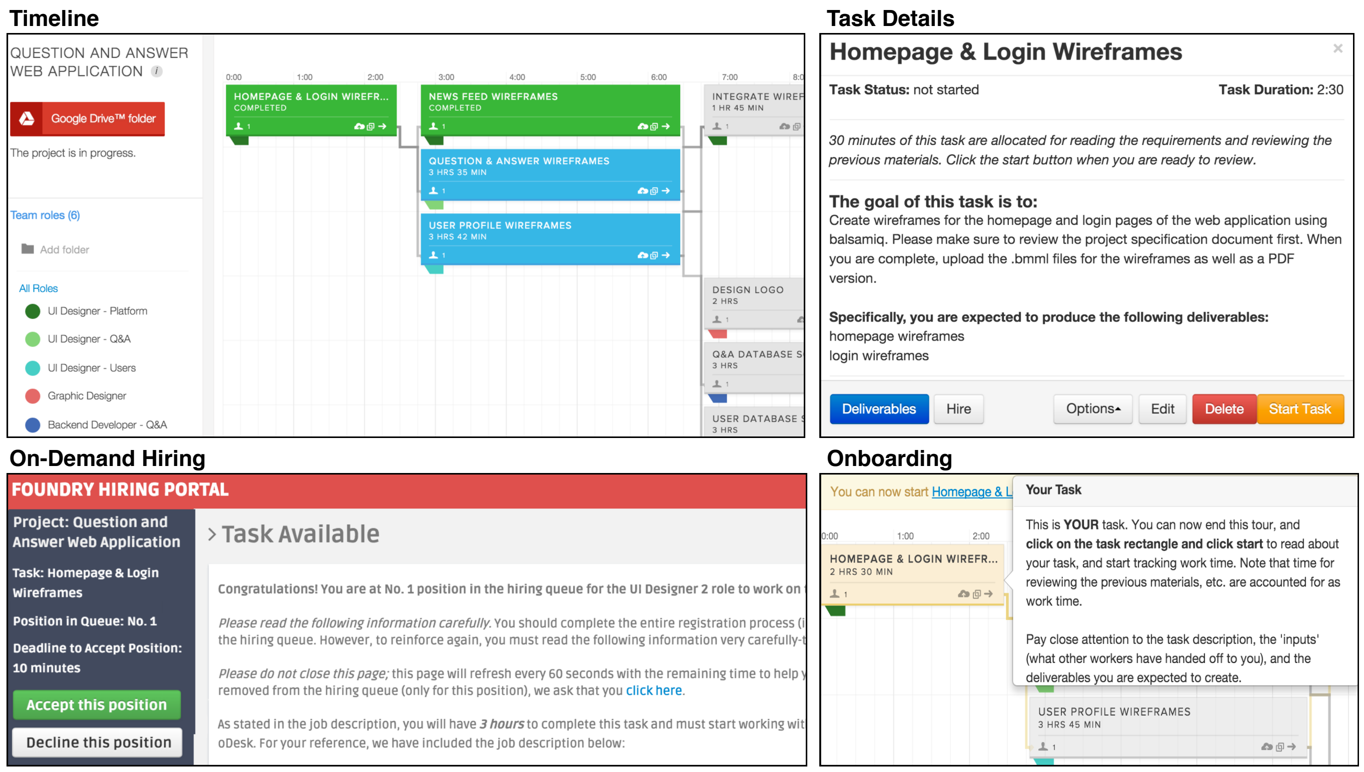
In this section, we introduce flash organizations and our system, Foundry, which enables them. Flash organizations draw together 1) the structure and coordination techniques of traditional organizations to enable open-ended and complex work, and 2) the scale and computational management abilities of crowdsourcing. We describe two problems central to the design of flash organizations, and our solutions to them: asset specificity addressed through computational role structures, and structure adaptation addressed through version control of organizational structures. We enacted these contributions in Foundry, a web platform that enables authoring, populating, and adapting organizational structures (Figure 2).

## Computational organizational structures

Flash organizations encode computational structures inspired by organizations. However, traditional organizations premise their organizational structures on *asset specificity* [83], the value that comes from people working together over time. Asset specificity accounts for workers becoming more and more in sync with teammates over time and improving their ability to coordinate and solve problems together. Clearly crowds, with transient participants assembled on-demand through open calls, do not offer asset specificity.

To address asset specificity, we draw on research on temporary organizations such as disaster response teams and movie crews. These temporary organizations coordinate successfully even without asset specificity [2, 8] by relying heavily on *role structures*, which are activity-based positions that can be assumed by anyone with necessary training [2, 79]. Example role structures for a film crew include main grip and director; example roles for a disaster response team include strike team leader and firefighter. Role structures encode responsibilities and interdependencies; boom operators and grips know how to coordinate by virtue of their roles.

We thus designed flash organizations around computational role structures. Each role represents a position for a crowd



**Homepage & Login Wireframes**

**Task Status:** not started **Task Duration:** 2:30

*30 minutes of this task are allocated for reading the requirements and reviewing the previous materials. Click the start button when you are ready to review.*

**The goal of this task is to:**
Create wireframes for the homepage and login pages of the web application using balsamiq. Please make sure to review the project specification document first. When you are complete, upload the .bmml files for the wireframes as well as a PDF version.

**Specifically, you are expected to produce the following deliverables:**
homepage wireframes
login wireframes

**Project: Question and Answer Web Application**

**Task: Homepage & Login Wireframes**

**Position in Queue: No. 1**

**Deadline to Accept Position: 10 minutes**

> Task Available

Congratulations! You are at No. 1 position in the hiring queue for the UI Designer 2 role to work on

*Please read the following information carefully.* You should complete the entire registration process (i
the hiring queue. However, to reinforce again, you must read the following information very carefully-t

*Please do not close this page;* this page will refresh every 60 seconds with the remaining time to help y
removed from the hiring queue (only for this position), we ask that you click here.

As stated in the job description, you will have ***3 hours*** to complete this task and must start working wit
oDesk. For your reference, we have included the job description below:

You can now start Homepage & L

**Your Task**

This is **YOUR** task. You can now end this tour, and **click on the task rectangle and click start** to read about your task, and start tracking work time. Note that time for reviewing the previous materials, etc. are accounted for as work time.

Pay close attention to the task description, the 'inputs' (what other workers have handed off to you), and the deliverables you are expected to create.

**Figure 2: Foundry supports the authoring of flash organizations. The Foundry timeline (top left) displays all roles and current tasks. The task description (top right) displays task requirements and allows the leader to hire a worker to fill the role. When hiring for a new role (bottom left), Foundry notifies qualified workers; those who respond enter a hiring queue. Once hired, they are oriented to the purpose of the organization and upstream and downstream tasks (bottom right).**

worker in the organization, and it specifies the expertise required to fill the position (e.g., audio editor, AngularJS programmer). Roles enable automatically hiring from an online labor market such as Upwork. Flash organizations' role structures enable open call hiring, clarify what workers are supposed to do, and specify with whom they should communicate.

A flash organization coordinates these roles by arranging them into a hierarchy, which encodes authority and decisions rights [82]. Hierarchies enable flash organizations to take action with centralized, coordinated purpose, much like a director has executive authority on a movie set. Hierarchy means that flash organizations can be formally represented as a tree of nested role structures. The role structure encodes interdependencies, and the nesting encodes hierarchy and decision rights. Leaf nodes are roles representing workers, which can be nested into teams. Teams can optionally be led by a team lead. For example, a team might include a set of workers with expertise in interface design, nested under a UI Design Lead. At the top of the tree is the organization's leader.

The flash organization's hierarchy (Figure 1 top) determines the actions that each worker can perform. The goal is for information to flow up to the central leader so that decisions can be made with awareness of the state of the entire organization. When a worker submits a task in the system, the hierarchy one level above is alerted, and then reviews and accepts it or returns it with feedback for revision. The leader has full formal decision rights, or delegates rights to team leads.

To create a role-based hierarchy in Foundry, users add roles (e.g., "User Interface Designer") and link the role to at least one skill tag listed in Upwork. Foundry utilizes these keywords to automatically post positions to the relevant experts on the Upwork marketplace. For example, a web engineering role can specify the Upwork "node.js" tag, and Foundry queries Upwork for workers who match. The leaders then begin Foundry's on-demand hiring process to fill the role.

Once a role is created, it can be assigned *tasks*, the basic unit of work in Foundry (Figure 2). Foundry tasks are parametrized by desired duration, description, required inputs from other tasks, required outputs to other tasks, and the individual directly responsible for ensuring the task is completed [70]. A task timeline visualizes sequencing. Workers can start, pause, and complete tasks using the Foundry interface. For each role, Foundry highlights their upcoming tasks and shows the remaining time for any active tasks. When a worker submits a task, they upload the file to Foundry, and answer documentation questions to record decisions or other information to the organizational record. Foundry visualizes the role hierarchy by organizing tasks into rows by team (Figure 1).

Finally, Foundry draws on CSCW best practice for virtual organizations. A timeline interface supports shared awareness of progress; text and video chat, availability indicators (present, offline, idle), and Slack integration support coordination (Figure 2). Foundry automatically publishes major notifications such as tasks started or completed, and pull requests issued.

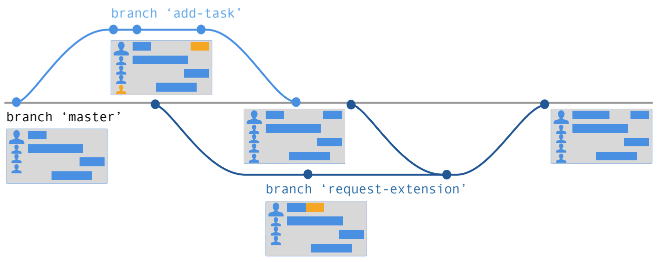


**Figure 3: Workers can branch the current organizational structures, make any desired edits, and then issue a pull request for review. This mechanism enables the organization to continuously adapt.**

## Reconfigurable organizational structures

In contrast to crowdsourcing efforts using workflows, flash organizations are designed to adapt throughout the work process. To enable this adaptation, the organizational structures must be modified as new opportunities or challenges arise. For example, when a movie crew encounters unexpected weather, they change which scene they are shooting, quickly activate new parts of the organization, or redeploy experts to fill needed roles [2]. Likewise, when a disaster response crew finds unexpected materials at the scene of a fire, a firefighting team must quickly reconfigure itself to mount a hazardous materials response [8]. Flash organizations likewise must be able to adapt to changing conditions by reconfiguring their structures.

However, for adaptation to be feasible, flash organizations need to enable distributed workers from across the organization to update the organizational structures in real-time. Members of an engineering team, for example, might be adjusting deliverables at the same time as a quality assurance lead is changing tasks' dependencies. This creates pressure to adapt rapidly, but flash organizations must also not lurch without centralized purpose in reaction to each new adaptation. A single shared, multi-author organizational structure (e.g., Google Docs) would be reactive but susceptible to uncoordinated changes; distributed structures ensure consistency but require significant effort to recombine. Likewise, a globally-writeable organizational structure (e.g., a wiki) enables changes to come from any worker, but can cause organizational chaos if workers disagree and engage in edit wars [48]; a locally-writeable structure accessible only to the leader centralizes control but silences good ideas from lower in the hierarchy. Flash organizations ideally require an approach that allows workers to explore changes quickly in a sandbox, and then ask people up the hierarchy to quickly review and merge them.

Flash organizations enable reconfiguration through a technique inspired by version control, enabling workers to branch, merge, and issue pull requests for the organizational structures that define the organization (Figure 3). There are many flavors of version control, for example distributed (git) vs. centralized (Subversion), and changes pushed directly (git and Subversion) or through a review process (GitHub pull requests and Subversion with patch files). We chose our model based on flash organizations' design requirements. There is only one instance of each flash organization, so a decentralized model with multiple copies is unnecessary. However, to coordinate changes, flash organizations require a model that supports review and automatic merging. So, at a high level, flash organizations enable a contributor to *branch* (copy) the organization's current state and edit it while the system tracks the differences from the *master* (original), then merge any changes when the branch is ready via a *pull request* where other team members review the changes and decide to accept or reject them.

Foundry adapts this model to enact both top-down organizational changes as well as bottom-up changes driven by workers. Any member of a flash organization can branch the organization on Foundry to create an editable copy that retains a link to the master branch. The member then edits organizational structures to indicate desired changes, and Foundry highlights a diff of the changes relative to the original organization. Within the branch, the worker can edit any organizational structure including roles, teams, and task details. When desired, the member *pulls from master* to automatically merge in changes that occurred in the master since they branched. For minor adaptations (e.g., adding time), Foundry also provides a form to submit a simple pull request by filling in information for common types of adaptation.

When ready, the organization member sends the proposed changes in the branch via a pull request for review one level up the role hierarchy. Pull requests automatically create alerts via a Slack integration. The alerts appear in a shared organization-wide Slack channel, which is visible and searchable by all members. The requester, reviewer, and other members can discuss the proposed changes in the Slack channel. In this way, the Foundry's visualization of the organizational structures and the pull requests function as boundary objects that people use to negotiate and develop shared understanding of future plans. The reviewer ultimately decides whether to accept and merge the pull request back into the master organization. Foundry then automatically merges changes back into the master branch and issues an alert in Slack. If there are conflicts between the master and member's branch (e.g., the member edited a task that task had been deleted on the master branch), the conflicts are returned to the reviewer to resolve.

As in software version control, implementing this approach requires merging the branch and the master organizational structures. Typical version control operates using a three-way merge on unstructured text such as program code [59]. Three-way merges require tracking history of each version, so that the algorithm can identify a common ancestor of the master and the branch in order to perform the merge. So, Foundry maintains ancestry history for organizational structures: as in source code, the parent is the version of the organizational structure that was branched or edited. However, three-way merges require diffs, and most diffs are designed for text instead of hierarchically structured objects like Foundry's organizational structures. So, we use a diff algorithm designed for structured objects (e.g., arrays) and hierarchical objects (e.g., JSON, which Foundry uses) [72]. With this infrastructure, the three-way merge algorithm can return insertions, deletions, and edits, which Foundry manages automatically, as well as any conflicts, which Foundry returns to the user to resolve. Following this branch-and-merge process, Foundry hires newly required workers and notifies members of changes.

### On-demand hiring of expert crowd workers

Foundry populates the organizational structures using *on-demand hiring* (Figure 2). On-demand hiring is entirely automated and enables flash organizations to hire relevant experts from expert crowdsourcing marketplaces such as Upwork within about fifteen minutes on average. Foundry enables on-demand hiring through worker panels that have been pre-vetted via a skill-based qualification task. These panels are a retainer pool of high-quality workers [5]. Example Foundry panels include Android application development, graphic design, quality assurance testing, and video animation.

When a new role is added to the organizational structure, Foundry e-mails all workers on the relevant panel to notify them that a position is immediately available on a first-come, first-served basis. Workers click a link in the email to indicate interest, entering Foundry's hiring queue. The first qualified worker to arrive receives first place in the queue, and has ten minutes to read the details, then choose whether to accept. Once a worker accepts, they begin working. In some situations, however, a leader wants greater control in hiring. In these cases, they can perform more traditional *warm hiring* by inspecting all members of the panel, interviewing, and then making an offer to a specific individual.

We formed panels by posting skill-based qualification tasks [40, 60] for each panel. Upwork workers completed a 1–3 hour task (e.g., simple Android development, logo design, QA on an existing website) to apply for the panel. Their submissions were reviewed by an expert reviewer such as a highly rated domain expert on Upwork. If the worker's submission was of sufficiently high quality, the worker was added to the panel. We envision that as the reputation systems on platforms such as Upwork improve [24, 36], the panel role could eventually be played by the platform itself.

Arriving in the middle of a fully functional organization requires workers to quickly learn their specific responsibilities and interdependecies, and organizational goals. Foundry onboards new workers by orienting them to their role responsibilities and their position in the organization through a guided walkthrough (Figure 2 bottom right). Foundry then calls out relevant inputs, upcoming tasks, and the description of the worker's first task. This process takes under five minutes.

## EVALUATION

Do flash organizations enable crowds to mount large-scale coordinated efforts toward complex and open-ended goals? In this section, we explore this question by reporting results from a system deployment where three flash organizations pursued goals that have remained open challenges for crowds: open-ended product design, software development, and game design. These projects represent more open-ended and complex goals than past targeted successes such as crowdsourced interface prototyping [70], code debugging [11, 50], and ideation [1]. Our evaluation strategy is inspired by prior work in crowdsourcing, which has likewise demonstrated proof of concept goals via systems operating on real tasks (e.g., [6, 43, 49]).

Because this paper's thesis is an existence claim — that flash organizations *can* coordinate and adapt to complete open-ended, complex goals — rather than a comparative "better than" claim, we opted for a field study deployment to establish whether flash organizations are in fact capable of complex work, and if so, how they achieve it.

### Method

We recruited three leaders from outside our research team to run flash organizations. We sought leaders who had unique complex goals to pursue and who represented different expertise. None of the leaders were experts in crowd work or Upwork. We provided each leader with a budget, Foundry, and a deadline of six weeks to achieve their goal. Leaders kept ownership of all created products and intellectual property.

The three resulting organizations spanned software, product, and game design. The first organization, EMS Trauma Report, designed and led by a medical student, used the crowd to create a prototype Android mobile and web application for emergency medical technicians (EMTs) to report trauma injuries from an ambulance en route to the hospital. The second organization, True Story, designed and led by a team of crowdfunded card game makers, used the crowd to design, manufacture, and playtest a storytelling card game and an accompanying mobile application. The third organization, Enterprise Workshop Planning Portal, designed and led by a member of a technology lab at the Accenture software consulting firm, used the crowd to create an enterprise web portal to administer client workshops.

All decisions were made by the organizational leaders, including the creation and execution of roles, teams, and tasks. Leaders could use Foundry to delegate decisions to team leads and workers. Their customized organizational structures were automatically populated with diverse crowd workers including graphic designers, poets, and programmers. Table A7 in the Appendix reports the panels accessible to the leaders, and the number of applicants in each panel hiring process. We paid Upwork workers their posted profile hourly wage. When necessary, we aided the leaders in using Foundry, but did not make any organizational decisions on their behalf.

During the deployment, we tracked the organizational structures that each leader developed, the experts they hired and the time elapsed when hiring them, and the number and type of organizational adaptations. We conducted interviews with 47 participants, including the leaders, team leaders, and workers. We also recruited three neutral reviewers to assess the quality of each deliverable. These reviewers were expert in the respective product domains. They judged whether the final deliverables met the intended goal and were at least average quality as compared to similarly scoped products.

## RESULTS

All three leaders spun up and led an organization to complete their goals within six weeks, convening workers on-demand in fourteen minutes on average. Each organization successfully completed its goal (Figure 4) to the satisfaction of the leader and received an acceptable quality rating by the three expert reviewers. The organizations collectively comprised 93 crowd workers, including 22 team leads and 24 teams (Figure 5). These workers completed 639 tasks across 3,261 person-hours

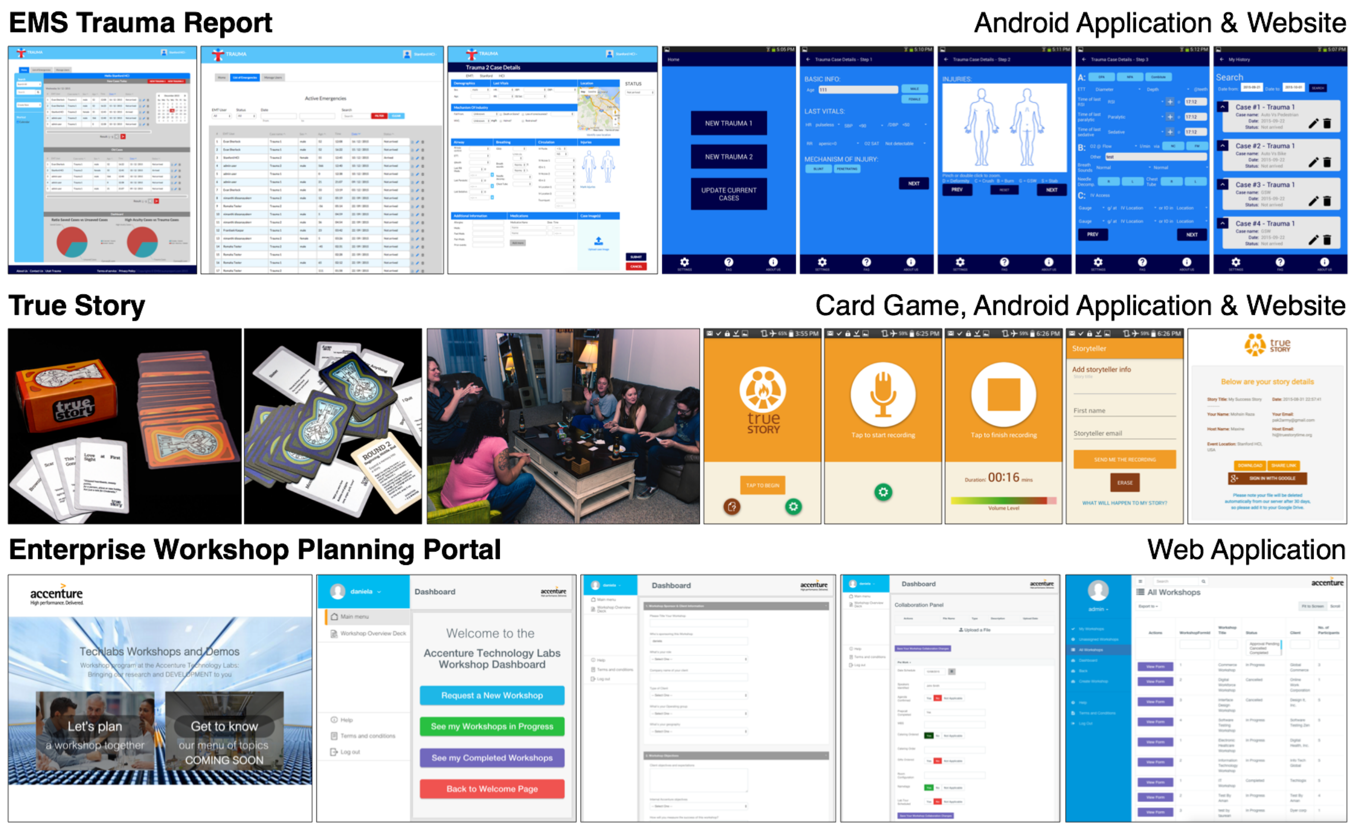

EMS Trauma Report
Android Application & Website
True Story
Card Game, Android Application & Website
Enterprise Workshop Planning Portal
Web Application
NEW TRAUMA 1
NEW TRAUMA 2
UPDATE CURRENT CASES
Tap to start recording
Tap to finish recording
Welcome to the Accenture Technology Labs Workshop Dashboard
Request a New Workshop
See my Workshops in Progress
See my Completed Workshops
Back to Welcome Page
Techlabs Workshops and Demos
Let's plan a workshop together
Get to know our menu of topics COMING SOON

**Figure 4: Three flash organizations successfully developed:(top) a tablet application and web portal for emergency medical responders; (middle) art, content, and a supporting application for a storytelling card game; and (bottom) an enterprise IT portal for consultant workshop planning.**

of work time. Altogether, the organizations wrote 52,000 lines of software code, including two mobile applications and three full-stack web applications, and created two illustrated 80-card decks. The median task across organizations lasted 3.05 hours, and the median daily concurrent work time was 14.1 hours by a median five workers. The organizations spent $46,191 (EMS Report), $6650 (True Story), and $36,604 (Enterprise Workshop Planning Portal), with the interquartile range of workers' profile wages $15–$30 per hour. We first present a case analysis of the adaptation and hiring involved and then present quantitative measures of these activities.

### *EMS Trauma Report*

The EMS Trauma Report organization developed an Android application for emergency medical technicians (techs) to use a touch-enabled tablet from an ambulance to send advance reports while en route to the hospital. The application allowed techs to rapidly enter vital information such as demographics, mechanism of injury, whether the patient is intubated, heart rate and blood pressure, location of the trauma event, and a photograph. The data were then uploaded to a secure hospital web site displaying a filterable overview-plus-detail list of all incoming trauma cases, as well as an automatically-updating GPS location of the ambulance.

The EMS leader began by warm-hiring an Android developer, and asking her to craft a plan for the organization. She hired a user interface design team, and both team leads then decided to bring on two more developers, hiring each on-demand in about 8 minutes. Together they iterated on building out prototypes of the leader's sketches. Next, the Android lead, who became the de-facto organization leader, spun up a front-end engineering team, hiring two developers to build the client application and user interface. These groups worked together on features such as the interaction flow for low-acuity trauma cases. Pull requests laid out new tasks and team members.

At this point, many Android and front-end team members noted that they could not do what they needed to do without a back-end. The EMS leader then hired three back-end developers on-demand in 17, 27, and 60 minutes respectively. After a day, the team hierarchy was reconfigured to make a particularly skilled worker the team lead.

The team produced an early prototype, prompting the EMS leader to spin up a user testing team. He took the prototype to local users and sent their feedback to this team. User feedback indicated that the system needed to be redesigned to accommodate high-acuity trauma cases. The EMS leader sketched out wireframes for the revised design, which team leads used to create pull requests for 15 additional tasks to cover high-fidelity mocks, engineering, testing and debugging.

During this same period, the EMS leader wondered about compliance with federal Health Insurance Portability and Accountability Act (HIPAA) regulations. He wanted to hire someone to oversee HIPAA compliance, but Upwork did not have workers with that expertise. He hired a web security engi-

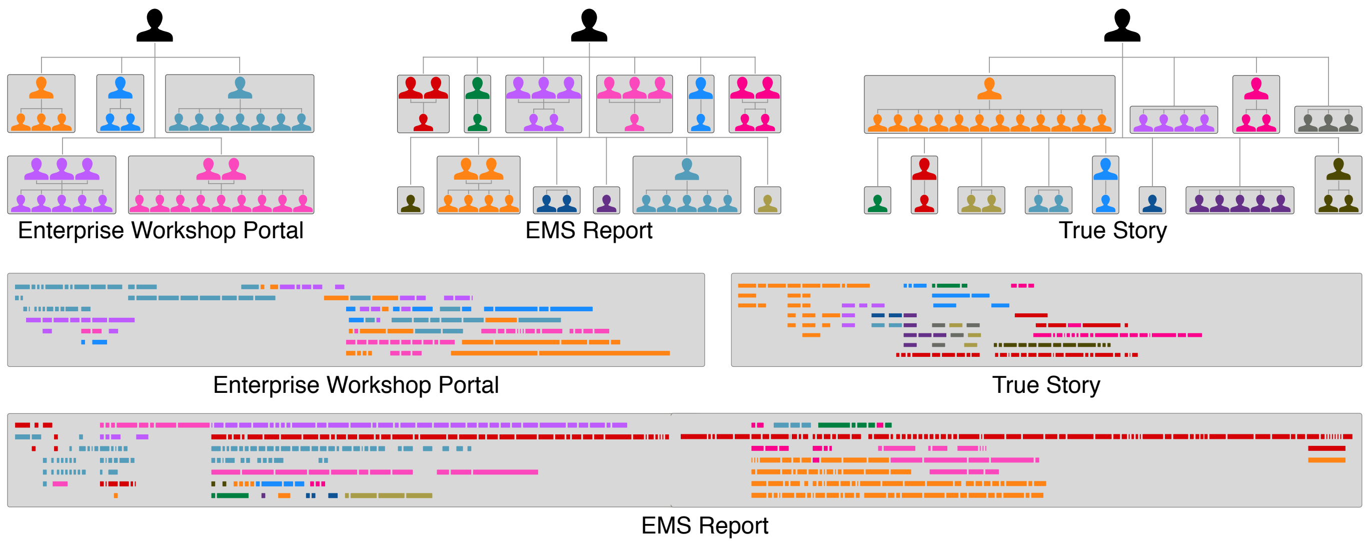


**Figure 5: Final organizational structures and task timelines for the three flash organizations. Colors indicate different roles and corresponding tasks (details in Appendix). Structures varied from flat to nested hierarchies, and included 24 teams and 639 tasks across 3,261 person-hours of work time.**

neer in Egypt to train himself on HIPAA policy and connected him with the local university compliance officer.

The EMS leader spun up a marketing team on-demand to create materials to pitch the app to funders and users. While creating marketing materials, a worker proposed replacing the hiring web page with a feature overview page. The team agreed, and he created pull requests to create the new page.

### *True Story*

The True Story (TS) organization produced a storytelling card game including a deck of cards. Each card in the final game (`http://truestorytime.org/game/`) is a prompt for players to tell a story from their life, for example "Unknown Territory" or "Fake It Till You Make It". The organization developed an artistic style for the card decks, and a short poem on each card that corresponds with the prompt (e.g., "Crushing": *Subtle looks, pounding pulse / However long the hover lasts / Between friend zone and fun zone*). An Android application was also made for use in the game to record stories as they are told.

The TS leaders first used on-demand hiring to quickly hire a team of 12 poets to write a short poem to print on each card. They then realized they wanted an integrated tone across the poems and so hired a new poet to review the poems and create a integrated set. On-demand hiring enabled the 160 cards to be created quickly and provided a range of creative ideas.

The TS leaders also divided out the design and manufacturing of the actual cards. They hired different teams to design the back and front of the cards, the card packaging, and a game logo. The TS leaders then decided to playtest their game. They hired a playtesting lead on-demand, who hired playtesters to organize their friends for game nights and record the proceedings. She shipped test card decks to the playtesters and synthesized their results for the TS leaders.

As the deadline neared, the TS leaders decided to make a mobile application and website that could record the stories told in the game. To achieve this goal in one week, the TS leaders, team leads and workers submitted over 50 pull request changes to add a series of new roles and tasks. Creating a mobile application required hiring an entire new team with Android development experience — a set of skills non-overlapping with the existing organization and leaders' expertise. The TS leaders hired a team lead who created roles and assigned tasks to design, test and produce the mobile app and website.

### *Enterprise Workshop Planning Portal*

The Enterprise Workshop Planning Portal (ENT) organization designed and engineered a web application for Accenture to administer client workshops. The organization began with vague requirements, then iterated toward a spec. The final system first asks workshop organizers to enter information about clients, then build a schedule for the session and monitor progress via a dashboard. The ENT organization had to coordinate with Accenture employees to maintain brand consistency and understand the teams' needs.

The ENT leader began by interviewing and warm-hiring the design and back-end team leads in 30 minutes and 11 hours, respectively. Each team lead then hired 3 team members on-demand in an average of 16 minutes. The back-end team lead struggled to make a plan, so the ENT leader reconfigured the hierarchy, asking him to continue to code but asking another team member to lead the team. Later, the ENT leader did not like the mockups created by a member of the design team, but appreciated his leadership expertise, so used Foundry to move that UI Designer into a team lead role and assign his remaining design tasks to three other UI Designers who had delivered higher quality mocks.

Quality assurance and user testing teams were hired on-demand as the project neared completion. The QA experts were hired in an average of 18 minutes each. While they were waiting for the developers to finish implementing features for them to test, the team lead used Foundry to initiate pull requests that prepared all of the quality assurance test cases to be completed and standardize their reporting.

| | Median hiring time | Pull requests | Leaders | Team leads | Workers |
|---|---|---|---|---|---|
| EMS Report | 13min40s | 335 | 7.2% | 92.8% | 0.0% |
| True Story | 12min30s | 113 | 21.2% | 47.8% | 31.0% |
| Enterprise | 15min13s | 118 | 66.9% | 17.9% | 15.3% |

**Table 2: Automated hiring and organizational reconfigurations.**

The ENT leader asked users in his organization for prototype feedback, which prompted requirements to be redesigned, delaying the project. Workers issued a series of pull requests to adjust project scope and implementation goals. To build out the desired interaction design, the front-end team shifted from plain HTML to AngularJS, leading to a series of pull requests for front-end engineering. The ENT leader then felt the development team was understaffed for meeting the goal he had in mind. He used Foundry to automatically hire three more front-end and three back-end developers. Days later, one of the AngularJS developers had a family emergency. The team needed more AngularJS expertise, so he used Foundry to request front and back-end developers who knew AngularJS. At this point, the leader felt at risk for missing budget and schedule goals, so all teams revised work plans using pull requests to scope a less ambitious deliverable.

### *Results: Reconfiguring organizational structures*

Each organization continuously adapted to changing conditions by adding people, tasks, teams, and time, and by revising groupings, hierarchy, and task requirements. These on-demand adaptations, which included 113 pull request changes in True Story, 118 in Enterprise and 335 in EMS Report (Table 2), resulted in 566 pull request changes. They came from leaders (9.4%), team leads (68.2%), and team members (22.4%). Adaptation was continuous over the duration of each organization, with a median of four changes per day per organization. Changes were both top-down per the leaders' directives, and bottom-up per workers' initiative.

Top-down adaptations allowed quick realignment whenever the leader needed to react to new information or unexpected feedback. Table 3 summarizes examples, including adding new roles and reconfiguring hierarchies (e.g., True Story spinning up a mobile application development team with one week until the deadline). Such top-down adaptations are more reminiscent of traditional requester-driven crowdsourcing models. Flash organizations also allowed crowd workers to initiate reconfigurations bottom-up. Examples in Table 3 include adding tasks that had not been anticipated (e.g., EMS workers proposing revised content for the application's web page).

Flash organizations' adaptations were not without issue. For one, tracking adaptations in Foundry as the changes became more fine-grained (e.g., bug fixes) was less useful for workers, who found little need to update the task list and instead engaged in real-time teaming to rapidly identify, claim, and patch bugs. A second issue: some workers felt the ability to adapt came at the expense of careful planning; they wished the leaders engaged in a more extensive planning process with fewer and smaller adaptations throughout.

### *Results: On-demand hiring*

Seventy five workers were hired automatically across each organization (Table A1 in the Appendix). Leaders used this on-demand hiring process to fill new organizational structures in a median of 13.7 minutes. In contrast, (manual) warm hires took much longer, a median 15 hours. These hiring processes unfold on significantly different scales: automated hires in 14 *minutes*, warm hires in 15 *hours*, and traditional organization hiring processes in 14–25 *days* [17].

| | |
|---|---|
| **Bottom-up change: add new tasks** | EMS Trauma Report |
| One of the marketers realized that the application's hiring page would be better off as a feature overview. Other workers agreed, so he created four pull requests adding tasks to redesign the page and its content. | |
| **Bottom-up change: add new tasks** | Enterprise Workshop Portal |
| As the engineering teams completed their milestones, the quality assurance team took the initiative to create pull requests with tasks that would coordinate the upcoming engineering team's testing process. | |
| **Top-down change: reconfigure hierarchy** | EMS Trauma Report |
| A back-end engineer showed particular initiative and skill. The EMS leader reconfigured his role in the team hierarchy, making him the team lead. | |
| **Top-down change: add new role** | True Story |
| The True Story leaders sourced card content from team of poets. Upon review, they decided to create a new role on the team to integrate content. | |

**Table 3: The flash organizations used top-down pull requests per leaders' directives, and bottom-up pull requests per workers' initiative.**

| | |
|---|---|
| **A team of poets** | True Story |
| The True Story leaders hired a team of twelve poets on-demand. They wrote creative content for the game cards. | |
| **HIPAA consultation on-demand** | EMS Trauma Report |
| The EMS leader hired a web security engineer in Cairo to train himself on American HIPAA privacy laws and advise the team. | |
| **Rapid team expansion** | Enterprise Workshop Portal |
| The ENT leader expanded engineering capacity by hiring four new front-end and back-end engineers on-demand in as little as 7 minutes each. | |

**Table 4: Organizations used on-demand hiring to spin up expertise.**

The organizations used on-demand hiring to quickly hire needed expertise and to source diverse ideas. Examples are summarized in Table 4, including True Story hiring a team of 12 poets and a "Chief Poetry Officer" to create game content, and EMS hiring a security engineer to train himself on American HIPAA privacy laws.

Although flash organizations could in theory recruit new crowd workers for each new task, the organizations in practice accreted members over time. Leaders used Foundry's hiring functionality to rehire members for new tasks. Rehiring minimized the necessary onboarding, and meant that organization members inhabited several different roles in the organizational structure at different points in time. On-demand hiring produced other challenges. There was a somewhat unpredictable skill fit of the hired crowd experts. Reliable reputation signals remain an issue for online labor markets [36, 69]. Future research can aim to improve crowd platforms' reputation systems, and in particular to identify professionals with similar styles or skills in a domain.

## DISCUSSION

Like many other social structures, the nature of work is being reshaped by the internet, computation and algorithms. Responsibilities that used to be the domain of human managers are becoming the domain of computational systems. To date these systems have focused on distributed and independent work [7]. In this paper, we envision a future in which computational systems instead orchestrate organizations that achieve complex and open-ended goals requiring diverse expertise. We present a field deployment of our system in which flash organizations automatically hired experts from the crowd into role-based organizational structures and reconfigured the structures as work progressed.

Flash organizations advance crowdsourcing research by introducing organizational structures as a new approach for coordinating online workers assembled through open call. This approach adapts the coordination affordances of organizations for computational systems and crowds. In doing so, flash organizations open a set of goals previously out of reach for crowdsourcing.

Reciprocally, flash organizations also advance research on organizational design in three main ways. First, flash organizations draw on the principles of role-based coordination that enable temporary organizations, but represent the first example of a computational system that encodes and reconfigures these role structures. Future research can explore the social and behavioral dynamics introduced by this approach. Second, flash organizations convene expertise near-instantaneously rather than reconfiguring fixed groups of employees like traditional organizations. This property means these temporary organizations can fluidly assemble participants from online labor markets, a novel capability compared to traditional organizations. But it introduces a trade-off between rapid hiring and workers' familiarity with each other and with the organizational context. Future research can explore this trade-off, and develop approaches for supporting familiarity [74] and helping with relevant context. Third, because flash organizations log all organizational activity, they offer an unprecedented opportunity to conduct data science on organizational structures and processes. Organizational structures are a powerful coordination mode in their own right, but coupled with large-scale data on tasks, roles, hierarchies, and customized workflows, they offer a powerful tool that can usher in an era of flexible networked collaboration.

Flash organizations advance a future of work that is increasingly mediated by computation and algorithms. As new such techniques are developed, continued conversation around desired social outcomes is necessary [25, 76]. If the sociotechnical ecosystem is not designed carefully, flash organizations (like other labor shifts through history) may drive down wages and impose rigid working conditions [39, 58]. Instead, we hope that flash organizations will offer crowd workers, who are currently predominantly piecework employees, the opportunity to join longer-term and more fulfilling projects. More broadly, flash organizations help envision a world that enables crowd workers to pursue long-term careers [42], including skill growth [77], access to labor collectives [73], and guarantees of stable income.

### Limitations

Flash organizations may not be appropriate for every kind of goal, and more research is needed to explore this design space and its failure points. For example, because flash organizations are nimble, they are likely to prove a good fit for early-stage organizations, or organizations that want to prototype new project-level efforts. However, efforts that require high asset specificity and rely on years of expertise, for example familiarity with Google's massive code base, may benefit from more traditional formats. Hybrid opportunities exist: large stable organizations could view their own employees as a crowd, engaging workers in primary projects while rotating them onto secondary on-demand projects to enable secondary teams to grow rapidly if needed.

Other tradeoffs and limitations exist: flash organizations reach a worldwide labor pool, but must therefore contend with cultural and timezone differences [34, 65, 68]; flash organizations enable near-strangers to coordinate effectively through roles, but strangers are less effective than familiar teams [38]; crowd platforms' hiring can be noisy due to over-inflated reputation systems [24, 36]. No organizational form is perfect for all situations [52], and flash organizations are better suited for situations where these tradeoffs matter less. Future designs will iterate on these issues.

We chose a field deployment as our evaluation strategy, which allowed us to demonstrate the feasibility, strengths, and limitations of the approach. Field study deployments are less able to isolate specific causal mechanisms than a randomized field experiment with a matched counterfactual. However, they are a strong fit when the phenomenon is nascent [21], and they provide a strong method for "thick" [53] descriptive accounts, including strengths and limitations.

## CONCLUSION

Through flash organizations, we envision a world in which anyone with an internet connection can assemble an organization from an online labor market and then lead that organization in pursuit of complex, open-ended goals. Engineering such a future would enable a crowd workforce to flexibly assemble and reassemble itself into collectives that rival modern organizations in their prevalence, impact and achievements. Toward this future, flash organizations contribute: (1) methods for computationally structuring crowds like organizations, with roles, teams, and hierarchies; (2) system infrastructure for authoring, hiring, and guiding flash organizations; (3) role-based coordination to enable crowd workers to coordinate via knowledge of each others' positions; (4) a version control model for reconfiguring flash organizations' structures to adapt the organization; and (5) a deployment demonstrating that these techniques enable crowds to achieve open-ended goals.

## ACKNOWLEDGMENTS

The authors thank Kartik Sawhney, Jare Fagbemi, Michael Kim, Maxine Fonua, Corey Garff, Matthew Kim and Jay Patel for their efforts on this project. Additional thanks to Stu Card, Aniket Kittur, Tom Malone, Pamela Hinds, Sharad Goel, Clark Barrett, and Anita Woolley for feedback on early drafts. This work was supported by the National Science Foundation (award IIS-1351131), Accenture Technology Labs, Microsoft FUSE Labs, the Stanford Cyber Initiative, the Stanford Institute for Research in the Social Sciences, and a Stanford Interdisciplinary Graduate Fellowship.

## REFERENCES

1. Ricardo Matsumura Araujo. 2013. 99designs: An Analysis of Creative Competition in Crowdsourced Design. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing*. 17–24.
2. Beth A. Bechky. 2006. Gaffers, Gofers, and Grips: Role-Based Coordination in Temporary Organizations. *Organization Science* 17, 1 (2006), 3–21. DOI: http://dx.doi.org/10.1287/orsc.1050.0149
3. Yochai Benkler. 2002. Coase's Penguin, or, Linux and The Nature of the Firm. *Yale Law Journal* (2002), 369–446.
4. Yochai Benkler. 2013. Peer Production and Cooperation. In *Handbook on the Economics of the Internet*. 1–29.
5. Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. 2011. Crowds in Two Seconds: Enabling Realtime Crowd-powered Interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, 33–42. DOI: http://dx.doi.org/10.1145/2047196.2047201
6. Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: A Word Processor with a Crowd Inside. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 313–322. DOI: http://dx.doi.org/10.1145/1866029.1866078
7. Jeffrey P. Bigham, Michael S. Bernstein, and Eytan Adar. 2015. Human-Computer Interaction and Collective Intelligence. In *Handbook of Collective Intelligence*. MIT Press, 57–84.
8. Gregory A. Bigley and Karlene H. Roberts. 2001. The incident command system: High-reliability organizing for complex and volatile task environments. *Academy of Management Journal* 44, 6 (2001), 1281–1299. DOI: http://dx.doi.org/10.2307/3069401
9. Shona L. Brown and Kathleen M. Eisenhardt. 1997. The Art of Continuous Change: Linking Complexity Theory and Time-Paced Evolution in Relentlessly Shifting Organizations. *Administrative Science Quarterly* 42, 1 (1997), 1–34. DOI: http://dx.doi.org/10.2307/2393807
10. John M. Carroll, Dennis C. Neale, Philip L. Isenhour, Mary Beth Rosson, and D. Scott McCrickard. 2003. Notification and awareness: Synchronizing task-oriented collaborative activity. *International Journal of Human Computer Studies* 58, 5 (2003), 605–632. DOI: http://dx.doi.org/10.1016/S1071-5819(03)00024-7
11. Yan Chen, Steve Oney, and Walter S. Lasecki. 2016. Towards Providing On-Demand Expert Support for Software Developers. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3192–3203. DOI: http://dx.doi.org/10.1145/2858036.2858512
12. Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. 2013. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA, 1999–2008. DOI: http://dx.doi.org/10.1145/2470654.2466265
13. Susan G. Cohen. 1997. What Makes Teams Work: Group Effectiveness Research from the Shop Floor to the Executive Suite. *Journal of Management* 23, 3 (6 1997), 239–290. DOI: http://dx.doi.org/10.1177/014920639702300303
14. Catherine Durnell Cramton. 2001. The Mutual Knowledge Problem and Its Consequences for Dispersed Collaboration. *Organization Science* 12, 3 (6 2001), 346–371. DOI: http://dx.doi.org/10.1287/orsc.12.3.346.10098
15. Richard Daft. 2015. *Organization Theory and Design*. Cengage Learning.
16. Peng Dai, Mausam, and Daniel S. Weld. 2010. Decision-Theoretic Control of Crowd-Sourced Workflows. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*. 1168–1174.
17. Steven J. Davis, R. Jason Faberman, and John C. Haltiwanger. 2013. The Establishment-Level Behavior of Vacancies and Hiring. *The Quarterly Journal of Economics* 128, 2 (5 2013), 581–622. DOI: http://dx.doi.org/10.1093/qje/qjt002
18. Edward L. Deci, Richard Koestner, and Richard M. Ryan. 1999. A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. *Psychological Bulletin* 125, 6 (1999), 627–668. DOI: http://dx.doi.org/10.1037/0033-2909.125.6.627
19. Jia Deng Jia Deng, Wei Dong Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2–9. DOI: http://dx.doi.org/10.1109/CVPR.2009.5206848
20. Paul Dourish and Victoria Bellotti. 1992. Awareness and Coordination in Shared Workspaces. In *Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work*. ACM, New York, NY, USA, 107–114. DOI: http://dx.doi.org/10.1145/143457.143468
21. Amy C. Edmondson, Harvard Business School, and Stacy E. Mcmanus. 2007. Methodological Fit in Management Field Research. *Academy of Management Review* 32, 4 (2007), 1155–1179. DOI: http://dx.doi.org/10.5465/AMR.2007.26586086
22. Ethan Fast and Michael S. Bernstein. 2016. Meta: Enabling Programming Languages to Learn from the Crowd. In *Proceedings of the 29th Annual ACM Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2984511.2984532

23. Roberto M. Fernandez and Nancy Weinberg. 1997. Sifting and Sorting: Personal Contacts and Hiring in a Retail Bank. *American Sociological Review* 62, 6 (1997), 883–902.

24. S S Gaikwad, D Morina, A Ginzberg, C Mullings, S Goyal, D Gamage, C Diemert, M Burton, S Zhou, M Whiting, K Ziulkoski, A Ballav, A Gilbee, S S Niranga, V Sehgal, J Lin, L Kristianto, J Regino, N Chhibber, D Majeti, S Sharma, K Mananova, D Dhakal, W Dai, V Purynova, S Sandeep, V Chandrakanthan, T Sarma, S Matin, A Nassar, R Nistala, A Stolzoff, K Milland, V Mathur, R Vaish, and M S Bernstein. 2016. Boomerang: Rebounding the Consequences of Reputation Feedback on Crowdsourcing Platforms. In *Proceedings of the 29th Annual ACM Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2984511.2984542

25. Mary L. Gray, Siddharth Suri, Syed Shoaib Ali, and Deepti Kulkarni. 2016. The Crowd is a Collaborative Network. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 134–147. DOI: http://dx.doi.org/10.1145/2818048.2819942

26. Jonathan Grudin. 1994. Groupware and Social Dynamics: Eight Challenges for Developers. *Commun. ACM* 37, 1 (1994), 92–105. DOI: http://dx.doi.org/10.1145/175222.175230

27. Carl Gutwin and Saul Greenberg. 2002. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work* 11, 3-4 (2002), 411–446. DOI: http://dx.doi.org/10.1023/A:1021271517844

28. Daniel Haas, Jason Ansel, Lydia Gu, and Adam Marcus. 2015. Argonaut: Macrotask Crowdsourcing for Complex Data Processing. In *Proceedings of the VLDB Endowment*, Vol. 8. 1642–1653. DOI: http://dx.doi.org/10.14778/2824032.2824062

29. J. Richard Hackman. 1987. The Design of Work Teams. In *Handbook of Organizational Behavior*, Jay W. Lorsch (Ed.). Prentice Hall, Englewood Cliffs, NJ.

30. Nathan Hahn, Joseph Chang, Ji Eun Kim, and Aniket Kittur. 2016. The Knowledge Accelerator: Big Picture Thinking in Small Pieces. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2258–2270. DOI: http://dx.doi.org/10.1145/2858036.2858364

31. Kieran Healy and Alan Schussman. 2003. The ecology of open-source software development. *Unpublished manuscript, January* 29 (2003), 2003.

32. Benjamin Mako Hill. 2013. *Essays on volunteer mobilization in peer production*. Ph.D. Dissertation. Massachusetts Institute of Technology.

33. Benjamin Mako Hill and Andres Monroy-Hernandez. 2013. The Cost of Collaboration for Code and Art: Evidence from a Remixing Community. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*. ACM, New York, NY, USA, 1035–1045. DOI: http://dx.doi.org/10.1145/2441776.2441893

34. Pamela Hinds, Lei Liu, and Joachim Lyon. 2011. Putting the Global in Global Work: An Intercultural Lens on the Practice of Cross-National Collaboration. *The Academy of Management Annals* 5, 1 (6 2011), 135–188. DOI: http://dx.doi.org/10.1080/19416520.2011.586108

35. Pamela J. Hinds and Cathleen McGrath. 2006. Structures that Work: Social Structure, Work Structure and Coordination Ease in Geographically Distributed Teams Pamela. In *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work (CSCW '06)*. ACM, New York, NY, USA, 343–352. DOI: http://dx.doi.org/10.1145/1180875.1180928

36. John J. Horton, Leonard N. Stern, and Joseph M. Golden. 2015. Reputation Inflation: Evidence from an Online Labor Market. (2015).

37. Jeff Howe. 2008. *Crowdsourcing: How the power of the crowd is driving the future of business*. Century.

38. Robert S. Huckman, Bradley R. Staats, and David M. Upton. 2009. Team Familiarity, Role Experience, and Performance: Evidence from Indian Software Services. *Management Science* 55, 1 (2009), 85–100. DOI: http://dx.doi.org/10.1287/mnsc.1080.0921

39. Lilly C. Irani and M. Six Silberman. 2013. Turkopticon: Interrupting Worker Invisibility in Amazon Mechanical Turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 611–620. DOI: http://dx.doi.org/10.1145/2470654.2470742

40. Stephane Kasriel and Jacob Morgan. *Hire Fast & Build Things: How to recruit and manage a top-notch team of distributed engineers*.

41. Daniel Katz and Robert Kahn. 1978. *The Social Psychology of Organizations*. John Wiley & Sons, New York.

42. Aniket Kittur, Jeffrey V. Nickerson, Michael S. Bernstein, Elizabeth M. Gerber, Aaron Shaw, John Zimmerman, Matthew Lease, and John J. Horton. 2013. The Future of Crowd Work. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work (CSCW '13)*. ACM, New York, NY, USA, 1301–1318. DOI: http://dx.doi.org/10.1145/2441776.2441923

43. Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. 2011. CrowdForge: Crowdsourcing complex work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2047196.2047202

44. Anand Kulkarni, Matthew Can, and Björn Hartmann. 2012a. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12)*. ACM, New York, NY, USA, 1003–1012. DOI: http://dx.doi.org/10.1145/2145204.2145354

45. Anand Kulkarni, Philipp Gutheim, Prayag Narula, David Rolnitzky, Tapan S. Parikh, and Björn Hartmann. 2012b. MobileWorks: Designing for Quality in a Managed Crowdsourcing Architecture. *IEEE Internet Computing Magazine* 16, 5 (2012), 28. DOI: http://dx.doi.org/10.1109/MIC.2012.72

46. Walter S. Lasecki, Juho Kim, Nick Rafter, Onkur Sen, Jeffrey P. Bigham, and Michael S. Bernstein. 2015. Apparition: Crowdsourced User Interfaces That Come To Life As You Sketch Them. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2702123.2702565

47. Walter S. Lasecki, Christopher Miller, Adam Sadilek, Andrew Abumoussa, Donato Borrello, Raja Kushalnagar, and Jeffrey P. Bigham. 2012. Real-time captioning by groups of non-experts. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '15)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2380116.2380122

48. Walter S. Lasecki, Kyle I. Murray, Samuel White, Robert C. Miller, and Jeffrey P. Bigham. 2011. Real-time crowd control of existing interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA. DOI:http://dx.doi.org/10.1145/2047196.2047200

49. Walter S. Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F. Allen, and Jeffrey P. Bigham. 2013. Chorus: A Crowd-Powered Conversational Assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2501988.2502057

50. Thomas D. LaToza, W. Ben Towne, Christian M. Adriano, and Andre van der Hoek. 2014. Microtask Programming: Building Software with a Crowd. In *Proceedings of the 27th Annual ACM symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 43–54. DOI:http://dx.doi.org/10.1145/2642918.2647349

51. Edith Law and Haoqi Zhang. 2011. Towards large-scale collaborative planning: answering high-level search queries using human computation. In *AAAI Conference on Artificial Intelligence*. 1210–1215. DOI: http://dx.doi.org/10.1007/s00247-004-1242-4

52. Paul R. Lawrence and Jay W. Lorsch. 1967. Differentiation and Integration in Complex Organizations. *Administrative Science Quarterly* 12, 1 (1967), 1–47. DOI:http://dx.doi.org/10.2307/2391211

53. Yvonna S. Lincoln and Egon G. Guba. 1985. *Naturalistic inquiry*. Vol. 75. Sage.

54. Chris J. Lintott, Kevin Schawinski, AnÅ¿e Slosar, Kate Land, Steven Bamford, Daniel Thomas, M Jordan Raddick, Robert C. Nichol, Alex Szalay, Dan Andreescu, Phil Murray, and Jan Vandenberg. 2008. Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey âŸĚ. *Monthly Notices of the Royal Astronomical Society* 389, 3 (9 2008), 1179–1189. DOI: http://dx.doi.org/10.1111/j.1365-2966.2008.13689.x

55. Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 2010. TurKit: Human Computation Algorithms on Mechanical Turk. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 57. DOI: http://dx.doi.org/10.1145/1866029.1866040

56. Kurt Luther, Kelly Caine, Kevin Ziegler, and Amy Bruckman. 2010. Why it works (when it works): success factors in online creative collaboration. In *Proceedings of the 16th ACM International Conference on Supporting Group Work (GROUP '10)*. ACM, New York, NY, USA. DOI:http://dx.doi.org/10.1145/1880071.1880073

57. Thomas W. Malone and Kevin Crowston. 1994. The interdisciplinary study of coordination. *Comput. Surveys* 26, 1 (1994), 87–119. DOI: http://dx.doi.org/10.1145/174666.174668

58. David Martin, Benjamin V. Hanrahan, Jacki O'Neill, and Neha Gupta. 2014. Being A Turker. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*. ACM, New York, NY, USA, 224–235. DOI: http://dx.doi.org/10.1145/2531602.2531663

59. Tom Mens. 2002. A state-of-the-art survey on software merging. *IEEE Transactions on Software Engineering* 28, 5 (2002), 449–462. DOI: http://dx.doi.org/10.1109/TSE.2002.1000449

60. Tanushree Mitra, C.J. Hutto, and Eric Gilbert. 2015. Comparing Person- and Process-centric Strategies for Obtaining Quality Data on Amazon Mechanical Turk. In *Proceedings of the ACM CHI'15 Conference on Human Factors in Computing Systems (CHI '15)*, Vol. 1. 1345–1354. DOI: http://dx.doi.org/10.1145/2702123.2702553

61. Michael Nebeling, Alexandra To, Anhong Guo, Adrian A. de Freitas, Jaime Teevan, Steven P. Dow, and Jeffrey P. Bigham. 2016. WearWrite: Crowd-Assisted Writing from Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. 3834–3846. DOI: http://dx.doi.org/10.1145/2858036.2858169

62. Tsedal B. Neeley, Pamela J. Hinds, and Catherine D. Cramton. 2012. The (Un)Hidden Turmoil of Language in Global Collaboration. *Organizational Dynamics* 41, 3 (2012), 236–244. DOI: http://dx.doi.org/10.1016/j.orgdyn.2012.03.008

63. Gerardo A. Okhuysen and Beth A. Bechky. 2009. Coordination in Organizations: An Integrative Perspective. *Academy of Management Annals* 3, 1 (1 2009), 463–502. DOI: http://dx.doi.org/10.1080/19416520903047533

64. Gary Olson and Judith Olson. 2000. Distance Matters. *Human-Computer Interaction* 15, 2 (2000), 139–178. DOI:http://dx.doi.org/10.1207/S15327051HCI1523{_}4

65. Judith S. Olson and Gary M. Olson. 2014. How to make distance work work. *Interactions* 21 (2014), 28–35. DOI: http://dx.doi.org/10.1145/2567788

66. Wanda J. Orlikowski. 2000. Using technology and constituting structures: A practice lens for studying technology in organizations. *Organization Science* 11, 4 (2000), 404–428.

67. Galen Pickard, Wei Pan, Iyad Rahwan, Manuel Cebrian, Riley Crane, Anmol Madan, and Alex Pentland. 2011. Time-critical social mobilization. *Science* 334, 6055 (2011), 509–512.

68. Katharina Reinecke, Minh Khoa Nguyen, Abraham Bernstein, Michael Näf, and Krzysztof Z. Gajos. 2013. Doodle Around the World: Online Scheduling Behavior Reflects Cultural Differences in Time Perception and Group Decision-Making. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work (CSCW '13)*. ACM, New York, NY, USA, 45–54. http://doi.acm.org/10.1145/2441776.2441784

69. Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. 2000. Reputation systems. *Commun. ACM* 43, 12 (2000), 45–48. DOI: http://dx.doi.org/10.1145/355112.355122

70. Daniela Retelny, Sebastien Robaszkiewicz, Alexandra To, Walter S. Lasecki, Jay Patel, Negar Rahmati, Tulsee Doshi, Melissa Valentine, and Michael S. Bernstein. 2014. Expert Crowdsourcing with Flash Teams. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2642918.2647409

71. Horst W. J. Rittel and Melvin M. Webber. 1973. Dilemmas in a General Theory of Planning. *Policy Sciences* 4, 2 (1973), 155–169. DOI: http://dx.doi.org/10.1007/BF01405730

72. Philip J. Roberts. 2017. No Json-delta: a diff/patch pair for json-serialized data structures. (2017).

73. Niloufar Salehi, Lilly C. Irani, Michael S. Bernstein, Ali Alkhatib, Eva Ogbe, Kristy Milland, and Clickhappier. 2015. We Are Dynamo: Overcoming Stalling and Friction in Collective Action for Crowd Workers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1621–1630. DOI: http://dx.doi.org/10.1145/2702123.2702508

74. Niloufar Salehi, Andrew McCabe, Melissa Valentine, and Michael S. Bernstein. 2017. Huddler: Convening Stable and Familiar Crowd Teams Despite Unpredictable Availability. In *Proceedings of the 20th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '17)*. ACM, New York, NY, USA. DOI:http://dx.doi.org/10.1145/2998181.2998300

75. Charles M. Schweik and Robert C. English. 2012. *Internet success: a study of open-source software commons*. MIT Press.

76. Ben Shneiderman. 2008. Science 2.0. *Science* 319, 5868 (2008), 1349–1350.

77. Ryo Suzuki, Niloufar Salehi, Michelle S. Lam, Juan C. Marroquin, and Michael S. Bernstein. 2016. Atelier: Repurposing Expert Crowdsourcing Tasks as Micro-internships. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2858036.2858121

78. Jaime Teevan, Shamsi T. Iqbal, and Curtis Von Veh. 2016. Supporting Collaborative Writing with Microtasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2858036.2858108

79. Melissa A. Valentine and Amy C. Edmondson. 2015. Team Scaffolds: How Mesolevel Structures Enable Role-Based Coordination in Temporary Groups. *Organization Science* 26, 2 (2015), 405–422. DOI: http://dx.doi.org/10.1287/orsc.2014.0947

80. Andrew H. Van de Ven, Andre L. Delbecq, and Richard Koenig. 1976. Determinants of Coordination Modes within Organizations. *American Sociological Review* 41, 2 (1976), 322. DOI:http://dx.doi.org/10.2307/2094477

81. Vasilis Verroios and Michael S. Bernstein. 2014. Context Trees: Crowdsourcing Global Understanding from Local Views. In *Second AAAI Conference on Human Computation and Crowdsourcing (HCOMP '14)*. AAAI Press, 210–219.

82. Max Weber, Alexander Morell Henderson, and Talcott Parsons. 1947. *The theory of social and economic organization*. Oxford University Press.

83. Oliver E. Williamson. 1979. Transaction-cost economics: the governance of contractual relations. *Journal of Law and Economics* 22, 2 (1979), 233–261.

84. Lixiu Yu, Aniket Kittur, and Robert E. Kraut. 2016a. Distributed Analogical Idea Generation with Multiple Constraints Lixiu. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 1236–1245. DOI: http://dx.doi.org/10.1145/2556288.2557371

85. Lixiu Yu, Aniket Kittur, and Robert E. Kraut. 2016b. Encouraging "Outside-the-box" Thinking in Crowd Innovation Through Identifying Domains of Expertise. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 1214–1222. DOI: http://dx.doi.org/10.1145/2818048.2820025

86. Haoqi Zhang, Edith Law, Robert C Miller, Krzysztof Z Gajos, David C Parkes, and Eric Horvitz. 2012. Human Computation Tasks with Global Constraints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 217–226. DOI: http://dx.doi.org/10.1145/2207676.2207708

87. Yue Maggie Zhou. 2013. Designing for Complexity: Using Divisions and Hierarchy to Manage Complex Tasks. *Organization Science* 24, 2 (2013), 339–355. DOI: http://dx.doi.org/10.1287/orsc.1120.0744