No Workflow Can Ever Be Enough: How Crowdsourcing Workflows Constrain Complex Work

DANIELA RETELNY, Stanford University
MICHAEL S. BERNSTEIN, Stanford University
MELISSA A. VALENTINE, Stanford University

The dominant crowdsourcing infrastructure today is the workflow, which decomposes goals into small independent tasks. However, complex goals such as design and engineering have remained stubbornly difficult to achieve with crowdsourcing workflows. Is this due to a lack of imagination, or a more fundamental limit? This paper explores this question through in-depth case studies of 22 workers across six workflow-based crowd teams, each pursuing a complex and interdependent web development goal. We used an inductive mixed method approach to analyze behavior trace data, chat logs, survey responses and work artifacts to understand how workers enacted and adapted the crowdsourcing workflows. Our results indicate that workflows served as useful coordination artifacts, but in many cases critically inhibited crowd workers from pursuing real-time adaptations to their work plans. However, the CSCW and organizational behavior literature argues that all sufficiently complex goals require open-ended adaptation. If complex work requires adaptation but traditional static crowdsourcing workflows can't support it, our results suggest that complex work may remain a fundamental limitation of workflow-based crowdsourcing infrastructures.

CCS Concepts: • **Human-centered computing → Human computer interaction (HCI)**; **Collaborative and social computing**;

Additional Key Words and Phrases: Crowdsourcing; workflows; human computation

ACM Reference format:
Daniela Retelny, Michael S. Bernstein, and Melissa A. Valentine. 2017. No Workflow Can Ever Be Enough: How Crowdsourcing Workflows Constrain Complex Work. *Proc. ACM Hum.-Comput. Interact.* 1, 2, Article 89 (November 2017), 23 pages.
https://doi.org/10.1145/3134724

1 INTRODUCTION

Crowdsourcing succeeds by decomposing goals into small tasks to be completed by independent workers and then algorithmically recombining the contributions to create a final result. To coordinate and integrate work, most crowdsourcing approaches rely on *workflows* [10, 44, 67]: pre-specified sets of decomposed tasks that are sequenced and integrated by computation to achieve a final goal.

A central mission of crowdsourcing research has been to design workflows for increasingly complex crowdsourcing goals [33]. Examples include document editing [9], email management [34],

Authors' contact information: Daniela Retelny, Stanford University, Department of Management Science & Engineering, dretelny@alumni.stanford.edu; Michael S. Bernstein, Computer Science Department, msb@cs.stanford.edu; Melissa A. Valentine, Department of Management Science & Engineering, mav@stanford.edu.

2573-0142/2017/11-ART89 $15.00
https://doi.org/10.1145/3134724

text translation [29] and software development [39]. Crowdsourcing workflows are powerful because they build expert knowledge into software, allowing people around the world to contribute meaningfully [10].

However, it remains unknown whether there is a limit to what can be encoded into traditional static crowdsourcing workflows, and in particular why many complex goals have remained a struggle to achieve [32]. Workflows for complex goals such as invention [69–71], writing [1, 30, 33, 48, 60] and product development [39, 52, 72] today only succeed in limited scenarios and with specific constraints. Challenges include poor early results that derail later stages [44], uncoordinated contributions that make inconsistent changes [9, 33], and workers who do not have sufficient global context to make effective decisions [16, 30, 36]. Crowd-written blogs, articles, and stories, for example, repeat content and have an inconsistent voice [1, 9, 30, 36]. Other domains struggle with similar issues. Why are such challenges so pernicious?

In this paper, we investigate the source of this challenge by exploring the role that workflows play in crowdsourcing complex work. To do so, we perform a mixed-methods study of crowd workers tasked with a complex and interdependent goal. Our analysis focuses on the impact that the workflow has on the crowd's coordination toward that goal, specifically how it supports the workers in coordinating toward their goal and how it constrains them from doing the work they need to do.

We focus our case study on *flash teams* of expert crowd workers [52], which represent a state-of-the-art crowdsourcing workflow architecture. Flash teams are modular teams composed of diverse experts (e.g., programmers, designers) from the crowd, assembled and guided by a computational platform to achieve a complex, interdependent goal. The flash teams platform might, for example, convene a UI designer, UX researcher, and two software engineers and guide them through a series of tasks and handoffs: a low-fidelity mockup, heuristic evaluation, revised mockup, functional prototype, user study, and so on. Prior work has shown that flash teams benefit from using these structured workflows compared to more open-ended coordination approaches [52]. Here, we use flash teams to study in more detail what the workflows do, and do not, support.

In our study, we convened six flash teams to pursue the user-centered design of a small-scale mobile web application, representing the broader class of design and engineering crowdsourcing workflows (e.g., [37, 72]). To understand the effect of the workflow, we randomly assigned the six teams into either a workflow-based or role-based team condition. The workflow-based teams were given a pre-specified workflow to follow, which fully decomposed all of the tasks and dependencies for the goal as in the crowdsourcing literature (e.g., [52]). In contrast, the role-based teams were given a minimally specified work plan, which did not define any individual tasks or dependencies, allowing the crowd workers to self-manage to complete the goal. The teams in both conditions had the same role structures [4, 63] and hired three crowd workers with the exact same expertise, including a UI Designer, UX Researcher, and Web Developer. To fill these roles, we recruited 22 crowd workers with the required expertise from Upwork and randomly assigned them into teams. We then used an inductive mixed-method approach across more than 108 hours of chat transcript and log data to compare how the workflow-based and role-based teams, which collectively completed more than 40 intermediate and final deliverables, enacted and adapted their work structures.

Our analysis of the flash teams indicates that while crowdsourcing workflows serve as useful coordination artifacts, they can inhibit *adaptation* behaviors that lie beyond the workflow's pre-specified bounds. The role-based teams typically adapted their work structure, deliverables and goals throughout the project, but the workflow-based teams were bound to the exact plan specified by the workflow and often struggled with even the smallest contingency. For example, while the role-based teams fluidly revised design features as they went, the workflow-based teams had ideas but could not revise even simple interface elements, such as adding pagination and sorting features, because of the pre-specified constraints enforced by the workflow. The tradeoff, however, was that the role-based teams lacked structure and constraints, which led to major coordination challenges such as the development proceeding before usability research on the low-fidelity prototype had occurred. Workflow-based teams might struggle to adapt, while role-based teams might adapt haphazardly because they had no guiding work plan.

This lack of adaptation can help explain why crowds have struggled with complex work. As Lucy Suchman articulated in seminal CSCW research, predefined *plans* are insufficient to describe human behavior toward complex goals [58]. Instead, complex goals require *situated action*: constant modification and replanning in reaction to the current situation. Similar concepts recur across literatures: Schön calls this continuous adaptation the reflective practitioner [56]; organizational behavior lists it as a core feature of reciprocal interdependence [62]. The more complex the goal, the more necessary adaptation becomes [21, 23, 63, 65].

Despite this recognition by Suchman and others that complex goals require adapation, crowdsourcing's model has remained predicated on pre-specified plans via workflows, because this is the most straightforward to embed into an algorithm. Most workflows can only adapt in response to events, contingencies, or outcomes that have been predicted and "pre-programmed" a priori. While advanced crowdsourcing workflows allow the crowd to define tasks for itself before working [30, 36], and enable advanced control flow such as looping [16, 30] or conditional selection between many options [18, 42], they cannot yet add entirely new, unforeseen behaviors as work proceeds. While pre-defined crowdsourcing workflows are important coordination aids, they can be a double-edged sword that limits crowdsourcing from engaging in the adaptations necessary to achieve complex goals.

To date, researchers and practitioners have sought to expand the boundary of complex crowd work without a guiding theory of what will work and what may not. Our work clarifies that crowdsourcing workflows will remain a good fit when contingencies are unlikely and adaptation is unnecessary. However, if the work is the kind where even an expert often reflects, revises, and adapts, then workflows will struggle to succeed. We offer design changes to crowdsourcing platforms that may ameliorate these issues: for example, a panic button that allows workers to call a requester's attention to a task if a contingency has arisen, and a revise button that allows a worker to return a task to an upstream stage in the workflow with comments. More aggressively, alternative crowdsourcing approaches may succeed: those which sacrifice some automation for improved ability to adapt, for example reflection-based [30] or organization-based [64] approaches.

Our contributions comprise:

- The first study of what crowd workers experience when they are "inside a workflow."
- A comparison of workflow-based crowdsourcing to role-based crowdsourcing for complex goals.
- An analysis identifying open-ended adaptation as a core limitation of crowdsourcing workflows.

Similar to Kittur et al. [32] and Gray et al. [26], this research focuses on understanding and incorporating both the technical and social needs of crowd workers. We aim to shift the conversation from pre-defined tasks and dependencies common in workflow-based approaches towards a more holistic view that encompasses the practices and processes necessary for crowds to more effectively collaborate and complete much larger and more complex goals [64]. Furthermore, the insights from our analysis provide crowdsourcing researchers and practitioners with a deeper understanding of the needs and practices of crowd workers, which we hope will lead to new crowdsourcing approaches and systems better equipped to support complex crowd work.

2 RELATED WORK

To answer the question of why crowds struggle with complex goals, it is important to understand how crowdsourcing operates. In this paper, our primary focus is on paid crowdsourcing, which relies on extrinsic incentives, such as money and reputation scores, to motivate and reward workers [10]. Below we summarize research on workflows as coordination artifacts in crowdsourcing and organizations as well as research on crowdsourcing complex work.

2.1 Workflows as Coordination Artifacts

The use of workflows as coordination artifacts dates back to some of the earliest work on organizational behavior [47, 65], some of which was published over half a century ago. Since then, workflows have been adopted as coordination artifacts in a range of domains, including artificial intelligence and crowdsourcing.

2.1.1 Crowdsourcing Workflows. Crowdsourcing brings together large groups of people to solve large-scale problems [6, 28]. Such efforts are enabled by decomposing goals into much smaller tasks that can be executed by independent workers and recombined upon completion. Many of these systems rely on crowdsourcing workflows, which are a set of pre-defined tasks that have been decomposed and are computationally sequenced and assigned to distributed workers to complete [43, 44, 67]. Crowdsourcing workflows are also referred to as human computation algorithms or crowd algorithms [10, 16, 40, 43]. Tasks are "what a worker is asked to do" [67], such as answer a multiple choice question or label an image. Larger tasks are often decomposed into smaller subtasks, either by the system [33] or by other workers [36]. While tasks are typically given to workers, some such as image labeling [49] and translation [29] may be automatically completed by computers or a combination of humans and machines [51, 66].

Workflows play a central role in the success of crowdsourcing. For example, the *Find-Fix-Verify* workflow enabled high-quality text editing where a single non-decomposed task had a 30% error rate [9]. Likewise, *iterate-and-vote* workflows enabled surprising feats like messy handwriting recognition [43], and *map-reduce* workflows have shown promise in writing [33]. These workflows serve as examples of algorithmic approaches for decomposing larger goals into smaller tasks and subtasks. Whereas a single task might result in a range of outcomes, a decomposed workflow can carefully scaffold each step.

Not all workflows follow a single deterministic path. One of computation's main contributions to crowdsourcing and labor has been that workflows can in fact be Turing-complete, enabling more dynamic workflows. Examples of past contributions include support for iteration between tasks and subtasks in crowdsourcing workflows [18, 44], techniques for real-time crowd powered workflows [8, 11, 38] as well as systems for requesters and workers to collaboratively design workflows [36] and provide timely, task-specific feedback [20]. However, in all cases, these workflows are statically pre-defined before they run, or could not be revised without requester intervention. While newer approaches to coordinating crowd work, such as recent work on flash organizations [64], have started to use adaptive workflows, there is still a lot of work to be done in order for crowds to be able to edit these workflows as the work proceeds.

Researchers have leveraged the success of workflows into new tools, frameworks and algorithms for workflow creation. These toolkits enable programatic description of workflows [2, 44] and decompositions of work into smaller partitions [33, 36]. Artificial intelligence algorithms rooted in traditional planning models provide additional tools for dynamically creating and managing these workflows [42, 52, 67].

2.1.2 Workflows in Organizations. Crowds are not the only groups of people to use workflows: they are common in organizations and other domains as well. Organizational behavior and CSCW research has empirically and theoretically examined workflows, which are also referred to as plans [58], as coordination tools. At a high level, much of this research demonstrates that workflows play an important role for formal (e.g., programmed) modes of coordination [47, 50, 65] and in helping to scaffold goals [58]. Specifically, workflows are used in situations where interdependencies and contingencies are predictable environments and goals have low uncertainty [3]. In these settings, research suggests that the role of workflows is to define responsibility for tasks, allocate resources (e.g., scheduling), develop agreement (e.g., orienting actions) [50].

Workflows are also not the only available coordination artifact. For example, peer production [5, 7] relies on shared repositories, such as wikis and shared code repositories. These repositories enable emergent coordination of complex goals among volunteers who are often globally distributed. In contrast, traditional organizations, which are composed of bounded employees, coordinate work through roles, teams and hierarchies [12, 17]. This enables the organizations to assign responsibilities, integrate efforts and adapt goals in response to the work environment. More recently, in response to the critique that organizations are too static and bureaucratic, organizations have adopted more agile coordination methods, such as scrum [19], as well as flat organizational structures, such as holacracy [54].

This research highlights many of the limitations of workflows for coordinating complex work. Specifically, this work calls out that the predictability of the work environment, the attributes of the goal and work structure and the social dynamics and practices surrounding crowdsourcing workflows play a critical role in their enactment as well as their effectiveness as coordination artifacts. This research also extends the existing literature by exploring the role of crowdsourcing workflows and how they are enacted in a new context. Whereas most research on workflows has focused on traditional teams, this research explores workflow enactment among teams of online crowd workers. Unlike traditional teams, online teams of crowd workers come together for short periods of time and are composed of individuals who have never worked together. This introduces new challenges and questions that have yet to be explored.

This paper pushes back on the crowdsourcing literature zeitgeist of creating workflows for increasingly complex goals, instead arguing that the effectiveness of crowdsourcing workflows does not necessarily extend to complex and interdependent goals. This argument poses a question for crowdsourcing designers, since it means that workflows cannot be pre-defined, debugged, and then executed autonomously by workers. Instead, as shown in this study, workflows and plans must be interpreted by workers and adapted flexibly to local circumstances, interactions and work environments.

2.2 Crowdsourcing Complex Work

Crowdsourcing is known to work best for problems that can be decomposed and require little expertise to complete [10]. However, Kittur et al. posed the question: could crowdsourcing be used for complex work as well [33]? Following their work, a series of papers proposed workflows and techniques for crowdsourcing complex goals [33, 36, 52]. CrowdForge [33] and Turkomatic [36] incorporated crowds in the workflow design and planning process. The workflows created using CrowdForge's *map-reduce* framework allowed crowds to author their own subdivided tasks to write articles, research purchase decisions, and conduct basic science journalism. Turkomatic's *price-divide-solve* algorithm, which was recursively editable, enabled microtask workers to complete a range of complex goals, including essay writing, itinerary planning and java programming. These outcomes were not perfect — Crowdforge's crowdsourced science articles, for example, tended to overlook a major empirical result, and Turkomatic's workflows required continuous requester monitoring and intervention — but these systems provided compelling evidence that complex outcomes could be possible.

Complex work also requires a transition from microtask workers from platforms such as Amazon Mechanical Turk to expert crowd workers from platforms such as Upwork. For example, flash teams introduced an approach for gathering teams of experts from the crowd and enabling them to quickly complete complex and interdependent goals, such as web development and video production [52]. To achieve goals of this nature, flash teams decompose the work into a sequence of linked tasks and handoffs, which are completed by diverse experts from the crowd who are computationally managed. Other crowd work systems have hired experts for complex goals, such as writing [48], software development [15, 24], mentorship and skill development [59] and physical world tasks [61]. Similar to microtask crowdsourcing systems, however, most of these expert crowdsourcing systems rely on workflows of decomposed tasks. Therefore, while expert crowds shifted the complexity ceiling of crowdsourced outcomes, their process is still identical to microtask crowds.

Achieving complex goals remains an open challenge for crowdsourcing. Several researchers have noted the challenges faced when decomposing goals and designing a workflow for highly complex and interdependent goals [10, 31, 36] such as software development [57]. To overcome these challenges, crowdsourcing researchers have explored other approaches to enable crowds to coordinate more effectively and achieve more complex goals. For example, research has shown that providing feedback to crowd workers can lead to better results [20] and that balancing personalities of crowd workers can lead to more effective crowd teams with higher quality outputs and higher quality communication [45]. Recent research has also shown that teams of crowd workers that have experience working together are more effective [55].

Despite targeted successes in brainstorming [14, 69], writing [30, 60] and prototyping [52], research efforts have struggled to provide general solutions to the broader wicked problems [53] of innovation, creativity and product development. Recent research on flash organizations, for example, demonstrated an approach for achieving complex and open-ended goals by encoding coordination into a hierarchical organizational structure with de-individualized roles rather than workflows, introducing a version control-style technique for adapting as work proceeds [64]. It has remained unknown whether organizational structures are a prerequisite for complex work with crowds, or if a workflow could have achieved the same goals.

Whereas crowdsourcing relies heavily on workflows, organizations use them as one tool among many. This tension motivates our study to understand how a workflow impacts the practice of crowd work. In order to do this, we modulate the existence and absence of a workflow and observe the coordination patterns of the crowd workers.

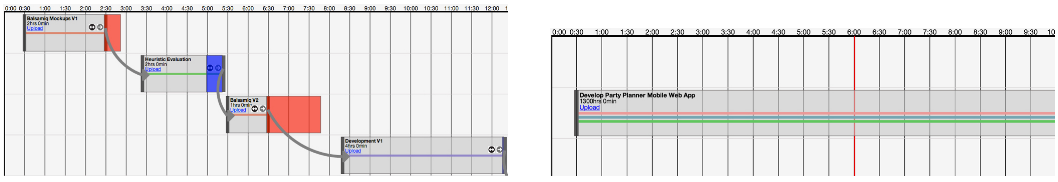
3 METHOD

To understand the impact workflows have on crowd coordination, as well as the challenges encountered with and without workflows, we conducted in-depth case studies of six *flash teams* [52] of crowd workers from Upwork who were recruited as part of a deployment study. We selected expert crowd work as our domain and user-centered design as our goal because it represents a current state-of-the-art for crowd work.

All teams were tasked with the same goal, which was to create a party planning task manager mobile web app. This goal has several desirable characteristics for the purpose of our investigation. First, user-centered design of a web site represents the current state-of-the-art in prior work [52] for complex-based crowdsourcing workflows, both in terms of goal complexity and workflow complexity. Second, the goal is in a useful middle ground: not so easy that success is guaranteed, but also not so difficult that it cannot be achieved via crowds. This middle-ground characteristic makes it likely that the workers will need to deal with challenges and contingencies, shedding light on where the workflow may be struggling. Third, the goal involves heterogeneous workers: designers, usability experts, and programmers. Crowd work increasingly involves experts of different expertise (e.g., [15, 48, 59, 64]). Complex goals involve such combinations of expertise. The boundaries between tasks requiring different expertise again provide stress on the workflow, since workers will need to make their discipline's assumptions explicit.

All of the teams were composed of the same three roles, which included a UI Designer, UX Researcher and Web Developer. These roles were defined by first identifying the tasks that needed to be completed in order to complete the party planning task manager mobile web application and then identifying the skill and roles needed to complete each of the tasks. The party planner context was chosen because it represents a relatively simple design goal with initial sketches inspired by undergraduate student HCI projects.

We authored and deployed the role and work structures using Foundry [52], a computational platform for supporting interdependent crowd work. On Foundry, each worker has a specific role (e.g., "UI Designer") and each task is assigned to one or more roles (e.g., "Design low-fidelity mockups" is assigned to UI Designer). Foundry was used by the teams to access the role and work structures and collaborate using the built-in chat and shared Google Drive folder for the project.



(a) Pre-specified workflows enacted by the workflow-based teams

(b) Minimally specified work plans enacted by the role-based teams

Fig. 1. Screenshots of the pre-specified workflows (a) and minimally specified work plans (b) on Foundry, which were enacted by the workflow-based and role-based teams, respectively. The pre-specified workflows decomposed tasks and dependencies, specified the task durations and could be computationally tracked and managed. The minimally specified work plans consisted of one unstructured 13-hour task assigned to all roles with a duration of 13 hours.

Using hiring criteria from prior work [52], we recruited a total of 22 workers from Upwork during the study. We initially hired 18 workers to fill the three roles on each of the six teams. Specifically, we posted three job postings on the Upwork marketplace for UI Design, UX Research and Web Developer jobs. The postings described the project as well as the skills required to fill the role. We then did brief interviews with qualified candidates and hired the workers that met the job criteria. After hiring workers to fill the roles on the teams, we randomly assigned the teams into either a *workflow* or *role-based* condition, resulting in three teams per condition. During the study, we hired four additional workers to replace the workers who didn't show up or left the team. Three of the replaced workers were in the workflow-based condition and one was in the role-based condition. All workers were informed that this was an IRB approved research study and agreed to participate. On average, the workers had a rating of 4.66/5, 767 total work hours and an hourly rate of $22.22. The workers were mostly male (19 males, 3 females) and came from 8 countries.

When the study started, we sent an email to each team describing the high level goal of the project and instructed the teams to follow the user-centered design process, starting with the rough sketch, and provided them with high-level guidance on the deliverables expected. This email reiterated the roles, expected deliverables and deadline. Each worker also received a separate email with a unique link to the team on Foundry. When workers arrived to Foundry, they saw the list of team roles, the status of the project, all chat messages and either a pre-specified workflow or minimally specified work plan depending on the their team's condition. By clicking on each task in Foundry, the workers could view more information about the expectations, roles, and deliverables for that task. We gave the teams a target deadline of 13 hours to complete the goal; but since we were interested in studying the execution of the work rather than whether they completed on time, we ultimately gave them as much time as needed.

The teams were randomized across two conditions. The *workflow-based teams* were provided with pre-specified workflows on Foundry (Figure 1a), which were designed to mimic traditional crowdsourcing workflows. Specifically, Foundry displayed each worker's role and an interactive workflow that decomposed, defined and tracked the status of all tasks and dependencies. Each task in the workflow specified the input and output, the worker responsible, the duration and other metadata along with the dependencies with prior and subsequent tasks. The workflow required the following tasks, assigned to the following roles: (1) low-fidelity mockup, assigned to the UI Designer; (2) heuristic evaluation, assigned to the UX Researcher; (3) web prototype, assigned to the Web Developer; (4) user study, assigned to the UX Researcher; (5) revised design, assigned to the UI Designer; and (6) revised web prototype, assigned to the Web Developer. While there is flexibility in roles in practice — for example, some designers can act as UX Researchers as well — this division of labor is a common one in user-centered design.

The *role-based teams* served as the baseline comparison condition, allowing us to isolate and evaluate the effect of the workflows. The main difference between the conditions was that the role-based teams were not provided with a pre-defined workflow. Instead, Foundry displayed one large task block (Figure 1b). We refer to this unstructured block as a minimally specified work plan since it lacked most of the key affordances of crowdsourcing workflows but still included some basic information, such as roles and project duration. The key difference, however, was that the role-based teams did not see decomposed or pre-defined tasks and dependencies. Instead, the role-based teams self-managed to complete the task.

Throughout each project, we monitored the chat to ensure teams were working and to answer workers' questions. We were equally responsive to all of teams across both conditions. Furthermore, in cases where team members didn't show up or left the team during the project, we immediately hired a replacement from Upwork and notified the team through the chat.

While the findings reported in earlier work [52] indicated that the pre-specified teams completed the projects in half the time and required less coordination than the minimally specified teams, our current analysis aims to understand *how* the teams in these different conditions enacted and adapted the workflows as well as the challenges encountered.

3.1 Data and Analysis

The data that informs this analysis includes real-time observation notes and archival data. During the projects, we took detailed notes about tasks, interactions and behaviors. After the projects were finished, we exported the task, role and chat log data from Foundry for all six of the teams. We also saved copies of the final deliverables, task outputs and any other artifacts that were created. Finally, we exported all of the work diary data from the Upwork time tracker, which takes screenshots and measures worker activity approximately every ten minutes while the workers are logging work time. Given that all hourly workers must use the time tracker in order to get paid, the work diary data provides a first-hand view of the tasks completed and the work structures employed.

In addition to the observation, archival and deliverable data, we emailed the workers a final follow up survey, which all 22 workers completed. The survey included questions about the workers' experiences, such as what went well, the challenges encountered and their overall perceptions of how they performed both individually and as a team.

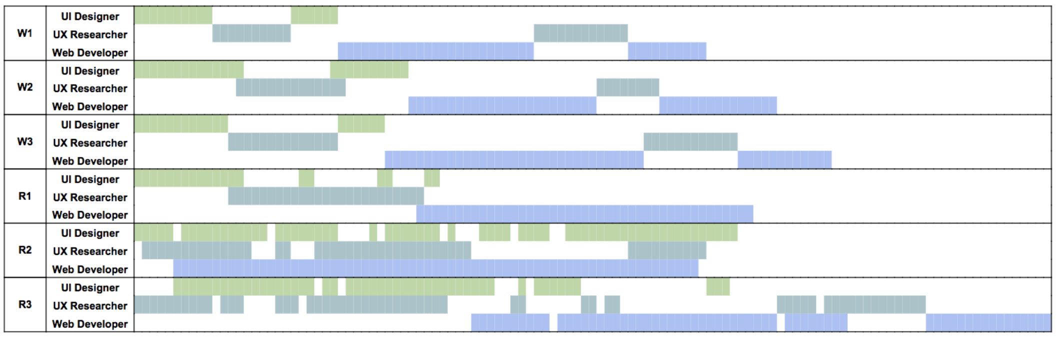


Fig. 2. The work structures enacted by the six crowd teams based on the Upwork work diary data. Each block represents a continuous period of time logged by the worker in that row. The structures enacted by the workflow-based teams (W1-W3) aligned with the workflows defined on Foundry whereas the structures enacted by the role-based teams (R1-R3) were all different.

We first open-coded and analyzed the chat and open-ended survey data for each team individually using the NVivo qualitative data analysis software. We compared themes that emerged across teams and conditions to develop conceptual insights [22] and identify common behaviors, practices and challenges [25]. We then compared the patterns that emerged across the two conditions to understand the similarities and differences between the workflow-based and role-based teams. We iterated between the emergent themes in our data, our research question and relevant literature until we reached an understanding of how the teams enacted and adapted their work structures.

Once the qualitative analysis was complete, we analyzed the work diary data from Upwork to better understand what might explain the differences we were seeing. We analyzed all of the timestamps, memos and screenshots, which captured what workers were doing at each moment and enabled us to reconstruct a timeline of what actually happened in the projects. This data also allowed us to conduct other analyses, such as evaluating concurrent work, active work time and dependency structures in the teams. We ran one-tail t-tests to compare the differences between the workflow-based and role-based teams.

4 ENACTING AND ADAPTING WORKFLOWS

In this section, we share the results from our in-depth case studies of the workflow-based and role-based teams. Our analysis suggests that neither pre-specified workflows nor minimally specified work plans are fully sufficient for orchestrating complex and interdependent crowd goals. Goals of this nature require: 1) coordinating multiple interdependent contributions from diverse workers [32, 50]; and 2) adapting work structures and deliverables in response to unplanned contingencies and opportunities that emerge [13, 58]. Specifically, our results suggest that pre-specified workflows better support coordination and require less communication but inhibit teams from adapting their work structure or goals as work proceeds. In contrast, minimally specified work plans are easier to adapt but make coordination more difficult, require more communication and don't ensure that the most effective adaptations are pursued.

We organize our findings as follows. First, we show how the workflow-based and role-based teams enacted the work structures and discuss their effectiveness as coordination artifacts. Next, we demonstrate how both sets of teams adapted their work structures as they worked towards their final goal.

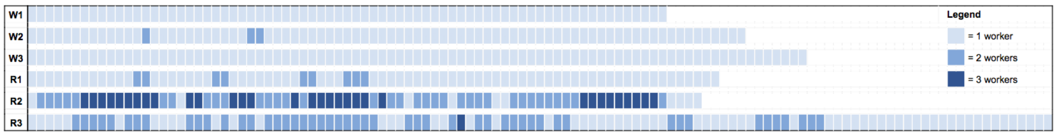


Fig. 3. A heatmap of the concurrent number of active workers in each team. There were between one (light blue) to three (dark blue) workers active in each team during any given 10 minute interval. The workflow-based teams (W1-W3) had very few concurrent workers whereas the minimally specified teams (R1-R3) had concurrent workers much more frequently.

4.1 Workflows Support Coordination

Our analysis suggests that there were distinct differences in the role and effectiveness of the pre-specified workflows and the minimally specified work plans as coordination artifacts. The pre-specified workflows (Figure 1a) encoded best practices, preventing the workflow-based teams from making obvious missteps. The pre-specified workflows also clarified the order of tasks and the relationships between tasks, reducing the amount of explicit coordination and communication needed between workers. In contrast, the role-based teams, which used minimally specified work plans (Figure 1b) had to decide amongst themselves how to proceed. While this increased flexibility, it could also lead to inefficient coordination and even incorrect work.

4.1.1 Encoding Best Practices. In this section, we detail how the pre-specified workflows enacted by the workflow-based teams encoded best practices, reducing coordination around planning and avoiding unintended behaviors. By decomposing goals into actionable tasks with clear outputs and constraints, the pre-specified workflows helped orient and integrate actions in the team and enabled consistent and replicable outcomes.

We compared the Foundry work structures (Figure 1) to the enacted work structures captured in the Upwork work diaries (Figure 2). Whereas the role-based teams all enacted different work structures, the workflow-based teams all enacted similar work structures that closely aligned with the workflows defined on Foundry. As shown in Figure 2, all of the workflow-based teams (e.g., Teams W1-W3) completed the same set of tasks, in the same order, by workers in the same role. In contrast, the role-based teams (e.g., Teams R1-R3) all enacted vastly different work plans, resulting in workers completing tasks in different orders and at different times, sometimes working simultaneously and other times working independently.

The inconsistent work structures enacted by the role-based teams serve as illustrative examples of crowds trying to coordinate without predefined structures. For example, the pre-specified workflows encoded a common practice in user-centered design: use a heuristic evaluation to improve a low-fidelity mockup, and a user study on a functional prototype. Without this defined structure, however, Team R1 decided to conduct the heuristic evaluation and user study at the same time on the low-fidelity mockup and then performed no user studies on the functional prototype. This led to back-to-back development of the two versions of the mobile web app with no user feedback or expert evaluation in between. Instead, after the developer created the first version of the mobile web app, he iterated on it again based on his own opinions instead of user input, resulting in the revision having the same usability problems as the original. Similarly, Team R2 completed the heuristic evaluation while development was already in progress, which was too late to revise the design. This team also completed the user testing at the end of the project, which was too late for the results to be incorporated. Roles were also more fluid: the UX Researcher in Team R2 created UI specifications and several revised mockups throughout the project, both of which were the domain of the UI Designer in the workflow-based teams.

Team	**Total Cost**	**Average Concurrency**	**Chat Word Count**	**Active Work Time**	**Duration (with no breaks)**
W1	$209.77	1.00	3,801	10:53:29	10:53
W2	$352.41	1.04	2,859	13:19:26	12:49
W3	$206.91	1.00	3,362	13:29:41	13:30
R1	$340.27	1.11	4,292	14:15:15	12:45
R2	$603.69	2.29	7,745	28:51:59	12:22
R3	$427.37	1.44	7,237	27:25:35	18:46

Table 1. On average, the workflow-based teams (W1-W3) were significantly cheaper than the role-based teams (R1-R3), had fewer concurrent workers, needed less communication and required fewer work hours.

4.1.2 Influencing Interdependence Styles. In the pre-specified workflow teams, a specific interdependence style emerged. Interdependence, which is inevitable in complex work, plays an important role in determining how to effectively decompose goals, organize teams and structure interactions [65]. For example, goals that can be modularized lend themselves to sequential interdependence, where intermediate results are handed off step-by-step. In contrast, goals that are harder to decompose benefit from a different set of interdependencies and work structures, such as synchronous coordination and mutual adjustment [65]. Given that a critical aspect of coordination is how integration of work occurs under conditions of task interdependence [23, 50], understanding the dependency structures that emerge under different constraints is important for supporting complex crowd work.

To evaluate and compare the dependency structures across teams and conditions, we calculated the average number of concurrent workers for each 10 minute interval in each of the teams (Table 1) and created a heatmap of the dependency structures that emerged (Figure 3). The heatmap highlighted differences in the amount of concurrent work that occurred across the conditions and demonstrated that the role-based teams resulted in more concurrent work compared to the workflow-based teams. Specifically, while the dependency structures varied across the role-based teams, on average in each team there was more than one worker active at any given time (μ = 1.6 workers, SD = 0.6), indicating direct synchronous coordination. In contrast, the workflow-based teams rarely worked concurrently (μ = 1.0 workers, SD = 0.0), indicating that the workflows helped enable sequential interdependence. A one-tail t-test evaluating whether the minimally specified plans enacted in the role-based teams resulted in more concurrent work trended toward significance (t(4)=1.7, p = .08), but with a large effect size of $d = 1.0$, a larger sample would likely bear out the effect.

The trends in the data suggest that, in the absence of structured workflows, workers default to concurrent work and mutual adjustment. These trends also shed light on important implications of concurrent work. Specifically, there were strong positive correlations in our data between the amount of concurrent work and total active work time (R = 0.87), cost (R = 0.92) and communication (R = 0.87). Furthermore, the total active work time, cost and chat log word counts in the role-based teams were all significantly higher in comparison to the workflow-based teams (all p < .05). In fact, our analysis of the chat logs revealed that the role-based teams (μ = 6,425 words, SD = 1,864.33) required almost twice the amount of communication than the workflow-based teams (μ = 3,341 words, SD = 471.36). Workers in the workflow teams reported feeling that "coordination between team members was not necessary on this project" and that the project "could have been done without even talking to each other."

4.1.3 Minimizing Coordination Challenges. Whereas we observed few coordination challenges for the workflow-based teams, the same was not the case for the role-based teams. Even though the workers all had the skills needed to complete the project, the lack of a proper scaffold and constraints in the minimally specified work plans resulted in many observed coordination challenges. For example, role-based team R3 spent two hours arguing over who had responsibility for what, with the Web Developer adamant that the UI Designer should own the front-end code, despite this typically being the Web Developer's responsibility. The team ultimately fired the Web Developer and hired a new one. Equally challenging on the other side of the coordination spectrum, the UI designer in role-based team R1 started working without ever checking in with the other team members. The Web Developer and UX Researcher, who remained unaware of what the UI Designer was doing, were stuck discussing the project and waiting for several hours until the UI Designer returned. This lack of visibility also slowed down teams unnecessarily: the UI Designer in role-based team R3 finished the mockups but did not notify the rest of the team. Eventually, the UX Researcher, who was waiting on the mockups to start his task, asked the UI Designer about the status of the mockups only to find out they were already completed.

All of the role-based teams attempted to informally create work plans via the chat at some point during the project, highlighting the need for structure when coordinating complex and interdependent work. The plans the teams created either helped or inhibited effective coordination depending on when they were created and how they were managed. For example, the role-based team with the smoothest coordination and fewest incidents created a very specific plan at the start of the project, which defined all of the tasks, assigned them to workers and specified their expected duration. As the work proceeded, this team continued to revise their plan based on the status and outcomes of deliverables and other factors such as the remaining time. Much of this team's planning process, however, was a result of the effort of the UX Researcher who emerged as a team lead. Throughout the project, this UX Researcher created documents such as UI and development specs to help structure the work and communicate responsibilities and goals.

In contrast, the other two role-based teams created ad-hoc work plans throughout the project without ever establishing an initial plan at the beginning. Unsurprisingly, these teams encountered many coordination challenges. For example, one of the UX Researchers decided to complete the heuristic evaluation and user testing at the same time and simultaneously work with the UI Designer to refine the designs, which is uncommon in user centered design. This UX Researcher also proposed that the Web Developer start coding the web application in parallel so there would be no bottleneck, which resulted confusion and frustration and ultimately led to the developer and UX Researcher quitting the team. When expressing his reason for quitting, the UX Researcher wrote: *"I didn't feel respected or acknowledged for my work. I stayed up until 7am for you and only logged 4 hours pay."* Unlike the role-based team that compensated for the lack of a scaffold by establishing a plan and revising it throughout the project, this team failed to create the structure needed to help orient their actions effectively.

4.2 Workflows Inhibit Adaptation

The prior section suggested that workflows bring important benefits by structuring worker coordination. In this section, we focus on the central limitation of the workflows in our study: that *pre-specified workflows inhibited crowd teams from adapting as they worked.* Even though the initial decomposition of tasks provided a guide, the workflows could not account for all possible contingencies and outcomes that might occur.

We observed two main challenges that workflows present, making it difficult for crowds to adapt. Specifically, in contrast to minimally specified work plans, the rigid structure imposed by pre-specified workflows: 1) made it difficult for crowds to *respond to contingencies*; and 2) inhibited the crowd workers from *pivoting when a better opportunity* presented itself. Characterizing these challenges is critical if we hope to enable crowds to take on complex goals. Below we elaborate on each of these challenges and demonstrate how they prevented the workflow-based teams in our study from adapting their work structures and goals.

4.2.1 Responding to Contingencies. Even though the workflows in Foundry were quite detailed in their directions, they could not take into account all possible outcomes, making it difficult for workers to adapt when unexpected contingencies occurred. While all of the teams completed their goals and produced functional mobile web applications, they encountered a range of contingencies throughout the projects. Here we refer to *contingencies* as unexpected challenges or occurrences that prevented the work plan from proceeding. Examples of contingencies included workers not showing up or quitting, as well as problematic decisions, changes and assumptions that resulted in incompatible deliverables. While the pre-defined workflows accounted for as many contingencies as possible, it is in practice impossible to anticipate all possible contingencies. We will describe how the workflow-based teams struggled to adapt in response to many such contingencies, whereas the role-based teams more easily overcame them and moved on.

We describe how teams in each condition responded to two illustrative contingencies. One type of contingency was related to hiring: teams in both conditions encountered contingencies wherein workers did not show up, or quit before completing their task. When this happened, the workers needed to be replaced and schedules needed to be revised, which had downstream effects on the rest of the workers and tasks. Even though online crowd platforms such as Upwork allow workers to be replaced quickly, the team had in most cases already been impacted. For example, the developer in one of the workflow-based teams showed up two hours late and then said she was in bed and would return in 30 minutes. This worker never returned and had to be replaced, which caused the development task to get delayed and had further downstream effects: for example, when the UX Researcher arrived for the user testing task, the developer still had two more hours of work.

A second common contingency was incompatible deliverables caused by workers' decisions, changes, assumptions and errors, all of which are difficult for workflows to predict a priori. Some resulted from workers' deliberate decisions (e.g., deciding to use a different programming language than requested), and others were caused by errors or incorrect assumptions (e.g., completing the heuristic evaluation assuming a native phone application instead of a mobile web application).

The teams that were constrained to follow workflows struggled to deal with such contingencies. Because of the rigid structure imposed by pre-specified workflows, decisions and changes that broke the assumptions and specifications embedded in the workflow resulted in incompatible deliverables and caused problems for subsequent workers and tasks. For example, when two of the developers in the workflow-based teams decided to code the mobile web applications using server-side programming languages (e.g., PHP) instead of the client-side languages specified in the workflow (plain HTML and Javascript), the deliverables that were produced were incompatible with the requirements of the dependent tasks, which specified that the deliverable use only client-side web technology. Because workers could not change the workflow, both teams ultimately ended up hosting their deliverables on a temporary server so that the application was at least accessible to later workers and the requesters without needing to set up their own server. So, violating the workflow's embedded assumptions caused multiple downstream issues.

Workers' errors or poor decisions also resulted in incompatible deliverables. Often, the changes required for the next task's worker to massage the deliverable into shape were outside the workflow's bounds, making it more difficult for the team to react. For example, one of the UX Researchers incorrectly assumed that the heuristic evaluation should be completed for a native application (e.g., iPhone and Swift) instead of a mobile web application (e.g., responsive HTML and CSS), which have different usability and design standards. He also ignored the task directions to use Nielsen's Heuristics to evaluate the interface, instead using his own opinion and judgment, as he revealed in the team chat after the task was complete:

> "And one more thing. I did not follow any heuristic approach, I followed what makes it easy for the user to use the app. I have the capacity to hypothesize about the reasons behind the actions that people take. I am able to see things from another person's perspective. And most UX designers do more than think about what people do. It's a purely intellectual job. Hope you understand."

The consensus was that the UX Researcher's decisions led to a mostly unusable heuristic evaluation deliverable. Unfortunately, the workflow specified that the UI Designer had to implement changes for all the usability errors from the heuristic evaluation. The UI designer could not propose an alternative workflow, but instead had no choice but to do exactly what the workflow asked (i.e., create the revised mockups) using the information in the incorrect deliverable, which the UI Designer described as "childish" in the survey.

In contrast, the concurrent and emergent work structure adopted by the role-based teams, along with the lack of constraints in their work plans, on several occasions helped workers catch missing deliverable details and incompatibilities before it was too late. For example, when one of the UI Designers finished the first set of mockups, the UX Researcher quickly reviewed them before conducting the heuristic evaluation. In this review, the UX Researcher noticed that the wireframes were missing error messages for incomplete fields, which allowed the UI Designer to quickly revise the mockups before the heuristic evaluation. In a different role-based team, when the revised mockups did not resolve the issues documented in the UX Researcher's evaluation, the UX Researcher and the UI Designer went through and resolved each of the issues together in the chat. This fluid and emergent behavior, which did not occur in the workflow-based teams since tasks and dependencies were predefined and sequential, also emerged as a theme in the survey responses. One worker, for instance, mentioned that:

> "There was constant communication through Foundry; Feedback and iterations were happening right away and were making it possible to execute the project well."

Taken together, the examples above illustrate some of the contingencies that were encountered in the teams as well as how the constraints in the pre-defined workflows inhibited the workflow-based teams from adapting in response. While it might be possible to design workflows for any single one of these issues or even a set of the issues, in aggregate they are too numerous and varied to design for.

4.2.2 Pivoting When Better Opportunities Emerge.

In addition to making it difficult for crowd workers to adapt in response to contingencies, the constraints and structure enforced by workflows inhibited crowds from considering alternative solutions — more importantly, pivoting when better opportunities emerged. For example, when a work structure or process better suited for the current context emerged, in several instances the workflow-based teams tried to change course but couldn't. In other cases, workflow-based teams attempted to improve their applications by adding features that they felt were important, but were limited because the workflow did not allow it. These outcomes, which are described below, highlight crowdsourcing workflows' inability to dynamically change course when better work plans or goal-related opportunities emerge, both of which are critical to supporting complex work.

The workflow-based teams and role-based teams all faced situations in which their work structures or plans were not best suited for their local context or current status. When these mismatches occurred, the teams attempted to adapt with different levels of success. For example, when the

UI Designer in one of workflow-based teams did not show up and had to be replaced, the revised mockup task got delayed, which had downstream effects on the subsequent development task. To try and minimize the delays and keep the project on schedule, the developer asked if he could work in parallel with the UI Designer. While this parallel work structure was likely better suited for the team's current situation, it was not possible to change the workflow's pre-specified structure, forcing the developer to wait until the revised mockups task was completed.

In contrast to the workflow-based teams, the lack of structure and constraints in the role-based teams made it easier for them to adapt their work plans when better alternatives were available. For example, the role-based teams would internally establish informal deadlines and responsibilities and adjust them as needed. Furthermore, when time was running out or workers were overwhelmed, workers in the role-based teams adapted their work structure by prioritizing the key tasks and collaborating to make sure they all got done. However, while the role-based teams could more easily revise their work plans, they did not always adapt in the most effective way. For example, one of the developers attempted to change the spec of the project away from an application that could run on any phone (a web application) to one that only worked on iPhones (via PhoneGap), which was not at all in line with project specification and would have resulted in an incompatible final deliverable that would not have met the requester's requirements. These ineffective work structure adaptations led to coordination challenges and other inefficiencies in the team, emphasizing the need for a balance between structure and flexibility.

In addition to revising their work structures and plans, the teams in both conditions tried to adapt their goal when they identified new opportunities and improvements such as missing features. Specifically, the workflow-based teams attempted to add features such as delete, search, pagination and sort to their mobile web applications. While these features were outside the scope of the project specifications or workflows, the workers felt they would improve the usability of the application and should be added. In all cases, however, the workflow-based teams were unable to successfully add these features in large part because of the strict constraints embedded in the workflows.

While the role-based teams attempted to add fewer features, they were more successful when doing so. Decisions to add new features in the workflow-based teams were made by *individual workers*, but the role-based teams typically made these feature decisions *collectively* and worked together to achieve them. Specifically, two of the three role-based teams successfully added a delete feature. One of the teams also added a data storage solution (written in Javascript), which they mentioned was needed to make the application functional. This team also wanted to add several other new features, redistributing their time and ultimately prioritizing the features they deemed most important and feasible within the constraints of the project.

Reflecting on the differences, one reason the workflow-based teams were less successful was that made decisions individually, without considering whether they were possible within the constraints and available resources of downstream tasks. However, the strict constraints enforced by the workflow, such as the pre-specified time allotted to each task, prevented downstream workers from redistributing time or resources to other tasks or exploring alternative work paths. This led workers to focus on completing goals within the constraints set by the workflow, which often meant removing new features. Workers could optimize myopically for their own tasks, rather than globally for the final outcome.

Taken together, these findings suggest that crowdsourcing workflows restrict crowds from revising their structure in light of new opportunities. Specifically, by mapping out sequences of tasks and dependencies in advance and not allowing workers to edit them as the work proceeds, workflow-based crowdsourcing approaches assume that there is an optimal sequence of tasks and dependencies and that complex goals can be fully specified, which are rarely the case [27, 41, 58].

5 DISCUSSION

This paper sheds light on the role of crowdsourcing workflows as well as some of the reasons why crowds struggle with complex work. Our analysis suggests that while pre-specified workflows can help crowds avoid errors, coordinate tasks, and manage dependencies, these same structures can also inhibit adaptation. In contrast, while minimally specified work plans can reduce some of adaptation barriers, they can complicate coordination and do not necessarily ensure that the most effective adaptations are the ones pursued.

These findings highlight the limitations of workflow-based crowdsourcing approaches and demonstrate the need for new crowdsourcing approaches better equipped to support complex and loosely defined goals, which require emergent coordination and adaptation as the work proceeds. Below we discuss these limitations and provide recommendations for new approaches and future research.

5.1 Crowds and Situated Actions

We can now return to the question motivating this paper: why do crowds so often struggle with complex goals? While there are many other coordination approaches that don't use workflows — mostly outside of crowdsourcing — the dominant infrastructure in crowdsourcing relies on workflows and human computation algorithms. Our analysis identified that crowd workers struggled against the constraints of the workflows, which prevented *adaptation* in the face of errors and better alternatives. Any foreseeable adaptations can, of course, be built into the workflow. Many workflows involve if-else logic, sanity checks, and other methods of managing predictable issues. But unforeseeable adaptations happen, even with goals as constrained as interface iteration given a low-fidelity sketch.

As Suchman argued in *Plans and Situated Actions* [58], no plan for a complex goal can cover all issues, or even cover enough issues to be meaningfully complete. Suchman critiqued computing, artificial intelligence and CSCW for designing based on an assumption that complex human behavior could be achieved through pre-specified plans. Human behavior in complex situations, she argued, is not planned cognitively and then executed, but instead is reflectively revised through continuous interaction with the world. These insights and critiques influenced CSCW to stop attempting to code all human behavior into rules [68].

Recent work in organizational behavior reinforces this message: workflows are limited tools when work is unpredictable; when it has changing interdependencies; or when it occurs in high velocity, time-constrained environments [3, 13, 23]. Specifically, in these environments work plans cannot specify all information in advance and therefore must be adapted as goals evolve [13] and contingencies unfold [23, 58]. Given that complex work increasingly operates under these circumstances [13], organizational behavior researchers have emphasized the need to support both formal and informal (e.g., emergent) coordination [3, 23, 35, 65].

Open-ended adaptation, then, may prove to be the Achilles' heel of crowdsourcing via workflows. This literature, and our results, would predict that no crowdsourcing workflow can ever be enough to achieve truly complex goals. Given that workflows are most useful to the extent one is able to anticipate the tasks, dependencies and contingencies [27, 65], crowdsourcing workflows are only as effective as the contingencies and adaptations they manage to encode. And the more complex the goal, the fewer adaptations can be readily accounted for. Therefore, crowdsourcing workflows consisting of predefined work plans will likely remain fundamentally incompatible with complex work.

At its core, this result indicates a double-edged sword of crowdsourcing workflows. Without workflows, as in the role-based teams, the crowd commonly makes elementary mistakes such as ignoring core principles of user-centered design. Workflows prevent these undesired departures from best practice. Ironically, in doing so, workflows also prevent any desirable departures and adaptations. So, by preventing bad outcomes, crowdsourcing researchers and practitioners are also preventing good outcomes.

Organizational behavior research deepens this understanding. Specifically, this tension is similar to the formal-informal coordination paradox noted in prior work, which suggests that neither formal nor informal (e.g., improvised) modes of coordination fully satisfy the coordination needs of high-velocity and unpredictable organizations [12, 13, 23]. In their current form, crowdsourcing workflows are akin to the formal coordination processes and artifacts used in organizations [46, 50]. While these formal approaches have benefits, such as clear responsibilities and division of labor [23, 46], they also have important known limitations [50]. For example, formal coordination approaches struggle in uncertain environments or in situations that require rapid action [23]. However, as in our study, self-organization like the role-based teams come at the expense of increased coordination challenge.

5.2 Recommendations for Crowdsourcing Platforms

Taken together, these insights from prior work combined with the findings from the case studies lay out a new set of design requirements and recommendations for crowdsourcing platforms. Specifically, we argue that future crowdsourcing workflows and systems need to support both informal and formal coordination to allow crowds to respond to unplanned contingencies and contribute new ideas and solutions. The question then becomes, how do we support both emergent and formal coordination in the context of complex crowd work?

One possible approach would be to let crowd workers design their own workflows that are specific to the people and context of the problem being solved. Just as a designer does not follow the exact same process twice, this would allow workers to author tweaks before launching a workflow. This would allow the crowd workers to create the structure needed for effective coordination without forcing them to follow a specific pre-defined structure that might not be best suited for their work environment. For example, rather than providing teams with a pre-specified workflow or no workflow, crowdsourcing systems could provide them with a blank workflow with one or two initial tasks that require team members to create a work structure to achieve the desired final goal within the constraints specified. As they create the workflow, the system or requester could provide them with suggestions and guidance to ensure that the team produces a high quality and reliable workflow.

Our findings also highlight the need for approaches that allow teams of crowd workers to collectively adapt their work structures in response to new ideas, evolving goals and unplanned contingencies. Based on the outcomes and takeaways of our case studies, we propose different techniques for enabling collective adaptation, which future crowd work systems and research should explore. First, we suggest combining workflows with some of the techniques and activities used in team coordination approaches. For example, workflow-based crowdsourcing approaches could incorporate short meetings at regular intervals (e.g., daily or upon reaching specific milestones), which are known to help teams synchronize efforts, discuss obstacles and self-organize [19].

In addition to collective adaptations, crowdsourcing workflows should also support ongoing reconfigurations by individual crowd workers on the team while providing sufficient structure and guard rails to ensure the adaptations pursued are in line with the project goals and constraints. Existing collaboration and coordination systems as well as the version control-style reconfiguration techniques in flash organizations [64] demonstrate a few different approaches for achieving ongoing reconfigurations and adaptation among distributed individuals. For example, Github's branching and pull request model, Google Drive's permission system, and Quirky's user-based voting have all enabled different forms of collective action and orchestrated activity among distributed and interdependent individuals. It may be enough to change our infrastructure from pre-defined workflows to reconfigurable workflows that serve as editable work scaffolds for achieving a pre-defined complex goal. Future crowdsourcing research should explore whether some of these existing approaches would help crowds effectively reconfigure their structure and efforts as they work on complex goals.

However, shifting crowdsourcing infrastructures away from workflow-based solutions will come with its own costs. Whether reflection-based [30], organization-based [64], or other, these systems all give up some of their automation in order to achieve greater flexibility. For example, coordinating crowds like organizations [64] means entrusting individuals with the responsibility to make on-the-ground decision, and increases the amount of manual interpersonal coordination that is necessary. As a result, no two flash organizations will execute exactly the same way, making them less predictable. In contrast, workflows like Find-Fix-Verify [9] are attractive in part because they run predictably, at scale, and without intervention. Enabling adaptation increases coordination costs, trading off some of these qualities. The open question is: are these tradeoffs worth making?

Finally, our results shed light on the importance of communication for coordination and adaptation. Adapting requires communication, since all information needs to be available to make a good decision, and all workers need to be updated of the new plan. While some crowdsourcing platforms enable communication between workers or with requesters, most are designed to minimize communication. These findings demonstrate, however, that communication might help crowd workers better orchestrate their efforts and reconfigure their structure and activities when needed.

5.3 Implications for Design

In this section, we translate our design recommendations into concrete design changes to modern crowdsourcing platforms.

Limits to workflows will impact the vast majority of existing crowdsourcing tools, which focus on entirely pre-planned tasks and workflows. For example, the most common use of crowdsourcing platforms such as Amazon Mechanical Turk involves pre-designing the survey or task, which leads to errors when contingencies arise [10]. Workflow systems such as TurKit [44] and CrowdWeaver [31] percolate and amplify these errors across stages of the workflow [43], much like the Game of Telephone. Some systems allow for more open-ended coordination, for example posting and claiming tasks to a collective board as in Apparition [37], though they are limited to the forms of coordination that have been pre-programmed into them. For example, Apparition only has support for independent tasks and limited communication between workers. Non-crowdsourcing workflow tools exist in tools such as Asana and Slack, but like crowdsourcing tools, they typically expect administrative intervention to adapt or deal with contingencies. What is most clearly missing is the facility to for crowd workers to *adapt* a plan as it is executed [64].

How could modern crowdsourcing platforms such as Amazon Mechanical Turk improve their designs? A small design change with significant power would be to give workers a panic button as part of the platform's task interface — a mechanism to note when the workflow or task design is preventing them from doing what is necessary for the work to succeed. The panic button would pause either the current task or the entire workflow, allowing the requester to inspect and repair as needed, or to route that task manually around the workflow.

A second approach would be to allow workers to return a task for revision to an earlier stage in the workflow. Many crowdsourcing workflow tools do or could retain information about a task's upstream workflow stages. In this situation, a worker could indicate that the task seems to have been routed to them in error, or that there were mistaken assumptions that a previous worker made, and push the task back to a prior stage in the workflow.

More aggressively, platforms could allow requesters to pay workers to act as project managers and deal with these issues on their behalf. The worker's role would be to occasionally monitor the stream of results for evidence of contingencies, reviewing the results before returning them to the requester. They would also have the power to suggest alternatives to the requester, and if desired, author a new task design to handle it. This worker PM could also deal with any uses of the panic button or revision requests.

Finally, the workflow decisions could be directly exposed to workers. For example, when a worker is going to submit a task, the platform could show the next possible phases in the workflow, including the algorithm's default choice for what comes next, and allow the worker to edit that the default selection if it is not the right one.

5.4 Limitations

Our study has several important limitations. These limitations derive from (1) the specific workflow being studied, and (2) our sample. In this section, we reflect on how these decisions impact the generalizability of the results.

While Upwork and expert design tasks are currently the state-of-the-art in crowdsourcing and crowd research [15, 24, 48, 52, 59, 61], it is possible that the effects will differ with alternative goals or workflows. In particular, it is possible that these results do not hold with single-expertise teams such as with only software engineers, or that these results are particular to the combination of expertise types that we recruited. The results may be particular to expert macrotask crowdsourcing, and not apply to other types of crowdsourcing such as microtask crowdsourcing on Amazon Mechanical Turk. The most likely difference would be that some configurations of workers and workflows will observe a smaller proportion of errors, contingencies, and needs to adapt. The more focused the goal, the less variation there will need to be in the task. In contrast, our workflow likely accentuated such issues. However, even microtask systems have variously observed similar issues (e.g., [43]), so we would expect similar patterns to emerge even with microtask work.

Second, our study only observed one workflow, focused specifically on user-centered design. There are, of course, a wide variety of crowdsourcing workflows in research and practice, targeting many different goals. The workflow in this study is non-standard in two ways — it uses expert crowd workers rather than microtask workers (e.g., from Mechanical Turk), and it forces collaboration between two different areas of expertise. We chose this workflow because it represented one of the most complex goals achieved via a crowdsourcing workflow to date. However, it is important to note that requiring multiple areas of expertise may have accentuated coordination problems, errors, and misunderstandings. It may also have allowed more autonomy than microtask workflows, because it can assume more expertise from the workers. To the extent that this workflow is representative of a broad class of modern expert crowdsourcing workflows that seek to achieve complex goals (e.g,. [15, 48, 59, 61]), it is likely that these results will generalize. However, there certainly exist workflows that will either lessen or increase the issues observed here. More research is necessary to test the boundary conditions.

We also recognize that there exist alternative workflows for the same goal, with perhaps more detail, iteration or tasks, that we could have studied. This also limits our generalizability. However, these results resonate with prior work [27, 50, 65], suggesting that these limits to crowdsourcing workflows may generalize. For example, organizational behavior research has demonstrated that no predefined workflow for complex and interdependent goals, regardless of the amount of detail or iteration included, can predict all possible contingencies and outcomes [27, 41]. So, alternative workflows may have avoided the specific breakdowns we observed, but likely would have stumbled across other instances of the same classes of breakdowns. Future research should explore these insights across different types of goals and workflows.

Finally, a third limitation is our sample. We pursued an in-depth case study analysis with a sample of six teams composed of 22 crowd workers from Upwork. Upwork crowd experts hail from many different areas worldwide, and we cannot claim complete coverage given our sample size. Likewise, a larger sample would have afforded the ability to perform quantitative statistical comparisons and to better understand the variation within conditions. As a result, we are limited to reporting variations across conditions that were very large in magnitude, either qualitatively or quantitatively. However, the present analysis was a deep investigation of over 75 hours of log data, intermediate results, and chat transcripts, which affords us a fair measure of certainty in the effects we described.

6 CONCLUSION

This paper explores why crowds struggle to achieve complex and interdependent goals. We conduct in-depth case studies of six interdependent crowd worker teams tasked with creating a mobile web application from a napkin sketch. By comparing teams enacting pre-specified and minimally specified workflows, our analysis sheds light on the strengths and limitations of crowdsourcing workflows with different affordances. Our findings suggest that while pre-defined crowdsourcing workflows are important coordination artifacts, they can stymie crowd workers' ability to adapt in response to unplanned contingencies or goal changes. Given that many complex goals cannot be fully decomposed in advance, the insights presented in this paper emphasize the need for new types of crowdsourcing workflows and approaches that allow crowds to reconfigure as they work on evolving goals. The insights also suggest that existing platforms could provide workers with the ability to flag tasks for contingencies, either for review by the requester or for return to prior stages of the workflow. More aggressive alternative strategies that coordinate crowds as teams or organizations rather than workflows may also succeed.

ACKNOWLEDGMENTS

Thank you to the Upwork workers who participated. This work was supported by the National Science Foundation (award IIS-1351131), DARPA, the Stanford Cyber Initiative, the Stanford Institute for Research in the Social Sciences, and a Stanford Interdisciplinary Graduate Fellowship.

REFERENCES

[1] Elena Agapie, Jaime Teevan, and Andres Monroy-Hernandez. 2015. Crowdsourcing in the field: A case study using local crowds for event reporting. In *Third AAAI Conference on Human Computation and Crowdsourcing (AAAI '15).* AAAI Press.

[2] Salman Ahmad, Alexis Battle, Zahan Malkani, and Sepander D. Kamvar. 2011. The Jabberwocky Programming Environment for Structured Social Computing. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11).* ACM, New York, NY, USA, 53–63. https://doi.org/10.1145/2047196.2047203

[3] Linda Argote. 1982. Input Uncertainty and Organizational Coordination in Hospital Emergency Units. *Administrative Science Quarterly* 27, 3 (1982), 420–434. https://doi.org/10.2307/2392320

[4] Beth A. Bechky. 2006. Gaffers, Gofers, and Grips: Role-Based Coordination in Temporary Organizations. *Organization Science* 17, 1 (2006), 3–21. https://doi.org/10.1287/orsc.1050.0149

[5] Yochai Benkler. 2002. Coase's Penguin, or, Linux and The Nature of the Firm. *Yale Law Journal* (2002), 369–446.

[6] Yochai Benkler. 2013. Peer Production and Cooperation. In *Handbook on the Economics of the Internet.* 1–29.

[7] Yochai Benkler, Aaron Shaw, and Benjamin Mako Hill. 2015. Peer Production: A Modality of Collective Intelligence. In *Handbook of Collective Intelligence*, Thomas W. Malone and Michael S. Bernstein (Eds.). MIT Press, 175–204. http://mako.cc/academic/benkler_shaw_hill-peer_production_ci.pdf

[8] 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 (UIST '11).* ACM, New York, NY, 33–42. https://doi.org/10.1145/2047196.2047201

[9] 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. https://doi.org/10.1145/1866029.1866078

[10] 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.

[11] Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samuel White, and Tom Yeh. 2010. VizWiz: Nearly Real-Time Answers to Visual Questions. In *Proceedings of the 23nd Annual ACM Symposum on User Interface Software and Technology (UIST '10).* ACM, New York, NY, USA, 333–342. https://doi.org/10.1145/1866029.1866080

[12] 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. https://doi.org/10.2307/3069401

[13] 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. https://doi.org/10.2307/2393807

[14] Joel Chan, Steven Dang, and Steven P. Dow. 2016. Improving Crowd Innovation with Expert Facilitation. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16).* ACM, New York, NY, USA, 1221–1233. https://doi.org/10.1145/2818048.2820023

[15] 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. https://doi.org/10.1145/2858036.2858512

[16] 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. https://doi.org/10.1145/2470654.2466265

[17] Richard Daft. 2015. *Organization Theory and Design.* Cengage Learning.

[18] 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.

[19] Pete Deemer, Gabrielle Benefield, Craig Larman, and Bas Vodde. 2012. The Scrum Primer: A Lightweight Guide to the Theory and Practice of Scrum. (2012), 20 pages. https://doi.org/10.1093/ejo/cjq172

[20] Steven P. Dow, Anand Kulkarni, Scott R. Klemmer, and Bjorn Hartmann. 2012. Shepherding the Crowd Yields Better Work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12).* ACM, New York, NY, USA, 1013âĂŞ1022. https://doi.org/10.1145/2145204.2145355

[21] Amy C. Edmondson. 2012. *Teaming: How organizations learn, innovate, and compete in the knowledge economy.* John Wiley & Sons, San Francisco, California.

[22] Kathleen M. Eisenhardt. 1989. Building Theories from Case Study Research. *Academy of Management Review* 14, 4 (1989), 532–550. https://doi.org/10.5465/AMR.1989.4308385

[23] Samer Faraj and Yan Xiao. 2006. Coordination in Fast-Response Organizations. *Management Science* 52, 8 (2006), 1155–1169. http://mansci.journal.informs.org/content/52/8/1155.short

[24] 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. https://doi.org/10.1145/2984511.2984532

[25] Barney G. Glaser and Anselm L. Strauss. 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Aldine Publishing Co., New York.

[26] 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. https://doi.org/10.1145/2818048.2819942

[27] James D. Herbsleb and Rebecca E. Grinter. 1999. Splitting the organization and integrating the code: Conway's law revisited. In *Proceedings of the 1999 International Conference on Software Engineering (ICSE '99).* ACM, New York, NY, USA, 85–95. https://doi.org/10.1145/302405.302455

[28] Jeff Howe. 2006. The Rise of Crowdsourcing. *Wired Magazine* 14, 6 (2006).

[29] Chang Hu, Benjamin B. Bederson, Philip Resnik, and Yakov Kronrod. 2011. MonoTrans2: A New Human Computation System to Support Monolingual Translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, New York, NY, USA, 1133–1136. https://doi.org/10.1145/1978942.1979111

[30] Joy Kim, Sarah Sterman, Allegra Argent Beal Cohen, and Michael S. Bernstein. 2017. Mechanical Novel: Crowdsourcing Complex Work through Revision. In *Proceedings of the 20th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '17).* ACM, New York, NY, USA.

[31] Aniket Kittur, Susheel Khamkar, Paul André, and Robert E. Kraut. 2012. CrowdWeaver: Visually Managing Complex Crowd Work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12).* ACM,

New York, NY, USA, 1033–1036. http://dl.acm.org/citation.cfm?id=2145357

[32] 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. https://doi.org/10.1145/2441776.2441923

[33] 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. https://doi.org/10.1145/2047196.2047202

[34] Nicolas Kokkalis, Thomas Köhn, Carl Pfeiffer, Dima Chornyi, Michael S. Bernstein, and Scott R. Klemmer. 2013. EmailValet: Managing Email Overload through Private, Accountable Crowdsourcing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1291–1300. https://doi.org/10.1145/2441776.2441922

[35] Robert Kraut and Lynn Streeter. 1995. Coordination in Software Development. *Commun. ACM* 38, 3 (1995), 69–81.

[36] Anand Kulkarni, Matthew Can, and Björn Hartmann. 2012. 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. https://doi.org/10.1145/2145204.2145354

[37] 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 (CHI '15)*. ACM, New York, NY, USA. https://doi.org/10.1145/2702123.2702565

[38] 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, 23–33. https://doi.org/10.1145/2380116.2380122

[39] 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. https://doi.org/10.1145/2642918.2647349

[40] 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. https://doi.org/10.1007/s00247-004-1242-4

[41] Paul R. Lawrence and Jay W. Lorsch. 1967. Differentiation and Integration in Complex Organizations. *Administrative Science Quarterly* 12, 1 (1967), 1–47. https://doi.org/10.2307/2391211

[42] Christopher H. Lin, Mausam Mausam, and Daniel S. Weld. 2012. Dynamically switching between synergistic workflows for crowdsourcing. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI'12)*. AAAI Press, 87–93.

[43] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. 2010. Exploring Iterative and Parallel Human Computation Processes. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP '10)*. ACM, New York, NY, USA, 68–76.

[44] 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. https://doi.org/10.1145/1866029.1866040

[45] Ioanna Lykourentzou, Angeliki Antoniou, Yannick Naudet, and Steven P. Dow. 2016. Personality Matters: Balancing for Personality Types Leads to Better Outcomes for Crowd Teams. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, New York, NY, USA, 260–273. https://doi.org/10.1145/2818048.2819979

[46] Thomas W. Malone and Kevin Crowston. 1994. The interdisciplinary study of coordination. *Comput. Surveys* 26, 1 (1994), 87–119. https://doi.org/10.1145/174666.174668

[47] James G. March and Herbert A. Simon. 1958. *Organizations.* Wiley, Oxford, England.

[48] 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. https://doi.org/10.1145/2858036.2858169

[49] Jon Noronha, Eric Hysen, Haoqi Zhang, and Krzysztof Z. Gajos. 2011. Platemate: crowdsourcing nutritional analysis from food photographs. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA.

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

[51] Alexander J. Quinn and Benjamin B. Bederson. 2011. Human Computation: A Survey and Taxonomy of a Growing Field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 1403–1412. https://doi.org/10.1145/1978942.1979148

[52] 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. https://doi.org/10.1145/2642918.2647409

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

[54] Brian J. Robertson. 2015. *Holacracy: The new management system for a rapidly changing world.* Macmillan.

[55] 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. https://doi.org/10.1145/2998181.2998300

[56] Donald A. Schon. 2008. *The Reflective Practitioner: How Professionals Think In Action.* Basic Books. https://books.google.com/books?id=TyPLBQAAQBAJ

[57] Klaas-Jan Stol and Brian Fitzgerald. 2014. Two's company, three's a crowd: a case study of crowdsourcing software development. In *Proc. ICSE '14*. https://doi.org/10.1145/2568225.2568249

[58] Lucy Suchman. 2007. *Human-Machine Reconfigurations: Plans and Situated Actions* (2nd ed.). Cambridge University Press, Cambridge.

[59] 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. https://doi.org/10.1145/2858036.2858121

[60] 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. https://doi.org/10.1145/2858036.2858108

[61] Rannie Teodoro, Pinar Ozturk, Mor Naaman, Winter Mason, and Janne Lindqvist. 2014. The motivations and experiences of the on-demand mobile workforce. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*. ACM, New York, NY, USA, 236–247. https://doi.org/10.1145/2531602.2531680

[62] James D. Thompson. 1967. *Organizations in Action: Social Science Bases of Administrative Theory.* Vol. 48. 192 pages.

[63] 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. https://doi.org/10.1287/orsc.2014.0947

[64] Melissa A. Valentine, Daniela Retelny, Alexandra To, Negar Rahmati, Tulsee Doshi, and Michael S. Bernstein. 2017. Flash Organizations: Crowdsourcing Complex Work By Structuring Crowds As Organizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA. https://doi.org/10.1145/3025453.3025811

[65] 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. https://doi.org/10.2307/2094477

[66] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. 2008. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science* 321, 5895 (2008), 1465–1468. https://doi.org/10.1126/science.1160379

[67] Daniel S. Weld, Christopher H. Lin, and Jonathan Bragg. 2015. Artificial Intelligence and Collective Intelligence. In *Handbook of Collective Intelligence*. 89–114.

[68] Terry Winograd. 1987. A language/action perspective on the design of cooperative work. *Human–Computer Interaction* 3, 1 (1987), 3–30.

[69] Lixiu Yu, Aniket Kittur, and Robert E. Kraut. 2016. 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. https://doi.org/10.1145/2556288.2557371

[70] Lixiu Yu, Aniket Kittur, and Robert E. Kraut. 2016. 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. https://doi.org/10.1145/2818048.2820025

[71] Lixiu Yu and Jeffrey V. Nickerson. 2011. Cooks or Cobblers? Crowd Creativity through Combination. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 1393–1402. https://doi.org/10.1145/1978942.1979147

[72] Mengyao Zhao and Andre van der Hoek. 2015. A Brief Perspective on Microtask Crowdsourcing Workflows for Interface Design. In *2nd International Workshop on Crowdsourcing in Software Engineering*. 45–46. https://doi.org/10.1109/CSI-SE.2015.16

Received June 2017; revised August 2017; accepted November 2017