# EXPERT CROWDSOURCING WITH FLASH TEAMS AND ORGANIZATIONS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF MANAGEMENT SCIENCE AND ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Daniela Retelny
March 2017

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

**Michael Bernstein, Primary Adviser**

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

**Melissa Valentine, Primary Adviser**

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

**Pamela Hinds**

Approved for the Stanford University Committee on Graduate Studies.

**Patricia J. Gumport, Vice Provost for Graduate Education**

*This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.*

# Abstract

Paid online crowdsourcing marketplaces, such as Upwork, Freelancer and 99Designs, provide access to a global labor pool of millions of individuals with a range of expertise and experiences. By making it possible to hire workers within minutes, these marketplaces enable new ways of working and organizing. To date, however, most research has focused on microtask platforms, such as Amazon Mechanical Turk and CrowdFlower. While these platforms have enabled paid non-expert workers to complete goals like text shortening and image labeling, highly complex and interdependent goals, such as web development and design, remain out of reach. Goals of this nature require deep knowledge of the subject matter and cannot be decomposed into independent microtasks for anyone to complete.

This thesis shifts away from microtask work and introduces on-demand experts as a core component of crowdsourcing systems. In doing so, we propose and evaluate two new organizational forms, which are computationally-powered. Specifically, we present three projects demonstrating how crowdsourcing systems can assemble experts from the crowd into organizational structures and enable them to achieve complex and interdependent goals previously out of reach to microtask crowds.

The first project, flash teams, introduces a framework for dynamically assembling and computationally managing crowdsourced expert teams that can achieve goals such as web development, animated videos and online learning platforms. Flash teams are composed of structured crowdsourcing workflows that are parameterized by expertise. The team structures are lightweight, reproducible and scalable, which allow them be replicated, recombined, optimized and computationally generated. The second project explores the strengths and limits of the flash teams approach through an in-depth case comparison of interdependent expert crowd teams tasked with a creating a mobile web application. This project reveals that while computational work structures, such as crowdsourcing workflows provide coordination benefits, they inhibit crowds from adapting to contingencies and evolving goals as the work proceeds. To overcome this limitation and enable expert crowds to achieve open-ended goals, the third project introduces flash organizations, which are rapidly assembled and reconfigurable organizational structures composed of large groups of expert crowd workers. By

computationally recruiting experts on-demand and enabling them to adapt in real-time, we demonstrate how flash organizations can achieve large-scale open-ended goals, such as enterprise software applications, games and emergency response infrastructure.

Taken together, this thesis envisions a future of work in which digitally networked teams and organizations dynamically assemble from a globally distributed online workforce and computationally orchestrate their efforts to accomplish complex work.

# Acknowledgments

This thesis is dedicated to a long list of individuals who have supported me in so many different ways throughout graduate school, including:

- my husband, Andrew, who has been my number one supporter throughout graduate school, listened to countless practice talks and provided endless amounts of love and encouragement over the last four years, especially during some of the most stressful moments - I can't wait for what's to come, I love you.
- my parents, Gary and Angela, and my sister and brother-in-law, Jessica and Jason, whose constant encouragement, support, understanding and unconditional love got me to where I am today - there are no words to express my gratitude, admiration and appreciation for everything they have taught me and all of the opportunities they have made possible for me.
- my co-adviser, Michael Bernstein, who welcomed me into the Stanford HCI Group, took me on as his student, listened to my crazy interdisciplinary ideas and always encouraged me to think "big" - this thesis is very much an outcome of Michael's encouragement, support and guidance as well as the ambition he instilled in me.
- my co-adviser, Melissa Valentine, who has always been open to helping me explore new ways of bridging organizational behavior, human-computer interaction and computer science and more importantly, always encouraged me to follow my heart and interests regardless of what department or discipline they were in - this thesis would not have been possible without Melissa's unwavering support and dedication.
- my committee member and collaborator, Pamela Hinds, who accepted me to Stanford and the Center for Work, Technology and Organization (WTO), advised me for the first two years of graduate school and taught me a lot of what I know about research, teaching, writing and academic communities.
- my thesis chair and mentor, Jeff Hancock, who I have been fortunate to know since I was a freshman at Cornell and is responsible for getting me excited about computer-mediated communication, research and teaching - I am so happy that our paths crossed again at Stanford.

- my undergraduate adviser at Cornell and lifelong mentor, Geri Gay, who encouraged me to stick with Information Science when I was unsure if it was the right major for me and took a chance on me by letting me lead cutting-edge research projects as a Sophomore - I would not be where I am today without Geri's constant encouragement, support and wisdom.
- my flash teams and flash organizations collaborators who contributed many ideas, helped with data collection, and wrote many lines of code, including: Alexandra To, Negar Rahmati, Tulsee Doshi, Sebastien Robaszkiewicz, Jay Patel, Walter Lasecki, Michael Kim, Maxine Fonua, Maggie Goulder, Kartik Sawhney - these projects were truly a team effort and I am forever grateful for all of their hard work and passion about shaping the future of crowd work.
- my collaborators and colleagues at the Center for Work, Technology and Organization, many of whom have graduated and remained close friends, including: Bobbi Thomason, Daisy Chung, Joachim Lyon, Hatim Rahman, Zach Rodgers, Gonzalo Valdez and Ece Kaynak.
- all of my colleagues and friends at the Center for Work, Technology and Organization, the Stanford HCI Group, the Stanford Technology Venture Program (STVP), the Department of Management Science and Engineering and the Computer Science Department - I feel very fortunate to have been surrounded by such vibrant, innovative and supportive academic communities filled with so many incredible, interesting and inspiring people.
- all of the staff and admins who have supported me along the way, including: Lori Cottle, Tim Keeley, Juanita Winkleman, Jillian Lentz and Justyna Staresnick.
- all of the expert crowd workers from Upwork as well as the other participants of this research - without their time, efforts and contributions, none of this research would have been possible.

Finally, I would like to thank and express my appreciation and gratitude to all of the organizations that have funded my graduate education and research over the last five years, including: the Kleiner Perkins, Mayfield and Sequoia Capital Graduate Fellowship (through the Stanford School of Engineering), the Stanford Interdisciplinary Graduate Fellowship (SIGF), the Stanford Institute for Research in the Social Sciences (IRiSS) Computational Social Science Research Grant, the HPI-Stanford Design Thinking Research Grant, Accenture Technology Labs University Grant, the Microsoft Research FUSE Labs Peer Economy Research Award, the National Science Foundation and Upwork (formerly oDesk). This research would not have been possible without the generosity and support of these organizations over the last five years - thank you.

# Contents

**Abstract** **v**

**Acknowledgments** **vii**

**1 Introduction** **1**

1.1 Contributions . . . . . 3

1.2 Thesis Overview . . . . . 5

1.2.1 Chapter 3: Expert Crowdsourcing with Flash Teams . . . . . 5

1.2.2 Chapter 4: No Workflow Can Ever Be Enough: How Crowdsourcing Workflows Support and Constrain Complex Work . . . . . 5

1.2.3 Chapter 5: Flash Organizations: Crowdsourcing Complex Work By Structuring Crowds As Organizations . . . . . 6

**2 Related Work** **7**

2.1 Crowdsourcing . . . . . 7

2.1.1 Source of the Crowd . . . . . 8

2.1.2 Voluntary Crowdsourcing . . . . . 8

2.1.3 Paid Crowdsourcing . . . . . 9

2.1.4 Expertise of the Crowd . . . . . 10

2.2 Crowdsourcing Complex Work . . . . . 10

2.2.1 Interdependent Crowd Work . . . . . 11

2.2.2 Expert Crowd Work . . . . . 12

2.2.3 Crowdsourcing Workflows and Algorithms . . . . . 12

2.3 Organizational Behavior . . . . . 14

2.3.1 Team and Organizational Structures . . . . . 14

2.3.2 Distributed Work . . . . . 17

2.4 Conclusion . . . . . 18

**3 Expert Crowdsourcing with Flash Teams** **19**
3.1 Introduction . . . 19
3.2 Related Work . . . 21
3.2.1 Crowdsourcing With Experts . . . 22
3.2.2 Organizational Behavior . . . 22
3.3 Flash Teams . . . 23
3.3.1 Flash Team Composition . . . 23
3.3.2 Runtime and Coordination . . . 25
3.3.3 Computationally-Enhanced Flash Teams . . . 26
3.4 Example Flash Teams . . . 29
3.4.1 Napkin Sketch Design Team . . . 29
3.4.2 Animation Team . . . 31
3.4.3 Education Team . . . 32
3.4.4 Composite Team: Online Course Platform . . . 33
3.5 Field Experiment . . . 35
3.5.1 Method . . . 35
3.5.2 Results . . . 36
3.6 Discussion and Future Work . . . 37
3.7 Conclusion . . . 39
3.8 Acknowledgments . . . 39

**4 No Workflow Can Ever Be Enough: How Crowdsourcing Workflows Support and Constrain Complex Work** **40**
4.1 Introduction . . . 40
4.2 Related Work . . . 42
4.2.1 Workflows as Coordination Artifacts . . . 43
4.2.2 Crowdsourcing Complex Work . . . 45
4.3 Method . . . 47
4.3.1 Data and Analysis . . . 48
4.4 Enacting and Adapting Workflows . . . 50
4.4.1 Workflows Support Coordination . . . 50
4.4.2 Workflows Inhibit Adaptation . . . 54
4.5 Discussion . . . 58
4.5.1 Why Do Crowds Struggle with Complex Goals? . . . 58
4.5.2 Implications for Design . . . 59
4.5.3 Limitations . . . 61
4.6 Conclusion . . . 61

**5 Flash Organizations: Crowdsourcing Complex Work By Structuring Crowds As Organizations** **62**
5.1 Introduction . . . 62
5.2 Related Work . . . 64
5.2.1 Crowdsourcing Workflows . . . 65
5.2.2 Organizational Design and Distributed Work . . . 66
5.2.3 Peer Production . . . 67
5.3 Flash Organizations . . . 67
5.3.1 Computational Organizational Structures . . . 67
5.3.2 Reconfigurable Organizational Structures . . . 69
5.3.3 On-Demand Hiring of Expert Crowd Workers . . . 72
5.4 Evaluation . . . 72
5.4.1 Method . . . 73
5.5 Results . . . 74
5.6 Discussion . . . 79
5.6.1 Limitations . . . 80
5.7 Conclusion . . . 81
5.8 Acknowledgments . . . 81

**6 Discussion** **82**
6.1 Expert Crowd Work Tradeoffs . . . 82
6.1.1 Hiring . . . 82
6.1.2 Motivation and Incentives . . . 84
6.1.3 Speed and Scale of Expert Crowds . . . 85
6.1.4 Quality Control . . . 85
6.1.5 Goal Decomposition and Work Structures . . . 86
6.2 Uses of Expert Crowd Work . . . 87
6.2.1 Bounded, Decomposable and Routine Goals . . . 87
6.2.2 Aligning Incentives, Available Resources and Risk Factors . . . 88

**7 Conclusion and Future Work** **91**
7.1 Summary of Contributions . . . 91
7.2 Impact and Recent Developments . . . 92
7.3 Future Work . . . 93
7.3.1 Flexible Work Structures and Coordination Approaches . . . 93
7.3.2 Expert Crowd Work Tools, Platforms and Marketplaces . . . 94
7.3.3 Internal Crowds and Hybrid Organizations . . . 95
7.4 Looking Ahead . . . 96

**A Flash Organizations** **97**
A.1 Tables and Figures . . . . . 97

**Bibliography** **111**

# List of Tables

1.1 This thesis introduces techniques for crowdsourcing interdependent tasks with expert workers. Up until now, even though online crowdsourcing marketplaces provide access to experts, most research has focused on crowdsourcing tasks that can be completed by homogeneous workers. . . . . 4
1.2 An overview of the design axes and contributions of this thesis. Specifically, this thesis introduces and evaluates two generalizable approaches for crowdsourcing complex goals with expert crowd workers at both the team and organizational level. . . . . 4

3.1 Eight flash teams created mobile web applications, video animations, and an online course platform with three courses. . . . . 29
3.2 Time and cost comparison for the flash and control teams. On average, flash teams took half as many work hours than control teams. . . . . 36
3.3 Average work time by role for the flash and control teams. The flash teams finished the tasks for each role faster than the control teams. . . . . 37

4.1 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. . . . . 52

5.1 Coordination infrastructure and labor source in peer production, crowdsourcing, and traditional and flash organizations. . . . . 65
5.2 Automated hiring and organizational reconfigurations. . . . . 78
5.3 The flash organizations used top-down pull requests per leaders' directives, and bottom-up pull requests per workers' initiative. . . . . 78
5.4 Organizations used on-demand hiring to spin up expertise. . . . . 79

A.1 Hiring time for the automated hires in the three flash organization deployments. *Hiring time was defined as the time lapsed between when the automated hiring email was sent and first engagement occurred, defined as the earliest of logging into the Foundry timeline view or joining the organization's Slack collaboration tool. . . 97

A.2 Hiring time for the warm hires in the three flash organization deployments. *Hiring time was defined as the time lapsed between when the hiring request was made and first engagement occurred, defined as the earliest of logging into the Foundry timeline view or joining the organization's Slack collaboration tool. . . . . 98
A.3 The number and source of the organizational structure pull requests for the three flash organizations. *Calculated by dividing the total number of requests by the number of active days for the organization. . . . . 98
A.4 An overview of the number of tasks, workers, leads and total team members for each of the nine teams and three individuals in the EMS Report organization. *These numbers represent the total number of unique workers, leads and team members, respectively. Some people filled these positions for more than one team. . . . . 99
A.5 An overview of the number of tasks, workers, leads and total team members for each of the ten teams and two individuals in the True Story organization. *These numbers represent the total number of unique workers, leads and team members, respectively. Some people filled these positions for more than one team. . . . . 100
A.6 An overview of the number of tasks, workers, leads and total team members for each of the five teams in the Enterprise Workshop Portal organization. *These numbers represent the total number of unique workers, leads and team members, respectively. Some people filled these positions for more than one team. . . . . 100
A.7 The full set of panels utilized by the three flash organizations, as well as the number of workers in each stage of the panel creation pipeline. . . . . 101

# List of Figures


3.1 A flash team is a linked set of modular tasks that draw upon paid experts from the crowd, often three to six at a time, on demand. The napkin sketch design team follows the user-centered design process to create a series of prototypes and iterate based on feedback to produce a user-tested software prototype within a day. Multiple arrows indicate the beginning and end of pipelining; lighter bars indicate possible elastic growth. . . . . 20

3.2 The basic unit of a flash team is a *block*, which gathers one or more paid experts from the crowd to perform a focused task. Blocks can be connected via shared inputs and outputs. . . . . 24

3.3 Foundry allows end users to author and manage flash teams. In authoring mode, the user can specify the expertise for each block and enter details about each block's requirements. . . . . 25

3.4 Foundry's runtime mode: the design mockups finished 30 minutes late but implementation finished early. The current user test block, marked by a red playhead, involves both (orange and green) experts. . . . . 26

3.5 Sketched inputs and high-fidelity prototype results from the napkin sketch design teams. This team iterates from a sketch through low fidelity prototypes to produce a working, user-tested software prototype within a day. . . . . 30

3.6 The workflow for the animation flash team, which takes a high-level script outline as input and produces a short animated movie as output. . . . . 32

3.7 The animation team produced a short movie of a young Terry Winograd building a computer in his garage. . . . . 33

3.8 Course videos: singing from the diagram (top), portrait photography (left and bottom), and the Towers of Hanoi puzzle (right). . . . . 34

4.1 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. . . . . 48
4.2 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. . . . . 49
4.3 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. . . . . 50

5.1 Flash organizations are crowds that are computationally structured like organizations. They enable automated hiring of expert crowd workers into role structures and continuous reconfiguration of those structures to direct the crowd's activities toward complex goals. . . . . 63
5.2 Foundry supports the authoring of flash organizations. The Foundry timeline (top left) displays all roles and current tasks. The task description (top right) displays task requirements and allows the leader to hire a worker to fill the role. When hiring for a new role (bottom left), Foundry notifies qualified workers; those who respond enter a hiring queue. Once hired, they are oriented to the purpose of the organization and upstream and downstream tasks (bottom right). . . . . 68
5.3 Workers can branch the current organizational structures, make any desired edits, and then issue a pull request for review. This mechanism enables the organization to continuously adapt. . . . . 70
5.4 Three flash organizations successfully developed: (top) a tablet application and web portal for emergency medical responders; (middle) art, content, and a supporting application for a storytelling card game; and (bottom) an enterprise IT portal for consultant workshop planning. . . . . 73
5.5 Final organizational structures and task timelines for the three flash organizations. Colors indicate different roles and corresponding tasks (details in Appendix). Structures varied from flat to nested hierarchies, and included 24 teams and 639 tasks across 3,261 person-hours of work time. . . . . 75

A.1 Organizational structure (top) and workflow (bottom) for the EMS Report organization. The organization was composed of the CEO, nine teams, and three individual workers. The workflow shows the timing of the 390 completed tasks. Colors in the organizational structure and workflow indicate the respective organizational units, listed top right. . . . . 102

A.2 Organizational structure (top) and workflow (bottom) for the True Story organization. The organization was composed of the CEO, 10 teams, and two individual workers. The workflow shows the timing of the 122 completed tasks. Colors in the organizational structure and the workflow indicate the respective organizational units, listed top right. . . . . 103

A.3 Organizational structure (top) and workflow (bottom) for the Enterprise Workshop Portal organization. The organization was composed of the CEO and five teams. The workflow shows the timing of the 127 completed tasks. Colors in the organizational structure and workflow indicate the respective organizational units, listed top right. 103

A.4 Screenshots from the EMS Trauma Report Android Application. After opening the application, users can create an account or login with an existing account. After logging in, the application allows users to either create a new trauma case or update a current case. When adding a new Trauma case, the user will be presented with a different set of screens depending on whether it is a Trauma 1 or 2 case. The application automatically identifies the location of the case and continues to track the location of the device until the patient reaches the hospital. . . . . 104

A.5 Screenshots from the EMS Trauma Report website, which is monitored by staff at the hospital. After logging in, users are shown a dashboard with new and old cases as well as relevant statistics. Similar to the mobile application, users can create new Trauma 1 and 2 cases and update existing cases. Users with administrative privileges have the ability to manage and update all user accounts. . . . . 105

A.6 Photographs of the True Story game. The organization created two separate card decks, which included a family friendly version (the blue deck) and an adult only version (the red deck). Each card deck consisted of 80 story prompt cards and an instruction card for each of the three rounds in the game (e.g., 3 instruction cards total). Each story prompt card featured a topic and a descriptive quote aimed at helping game players recall a relevant story related to the prompt. The back of all cards in the deck had the same graphic, which was either blue or red depending on if it was the family friendly or adult only deck. . . . . 106

A.7 Screenshots of the True Story mobile application and website. When opening the application for the first time, the user is presented with a brief four-screen tutorial. They can always view the tutorial again by clicking on the question mark icon on the main screen. The application allows users to record the audio of their stories and upload it to the True Story web server. Once users upload their story, they receive an email with the link to access their story on the True Story website. From there, users have the option to download the story, save the story to their Google Drive account or share the link to their story with others. Users can also access their previous recordings from the last 30 days. . . . . 107

A.8 Screenshots of the enterprise workshop portal worker view. After logging in, users are taken to the portal welcome page. From there, they can access the workshop dashboard screen. On the dashboard page, users can request a new workshop and see their in progress and completed workshops. When requesting a new workshop, users are presented with a form with six different sections. In order to submit the form, they must answer all of the required questions across all sections. Once the user has submitted the workshop request form, the status of the request is "pending" until it has been approved and assigned by an administrator. During this time, users can view and edit their request forms. Once the request has been approved, the status updates to "in progress" and the user is given access to the "collaboration panel" for the workshop, which allows them to update the status of specific tasks, such as whether the catering has been ordered, as well as upload relevant files, such as an agenda or menu. . . . . 109

A.9 Screenshots of the enterprise workshop portal administrative view. After logging in, administrative users are taken to the portal welcome page. From there, they can access the workshop dashboard screen. On the dashboard page, administrative users can request a new workshop, see their in progress and completed workshops and access the content management system. The CMS allows administrative users to view and assign unassigned workshops as well as access, edit, filter and export any of the other workshops in the system. . . . . 110

# Chapter 1

# Introduction

Over the last decade, paid online labor marketplaces have emerged and enabled a new type of computational labor economy that has broadened crowdsourcing from volunteerism (e.g., Wikipedia) to paid work. These new mediated labor markets have enabled paid, online crowd workers to quickly come together and complete tasks that would otherwise be extremely difficult for a single individual to achieve at the same scale. As a result, we are starting to see the emergence of new organizational forms composed of individuals drawn from a global labor pool of workers with diverse expertise, skill levels, backgrounds and perspectives [19, 95].

Crowd work today is effective for simple independent tasks, but many real-world tasks require significantly more complexity and coordination among workers and requesters [95]. For paid, online crowd work to reach its potential, crowds need to have the ability to complete creative, interdependent and highly complex tasks. To date, a majority of crowdsourcing platforms and approaches are designed to recruit amateurs from crowd marketplaces, such as Amazon Mechanical Turk, to accomplish microtasks that require little training, including document editing [17], image labeling [159], nutrition analysis [136], translation [79] and visual question answering [20]. While these systems start to demonstrate the power of the crowd, they are not designed to support the breadth of expertise and interdependencies shared between crowd workers or ease the burden encountered by requesters when managing complex open-ended projects, such as design and video production. Specifically, when task units are complex and interdependent, they become much harder to decompose into homogenous microtasks, especially when they require a certain level of expertise to complete.

To drive crowdsourcing past this complexity boundary, this thesis shifts away from microtask work and introduces expert crowds as a core component of crowdsourcing systems. Specifically, *this thesis proposes that combining expert crowds with computational techniques and organizational structures can enable crowds to achieve complex and interdependent goals quickly, reliably and at scale*. By introducing experts from online labor markets, such as Upwork, into crowdsourcing systems and orchestrating their efforts through organizational structures and algorithmic techniques

embedded in computational systems, we open up a new set of complex goals that can be achieved through crowd work while maintaining the scale and computational affordances of crowdsourcing.

Expert crowdsourcing currently lacks generalizable techniques for guiding crowds through complex and interdependent tasks. While there are marketplaces that provide access to experts, most research has focused on techniques for crowdsourcing with non-expert workers (see Table 1.1). Overcoming these challenges requires a multidisciplinary approach with contributions from both (1) behavioral science, such as methods and theories from organizational behavior, which demonstrate how to manage team structures, how people are motivated and the way work is organized; and (2) computer science, which provides insights and frameworks for developing the technology necessary to investigate and extend our capability in this area. This research aims to fill this critical gap by building on previous research from organizational behavior, human-computer interaction and computer science to develop generalizable techniques and a computational system designed to support the creation, coordination and management of interdependent expert crowds at both the team and organization level.

This thesis presents three projects, which introduce and evaluate two generalizable approaches along with a computational platform for crowdsourcing complex work with experts. The first project, flash teams, is a framework for dynamically assembling and computationally managing crowdsourced expert teams. Flash teams are composed of structured crowdsourcing workflows that are parameterized by expertise. By breaking open the expertise barrier and organizing crowd workers into computationally managed team structures, flash teams enable crowds to achieve complex and interdependent goals, including mobile web applications, animated videos and an online MOOC platform with content, quickly and reliable. Furthermore, because flash teams have lightweight, reproducible and scalable team structures, they can be replicated, recombined, optimized and computationally generated. These affordances allow flash teams to be authored with little effort and even scale to other types of complex goals.

In the second project, we evaluate the extent to which the structured crowdsourcing workflows introduced by the flash teams support and constrain complex work. To do this, we conduct an in-depth case comparison of flash teams enacting pre-specified crowdsourcing workflows and role-based teams enacting minimally specified work plans. Our analysis reveals that while crowdsourcing workflows provide coordination benefits, they inhibit crowd workers from adapting to contingencies and evolving goals as the work proceeds. Given that many complex goals cannot be decomposed in advance, this finding highlights a critical limitation of existing crowdsourcing approaches and demonstrates that using traditional crowdsourcing techniques with experts is not sufficient for larger, open-ended tasks.

In order to fully achieve our vision of expert crowd work, it is critical for expert crowds to be able to accomplish complex and *open-ended* work. Therefore, the third project, flash organizations,

overcomes the limitations of pre-defined computational workflows by structuring crowds as organizations instead of strictly relying on crowdsourcing workflows. To achieve this, flash organizations introduce a framework for creating rapidly assembled and reconfigurable organizational structures composed of large groups of expert crowd workers. This structure then allows flash organizations to recruit members on-demand through an instant hiring approach and adapt in real-time using an approach inspired by distributed version control. Our deployment of three flash organizations demonstrates that these reconfigurable organizational structures and computational techniques enable expert crowds to achieve highly complex and open-ended goals similar to those achieved by traditional, offline organizations.

These approaches for expert crowd work are manifested in Foundry, a computational platform we created for authoring and managing teams and organizations of expert crowd workers. Foundry enables project requesters to create the flash team and flash organization structures, recruit experts from the crowd and oversee all tasks. Foundry also takes on managerial responsibilities for crowd workers by providing them with all of the information needed to get onboarded and complete their task as well as by facilitating coordination and collaboration among team members through structured interactions, handoffs and real-time task tracking.

## 1.1 Contributions

The core contributions of this thesis are twofold. First, it introduces and evaluates a novel combination of expert crowds and computational techniques for completing highly complex and interdependent goals, which crowds previously struggled to achieve. To date, most crowdsourcing approaches and systems have focused on decomposing goals into independent microtasks that can be completed by non-expert workers and recombined to create larger deliverables. This thesis shifts away from microtasks and opens up a wide range of complex and expert-based goals that can be achieved through crowdsourcing. Second, this thesis combines literature, theories and methods from organizational behavior, human-computer interaction, computer science and related fields to inform the design and understanding of crowdsourcing techniques and the computational system in which these techniques are embedded. These two contributions have implications for scaling the capabilities of crowds, structuring distributed expert crowd work and designing crowdsourcing systems.

Taken together, this thesis advances a vision of expert crowd work that accomplishes complex, interdependent goals, such as engineering and design. Completing these goals requires domain specific knowledge and skills, making them difficult to decompose into simple microtasks that anyone can complete. Furthermore, complex goals cannot be systematically decomposed into independent tasks that can be recombined at the end. These goals are highly interdependent and therefore require different organizational forms than those typically used in microtask crowdsourcing. Therefore, as shown in Table 1.1, this work moves away from independent homogeneous workers and introduces

interdependent expert workers into crowdsourcing systems.

| | **Independent Tasks** | **Interdependent Tasks** |
|---|---|---|
| **Homogeneous Workers** | Soylent [17], VizWiz [20], CrowdForge [96] | Legion [105], Apparation [103] |
| **Expert Workers** | Upwork [166], Freelancer [54], 99Designs [1] | *Flash Teams (Chapter 3), Flash Organizations (Chapter 5)* |

**Table 1.1: This thesis introduces techniques for crowdsourcing interdependent tasks with expert workers. Up until now, even though online crowdsourcing marketplaces provide access to experts, most research has focused on crowdsourcing tasks that can be completed by homogeneous workers.**

Interdependent goals vary in scale and complexity. For example, some goals can be broken down into smaller components and assigned to specific roles in advance [148]. In contrast, as the scale and complexity of goals increase, they become more open-ended and neither tasks nor roles can be completely specified in advance. To enable expert crowds to achieve goals along the spectrum of scale and complexity, this thesis introduces generalizable techniques at both the team and organization level. These techniques are manifested in a computational platform, Foundry, which demonstrates how these approaches can be embedded into interactive systems that can then be used by expert crowd workers and requesters alike. Finally, as shown in Table 1.2, in addition to introducing new techniques and infrastructure for expert crowd work, this thesis develops an understanding of how they are used in practice, which has important social and technical implications.

| Level | **Design** | **Understand** |
|---|---|---|
| **Team** | Flash Teams (Chapter 3) | Work structure enactment (Chapter 4) |
| **Organization** | Flash Organizations (Chapter 5) | |

**Table 1.2: An overview of the design axes and contributions of this thesis. Specifically, this thesis introduces and evaluates two generalizable approaches for crowdsourcing complex goals with expert crowd workers at both the team and organizational level.**

Flash teams and organizations point towards a future in which experts can dynamically come together and achieve goals that previously could only be achieved by traditional teams and organizations. Furthermore, enabling digitally networked teams and organizations has the potential to drastically increase the size of the global workforce and raise the complexity ceiling currently limiting the types of goals crowds can achieve. Given that expert freelance workers are the fastest growing component of the economy [44] and that technology continues to break down geographical and global barriers [95], the coordination infrastructure and computational systems introduced in this thesis takes online work one step closer to achieving a future in which complex expert work can happen anytime and anywhere.

## 1.2 Thesis Overview

This thesis begins with Chapter 2, which presents an overview of the related work and a discussion of the research challenges encountered, both of which motivate this research. The main contributions are then divided into three chapters (see Table 1.2), which are summarized in the subsections that follow. Chapter 3 introduces flash teams [148], which bring together expert crowd workers to complete complex goals using role-based team scaffolds [169] and modular workflows. Chapter 4 evaluates the extent to which pre-specified and minimally specified work structures support and constrain complex crowd work. Chapter 5 then introduces flash organizations, which overcome the adaptation challenges revealed in Chapter 4 and allow crowds to reconfigure their structure as the work proceeds. To conclude, Chapter 6 lays out the opportunities for expert crowd work, sheds light on design tradeoffs and implications and highlights some of the challenges and limitations of expert crowd work. Finally, Chapter 7 wraps up with a summary of the contributions of this research as well as future research directions.

### 1.2.1 Chapter 3: Expert Crowdsourcing with Flash Teams

This chapter introduces *flash teams*, a framework for dynamically assembling and managing interdependent paid experts from the crowd. Flash teams consist of sequences of linked modular tasks and handoffs that can be computationally managed. This enables interactive systems to reason about and manipulate these teams' structures. For example, flash teams can be recombined to form larger collectives and authored automatically in response to a user's request. Flash teams can also hire more people elastically in reaction to task needs, and pipeline intermediate output to accelerate completion times. To enable flash teams, this research also presents Foundry, an end-user authoring platform and runtime manager. Foundry allows users to author modular tasks, then manages teams through handoffs of intermediate work. We demonstrate that Foundry and flash teams enable crowdsourcing a broad class of goals including design prototyping, course development, and film animation, in half the work time of traditional self-managed teams.

### 1.2.2 Chapter 4: No Workflow Can Ever Be Enough: How Crowdsourcing Workflows Support and Constrain Complex Work

Crowdsourcing workflows coordinate distributed workers by decomposing goals into small independent tasks. However, complex goals such as design and engineering have remained difficult to crowdsource. This chapter explores the reasons for this apparent complexity limit by conducting in-depth case studies of 22 Upwork workers in six teams that were 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 these teams enacted and adapted crowdsourcing workflows. We find that while workflows served as useful coordination

artifacts, they inhibited crowd workers from pursuing important real-time adaptations to their work plans. Unfortunately, complex goals often require open-ended adaptation as work proceeds, suggesting that workflow-based crowdsourcing approaches may prove fundamentally incompatible with complex goals. We conclude with a discussion of how crowdsourcing workflows and approaches can better support complex crowd work.

### 1.2.3 Chapter 5: Flash Organizations: Crowdsourcing Complex Work By Structuring Crowds As Organizations

As the scale and complexity of goals increase, it becomes much harder to modularize and define the specific actions and roles needed to achieve them. Traditional organizations, for example, regularly bring together large groups to collectively take on open-ended and highly complex goals. Therefore, new crowdsourcing techniques are needed to achieve our vision and enable expert crowds to achieve goals of similar scale and complexity.

This chapter introduces *flash organizations*, which are rapidly assembled and reconfigurable organizations composed of crowd workers. This technique arranges crowd workers into organizational structures such as roles, teams and hierarchies, automatically hires to populate the structures within minutes, and reconfigures the structures to responsively adapt the organization's activities. To support the creation and deployment of flash organizations, this research introduces several new features and advancements into Foundry, including the ability to instantly issue calls and hire experts for open roles, computationally onboard new workers, and dynamically reconfigure the organizational structures using a technique inspired by distributed version control. Our deployment demonstrates that flash organizations can successfully carry out highly open-ended and interdependent goals, such as software development, product design, and game production. Furthermore, this research demonstrates how digitally networked organizations can assemble global expertise near-instantaneously to accomplish complex work.

# Chapter 2

# Related Work

This thesis combines research from crowdsourcing, human-computer interaction and organizational behavior.

## 2.1 Crowdsourcing

*"Technological advances in everything from product design software to digital video cameras are breaking down the cost barriers that once separated amateurs from professionals. Hobbyists, part-timers, and dabblers suddenly have a market for their efforts, as smart companies in industries as disparate as pharmaceuticals and television discover ways to tap the latent talent of the crowd. The labor isn't always free, but it costs a lot less than paying traditional employees. It's not outsourcing; it's crowdsourcing."* - Jeff Howe [77]

The notion of crowdsourcing was coined in a 2006 WIRED article written by Jeff Howe [77]. In this article and the book that followed [78], Howe draws parallels between crowdsourcing and outsourcing. Specifically, he emphasizes that unlike before when companies would look to India or China for cheap labor, it no longer matters where workers reside as long as they are connected to the Internet. On his website [76], Howe defines crowdsourcing as "the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call."

Since the term was introduced a decade ago, crowdsourcing has grown rapidly both in industry and research communities across a range of disciplines [145]. Crowds have written the world's largest encyclopedia [94], discovered protein structures [31], solved math problems [33], helped digitize books [174] and developed software [108]. While these examples only capture a fraction of the crowds' achievements, they start to illustrate the different types of crowd, uses cases and applications. For example, crowds can be recruited through calls to open or known sets of contributors who can either be volunteers or paid. Furthermore, crowd workers can be non-experts and complete homogeneous

tasks or highly specialized and complete complex, skill-based goals. Each of these types of crowds, which we describe below, have different tradeoffs and require different recruitment and coordination techniques.

### 2.1.1 Source of the Crowd

Crowds can be recruited from many different pools of individuals, each of which requires a different type of call or recruiting strategy. We adopt the terminology used by Lasecki and colleagues [102], which refer to crowds as either open or known, depending on where they are recruited from and whether contributors are known to the requester.

Open crowds are calls that are available to any contributor or set of contributors [102]. These calls can be used to recruit crowds from paid marketplaces, such as Mechanical Turk and Upwork, as well as to recruit voluntary crowds who are intrinsically motivated to complete the task. Most known examples of crowdsourcing involve open crowds. For example, open crowds have edited text [17], produced the largest encylopedia [94], solved math problems [33], identified galaxies [115] and created web applications [148]. In all of these examples, contributors were recruited through open calls to open crowds unknown to the initial requester.

In contrast, known crowds are composed of individuals known to the requester in some way. These could be friends, colleagues or family as well as individuals who are known to be trusted [102]. Because of the established trust and familiarity between requesters and contributors, known crowds reduce some of the risks and challenges faced when recruiting and coordinating crowds. Examples of known crowds have included academic conference program committees [5, 28] and location-based targeted communities [67]. These known crowds have collaboratively created conference session agendas and graded programming exam questions, respectively.

### 2.1.2 Voluntary Crowdsourcing

Voluntary crowdsourcing, which has also been referred to as collaborative crowdsourcing [19], refers to crowds that are intrinsically motivated to contribute to a task. Prior research has found that voluntary crowds are motivated for a variety of reasons [153], including a personal interest in the topic, the opportunity to learn new skills or the desire to explore novel opportunities.

Volunteer-based crowds are responsible for some of the most well known crowdsourcing achievements. Volunteer crowds, for example, have written Wikipedia [94], which is the world's largest encyclopedia, as well as discovered protein structures [31], helped find a missing person [92], discovered new galaxies [115] and solved unresolved math problems [33]. Voluntary crowds have also been used to complete monotonous tasks that are traditionally less enjoyable. For example, by embedding repetitive tasks, such as labeling images, into a game [173] or incorporating tasks, such as transcribing text, into existing workflows [174], researchers have enabled voluntary crowds to label thousands of images and transcribe hundreds of millions of words.

### 2.1.3 Paid Crowdsourcing

In contrast to voluntary-based crowdsourcing, which relies on intrinsically motivated contributors, paid crowdsourcing, also known as *crowd work* [95], typically involves extrinsically motivated workers who are paid for their contributions. Specifically, in *The Future of Crowd Work*, Kittur and colleagues [95] define paid, online crowd work as "the performance of tasks online by distributed crowd workers who are financially compensated by requesters (individuals, groups, or organizations)." In most cases, paid crowd workers are recruited by requesters to complete a specific set of tasks or goal [19]. While monetary compensation is often the driving incentive and motivator for paid crowd workers, previous research has found that it is not the only motivator [6, 164, 7].

By allowing crowd workers to get compensated for their contributions, paid crowdsourcing has expanded the possibilities for the types of goals that can be completed by the crowd. These goals are typically achieved by decomposing larger tasks into smaller units of subtasks that can be completed by workers on crowdsourcing marketplaces and platforms, such as Amazon Mechanical Turk. While Mechanical Turk is widely considered the most popular and well known crowdsourcing marketplace, many other crowdsourcing platforms exist [109, 95, 168]. For example:

- Upwork, previously known as Elance-oDesk, is a crowd marketplace for recruiting freelancers with a range of expertise;

- CrowdFlower is a microtask crowd platform for recruiting workers to complete a range of data enrichment tasks;

- MobileWorks is a microtask platform focused on providing work opportunities to underemployed communities in the United States and the developing world [100];

- 99Designs is a platform for hiring designers directly or through competitions;

- Topcoder is a competition-based marketplace that provides on-demand access to designers, developers and data scientists;

Paid crowds recruited from these platforms have achieved a wide range of goals. One of the most common use case of paid crowdsourcing has been to complete repetitive and large-scale data related tasks, such as labeling and annotation [20], clustering [29], translations [79] and transcriptions [104]. As new frameworks and techniques have been developed, however, crowds have been able to take on more complex goals, such as writing and editing text [17, 163], nutritional analysis [136], prototyping interactive systems [103], video production and educational material [148], software development [108, 160] and design [120, 179].

### 2.1.4 Expertise of the Crowd

Just like any other work or collaborative environment, the skills and expertise represented among members of the crowd vary widely. While the source of the crowd, such as whether it is an open or known crowd or what platform workers were recruited from, can provide some signal of the expertise available, it does not always indicate the range and depth of skills available [102]. In particular, the research community has mostly converged on two categories of crowd workers, which include non-expert crowds and expert crowds [19, 95, 148]. As will be described below, the primary distinction between non-expert and expert crowds is whether workers have specific skills or expertise relevant to the task they are completing.

Non-expert crowds, which have also been referred to as general crowds [102], are not expected to have task-specific skills or knowledge. These workers are often recruited from microtask platforms, such as Mechanical Turk or CrowdFlower, for homogeneous microtasks that can later be aggregated and combined to create larger deliverables. Compared to expert crowd, non-expert crowds can typically be hired much more quickly and at a lower cost [102]. For example, the average task payment on Mechanical Turk ranges from $0.01 to a few dollars [81, 152], resulting in hourly wages of $2/hour with no benefits or worker protections [82, 95]. These low-pay reward structures have unfortunately led to the perception that crowd workers are homogeneous and unskilled [95], resulting in the mistreatment of crowd workers [84, 85] and more recently, has encouraged collective action efforts among workers [154].

In contrast, expert crowds have specialized skills relevant to the task they are performing. For example, online marketplaces such as 99Designs, allow you to hire designers for a wide range of design tasks. Other marketplaces, such as Upwork, provide access to millions of experts with vastly diverse skills, including software developers, accountants, video producers. In fact, as of 2014, there are 9.7 million experts and 2,700 skills available on Upwork [167]. In 2014, employers posted 2.8 million jobs and freelancers earned over 941 million dollars. 50% of the money spent was in the technology category and the other 50% was spent across a range of other categories, including (from highest to lowest spent) admin support, writing and translation, design and multimedia, mobile, sales and marketing, and finance and legal. On average, Upwork reports that it takes 3 days to hire workers from the marketplace compared to an average of 43 days in traditional organizations.

## 2.2 Crowdsourcing Complex Work

This thesis focuses on creating and managing teams and organizations composed of *open, paid, expert crowds* from Upwork. Whereas crowdsourcing is known to succeed with tasks that can easily be decomposed and require little expertise [19], this research explores the possibility of enabling much larger types of goals. In particular, the objective of this thesis is to combine computation with expert crowds to enable crowd workers to achieve *complex and interdependent goals*, many of which

have been out of reach up until now.

In this section, we summarize the research on complex crowd work and draw attention to the strengths and limitations of current systems and approaches. This research suggests that complex crowd work differs from traditional crowdsourcing in two key ways. First, complex goals are interdependent and can't be decomposed into independent self-contained subtasks. Second, many complex goals require specific skills or expertise and therefore can't be completed by homogeneous workers from platforms such as Mechanical Turk.

### 2.2.1 Interdependent Crowd Work

While crowdsourcing techniques thrive with tasks that can be decomposed into independent self-contained subtasks, complex goals are much more interdependent and therefore require significantly more coordination, time and effort [96, 122]. Specifically, complex goals cannot be decomposed into independent microtasks and later recombined to create the final deliverable. Instead, many complex goals can only be achieved by completing a set of interdependent tasks in a certain order by multiple dependent workers. This shift in complexity and interdependence has resulted in new techniques and systems designed to enable crowds to achieve larger scale goals, manage dependencies and integrate deliverables.

Most research that has explored complex crowd work with homogeneous workers has focused on techniques to decompose goals into sequences of independent tasks. For example, the frameworks and algorithms embedded in CrowdForge [96] and Turkomatic [100] relied on crowd workers to help decompose tasks and create workflows. By allowing workers to participate in the workflow design process and empowering them to decompose goals into small independent tasks, these systems enabled crowds to achieve interdependent goals, such as writing essays and articles, planning travel itineraries and researching purchases and academic department chairs.

More recent contributions, however, have introduced techniques to allow homogeneous workers to achieve complex goals by completing interdependent tasks. For example, Legion [105] introduced a system for gathering synchronous crowds to collectively control interfaces in real-time and Apparition [103] developed a system and approach to coordinate crowd workers in real-time as they translate designers' sketches and descriptions into interactive protoypes in real-time.

While existing crowdsourcing approaches have succeeded at completing interdependent tasks that can be completed by aggregating many smaller self-contained tasks, achieving truly interdependent goals remain a challenge. For example, while more routine goals can be codified into a sequence of specific tasks with pre-defined handoffs, task inputs and outputs, highly interedependent goals are more complex and uncertain and therefore require problem solving and interrelated roles [155]. Recent approaches have explored achieving interdependent goals using team-based coordination techniques. For example, results from studies conducted by Lykourentzou and colleagues [121] and Salehi and colleagues [155] found that balancing personalities in a team and increasing team familiarity,

respectively, can improve performance and team effectiveness on interdependent collaborative tasks. Although these studies have started to show the importance of thinking beyond computational and pre-specified workflows for interdependent crowd work, crowdsourcing larger scale and open-ended interdependent goals remain an open challenge and important area of research.

### 2.2.2 Expert Crowd Work

Many complex goals also require specific expertise to complete. For example, software development requires knowledge about programming languages [27] and producing educational videos requires knowledge about a specific topic as well as video production [148]. Unlike the interdependent goals in earlier work [96, 100, 103, 105], these types of goals cannot be completed by homogeneous non-expert workers.

In addition to the flash teams framework presented in Chapter 3, recent research has contributed expert crowd work systems and frameworks for complex goals, such as software development [27, 49], mentorship and skill development [162], writing [134] and physical world tasks [164]. While this research on expert crowds has expanded the types of goals crowd workers can achieve, the processes used to coordinate efforts are the same as the techniques used in microtask crowdsourcing. Specifically, microtask and expert crowd work approaches both operate under the assumption that all tasks for the project can be defined in advance, which is not always the case with complex goals [96, 160]. As a result, even though expert crowds can achieve more complex and skill-based goals, they face many of the same limitations as microtask crowds.

Completing highly complex goals continues to be an open challenge in crowdsourcing. For example, expert crowds still struggle with highly interdependent and open-ended goals, such as the wicked problems of innovation, creativity and product development [149]. Complex goals of this nature, which are dynamic and uncertain, can't be decomposed in advance and are therefore difficult to achieve with microtask crowdsourcing systems and frameworks. This limitation motivates much of our research, especially the reconfigurable organizational structures described in chapter 5.

### 2.2.3 Crowdsourcing Workflows and Algorithms

Since it became a field of study, a big focus of crowdsourcing research has been on coordination. Specifically, researchers have designed, built and evaluated a wide range of techniques, frameworks and systems for coordinating the contributions of many individual crowd workers. As will be described in this section, much of this research has been on efficient workflows and algorithms for task decomposition and integration.

As described earlier, crowdsourcing brings together large groups of individuals and tasks them with solving large scale problems [14, 77]. To enable crowds to achieve such large scale goals, crowdsourcing systems and techniques decompose the goals into much smaller tasks that can be executed by independent workers and recombined upon completion. These systems typically rely on

highly controlled crowdsourcing workflows, also known as human computation algorithms or crowd algorithms [19, 29, 110, 116].

Crowdsourcing workflows specify sequences of pre-defined and decomposed tasks, which provide a path for transforming an initial input into a desired final output. Each of the tasks in the workflow, which specify what workers are asked to do [176] are computationally assigned to distributed crowd workers to complete [19, 145]. For example, Soylent's *find-fix-verify* workflow partitions open-ended tasks into three stages, each of which is composed of a specific set of tasks [17]. The tasks in each phase are sent out to three to five workers on Mechanical Turk, whose responses are then aggregated. In the find stage, these aggregated responses are used to identify the problems most consistently cited whereas in the verify stage, the outputs from the fix stage are voted on.

While most tasks in crowdsourcing workflows are assigned to workers, some tasks may be completed by computers or a combination of humans and computers [145, 174]. For example, translation tasks can be completed by humans, machines or a combination [79]. Similarly, the combination of crowd and machine-powered tasks can be used to identify the nutritional content of a plate of food just as effectively as expert dietitians [136]. Crowdsourcing workflow systems, such as CrowdWeaver, allow task designers to author and integrate human and machine tasks into their workflows.

Crowdsourcing workflows are critical to the success of crowdsourcing and have enabled a wide range of goals. Specifically, the algorithmic approach embedded in workflows decompose larger goals into scaffolds that determine and define each step, resulting in more reliable and consistent outcomes. Examples of these algorithmic workflows include Turkomatic's *price-divide-solve* algorithm [100], Soylent's *find-fix-verify* design pattern [17] and CrowdForge's *map-reduce* paradigm [96]. For example, CrowdForge [96] introduced a workflow comprised of three types of subtasks, including partition, map and reduce, which collectively decompose and recombine goals. Partition tasks involve high level decomposition of the problem (e.g., create an outline of an article with section headers). Map tasks consist of individual tasks for each item in the partition (e.g., gather one fact for a topic in the outline). Finally, reduce tasks require workers consolidating all of the map task results into a single output (e.g., write a paragraph combining all of the facts gathered in the map tasks). This workflow enabled a wide range of tasks, including writing an article, researching purchase decisions and science journalism.

Given the important role workflows play in crowdsourcing, researchers have introduced frameworks and systems for creating and managing workflows [93, 96, 117]. For example, the CrowdForge framework provides a systematic approach for decomposing tasks and managing the dependencies between them [96]. In addition to creating workflows, researchers have created systems to ease the workflow authoring process and reduce the management overhead once the workflows are deployed. For example, the CrowdWeaver [93] system enabled task designers to visually create and manage crowd workflows using a graphical interface. The system also allowed workflow creators to create and reuse task templates and provided real-time tracking and notification of crowd factors and task

progress.

Finally, workflows are not required to follow one deterministic sequence of tasks. Over the last five years, researchers have introduced techniques and systems for creating more dynamic workflows. This work has introduced approaches for supporting iteration between tasks and subtasks in crowdsourcing workflows [36, 96, 117], techniques for real-time crowd powered workflows [16, 20, 104] as well as systems for requesters and workers to collaboratively design workflows [100] and provide timely, task-specific feedback [43]. While all potential paths and outcomes must still be predefined into workflows before they are run, these dynamic workflows have enabled higher quality outcomes [43, 100, 101] and more complex goals [96].

Three open challenges with workflows continue to persist. First, it can be difficult to determine and create the most effective appropriate workflow for a goal [19] and manage workflows once they are created [93]. Second, while workflows can be dynamic and support iterative work processes, all potential paths, contingencies and outcomes must be pre-programmed into the workflow before it is run, which prevents workflows from responding to scenarios that were not predicted or programmed a priori [96, 100]. Third, there are sociotechnical aspects and challenges to algorithmic management, which can lead to lower worker satisfaction and cooperation and fragmented social sensemaking [112].

The three projects in this thesis are in large part motivated by these challenges. Flash teams (chapter 3) introduce an approach for automatically generating effective workflows that provide the sequence of tasks needed to transform an initial input into a desired final output. The in-depth multi-method case studies in chapter 4 then explore the extent to which pre-defined crowdsourcing workflows support and constrain complex work as well as the sociotechnical implications and limitations of workflows. Finally, flash organizations (chapter 5) introduce techniques to overcome these limitations and enable interdependent crowds to reconfigure their structure and adapt in response to evolving goals.

## 2.3 Organizational Behavior

To enable expert crowd workers to achieve complex and interdependent goals, we draw on research from organizational behavior. Given that traditional teams and organizations regularly achieve complex and interdependent goals, understanding the factors and conditions that enable them to do so is highly relevant to expert crowd work. In this section, we focus on organizational behavior research that has explored the conditions and barriers to effective coordination.

### 2.3.1 Team and Organizational Structures

Decades of organizational behavior (OB) research has shed light on the important role team and organizational structures play in supporting coordination among interdependent contributors. This work has shown that even when workers are highly motivated to successfully complete their goals,

they often choose ineffective ways of organizing and coordinating their actions [66, 165]. Overcoming this challenge requires organizing workers so that their actions are aligned.

The division of labor allows for tasks to be broken down, completed and re-integrated by specialized individuals in parallel [66]. Coordination, the "integrating or linking together different parts of an organization to accomplish a collective set of tasks" [171], can occur via programming or feedback [124]. Coordination via programming uses prescribed roles, requiring little communication between task performers; coordination by feedback involves continuous adjustments and ongoing communication [171]. Flash teams rely mainly on programming coordination, but include open chat channels for feedback coordination. In contrast, flash organizations rely on both programming and feedback coordination.

Team and organizational structures facilitate interdependent work by encoding individual responsibilities and dependencies [23, 111]. These structures provide descriptions and templates for ongoing action [11] and define relationships that permit coordination and control of work [143]. Traditionally, teams and organizations relied on formal organizational structures to define responsibilities and coordinate efforts. More recently, however, work environments are much more volatile and dynamic, requiring flexible role-based structures and nonprogrammed coordination approaches [21, 48, 169]. Flash teams and flash organizations rely on both formal and role-based structures.

#### Formal structures

Formal organizational structures [111] enable coordination through defined roles [88], teams [30, 62] and hierarchies [34, 175, 185]. Roles define individual responsibilities [88] and dependencies, such as reporting relationships [21]. Roles are considered as a form of specialization and horizontal division of labor within organizations [23]. Organizations are composed of groupings of individuals, such as teams [30, 35, 62]. For organizations to perform well, each of the teams or groupings must successfully complete their tasks and all of the tasks across the organization must be coordinated [35]. Finally, hierarchies refer to the vertical division of labor in organizations [23]. More specifically, hierarchies establish routes of communication and help identify sources of knowledge and decision makers within an organization [34, 143, 175, 185].

In addition to roles, teams and hierarchies, formal organizational structures enable coordination through formalized rules and procedures. Formalization is defined as "the extent of written rules, procedures, and instructions [2, p. 62]" and is key component of bureaucracy. By articulating task and role requirements, formalized structures, such as workflows, help manage interdependencies, reduce role conflict and communicate goals and priorities in teams and organizations. As noted by many scholars, however, the effectiveness of formalization depends on the fit of the structure with the routineness, certainty and interdependencies of the task [2, 8, 48, 50]. For example, while a team or organization's interdependence can vary in terms of tasks, goals, feedback or rewards, Felps, Mitchell and Byington [50, p. 197] argue that "highly interdependent groups have more interaction

and the content of that interaction is more central to accomplishing the work task."

Previous research suggests that task interdependence impacts the type of coordination that is appropriate. Specifically, there are three major types of workflow interdependence (Thompson, 1967): pooled interdependence, when each subtask renders a contribution to the whole; sequential interdependence, when subtasks produce output that serve as inputs for other units; and reciprocal interdependence, when the outputs of all subtasks serve as the inputs for others. All coordination mechanisms trade off benefits (e.g. efficiency) with costs (e.g. information omission).

#### Role-based structures

Recent research has demonstrated the benefits to role-based team structures [12, 97] and lightweight team scaffolds [169], especially in fast-paced, uncertain and dynamic work environments [21]. These role-based work structures organize work around clearly defined roles rather than specific individuals with established relationships and shared knowledge.

By "deindividualizing" team roles [97], role-based structures increase accountability and can support temporary coordination among individuals who have never worked together before, such as emergency response teams [21] and movie crews [12]. For example, movie crews on film sets temporarily come together under intense time pressures, during which they need to coordinate complex and interdependent activities [12]. In her ethnography of four film sets, Bechky found that the key to enabling temporary fast-paced coordination is "negotiated reproduction of role structures [12, p. 4]." Specifically, by enacting ones role and satisfying the expectations of the role, individuals can come together temporarily and achieve interdependent goals beyond the confines of any particular setting or project.

More recently, Valentine and Edmondson [169] integrated research on role-based coordination and team effectiveness to create meso-level structures known as team scaffolds. Specifically, team scaffolds bound a set of deindividualized roles and give them collective responsibility. In their longitudinal study of a hospital emergency department, personnel were bounded by a physical space and a deindividualized set of roles rather than specific individuals. This deindividualized boundary, which contained an explicit set of clearly defined roles, enabled individuals to identify interdependent team members at any given time, even if they had never worked together before. Furthermore, by instilling collective responsibility for an entire task (e.g., responsibility for a pod-specific set of patients), the scaffolds led to personnel working interdependently to achieve their group-level goals rather than executing individual role-based tasks. Taken together, these three dimensions of team scaffolds (boundedness, role set, collective responsibility) enabled group-level coordination behaviors among a fluid set of personnel.

Given that individuals are frequently changing, team scaffolds are not teams in the traditional sense. Instead, Valentine and Edmondson emphasize that a team scaffold is "a stable structure that helps fluid personnel act like a team [169, p. 4]." As work continues to move online, structures

of this nature, which enable temporary coordination among interdependent workers or strangers, are becoming increasingly more important. Flash teams and flash organizations draw heavily from this research and adopt similar structures. Specifically, flash teams and organizations assemble diverse expert crowd workers into bounded sets of de-individualized roles, each of which have specific tasks and requirements. In addition to their individual tasks and responsibilities, the workers are given collective responsibility, which is to achieve a desired goal or set of goals within the given constraints. By drawing on and extending team scaffolds, flash teams and flash organizations enable crowd workers who are distributed around the world and have never met or worked together before to effectively coordinate their efforts and achieve complex and interdependent goals that were previously out of reach to crowdsourcing.

### 2.3.2 Distributed Work

Organizations are increasingly relying on globally distributed teams to acquire important skill sets [71]. Distributed teams typically involve members located in multiple locations who collaborate in technology-mediated settings [139] that vary in richness and lack important contextual information [32]. Rich interactions among geographically distributed teams are important especially if team members have a high degree of interdependence and geographic, organization and cultural boundaries must be spanned. Once members have established shared views of the task and strong relationships, their interactions can be shorter and occur via an array of media and communication channels [130, 139].

Online crowd work face many of the same obstacles of geographically distributed work. Research on distributed work has identified many of the factors that impact effective coordination when workers are not collocated [73, 130, 138]. For example, when workers are globally distributed, they often face challenges related to incompatible timezones [140, 146], languages [135], cultures [71] and misaligned incentives [59, 141].

Many of the challenges encountered in distributed work environments are magnified in crowd work. For example, online crowd work involves short term work, resulting in tremendous amounts of turnover among workers. This makes it even more difficult to establish mutual knowledge and shared context [32], build trust [86] and establish a shared identity [18, 73], all of which are known to improve team outcomes. Given that most of the workers in flash teams and flash organizations will have never collaborated before, we use the known best practices for virtual coordination to inform the design and feature set of Foundry, the computational platform we created to author, manage and coordinate expert crowd work. For example, Foundry supports loosely coupled work structures [72, 139], situational awareness [41, 60], current state visualization [24, 122], and rich communication tools [139].

## 2.4 Conclusion

Crowdsourcing research has established a range of computational techniques and systems for decomposing goals into microtasks completed by homogeneous workers, which are then recombined to produce a much larger outcome. While these efforts have enabled paid crowd workers to achieve many large-scale goals, they are not designed for expert crowd work or highly interdependent and complex goals. Decades of organizational behavior research, however, has contributed theories and insights on the structures that enable traditional teams and organizations to achieve complex and interdependent goals. To date, however, this research has focused on bounded teams and organizations with stable membership.

To enable complex and interdependent expert crowd work, this thesis draws on and integrates many of the ideas and contributions from both crowdsourcing and organizational behavior research. By bringing together the computational techniques from crowdsourcing with the coordination structures and organizational design principles from organizational behavior, we introduce new computationally enabled organizational forms and crowdsourcing systems that break through the expertise and complexity barriers limiting crowds.

# Chapter 3

# Expert Crowdsourcing with Flash Teams

**Most of the content in this chapter is from the following paper:**

> Daniela Retelny, Sebastien Robaszkiewicz, Alexandra To, Walter Lasecki, Jay Patel, Negar Rahmati, Tulsee Doshi, Melissa Valentine, Michael S. Bernstein. Expert Crowdsourcing with Flash Teams. *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, 75–85. New York, NY, USA: ACM. https://doi.org/10.1145/2642918.2647409

## 3.1 Introduction

Crowdsourcing systems coordinate large groups of people to solve problems that a single individual could not achieve at the same scale. Microtasking systems typically use highly-controlled workflows to manage paid, non-expert workers toward expert-level results (e.g., [17, 31, 96, 104, 136]). While these crowdsourcing approaches are effective for simple independent tasks, many real-world tasks such as software development and design remain largely out of reach. Such tasks require deep domain knowledge that is difficult to decompose into independent microtasks anyone can complete [95, 160, 182]. Unlocking these capabilities is critical to the vision of universal, on-demand crowd support for end users. For example, could we enable anybody with a napkin sketch of a design idea to ask the crowd to follow the user-centered design process and create a user-tested, high-fidelity prototype of that idea within twenty-four hours?

In this paper, we explore the feasibility of solving these complex, interdependent problems via *structured collaborations between experts from the crowd*. We aim to gather experts from online marketplaces such as Upwork [166] into small teams that can, for example, follow the user-centered

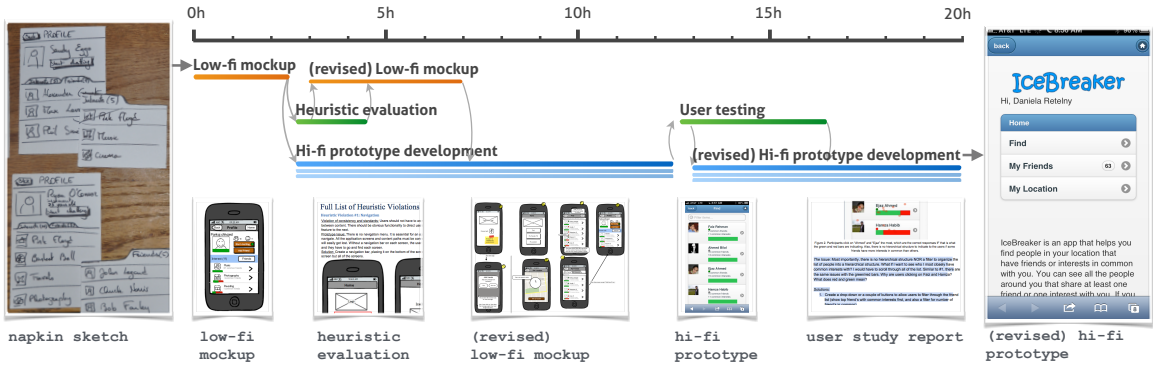



**Figure 3.1: A flash team is a linked set of modular tasks that draw upon paid experts from the crowd, often three to six at a time, on demand. The napkin sketch design team follows the user-centered design process to create a series of prototypes and iterate based on feedback to produce a user-tested software prototype within a day. Multiple arrows indicate the beginning and end of pipelining; lighter bars indicate possible elastic growth.**

design process to transform a napkin sketch into a fully user-tested prototype, create an animated movie from a prompt, or develop an entire online class platform complete with video content and quizzes — all in roughly one day.

Experts from the crowd, however, tend to work as isolated contractors, and microtask crowdsourcing techniques (e.g., [4, 17, 96, 106, 183]) cannot coordinate these experts because they do not effectively leverage participants' diverse skills and expertise. To enable end users to reliably crowdsource complex work, we take inspiration from organizational behavior research that suggests even temporary groups can coordinate complex work effectively, if they have an enabling team structure [169] to encode who is working together and who is responsible for which tasks [12, 23]. Our goal is to create such structures with far less manual effort and more scalable replicability than traditional organizations.

To achieve this goal, we frame expert crowd work around *sequences of linked tasks*. We hypothesize that these chains of focused work could maintain the coordinating strength of lightweight team structures while providing a representation that interactive systems can leverage to support collaboration, create new teams automatically, grow and shrink teams on demand, and combine teams into larger organizations.

We thus present *flash teams*, a framework for dynamically assembling and managing crowdsourced expert teams. Flash teams are a sequence of modular tasks that draw on paid experts from the crowd. Each task in a flash team requires an input (e.g., low-fidelity prototype) and produces an output of another type (e.g., heuristic evaluation). End users create workflows by linking these modular tasks together to pass each task's output as input to the next task. They then use a web application to launch and manage the team, monitoring the workflow if desired. For example, when creating a user-centered design flash team, the user might begin with a task where a UI designer

creates a low-fidelity mockup of the user's idea. The designer would pass the mockup on to a UX researcher for a heuristic evaluation. Tasks would proceed through the chain of a revised mockup, a high-fidelity prototype, a user study and a revised software prototype, crossing diverse expertise from one or more UI designers, UX researchers, and software developers.

Flash teams' modular task structure is transparent to computational systems, enabling those systems to leverage the structure to orchestrate crowd dynamics. For example, teams can be *combined*: since teams are modular, their component tasks can be composed like Lego blocks to form larger organizations that persist only for a day or an afternoon. Moreover, by leveraging automated path search through the space of previous teams' intermediate inputs and outputs, end users can *assemble a flash team on-demand by providing only the desired input and output*. During runtime, flash teams are *elastic*: they grow and shrink to meet task demands by hiring additional members from the crowd. Finally, their work can be *pipelined*: when in-progress results are enough for downstream tasks to begin work, the system passes in-progress results along to accelerate completion times.

We embody this approach in *Foundry*, a platform for end users to create flash teams and a runtime environment to manage them. Foundry allows users to create flash teams by directly authoring each task or forking a team that other users authored and then recruit from the Upwork[1] online labor marketplace [166]. For team members, Foundry takes on managerial responsibilities to walk team members through the workflow, notify them of any schedule shifts, scaffold the handoff process, and provide members with shared awareness [41]. For the end user, Foundry abstracts away low-level management effort while allowing for high-level oversight and guidance.

This research advances two important ideas. First, flash teams are the first to leverage the *scale of paid crowdsourcing for expert work*. This approach pushes beyond volunteer peer production systems with a vision of elastic and on-demand organizations that manage teams of paid experts from the crowd. Second, this paper offers a method of scaling expert crowd work through *computational management of an elastic workforce*. Flash teams enable complex work at crowd scale by automating the structures of traditional organizations. We introduce specific affordances such as structured handoffs, directly-responsible individuals (DRIs), and elastic growth, embedded within the Foundry platform. Empirical results from our field experiment demonstrate that flash teams can accomplish efficient and high-quality work.

## 3.2 Related Work

In this section, we connect two threads of research: crowdsourcing and organizational behavior.

[1] At the time this research was conducted, Upwork was known as oDesk. In 2015, oDesk and Elance merged and became known as Upwork.

### 3.2.1 Crowdsourcing With Experts

Crowdsourcing research aims to integrate large-scale, API-friendly labor marketplaces into software through open calls and task decomposition. Crowdsourcing has largely focused on tasks any individual can complete: many crowdsourcing platforms are built to accomplish tasks that require little training (e.g., Amazon Mechanical Turk) and recruit amateurs (e.g. FoldIt [31]). Consequently, most crowdsourcing workflows and algorithms [116] aim to structure non-expert contributions to produce expert-level performance. For example, MapReduce frameworks can guide crowds to write simple encyclopedia entries [96], and clustering workflows can produce expert-level taxonomies [29]. These workflows can even be authored by crowd members themselves [100]. Examples of microtask crowds pursuing expert performance include document editing [17], translation [79], and visual question answering [20]. AI techniques optimize these workflows [36]. Prior work that has gone beyond microtasks has still focused on recreating the performance of a single expert [105].

Prior work suggested that expertise might play a role in crowdsourcing [95]: flash teams instantiate a first step toward that goal. Other crowdsourcing campaigns already recruit experts, for example using workflow support tools [119] or helping solve planning problems [183]. Often these campaigns focus on a single expertise area such as math [33]. To date, these expert crowdsourcing efforts have been one-offs, needing to create an online presence to gather participants for each new goal. Our intent is to create a general approach and platform that can support a large number of tasks on demand. In doing so, we open the door to a range of higher-level workflows that can assume expert knowledge.

This work raises opportunities for platform design in expert crowdsourcing. Foundry draws inspiration from visual workflow tools [93, 100] and management tools such as Gantt charts to structure teams using a visual timeline language. It is possible to design for worker interest, honesty, and motivation (e.g., [6, 43]); Foundry thus aims to make collaborators visible and make the larger goal clear. Software concepts such as class hierarchies can integrate with business processes [123]: Foundry instantiates this vision with team structures.

### 3.2.2 Organizational Behavior

Organizational behavior (OB) research has identified the obstacles to team coordination as well as the conditions and structures that enable effective team coordination. Obstacles to effective team coordination include geographic dispersion, technology-mediated communication, and fluid (or changing) team membership [32, 71, 80, 139]. These conditions impede team coordination because team members lack a shared understanding of their overall task, and each person's responsibilities within the task [129]. Because their communication is all electronically-mediated, teams also struggle to engage in the rich communication necessary to build a shared understanding [32]. Teams of experts who are pulled from the crowd will face these same obstacles in the extreme.

Foundry's design is inspired by specific organizational behavior studies that show how to support coordination under the challenging working conditions confronted by crowdsourced expert teams.

First, Foundry's composeable team structures are motivated by organizational behavior studies of the purpose and functioning of team structures. Team structures enable coordination by encoding who is responsible for what work and which team members are interdependent [23]. Even in the absence of ongoing relationships, lightweight team structures can help even relative strangers coordinate effectively, when the structures set up shared space and shared work around strictly defined work roles [169]. Second, Foundry's modular design is influenced by management modularity theory which shows how system components must be "loosely coupled" and have a standardized interface so that they can connect and interact in a variety of configurations [10]. Third, Foundry's design includes several integration mechanisms such as pipelining, structured handoffs and DRIs that address key points of coordination (which organizational behavior research shows are often neglected) [66].

We argue that for teams of crowdsourced experts to be most successful, they need to quickly understand their shared work, interdependencies and respective roles in completing that work [133]. Our aim is to provide the value of team structures, modularity, and coordinating mechanisms (as identified in organizational behavior research), but to do so in ways that leverage automation, computation, and the scale and flexibility of the crowd.

## 3.3 Flash Teams

In this section we introduce *flash teams*, which are computationally-guided teams of crowd experts supported by lightweight, reproducible and scalable team structures. Flash teams aim to embed the techniques of high performing offline teams within a model that can take advantage of computation's ability to abstract, scale, and visualize progress. We take a position that expert crowd work can succeed via a linked set of modular tasks that interface with each other by publishing intermediate results. Each set of tasks is machine-understandable, which means that interactive systems can manage and manipulate the team structures so that flash teams can grow, adapt, and recombine into larger organizations.

### 3.3.1 Flash Team Composition

To structure expert crowd work, flash teams require small atomic actions called *blocks*. Each block represents one or more experts performing a task. It requires an input, instructions for what one or more members of the crowd should do, and specifies an output type (Figure 3.2). Blocks can then be connected to other blocks that can accept compatible inputs or outputs. Blocks aim to be self-contained so that they can be reused in other contexts. Inputs and outputs are represented as tags (e.g., `heuristic evaluation report`, `character art`, `voiceover script`, `voiceover audio files`), and any blocks that interface along the same input or output use the same tag. For example,


**Figure 3.2: The basic unit of a flash team is a *block*, which gathers one or more paid experts from the crowd to perform a focused task. Blocks can be connected via shared inputs and outputs.**

suppose the user is building a team to execute the user-centered design process. Then, they might create a heuristic evaluation block to take `low fidelity prototype` as input and produce `heuristic evaluation report` as output. That user could then connect the heuristic evaluation block to any other block that takes `heuristic evaluation report` as input, for example a block that revises a mockup based on feedback.

Flash teams exhibit distributed leadership [119] where the user (requester) may be coordinating with different workers in each block. To avoid a diffusion of responsibility, for each block, the project requester specifies a *directly responsible individual* [107], or DRI, who takes on a temporary management position for the duration of that task. For example, if a task involves a collaboration between three software developers, one of those developers acts as the DRI and coordinates workers, brokers agreements, and ensures the task is completed on time and at high quality.

Users assemble and manage a flash team by chaining blocks like interlocking puzzle pieces to transform a starter input into a desired final output. Blocks can execute in parallel or in serial but they typically must wait on all their inputs to begin work. The complete set of blocks determines which crowd experts need to be recruited — for example, a UI designer, UX researcher, and two software engineers — and when they are likely to be needed.

*Foundry.* Foundry (Figures 3.3–3.4) functions as both a flash team authoring environment with strongly typed handoffs and support for diverse expertise, and a runtime management platform. Users add blocks to the timeline as they would on a calendaring interface. Each block requires a title, a target length of time, input and output type tags, a description of the task (ideally with an example), and one or more of roughly 2,500 skills available on Upwork. To accelerate team creation, and based on an observation that many teams share some components, Foundry makes available a library of all blocks that previous teams have used for drag-and-drop. With Foundry, therefore, the requester can create new workflows from scratch using the library of task modules, fork a previously created workflow, or use Foundry's planner to formulate a workflow.

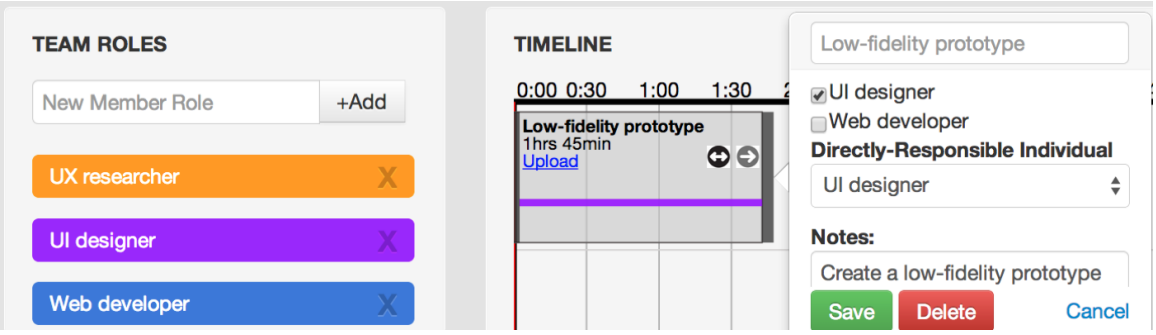



**Figure 3.3: Foundry allows end users to author and manage flash teams. In authoring mode, the user can specify the expertise for each block and enter details about each block's requirements.**

Most coordination between team members occurs via input/output handoffs. Foundry focuses the user on explicitly marking handoffs via arrows that connect tasks: this helps make dependencies visible. When each block is complete, worker(s) upload the output materials for their block to the team's shared folder, which Foundry automatically creates in Google Drive. When two blocks occur at the same time (e.g., a logo design block should be coordinating with the homepage layout block), users indicate that the blocks should be in synchronous collaboration.

### 3.3.2 Runtime and Coordination

Users either author a team from scratch, or borrow from a list of existing teams and edit or run them. Workers are recruited from Upwork using the skills described in the blocks. Upwork provides a worker rating system that makes filtering relatively simple. We have recruited workers for teams in as little as fifteen minutes, and this number could decrease as such platforms become more popular and optimized. When the user is happy with their team, they launch the team by clicking "Start" in the Foundry interface. Foundry switches to runtime mode and adds a timer playhead that starts advancing along the timeline on the first block. Each worker receives a unique link that logs them in to the Foundry runtime with their tasks highlighted in the timeline.

Foundry embeds the basic affordances of computer-supported cooperative work systems for distributed teams. While the flash team is working, Foundry's goal is to remove the requester from as much management responsibility as possible by leveraging what it knows about the flash team's structure. Requesters can monitor progress, provide feedback, and answer questions through the chat. They can also pause the process, tweak the team structure and resume.

Foundry always displays the overall progress through the timeline, the currently active block details, and when the logged-in worker's next task begins. When a worker marks a block as complete, Foundry notifies workers for the next blocks that their task is active. Additionally, Foundry maintains awareness through a chat window that allows workers to coordinate out-of-band and the

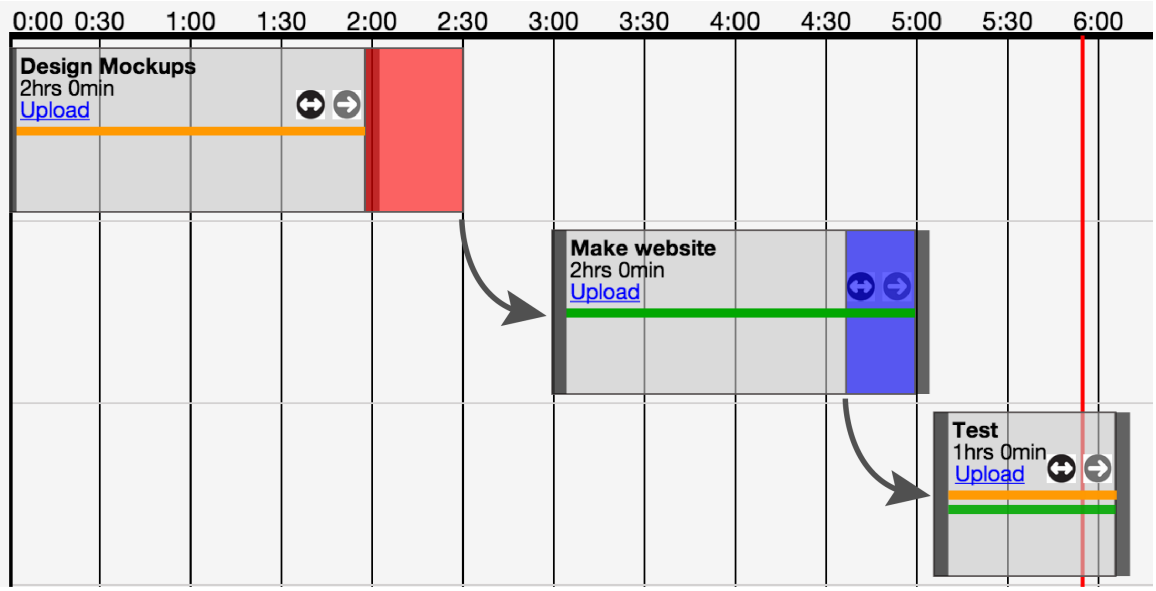



**Figure 3.4: Foundry's runtime mode: the design mockups finished 30 minutes late but implementation finished early. The current user test block, marked by a red playhead, involves both (orange and green) experts.**

user to provide feedback on the process as it occurs.

It is not uncommon for teams to require tweaks as they go: for example, a task may finish early or run late. If a task finishes early, Foundry recalculates the starting times for all downstream tasks and emails all impacted workers with a new estimated start time for their next task. Likewise, if a task runs late, Foundry emails the DRI with a link to a form that requests the estimated time to completion. When the worker fills out the form, downstream tasks are again recalculated and workers are notified of the new start times.

### 3.3.3 Computationally-Enhanced Flash Teams

Computation can leverage the machine-readable workflow representation of flash teams. For example, we can use a block's output tag to recommend a list of blocks that use the same tag as an input. In this section, we introduce mechanisms for Foundry to react, optimize and author flash teams on demand.

#### Modular combinations of teams

Because the blocks are modular and have clear boundaries, flash teams can be *combined* to create larger organizations. Entire teams can be abstracted into a block, much like one function in a computer program might contain many other function calls inside it. For example, to create a new platform for a massive open online course (MOOC), one flash team might combine several teams that

can design and implement software with additional teams that can develop educational materials and animated videos. Each team might be forked from previously used teams in Foundry and tweaked for the particular needs of the project.

Through combination, users can create not just teams but the equivalent of entire small organizations for an afternoon or a day. These combinations draw on the elastic nature of the crowd by instantiating multiple copies of a flash team, and multiple types of teams, with little downtime.

#### Path search support for team authoring

So far, Foundry assumes that end users are fluent with creating teams (e.g., authoring tasks and handoffs). Most people, however, are neither natural programmers nor managers. Instead, they have a goal in mind that they want to accomplish, but may not have a strong idea of how it might get completed.

For example, suppose a user wanted to embed themselves as a character in their favorite video game. The user might have no experience with art or video game modding, but have a `photograph` of themselves and knows that they want a `3D model` to replace the default game avatar with. Using this, the system explores the set of existing teams and realizes that the `photograph` can be used as `source material` for a fashion designer from a theatre production task who creates a `clothing sketch` according to the world's style. The `photograph` and `clothing sketch` can then be passed to an illustrator from an eBook task, who in turn creates a `2D drawing` of character art. The `2D drawing` is then used by a 3D modeling expert originally from a product mock-up team to create a `3D model` which can be imported into the game.

By leveraging blocks' shared input/output tags, Foundry can search through the space of novel team combinations. If the user knows what they want, they can tell Foundry their starter input (e.g., `character sketch`) from the list of previously-observed tags and the output they want (e.g., `3D model`). If there is a path through the known blocks that can solve the user's problem, Foundry will find it using an AI planner. The resulting teams are not whole components of previous teams, but subsets of blocks from many other teams that come together to enable a new goal. The user can review the team to ensure that all team blocks make sense in this new context.

This team was automatically generated on demand to fill in details that the user may not have been able to specify themselves. The members of the team were not drawn from prior teams with a focus on video games, but Foundry identified that previous blocks shared an input/output vocabulary and could thus interface with each other. It brought them together to effectively complete the desired task.

Because of teams' modularity, the number of possible teams grows combinatorially with the number of teams seen by the system. Foundry thus uses a planning algorithm to find a set of blocks that connect from original input to final output, if such a path exists. We represent all known blocks in the STRIPS planning language [52]. Each block becomes a STRIPS action with all required inputs

combined via a logical conjunction as the precondition, and all outputs listed as postconditions. An automated planner searches through transitions that connect a given starting state to a desired end state in order to find the shortest path. The general planning problem is NP-complete; however, boolean satisfiability solvers can scale these systems to thousands of actions [89].

#### Elasticity

Simple workflows cannot react to changing requirements: for example, a project may require additional help or a new skill. Because crowdsourcing platforms allow quick recruitment, the crowd can be an *elastic* resource, allowing teams to grow and shrink as needed. Elastic growth allows a single team abstraction to encompass a wide variety of actual runtime needs.

To enable elastic flash teams, the user specifies growth parameters for a block. The block may, for example, now allow up to two additional experts to join this task. If the user or the DRI feel that growth would be helpful, they can request the expansion and the user can approve it. One type of elasticity is replication: for example, the DRI for a software development task may ask for an additional developer to help accomplish the job on time. The second type of elasticity is specialization: if a team needs a specific skill, the DRI could recruit a team member with that skill (e.g., logo design). For instance, in one of the design teams, the team requested a Facebook API expert because the application required it. Users need to balance the temptation of growing a team against the increased coordination costs of larger groups.

#### Pipelining

Flash teams have been described as blocking operations: downstream tasks must wait for upstream tasks to complete before they can start. However, in many scenarios, early results may be enough for a downstream task to begin.

Flash teams can thus be *pipelined* by streaming intermediate results as they are ready, rather than waiting for the entire task to complete. To pipeline a handoff, the user indicates that a block can stream its output — for example, that the heuristic evaluation block can stream each violation as the worker identifies them. The user also indicates that the downstream task can accept streamed results — for example, that the revision of the low-fidelity mockup can fix each violation as it arrives. The DRI for the upstream block uses Foundry to indicate when the task is ready to start streaming, and Foundry launches the downstream task.

Pipelining has two beneficial effects: less waiting and increased collaboration. There is a risk that downstream tasks might need to change course when they see more of the input, but in our experience pipelining scenarios allow team members to work productively in parallel. In addition, instead of communicating through a single handoff, pipelining encourages synchronous feedback between team members, reducing the risk of communication failures or misunderstandings.

| **Task** | **Type of Team** | **Roles** | **Time** [*hh:mm*] | **Median Wage** | **Total Cost** |
|---|---|---|---|---|---|
| Happily App | Napkin Sketch | UX, UI, Dev | 31:30 | $26.85 | $744.48 |
| Eventick App | Napkin Sketch | UX, UI, Dev1, Dev2, Dev3 | 18:00 | $28.78 | $1,270.28 |
| Icebreaker App | Napkin Sketch | UI, UX1, UX2, Dev1, Dev2 | 23:10 | $31.38 | $1,200.97 |
| Animation | Animation | Director, Voiceover, Animator, Illustrator, Scriptwriter | 44:40 | $56.94 | $2,381.32 |
| Diaphragm MOOC | Education Content, Animation | Education, Content, Director, Animator, Actor, Voiceover | 19:20 | $30.14 | $1,597.32 |
| Photography MOOC | Education Content, Animation | Education, Content, Director, Animator, Actor, Voiceover | 19:00 | $21.77 | $741.58 |
| Tower of Hanoi MOOC | Education Content, Animation | Education, Content, Director, Animator, Voiceover | 11:30 | $18.52 | $446.49 |
| MOOC Platform | Napkin Sketch 1, Napkin Sketch 2, Napkin Sketch 3 | UX1, UI1, Dev1, UX2, UI2, Dev2, UX3, UI3, Dev3 | 13:00 | $29.14 | $1,015.80 |

**Table 3.1: Eight flash teams created mobile web applications, video animations, and an online course platform with three courses.**

## 3.4 Example Flash Teams

In this section, we present several flash teams and their results. We demonstrate how flash teams execute complex and creative tasks beyond the reach of paid microtasking and surpass the capabilities of traditional organizations via their scalable and flexible structure.

### 3.4.1 Napkin Sketch Design Team

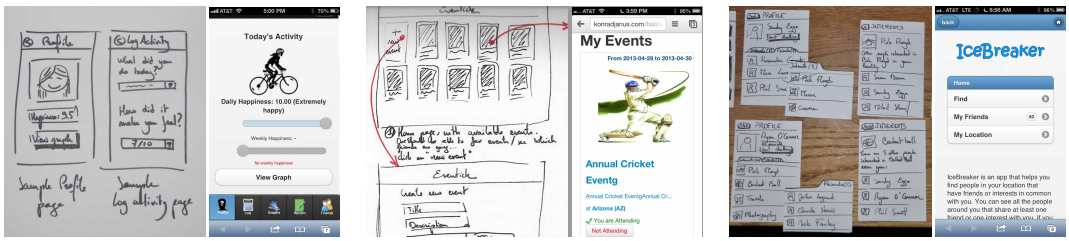
Rapid iteration is a core tenet of the user-centered design process, and prototyping many ideas in parallel leads to improved design outcomes [42]. Flash teams could enable a designer to generate many ideas as sketches in a notebook, then get them prototyped in parallel — from low-fidelity prototypes, through a heuristic evaluation and live user testing, to an iterated high-fidelity prototype — in less than a day. This rapid, parallel cycle could allow end users to quickly iterate toward high-quality ideas.

The *napkin sketch design team* carries out the entire user-centered design process starting from a

napkin sketch of the idea by combining techniques from HCI practice. Rather than acting as a turn-crank design solution or replacing designer insight, it enables designers to experiment and reflect on ideas quickly. The complete napkin sketch design team blocks are:

- *Low-fidelity prototype.* Produce a mockup of the sketched interface using a low-fidelity wire-framing tool. Input: `napkin sketch`. Output: `low-fi mockup`. Role: UI designer. Target time: 2.5hr.
- *Heuristic evaluation.* Report the violations of Nielsen's heuristics for a design. Input: `low-fi mockup OR high-fi prototype`. Output: `heuristic evaluation`. Role: UX researcher. Target time: 2hr. Can pipeline output.
- *Revised low-fidelity prototype.* Iterate on a mockup based on usability feedback. Input: `low-fi mockup AND (heuristic evaluation OR user study report)`. Output: `low-fi mockup`. Role: UI designer. Target time: 4hr. Can pipeline input.
- *Software prototype.* Create a hosted, high-fidelity web prototype to instantiate a mockup design. Input: `low-fi mockup`. Output: `high-fi prototype`. Role: Web developer (elastic 1-3).
- *User study.* Run at least three users through a prototype and create a document that summarizes the results and recommendations. Input: `high-fi prototype OR low-fi mockup`. Output: `user study report`. Role: UX researcher. Can pipeline output.
- *Revised software prototype.* Fix problems with the prototype that were identified by the user study. Input: `high-fi prototype AND user study report`. Output: `high-fi prototype`. Role: Web developer (elastic 1-3). Can pipeline input.

Depending on whether the same workers participate in multiple blocks or new workers come online for each step, this team involves three to ten experts in UI design, UX research and web programming.




**(a)** Happily, an emotion-tracking site.

**(b)** Eventick, a local event billboard site.

**(c)** Icebreaker: a local social network site.

**Figure 3.5: Sketched inputs and high-fidelity prototype results from the napkin sketch design teams. This team iterates from a sketch through low fidelity prototypes to produce a working, user-tested software prototype within a day.**

We ran the napkin sketch design team three times with inputs of varying complexity, all for prototypes of mobile web applications (Figure 3.5). We obtained the original application ideas and sketches from actual student projects in an Introduction to HCI course. The first design sketch was for Happily, a web application that helps users track their happiness throughout the day. The second was for Eventick, a mobile application where users can post local events and hear about events in their area. The third was for Icebreaker, a social networking site to help users find nearby individuals with common interests. These prototypes each contained roughly four working pages, a database backend, and content-specific requirements (e.g., Facebook API integration for Icebreaker).

Each flash team took advantage of more of flash teams' computational support than the previous iteration. Happily used a basic, sequential workflow, where one task occurred at a time and the next task could not begin until the previous one was completed. Eventick enabled pipelining: both heuristic evaluation and development began as soon as the first version of the mockup was complete. It also enabled elastic growth for replication, allowing the web development block to grow from one to three developers. Icebreaker was the most complex, enabling elastic growth for specialization: the DRI developer hired a developer specialized in Facebook API integration, and the team brought on a designer to create a logo for the homepage. Icebreaker also pipelined the user study results into the revised software prototype block.

These flash teams completed their prototypes in as little as eighteen hours of active work (Table 3.1). Workers were typically paid $25–$30 per hour, and costs ranged from roughly $750 to $1270 for up to five participants. The prototypes strongly reflected the designs articulated in the original napkin sketches, and many major design and usability problems had been ironed out through the heuristic evaluation and user study revisions. The initial napkin sketches and final prototypes are shown in Figure 3.5. We did not optimize the team to hand off across timezones, so actual wall clock time was often longer as team members needed to sleep and take breaks.

### 3.4.2 Animation Team

A quick review of YouTube videos is enough to realize that many people who have excellent ideas are not quite so excellent at communicating them. Unfortunately, for most of these people, hiring an expensive firm to bring their ideas to fruition is the only alternative. The animation flash team aims to make high quality video animations more accessible. It also exercises flash teams in a creative, non-engineering domain. The animation team takes a high-level script outline as input and produces a short animated movie as output. The core blocks of the animation team (Figure 3.6) included:

- *Script.* Write the video script. Input: `script idea`. Output: `script`. Roles: director, scriptwriter. Target time: 2hr.

- *Storyboard.* Use the script to create a shot-by-shot storyboard for the video. Input: `script`. Output: `storyboard`. Roles: director, animator. Target time: 2hr.

- *Character design.* Design the characters' visual appearance and develop their art assets. Input: `script`. Output: `character assets`. Role: illustrator (elastic 1-3). Target time: 4hr. Can pipeline output (draft/final designs).

- *Character animation.* Import and animate character assets from the storyboard. Input: `storyboard AND character assets`. Output: `animation character layer`. Role: animator (elastic 1-3). Target time: 7hr. Can pipeline input.

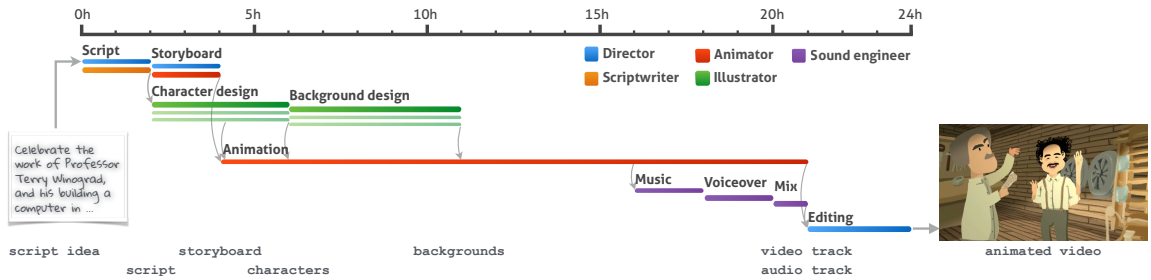



**Figure 3.6: The workflow for the animation flash team, which takes a high-level script outline as input and produces a short animated movie as output.**

To develop the flash team structure, we collaborated with a movie director from Upwork. The animation flash team consists of five roles: director, scriptwriter, illustrator, animator, and voiceover artist. Unlike the decentralized leadership of the napkin sketch team, the director role participates actively across the entire workflow. In the animation team we ran, the task was to create an animation that captures a memory from HCI researcher Terry Winograd's early life, where Terry builds a computer in his garage (Figure 3.7).

### 3.4.3 Education Team

While many people step into the role of teacher at times, most do so without aid or feedback from trained educators. In addition, while the web can teach a wide variety of concepts, there are always concepts that are not well covered or not explained clearly where a user might want some explanation. The education flash team allows anyone to generate a short curriculum to learn a topic. It takes as input the educational goal (e.g., "teach the basic mechanics of singing from the diaphragm") and outputs a script that teaches the subject and multiple-choice test questions to self-test the content. The main blocks of the education team are:

- *Script.* Brainstorm and write out a script for the lesson. Input: `lesson idea`. Output: `lesson script`. Roles: educator, topic area expert. Target time: 3hr.

- *Quiz.* Generate a quiz based on the content of the lesson script. Input: `lesson script`. Output: `lesson quiz`. Roles: educator, topic area expert. Target time: 1hr.

**Figure 3.7: The animation team produced a short movie of a young Terry Winograd building a computer in his garage.**

- *Entertainment value script improvements.* Edit the script so it is as engaging as possible. Input: `lesson script`. Output: `edited script`. Role: director. Target time: 1hr.

The team draws together three sets of skills: expertise in the topic area of the lesson, curriculum development, and multimedia direction. The director takes a final pass over the script to ensure that it is as engaging as possible.

An example script snippet from a flash team generating a lesson on singing from the diaphragm: *"When you breathe in, or inhale, your back muscles engage and expand. At the same time, your diaphragm lowers so your lungs can expand."*

### 3.4.4 Composite Team: Online Course Platform

The napkin sketch, animation and education teams are relatively focused in their goals. Could flash teams enable users to pursue more ambitious goals that combine multiple domains of expertise? Flash teams' modularity promises to allow for organizations composed of many smaller teams.

We created a composite flash team that combined and replicated all three other teams. Its goal was to create a prototype platform for online education of long-tail skills such as photographing portraits, singing from the diaphragm and solving the Tower of Hanoi puzzle (Figure 3.8).

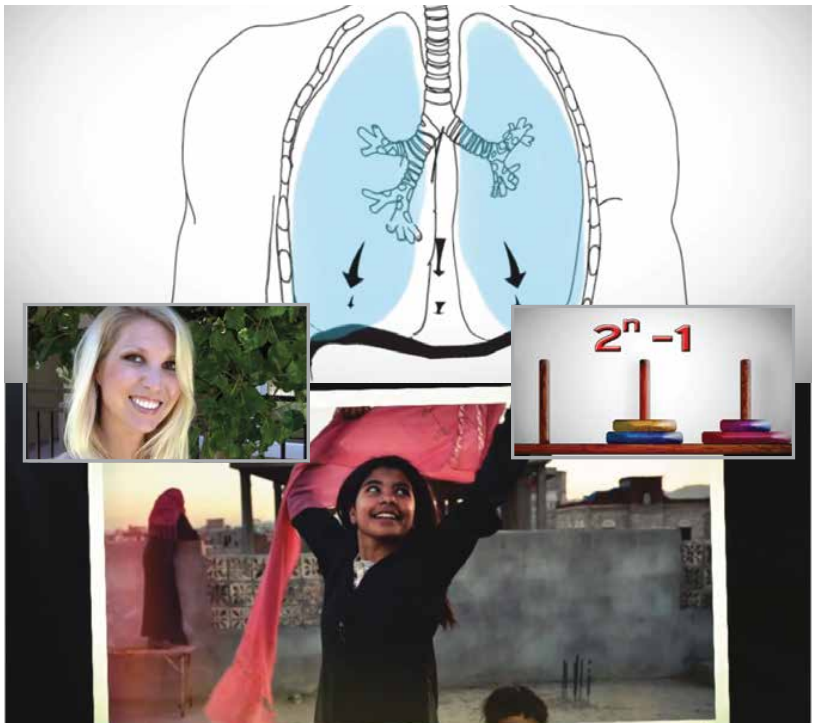



**Figure 3.8: Course videos: singing from the diagram (top), portrait photography (left and bottom), and the Towers of Hanoi puzzle (right).**

The composite team used abstracted versions of existing teams as single blocks, then tweaked their outputs and inputs. The blocks of the composite team included:

- *Homepage*. Design homepage for the site. Input: `napkin sketch`. Output: `high-fi prototype (homepage) AND design language`. Roles: napkin sketch team. Target time: 12hr. Can pipeline output (design language).
- *Class page*. Design the page to display a course. Input: `design language AND napkin sketch`. Output: `high-fi prototype (course)`. Roles: napkin sketch team. Target time: 12hr. Can pipeline input (design language).
- *Course content*. Write a script and quiz for the video. Input: `lesson idea`. Output: `multimedia-edited script`. Roles: education team. Target time: 4hr.
- *Course video*. Produce the video for the lesson. Input: `multimedia-edited script`. Output: `animation`. Roles: animation team. Target time: 16hr.

This team was composed of nine flash teams: three napkin sketch teams to create the three pages of the site, three education teams to write the video scripts and quizzes, and three animation teams to produce the videos. The first napkin sketch team pipelined the site design language. The

nine teams created the platform and three interactive lessons in *19 hours and 20 minutes* of active work for a budget of $3,801.19 (Table 3.1). Wall clock time was longer because not all teams began simultaneously. The talent for the singing video included a singer from the Frankfurt Opera.

## 3.5 Field Experiment

This paper claims that flash teams enable expert crowd members to come together and complete complex, interdependent tasks in short periods of time. We conducted a controlled experiment to compare flash teams to a self-managed team approach where Foundry brings together the crowd experts and lets them choose their own path. To minimize costs, we focused on collecting data with a small number (3) of teams in each condition, totaling 22 crowd experts.

### 3.5.1 Method

We recruited six napkin sketch design teams from Upwork by requesting workers with expertise in UI design, UX research, and web application development. Teams were assigned to either the flash team or the control condition, resulting in a total of three teams in each condition. We used the same job postings on Upwork, recruitment process and task for the teams in both conditions. To keep conditions comparable, the same set of Foundry features was available, including a shared folder via Google Drive and a chat window.

To recruit the experts, we conducted brief filtering interviews and accepted applicants with high ratings. All workers were paid their profile's hourly wage. Members of each team had a median of at least 250 working hours on Upwork and ratings over 4.2 out of 5 stars (average 4.66/5). We randomized the 22 workers into teams and tasked them with creating a simple party planning task manager from an input sketch that would allow end users to add, edit and complete tasks. We asked all teams to follow the user-centered design process and requested that the final mobile web app be completed within 13 hours.

The main difference between the two conditions was the existence of the flash team workflow, whose presence indicated a path to the team members, and allowed Foundry to support coordination via handoffs and notifications. For the flash team condition, Foundry clearly specified each block and the details associated with it, including the duration, team member, DRI, input and output. This enabled Foundry's support for handoffs, task status, time until next task, and email notifications. In contrast, for the control condition, the workflow on Foundry consisted of one long task and did not provide any details or information on how to complete it. This team was allowed to self-manage and make their own plan to complete the work in 13 hours.

Once each project started, we monitored the chat on Foundry to make sure the team was working and answer any questions that came up. We were equally responsive to the progress and questions of the experimental and control teams. If a team member left, we hired a replacement on Upwork

immediately. Though we told teams to complete in 13 hours, we allowed teams as much time as they needed to complete the task.

### 3.5.2 Results

While all six of the teams completed the task and their applications were above-bar for quality, the flash teams took roughly *half as many work hours*, followed the iterative design process more closely, and required less coordination.

| **Condition** | **Roles** | **Work Time** [*hh:mm*] | **Wall Clock** [*hh:mm*] | **Median Wage** | **Total Cost** |
|---|---|---|---|---|---|
| Flash | UI, UX1, UX2, Dev | 11:02 | 17:16 | $18.00 | $211.78 |
| Flash | UI, UX, Dev | 13:52 | 24:10 | $22.22 | $364.80 |
| Flash | UI1, UI2, UX1, UX2, Dev | 14:13 | 23:51 | $14.45 | $215.58 |
| Control | UI, UX, Dev1, Dev2 | 14:30 | 37:03 | $25.00 | $322.22 |
| Control | UI, UX, Dev | 29:20 | 12:24 | $16.67 | $612.79 |
| Control | UI, UX, Dev | 27:30 | 37:01 | $12.00 | $430.59 |

**Table 3.2: Time and cost comparison for the flash and control teams. On average, flash teams took half as many work hours than control teams.**

By active work time, even the *slowest* flash team completed the task faster than the *fastest* team in the control condition. As shown in Table 3.2, the cumulative number of hours worked for the flash teams ranged from 11hr2min to 14hr13min (mean = 13hr2min) and had little variance across teams (SD = 1hr45min). This small variance was consistent across all blocks: the average standard deviation across the six blocks' completion times was 12 minutes. In contrast, the cumulative number of hours worked for control teams ranged from 14hr30min to 29hr20min (mean = 23hr47min) and had tremendous variation (SD = 8hr5min). A one-tailed permutation test to compare the completion times in the two conditions was significant ($p = 0.05$), confirming that flash teams are significantly more efficient than self-managed teams of crowd experts. Wall clock time did not always align with total work time, since workers would sleep or take breaks.

The control teams took longer than the flash teams to finish each role's tasks (Table 3.3). The control teams spent 2.4x the time on UI Design, 1.9x the time on UX Research and 1.4x the time on Development, resulting in an additional 10hr44min in cumulative work. In observing the teams, we noted that team members would often invent work to do while waiting for a collaborator to finish a task. This overeager behavior, which we term *busy waiting*, particularly affected team members such as UI Design who had fewer hours of work that were officially required of them.

| **Role** | **Flash Team** [*hh:mm*] | **Control Team** [*hh:mm*] |
|---|---|---|
| UI Design | 2:59 ($\sigma$=0:55) | 7:20 ($\sigma$=3:39) |
| UX Research | 3:42 ($\sigma$=0:33) | 7:07 ($\sigma$=2:52) |
| Development | 6:21 ($\sigma$=0:46) | 9:20 ($\sigma$=2:01) |
| **Total** | **13:02 ($\sigma$=1:45)** | **23:47 ($\sigma$=8:05)** |

**Table 3.3: Average work time by role for the flash and control teams. The flash teams finished the tasks for each role faster than the control teams.**

By following the sequence of modular tasks on Foundry, flash teams followed the iterative design process more closely than the control teams. In theory, since the self-managed teams weren't restricted to a precise workflow, they could have iterated more times than the flash teams. However, these teams produced fewer, if any, iterations on the mockups or developed web app. The control teams inefficiently decomposed the tasks, combining separate tasks into a single task (such as performing the heuristic evaluation and user testing on the same mockup) and in the wrong sequence (for example, the summative user testing was executed on the low-fidelity mockup, and the software prototype was never evaluated).

The flash teams required less coordination than the control teams and were more able to take advantage of on-demand recruiting from the crowd. The control teams' success was largely dependent on individuals' project management skills. For example, one of the control teams did not coordinate at all and found the experience extremely frustrating. Another control team adopted a fully interdependent workflow in which the experts treated Foundry as a collaborative war room environment [126] and established social and informational awareness to support their coordination efforts [133]; they all worked together to produce a high-quality outcome at great financial cost. When workers disappeared due to other commitments, flash teams were robust and could reach out to the crowd for a replacement quickly. In contrast, when members of one of the control teams quit (in multiple occurrences, they grew too frustrated with the experience and left), the entire team's performance suffered. When one of the developers was replaced, for example, the new developer had to start from scratch and was forced to work on his own since the other team members had disappeared by that point.

## 3.6 Discussion and Future Work

Flash teams enable users to gather and coordinate paid experts from the crowd to complete complex and interdependent tasks quickly and reliably. Rather than try to recreate the strengths of in-person expert teams, this approach suggests a "beyond being there" vision of expert crowd work [74]. In particular, flash teams afford dynamic recruitment and coordination of on-demand expertise that is

extremely difficult in offline scenarios. Problems that might have plagued such teams, such as last minute dropouts, are relatively straightforward to solve in crowd work because another person can join on demand. Furthermore, flash teams can take advantage of timezone differences that could potentially allow them to carry on uninterruptedly for days or weeks.

Flash teams move beyond the typical conception of crowdsourcing as collecting multiple viewpoints on a single question. Rather than treating the crowd as redundant resources that cannot be fully trusted, flash teams view the crowd as an elastic, on-demand set of diverse and high-quality participants. The result is that flash teams often aim to gather experts with *different expertise* rather than redundant viewpoints. Even more ambitiously, flash teams can be combined to create new types of organizations with completely fluid boundaries — organizations that are composed of many smaller flash teams, each of which are spun up on demand, work in parallel, and disperse when complete.

Flash teams also have several important limits, which we intend to address in future work. While our controlled evaluation of flash teams was limited to napkin sketch design teams, in the future we plan to test other types of flash teams to better understand the types of tasks flash teams are well suited for. Furthermore, similar to traditional collocated and distributed teams, flash teams experience difficulties related to coordination and conflict. During informal interviews after the tasks were completed, flash team members indicated that they enjoyed working as a team, but wanted the opportunity to build more camaraderie — some members would finish their task and then just leave. One approach to improving team motivation would be to allow workers to find trusted colleagues and join into clusters that can be hired together.

It will be important for follow-on evaluations to compare flash teams to the same teams when led by project managers. Our current objective was to test flash teams against self-managing teams, which are the equally scalable alternative and therefore theoretically relevant for our paper. Research in organizational behavior has shown that self-managing teams are prevalent and effective. However, it is likely that many teams in future environments may want to consider human managers, which is a worthwhile tradeoff to understand.

Given that teams do not always go according to plan, future work should also explore issues related to runtime course correction and dispute resolution. When this happened in practice, the flash team tended to work within the existing task structure to resolve the issue. However, Foundry could provide more built-in support for veering off the path if the user allows it, or even algorithmic guides that allow for branching or looping. In addition, flash teams have disagreements like any other team. In one case, miscommunication and disagreement about artistic direction provoked a heated argument between experts on the animation team, causing the director to fire the illustrator and find a replacement.

Currently, the recruitment of each flash team requires a vetting process on Upwork to pre-clear a set of workers who are high-quality and available at the desired time. However, platforms such

as Upwork are moving toward more automated hiring procedures. We envision a future where a user could request an expert for a given wage and quality, the platform mediates to provide one either algorithmically or through crowdsourcing consultants. In the meantime, Foundry could build up lists of trusted experts for each flash team structure in order to provide a quick and trusted recruiting experience.

In the future, the crowd scale of flash teams could enable end users to make more data-driven decisions about collaboration and work, and empower the scientific study of teams. As the same flash team gets run multiple times, Foundry could display estimates for how long tasks tend to take in practice. Likewise, Foundry raises the opportunity to randomly perturb team structures at scale to run field experiments and A/B test collaboration structures.

## 3.7 Conclusion

This paper offers a vision of how computation and crowdsourcing can shape the future of creative, engineering, and analytical work. We introduce flash teams, which are linked sequences of modular goals for crowd experts that can be represented and interpreted by interactive systems. Flash teams benefit from computational authoring: end users can combine modular team elements to create larger organizations and generate teams on request through automated path search. Flash teams also benefit from computational management: they can grow and shrink on demand via elastic recruitment and pipeline results to accelerate completion times. Offline organizations can embed similar techniques into traditional teams, but flash teams open the door to doing so at a much larger scale than previously.

Flash teams offer an opportunity to shift the paid crowdsourcing narrative in both research and practice. Rather than aiming for redundant, independent judgments, flash teams envision a future of crowdsourcing with dynamic collaborations of diverse and interdependent participants. This future would enable traditional organizations to become far more reactive: a few end users with an idea, for example, could temporarily augment their team on demand as the need for certain skills and expertise becomes apparent. Ultimately, we aim to enable experts and amateurs alike to contribute skills they enjoy, on a set of tasks that they find interesting, and at a scale we are just beginning to glimpse.

## 3.8 Acknowledgments

Thank you to the Upwork workers who participated. This research was supported by the National Science Foundation under grant no. 1351131, the Hasso Plattner Design Thinking Research Program, the Precourt Energy Efficiency Center, and Upwork.

# Chapter 4

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

## 4.1 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 workers' contributions, most crowdsourcing approaches rely on *workflows* [19], also known as human computation algorithms [117]. Drawing on prior work [19, 176], we define a workflow as a pre-specified set of decomposed tasks that are sequenced and integrated by computation to reach a final goal. Workflow-based crowdsourcing approaches have enabled crowds to achieve many impressive goals, including document editing [17], email management [98], text translation [79] and software development [108].

However, if the goal of crowdsourcing is to achieve increasingly complex goals [95], many such goals have remained a struggle [96]. Complex goals such as invention [180, 181, 182], writing [3, 91, 96, 134, 163] and product development [108, 148, 184] are highly interdependent, uncertain and dynamic, making them difficult to decompose a priori into pre-specified workflows [45, 48]. Crowdsourcing goals in domains such as these encounter poor early results that poison later stages [117], uncoordinated contributions that make inconsistent changes [17, 96], and workers who do not have sufficient global context to make effective decisions [29, 91, 100]. Crowd-written blogs, articles, and stories,

for example, result in sections that repeat content and have an inconsistent voice [3, 17, 91, 100]. It has remained an open question whether these challenges could be overcome through sufficiently creative workflow design [95].

In this paper, we pursue a better understanding of *why* crowds struggle to achieve these complex and interdependent goals. Why are such challenges so pernicious, and to what extent are they fundamental limits of workflow-based crowdsourcing? We examine this question by studying how crowdsourcing workflows support and constrain crowd workers tasked with a complex and interdependent goal, in particular creating a user-centered mobile web application. User-centered design, which involves rapid iteration in response to user feedback, represents a state-of-the-art complex crowdsourcing goal [103, 148, 184], and it features both dependencies and changing requirements that require coordination across heterogeneous workers with diverse expertise [95].

To explore the challenges crowds face when pursuing complex goals using modern workflow-based approaches, we conducted in-depth case studies of six teams across two conditions. The objective of our comparison was to understand the impact that workflows have on crowds' coordination toward complex goals. In order to isolate and evaluate the effect of crowdsourcing workflows, 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 based on current best practice [148]. In contrast, the role-based teams were given a minimally specified work plan, which did not define any individual tasks or dependencies and therefore lacked the key affordances of crowdsourcing workflows.

All teams were given the same goal, which was to create a party planning task manager mobile web application starting from a rough sketch we provided. The teams in both conditions had the same role structures [12, 169] 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 to compare how the workflow-based and role-based teams, which collectively completed more than 40 intermediate and final deliverables over 108 hours of work time, enacted and adapted their work structures.

This research demonstrates that while crowdsourcing workflows serve as useful coordination artifacts, they inhibit important *adaptation* behaviors that lie beyond the workflow's pre-specified bounds. These insights extend organizational behavior research on coordination and interdependent work [45, 48, 169] to crowdsourcing and online crowd work. Specifically, the role-based teams easily 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 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. The workflow-based teams struggled to adapt, and the role-based teams adapted haphazardly because they had no guiding work plan.

That crowdsourcing workflows inhibit adaptation may help diagnose why crowds have struggled with complex work. Interestingly, however, the dominant paradigm in crowdsourcing continues to rely on pre-specified workflows for coordinating tasks and workers. While advanced crowdsourcing workflows allow the crowd to define tasks for itself before working [91, 100], and enable advanced control flow such as looping [29, 91] and conditional selection between many options [36, 113], they cannot yet add fundamentally new behaviors as work proceeds. For example, even if a workflow for storywriting allows the crowd to reflect on feedback and change anything about the story [91], the crowd cannot pivot to create a poem instead, because the workflow did not expect this adaptation.

Most workflows can only adapt in response to events or outcomes that have been predicted and "pre-programmed" a priori. Furthermore, organizational behavior research has also repeatedly shown that complex work is inherently dynamic and fundamentally *cannot* be pre-defined [45, 48]. Pre-defined workflows cannot capture the full range of situations that arise in interdependent and uncertain settings [171], such as emergency rooms [8] and high-velocity product innovation and technology firms [22, 90]. This work suggests that while predefined crowdsourcing workflows are important for coordination, they may simultaneously be preventing crowds from engaging in the adaptations necessary to achieve complex goals.

Similar to Kittur et al. [95] and Gray et al. [58], this research focuses on understanding and incorporating both the technical and social needs of crowd workers. Our work draws on literature from crowdsourcing, organizational behavior and CSCW and represents the first study of what it is like for crowd workers to be "inside a workflow" as they work on complex and interdependent goals. By doing this, 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. 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.

## 4.2 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 [19]. Below

we summarize research on workflows as coordination artifacts in crowdsourcing and organizations as well as research on crowdsourcing complex work.

### 4.2.1 Workflows as Coordination Artifacts

The use of workflows as coordination artifacts dates back to some of the earliest work on organizational behavior [124, 171], 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.

#### Crowdsourcing Workflows

Crowdsourcing brings together large groups of people to solve large-scale problems [14, 77]. 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 [116, 117, 176]. Crowdsourcing workflows are also referred to as human computation algorithms or crowd algorithms [19, 29, 110, 116]. Tasks are "what a worker is asked to do" [176], such as answer a multiple choice question or label an image. Larger tasks are often decomposed into smaller subtasks, either by the system [96] or by other workers [100]. While tasks are typically given to workers, some such as image labeling [136] and translation [79] may be automatically completed by computers or a combination of humans and machines [145, 174].

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 [17]. Likewise, *iterate-and-vote* workflows enabled surprising feats like messy handwriting recognition [116], and *map-reduce* workflows have shown promise in writing [96]. 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 [36, 117], techniques for real-time crowd powered workflows [16, 20, 104] as well as systems for requesters and workers to collaboratively design workflows [100] and provide timely, task-specific feedback [43]. 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 [170], have started

to use adaptive workflows, there is still a lot of work to be done in order for crowds to be able to effectively 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 [4, 117] and decompositions of work into smaller partitions [96, 100]. Artificial intelligence algorithms rooted in traditional planning models provide additional tools for dynamically creating and managing these workflows [113, 148, 176].

### 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, 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 [124, 137, 171] and in helping to scaffold goals [161]. Specifically, workflows are used in situations where interdependencies and contingencies are predictable environments and goals have low uncertainty [8]. 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) [137].

More recently, however, organizational behavior and CSCW research has demonstrated the limitations of workflows for work that is unpredictable, has changing interdependencies or occurs in high velocity or time-constrained environments [8, 22, 48]. Specifically, in these environments work plans cannot specify all information in advance and therefore must be adapted as goals evolve [22] and contingencies unfold [48, 161] Given that complex work increasingly operates under these circumstances [22], organizational behavior and CSCW researchers have emphasized the need to support both formal and informal (e.g., emergent) coordination [8, 48, 99, 171].

Finally, regardless of the predictability or stability of the work environment, workflows still must be interpreted and applied to the current context. For example, in Lucy Suchman's foundational book, *Plans and Situated Actions* [161], she critiqued computing, artificial intelligence and CSCW for designing based on an assumption that complex human behavior could be achieved through prespecified plans. Human behavior in complex situations, Suchman argued, is not planned cognitively and then executed, but instead is reflectively revised through continuous interaction with the world. That is, instead of making a plan and then executing it unreflectively, individuals update their beliefs and behaviors continuously as they act. These insights and critiques influenced CSCW to stop attempting to code all human behavior into rules [178]. Today, however, crowdsourcing and CSCW research has returned to this practice by designing, developing and adopting crowdsourcing workflows to coordinate workers and tasks.

To overcome the limitations of workflows, researchers and practioners from various domains have

moved away from using workflows as the primary coordination artifact. For example, peer production [13, 15] 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 [21, 34]. 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 [39], as well as flat organizational structures, such as holacracy [151].

In line with prior work, this reasearch highlights many of the limitations of workflows for coordinating complex work. Specifically, this work suggests that the predictability of the work environment, the attributes of the goal and work structure and the social dynamics and practices surrounding workflows play a critical role in their enactment as well as their effectiveness as coordination artifacts. This research also builds on existing literature by exploring the role of workflows and how they are enacted in a new context. Specifically, 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.

While crowdsourcing has mostly relied on workflows to coordinate work, this research demonstrates that the effectiveness of workflows does not necessarily scale to complex and interdependent goals. Goals of this nature cannot be completely defined in advance and often evolve as the work proceed. As a result, workflows can't just be created by algorithms or requesters and then executed autonomously by workers. Instead, as shown in this study, workflows and plans must be interpreted by workers and adapted to local circumstances, interactions and work environments.

### 4.2.2 Crowdsourcing Complex Work

Crowdsourcing is known to work best for problems that can be decomposed and require little expertise to complete [19]. However, Kittur et al. posed the question: could crowdsourcing be used for complex work as well [96]? Following their work, a series of papers proposed workflows and techniques for crowdsourcing complex goals [96, 100, 148]. CrowdForge [96] and Turkomatic [100] 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 important 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 [148]. Other crowd work systems have hired experts for complex goals, such as writing [134], software development [27, 49], mentorship and skill development [162] and physical world tasks [164]. Similar to microtask crowdsourcing systems, however, 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 [19, 93, 100] such as software development [160]. 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 [43] and that balancing personalities of crowd workers can lead to more effective crowd teams with higher quality outputs and higher quality communication [121]. Recent research has also shown that teams of crowd workers that have experience working together are more effective [155].

Despite targeted successes in brainstorming [26, 180], writing [91, 163] and prototyping [148], research efforts have struggled to provide general solutions to the broader wicked problems [149] of innovation, creativity and product development. Recent research on flash organizations, for example, demonstrated an approach for achieving complex and open-ended goals, including product design, software development and game production, with experts from the crowd [170]. To achieve goals of this nature, flash organizations encoded workers' roles into de-individualized roles and introduced techniques for enabling crowds to adapt their structure in response to highly interdependent and evolving goals. The limitation of this approach, however, is that by moving away from workflow-based coordination, in which the organizational structures are only defined at runtime, it becomes more difficult for computation to optimize and replicate the structures since what worked for one organization might not necessarily work for another organization. Furthermore, more research is still needed in order to effectively support crowds as they adapt their structures in realtime.

Whereas coordination in organizations and other contexts has evolved beyond the workflow, the dominant paradigm in crowdsourcing continues to be workflow-based coordination. Recent work, however, suggests that workflows may not be the best approach for achieving complex crowd work. This persistent tension motivates our study in this paper. Specifically, we seek to understand how

a workflow impacts the practice of crowd work. In order to do this, we will modulate the existence and absence of a workflow and observe the coordination patterns of the crowd workers.

## 4.3 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 teams of crowd workers from Upwork who were recruited as part of a larger study on expert crowd work [148]. 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, which was used in prior work [148], was designed to include the attributes of complex goals described earlier (e.g., heterogeneous tasks and workers, specific dependencies and changing requirements). We 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. We gave the teams a target deadline of 13 hours to complete the goal but ultimately gave them as much time as needed.

Using hiring criteria from prior work [148], we recruited 22 workers from Upwork and randomly assigned them into teams. 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. All of the teams were composed of the same three roles, which included a UI Designer, UX Researcher and Web Developer. We authored and deployed the role and work structures using Foundry [148], a computational platform for supporting interdependent crowd work. Foundry was also 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.

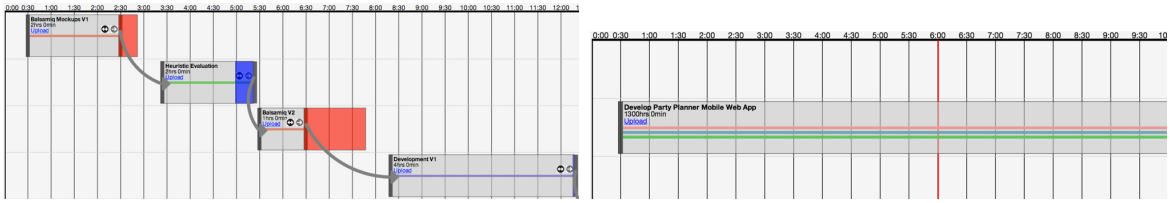
After hiring workers to fill the roles on the teams, we randomly assigned the teams into either a *workflow* or *role-based* condition, resulting in three teams per condition. The workflow-based teams were provided with pre-specified workflows on Foundry (Figure 4.1a), which were designed to mimic traditional 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, in order: (1) low-fidelity mockup by the UI Designer, (2) heuristic evaluation by the UX Researcher, (3) web prototype by the Web Developer, (4) user study by the UX Researcher, (5) revised design by the UI Designer, and (6) revised web prototype by the Web Developer.

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 4.1b). 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.

When the study started, we sent an email to each team describing the high level goal of the project and reiterating 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. 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 [148] 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.




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

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

**Figure 4.1: 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.**

### 4.3.1 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

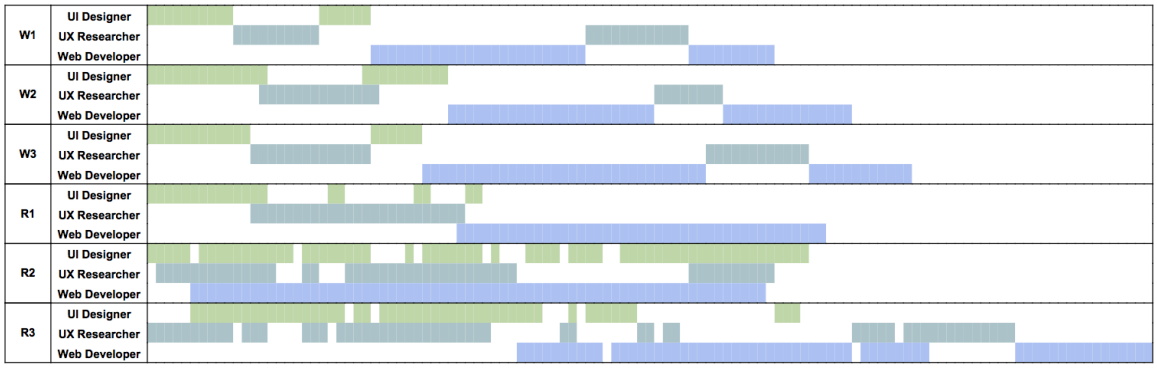



**Figure 4.2: The work structures enacted by the six crowd teams based on the Upwork work diary data. Each block represents a continuous period of time logged by the worker in that row. The structures enacted by the workflow-based teams (W1-W3) aligned with the workflows defined on Foundry whereas the structures enacted by the role-based teams (R1-R3) were all different.**

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.

We first open-coded and analyzed the chat and open-ended survey data for each team individually in NVivo. We compared themes that emerged across teams and conditions to develop conceptual insights [47] and identify common behaviors, practices and challenges [57]. 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.

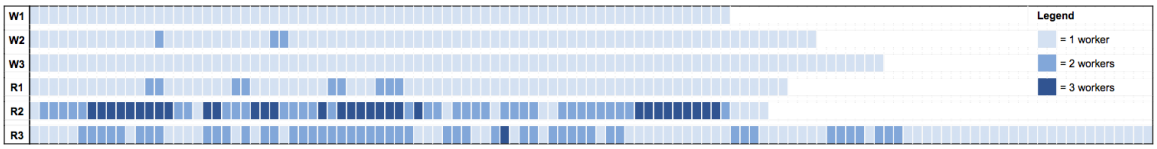



**Figure 4.3: A heatmap of the concurrent number of active workers in each team. There were between one (light blue) to three (dark blue) workers active in each team during any given 10 minute interval. The workflow-based teams (W1-W3) had very few concurrent workers whereas the minimally specified teams (R1-R3) had concurrent workers much more frequently.**

## 4.4 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 reveals 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 [95, 137]; and 2) adapting work structures and deliverables in response to unplanned contingencies and opportunities that emerge [22, 161]. This suggests that for crowds to take on complex goals, crowdsourcing workflows need to support both coordination and adaptation.

In this section, we describe how the workflows were enacted and adapted by the workers. Our findings shed light on the extent to which crowdsourcing workflows support and constrain coordination and adaptation. Specifically, our results show 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 adaptions 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.

### 4.4.1 Workflows Support Coordination

There were clear 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 4.1a) encoded best practices, ensuring that the workflow-based teams did not make obvious missteps. They 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 4.1b) had to decide amongst themselves how to proceed. This led to inefficient coordination and even incorrect work.

### Encoding Best Practices

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 workflows helped orient and integrate actions in the team and enabled consistent and replicable outcomes.

We compared the Foundry work structures (Figure 4.1) to the enacted work structures captured in the Upwork work diaries (Figure 4.2). 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 4.2, 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. Instead, after the developer created the first version of the mobile web app, he iterated on it again based on his own opinions. This revision was fairly similar to the first version, with some aesthetic differences (e.g., colors) but the same usability problems. 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 even mixed: 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.

### Influencing Interdependence Styles

Pre-specified workflows prompted a specific interdependence style. Interdependence, which is inevitable in complex work, plays an important role in determining how to effectively decompose goals, organize teams and structure interactions [171]. 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 [171]. Given that a critical aspect of coordination is how integration of work occurs under conditions of task interdependence [48, 137], understanding the dependency structures that emerge under different constraints is important for supporting complex crowd work.

| 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 4.1: 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.**

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 4.1) and created a heatmap of the dependency structures that emerged (Figure 4.3). The heatmap highlighted clear 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 ($\mu = 1.61$ workers, SD = 0.60), indicating direct synchronous coordination. In contrast, the workflow-based teams rarely worked concurrently ($\mu = 1.01$ workers, SD = 0.02) but rather relied on the workflows to manage the dependencies between workers and tasks, indicating sequential interdependence. A one-tail t-test evaluating whether the minimally specified plans enacted in the role-based teams resulted in more concurrent work was not significant ($t(4)=1.7$, $p = .08$) but future work calls for increasing the number of teams in the sample.

While not statistically significant, the trends in the data suggest that workers default to concurrent work and mutual adjustment unless the workflows override that default. Furthermore, trends in our data capture 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 ($\mu = 6{,}425$ words, SD = 1,864.33) required almost twice the amount of communication than the workflow-based teams ($\mu = 3{,}341$ 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."

### Minimizing Coordination Challenges

Whereas the workflow-based teams encountered few coordination challenges, 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 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 it being the Web Developer's domain. 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. These plans either helped or inhibited effective coordination depending on when they were created and how they were managed. For example, the most successful role-based team 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 incorrect and at odds with user centered design. This UX Researcher also proposed that the developer start coding the web application in parallel so there would be no bottleneck, which resulted in a lot of confusion and frustration and ultimately led to the developer and UX Researcher quitting the team. When expressing his reason for quitting, for instance, 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.

### 4.4.2 Workflows Inhibit Adaptation

In this section, we report how workflows inhibited crowd teams from adapting their work structure. Inhibiting adaptation is problematic for complex work, since complex work is highly interdependent [137, 171], uncertain [8] and non-routine [45], making it difficult (and often impossible) to fully decompose into pre-specified workflows a priori [48]. For example, even though complex goals such as creating a mobile web app can initially be decomposed into sequences of tasks, it is impossible to predict all possible contingencies and task results that might occur [68]. Furthermore, many complex goals can often be achieved in multiple ways and better alternatives might emerge as the work proceeds [22]. These results suggest that by forcing crowd workers to follow predefined workflows, we are doing a disservice to both workers and requesters and limiting the complexity of the goals that crowds can achieve.

We observed two main challenges that workflows present that make it difficult for crowds to adapt. Specifically, in contrast to minimally specified work plans, the rigid structure imposed by pre-specified workflows: 1) makes it difficult for crowds to *respond to contingencies*; and 2) prevents crowd workers from *pivoting when a better opportunity* presents itself. Given that contingencies (e.g. worker turnover and unexpected results) and pivots (e.g., adding missing features and redesigning an interface based on feedback) are inevitable in complex work [68, 111, 171], 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.

#### Responding to Contingencies

Even though the workflows were detailed, they could not take into account all possible outcomes and they restricted workers from adapting when unexpected outcomes occurred. While all of the teams completed their goals and produced functional mobile web apps, 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 [68, 137, 161]. As will be described below, 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 related to workers not showing up or quitting 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 app instead of a mobile web app).

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. Workflow-based teams could not revise the workflows to accommodate the changes and ultimately had to accept the deliverables as-is and move on. For example, when two of the developers in the workflow-based teams decided to code the mobile web apps 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.

Workers' incorrect assumptions about the tasks defined in the workflow also resulted in incompatible deliverables. Given that most assumptions and errors cannot be detected by the workflows, they often went unnoticed until the task was complete. By that point, the changes required were outside the bounds of what the workflow could support, preventing the teams from adapting. For example, one of the UX Researchers incorrectly assumed that the heuristic evaluation should be completed for a native app (e.g., an iPhone app) instead of a mobile web app (e.g., a responsive web site), which have different usability and design standards. He also incorrectly assumed that it would be acceptable to use his own opinion and judgment to construct a critique instead of using Nielsen's heuristics, which he only revealed after completing the task when he sent the following message in the chat:

> "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."

Overall, the UX Researcher's incorrect assumptions 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, 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. Furthermore, 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, 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 far too numerous and unpredictable to design for.

### Pivoting When Better Opportunities Emerge

In addition to preventing crowd workers from adapting in response to contingencies, our results demonstrate that the constraints and structure enforced by workflows inhibit crowds from considering alternative solutions and more importantly, pivoting when better opportunities or approaches emerge. For example, when a work structure or process better suited for the current context emerged, the workflow-based teams tried to change course but couldn't. Furthermore, crowd workers attempted to improve their mobile web apps by adding features that they felt were important but were limited by the predefined work process and constraints embedded in the workflow. 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 upcoming 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 allowed 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 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 apps. 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 due to the strict constraints embedded in the workflows. Specifically, some of the features (delete and search) were partially added to the final deliverables but could not be implemented in time whereas other features (pagination and sort) were added in earlier deliverables but removed by subsequent workers who felt that they were not allowed by the workflow.

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 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 app 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 workers could independently decide to add new features 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 optimized myopically for their own tasks, rather than globally for the final outcome.

Taken together, these findings demonstrate 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 [68, 111, 161].

## 4.5 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 certain types of work structures, such as pre-specified workflows, help crowds avoid errors, coordinate tasks, and manage dependencies, these same structures also inhibit adaptation, which is necessary for achieving complex goals. In contrast, while minimally specified work plans reduce some of the adaptation barriers, they make coordination difficult and do not 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.

### 4.5.1 Why Do Crowds Struggle with Complex Goals?

In this paper, we focus on the impact of crowdsourcing workflows and their ability to support complex crowd work. While there are many other coordination approaches, mostly outside of crowdsourcing, that don't use workflows, the dominant approach in crowdsourcing continues to rely on workflows. If we hope to enable crowds to take on complex goals, then we must understand the role crowdsourcing workflows play as well as their impact on coordination and adaptation. By comparing workflow-based crowdsourcing approaches to role-based approaches that rely on team-based coordination, we

empirically demonstrate the strengths and limitations of these prominent coordination approaches.

Taken together, our results suggest that no crowdsourcing workflow can ever be enough, because there is no such thing as a perfect workflow for complex work. Decades of organizational behavior and CSCW research have demonstrated that there is no right way to organize and coordinate complex work [111, 137, 171]. This research has emphasized that the uncertainty of the task and environment as well as the attributes of the task ultimately determine the best way to coordinate and design the work. Even still, however, it is impossible to predict all possible contingencies. Given that workflows are most useful to the extent one is able to anticipate the tasks, dependencies and contingencies [68, 171], crowdsourcing workflows are only as effective as the contingencies and adaptations they manage to encode. Therefore, crowdsourcing workflows in their current form appear to be fundamentally incompatible with complex work, explaining in part why crowds have struggled to achieve complex work with them to date.

At its core, this result indicates a double-edged sword of crowdsourcing workflows, or a *workflow paradox*. 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 [21, 22, 48]. In their current form, crowdsourcing workflows are akin to the formal coordination processes and artifacts used in organizations [122, 137]. While these formal approaches have benefits, such as clear responsibilities and division of labor [48, 122], they also have important known limitations [137]. For example, formal coordination approaches struggle in uncertain environments or in situations that require rapid action [48]. This research, like ours, has found that interdependent goals often require emergent coordination and adaptation [48] in order to respond to unplanned contingencies and evolving goals. However, as in our study, self-organization like the role-based teams come at the expense of increased coordination challenge.

### 4.5.2 Implications for Design

Taken together, these insights combined with the findings from the case studies have important design implications and emphasize the need for new crowdsourcing approaches and workflows that better support the challenges faced when completing complex and interdependent goals. Specifically, we argue that future crowdsourcing workflows and systems need to support both emergent and formal coordination in order 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? We discuss this question next.

We recommend exploring approaches for letting crowd workers design their own workflows or pieces of it based on the desired final goal and constraints, such as timing, budget and requirements. 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 [39]. Second, when major changes or contingencies occur, crowdsourcing workflows should provide a high priority call to action, such as a 'contingency' button. When activated, the system could temporarily suspend the workflow and immediately notify and gather the requester and all team members into a virtual team room. Once the requester and workers are in the virtual room, they can discuss the situation, agree on the best way to proceed and make mutually agreed upon changes to the workflow.

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 inline with the project goals and constraints. Existing collaboration and coordination systems as well as the reconfiguration techniques introduced by the flash organizations [170] demonstrate a few different approaches for achieving ongoing reconfigurations and adaptation among distributed individuals. For example, Github's branching and pull request model, Google Doc'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. Whereas flash organizations enabled reconfiguration by replacing workflows with computational role structures, our findings suggest the need for creating reconfigurable workflows that serve as editable work scaffolds for achieving a pre-defined 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.

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.

### 4.5.3 Limitations

Our study has several limitations, which can be addressed in future work. One limitation is our sample. We pursued an in-depth case study analysis of six teams composed of 22 crowd workers from Upwork. Future research can replicate and extend this study with a larger sample. In addition, while Upwork and expert design tasks are currently the state-of-the-art in crowdsourcing and crowd research [27, 49, 134, 148, 162, 164], it will be important to investigate the effects of workflows on other types of crowd work, such as Amazon Mechanical Turk. We suspect that similar results will hold, as the nature and effect of the workflows is similar with microtasks.

A second limitation is due to our focus on teams composed of three roles filled by experts, all of which completed the same goal. We also recognize that there are of course other possible workflows, with perhaps more detail, iteration or tasks, that we could have studied. Therefore, we can only generalize our results to similar types of teams and workflows. However, these results resonate with prior work on coordination and teams [68, 137, 171], suggesting that our results likely generalize. For example, no predefined workflow for complex and interdependent goals, regardless of the amount of detail or iteration included, can predict all possible contingencies and outcomes [68, 111]. As a result, when unplanned contingencies and edge cases not programmed into the workflow occur, the workflows will face the same adaptation challenges. Future research should explore these insights across different types of goals and workflows.

## 4.6 Conclusion

This paper explores why crowds struggle to achieve complex and interdependent goals. We conduct in-depth case studies of six interdependent crowd workers tasked with creating a mobile web app 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 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.

# Chapter 5

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

**Most of the content in this chapter is from the following paper:**

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

## 5.1 Introduction

Crowdsourcing mobilizes a massive online workforce into collectives of unprecedented scale. The dominant approach for crowdsourcing is the microtask workflow, which enables contributions at scale by modularizing and pre-specifying all actions [19, 117]. By drawing together experts [148] or amateurs [17], microtask workflows have produced remarkable success in robotic control [105], data clustering [29], galaxy labeling [115], and other goals that can be similarly pre-specified. However, goals that are open-ended and complex, for example invention, production, and engineering [95], remain largely out of reach. Open-ended and complex goals are not easily adapted to microtask workflows because it is difficult to articulate, modularize, and pre-specify all possible actions needed

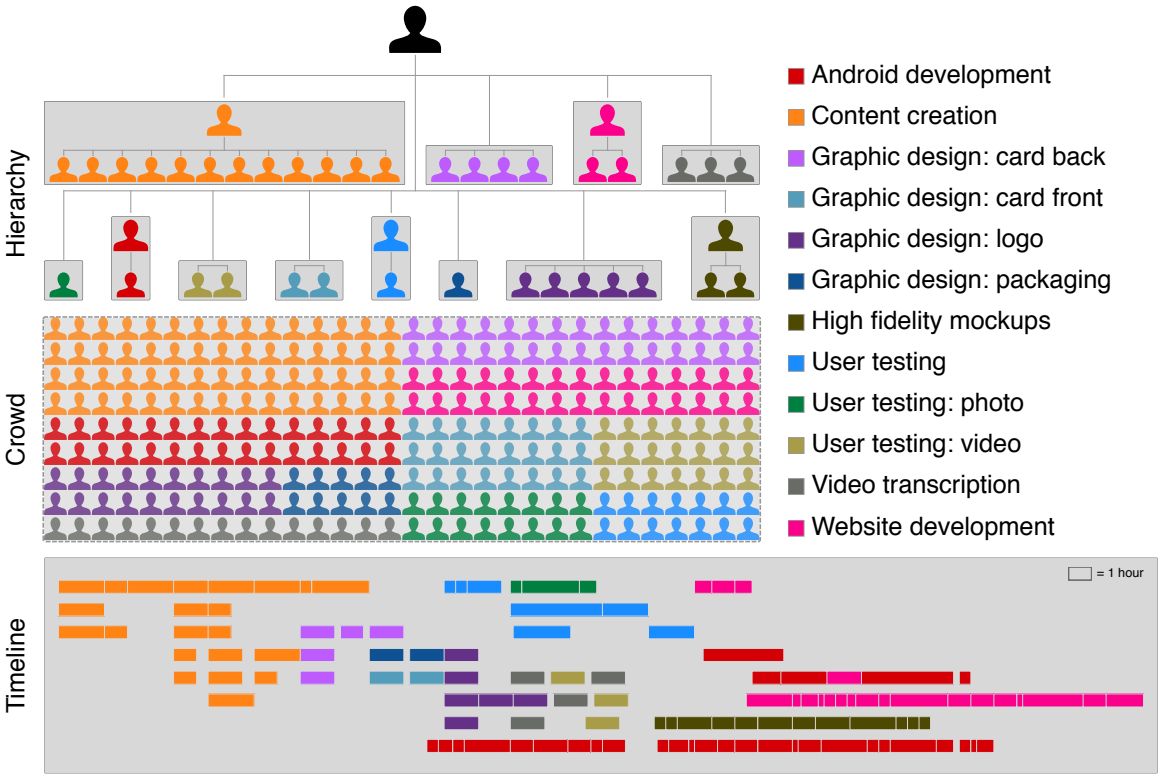



**Figure 5.1: Flash organizations are crowds that are computationally structured like organizations. They enable automated hiring of expert crowd workers into role structures and continuous reconfiguration of those structures to direct the crowd's activities toward complex goals.**

to achieve them [149, 171]. If crowdsourcing remains confined to only the goals so predictable that they can be entirely pre-defined using workflows, crowdsourcing's long-term applicability, scope and value will be severely limited.

In this paper, we explore an alternative crowdsourcing approach that can achieve far more open-ended and complex goals: crowds structured like *organizations*. We take inspiration from modern organizations because they regularly orchestrate large groups in pursuit of complex and open-ended goals, whether short-term like disaster response or long-term like spaceflight [21, 22, 137]. Organizations achieve this complexity through a set of formal structures — roles, teams, and hierarchies — that encode responsibilities, interdependencies and information flow without necessarily pre-specifying all actions [34, 177].

We combine organizational structures with computational crowdsourcing techniques to create *flash organizations*: rapidly assembled and reconfigurable organizations composed of online crowd workers (Figure 5.1). We instantiated this approach in a crowdsourcing platform that computationally convenes large groups of expert crowd workers and directs their efforts to achieve complex goals

such as product design, software development and game production.

We introduce two technical contributions that address the central challenges in structuring crowds like organizations. The first problem: organizations typically assume *asset specificity*, the ability for organization members to develop effective collaboration patterns by working together over time [177]. Clearly crowds, with workers rapidly assembled on-demand from platforms such as Upwork (`www.upwork.com`), do not offer asset specificity. So, our system encodes the division of labor into a de-individualized role hierarchy, inspired by movie crews [12] and disaster response teams [21], enabling workers to coordinate using their knowledge of the roles rather than their knowledge of each other.

The second problem: organizational structures need to be continuously reconfigured so that the organization can adapt as work progresses, for example by changing roles or adding teams [22, 137, 177]. Coordinating many workers' reconfigurations in parallel, however, can be challenging. So, our system enables reconfiguration through a model inspired by version control: workers replicate (branch) the current organizational structure and then propose changes (pull requests) for those up the hierarchy chain to review, including the addition of new tasks or roles, changes to task requirements, and revisions of the organizational hierarchy itself.

Enabling new forms of organization could have dramatic impact: organizations have become so influential as the backbone of modern economies that Weber argued them to be the most important social phenomenon of the twentieth century [175]. Flash organizations advance a future where organizations are no longer anchored in traditional Industrial Revolution-era labor models, but are instead fluidly assembled and re-assembled from globally networked labor markets. These properties could eventually enable organizations to adapt at greater speed than today and prototype new ideas far more quickly.

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

## 5.2 Related Work

In this section, we motivate flash organizations through an integration of the crowdsourcing and organizational design research literature, and connect their design to lessons from distributed work and peer production (Table 5.1).

| | Coordination Structures | Source of labor | Example outcomes |
|---|---|---|---|
| Peer production | Shared repositories (wikis, code) | Open call to volunteers | Wikipedia, Debian Linux [13] |
| Crowdsourcing | Computational microtask workflows | Open call to paid crowdsourcing marketplaces or volunteers | ImageNet [40], Galaxy-Zoo [115] |
| Traditional organizations | Roles, teams, hierarchy | Employees | Incident Command [21], Xerox [34] |
| Flash organizations | Computationally-reconfigurable roles, teams, hierarchy | Open call to paid crowdsourcing marketplaces | EMS Trauma Report, True Story, Enterprise Workshop Planning Portal |

**Table 5.1: Coordination infrastructure and labor source in peer production, crowdsourcing, and traditional and flash organizations.**

### 5.2.1 Crowdsourcing Workflows

Crowdsourcing is the process of making an open call for contributions to a large group of people online [19, 78]. In this paper, we focus especially on *crowd work* [95] (e.g., Amazon Mechanical Turk, Upwork), in which contributors are paid for their efforts. Current crowd work techniques are designed for decomposable tasks that are coordinated by workflows and algorithms [117]. These techniques allow for open-call recruitment at massive scale [144] and have achieved success in modularizable goals such as copyediting [17], real-time transcription [104], and robotics [105]. The workflows can be optimized at runtime among a predefined set of activities [36]. Some even enable collaborative, decentralized coordination instead of step-by-step instructions [103, 183]. As the area advanced, it began to make progress in achieving significantly more complex and interdependent goals [96], such as knowledge aggregation [64], writing [96, 134, 163], ideation [180, 181], clustering [29], and programming [27, 108].

One major challenge to achieving complex goals has been that microtask workflows struggle when the crowd must define new behaviors as work progresses [96, 100]. If crowd workers cannot be given plans in advance, they must form such action plans themselves [110]. However, workers do not always have the context needed to author correct new behaviors [29, 172], resulting in inconsistent or illogical changes that fall short of the intended outcome [100].

Recent work instead sought to achieve complex goals by moving from microtask workers to expert workers. Such systems now support user interface prototyping [148], question-answering and debugging for software engineers [27, 49, 108], worker management [61, 101], remote writing tasks [134], and skill training [162]. For example, flash teams demonstrated that expert workflows can achieve far more complex goals than can be accomplished using microtask workflows [148]. We in fact piloted the current study using the flash teams approach, but the flash teams kept failing at complex and open-ended goals because these goals could not be fully decomposed a priori. We

realized that flash teams, like other crowdsourcing approaches, still relied on immutable workflows akin to an assembly line. They always used the same pre-specified sequence of tasks, roles, and dependencies.

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

### 5.2.2 Organizational Design and Distributed Work

Flash organizations draw on and extend principles from organizational theory. Organizational design research theorizes how a set of customized organizational structures enable coordination [111]. These structures establish (1) roles that encode the work responsibilities of individual actors [88], (2) groupings of individuals (such as teams) that support local problem-solving and interdependent work [30, 62], and (3) hierarchies that support the aggregation of information and broad communication of centralized decisions [34, 185]. Flash organizations computationally represent these structures, which allows them to be visualized and edited, and uses them to guide work and hire workers. Some organizational designs (e.g., holacracy) are beginning to computationally embed organizational structures, but flash organizations are the first centralized organizations that exist entirely online, with no offline complement. Organizational theory also describes how employees and employers are typically matched through the employee's network [51], taking on average three weeks for an organization to hire [37]. Flash organizations use open-calls to online labor markets to recruit interested workers on-demand, which differs dramatically from traditional organizations and requires different design choices and coordination mechanisms.

Organizational design research also provides important insight into virtual and distributed teams. Many of the features afforded by collocated work, such as information exchange [139] and shared context [32], are difficult to replicate in distributed and online environments. Challenges arise due to language and cultural barriers [71, 135], incompatible time zones [140, 146], and misaligned incentives [59, 141]. Flash organizations must design for these issues, especially because the workers will not have met before. We designed our system using best practices for virtual coordination, such

as loosely coupled work structures [72, 139], situational awareness [41, 60], current state visualization [24, 122], and rich communication tools [139].

### 5.2.3 Peer Production

Flash organizations also relate to peer production [13]. Peer production has produced notable successes in Wikipedia and in free and open source software. One of the main differences between flash organizations and peer production is whether idea conception, decision rights, and task execution are centralized or decentralized. Centralization, for example through a leadership hierarchy, supports tightly integrated work [34, 185]; decentralization, as in wiki software, supports more loosely coupled work. Peer production tends to be decentralized, which offers many benefits, but does not easily support integration across modules [14, 70], limiting the complexity of the resulting work [13]. Flash organizations, in contrast, use centralized structures to achieve integrated planning and coordination, even across diverse disciplines. In addition, by tapping into online labor markets, flash organizations overcome peer production's struggles to attract volunteer contributors [65, 69, 118, 156]. In exchange for these benefits, flash organizations face the additional costs of negotiating contracts, motivation crowding [38], and paying for labor.

## 5.3 Flash Organizations

In this section, we introduce flash organizations and our system, Foundry, which enables them. Flash organizations draw together 1) the structure and coordination techniques of traditional organizations to enable open-ended and complex work, and 2) the scale and computational management abilities of crowdsourcing. We describe two problems central to the design of flash organizations, and our solutions to them: asset specificity addressed through computational role structures, and structure adaptation addressed through version control of organizational structures. We enacted these contributions in Foundry, a web platform that enables authoring, populating, and adapting organizational structures (Figure 5.2).

### 5.3.1 Computational Organizational Structures

Flash organizations encode computational structures inspired by organizations. However, traditional organizations premise their organizational structures on *asset specificity* [177], the value that comes from people working together over time. Asset specificity accounts for workers becoming more and more in sync with teammates over time and improving their ability to coordinate and solve problems together. Clearly crowds, with transient participants assembled on-demand through open calls, do not offer asset specificity.

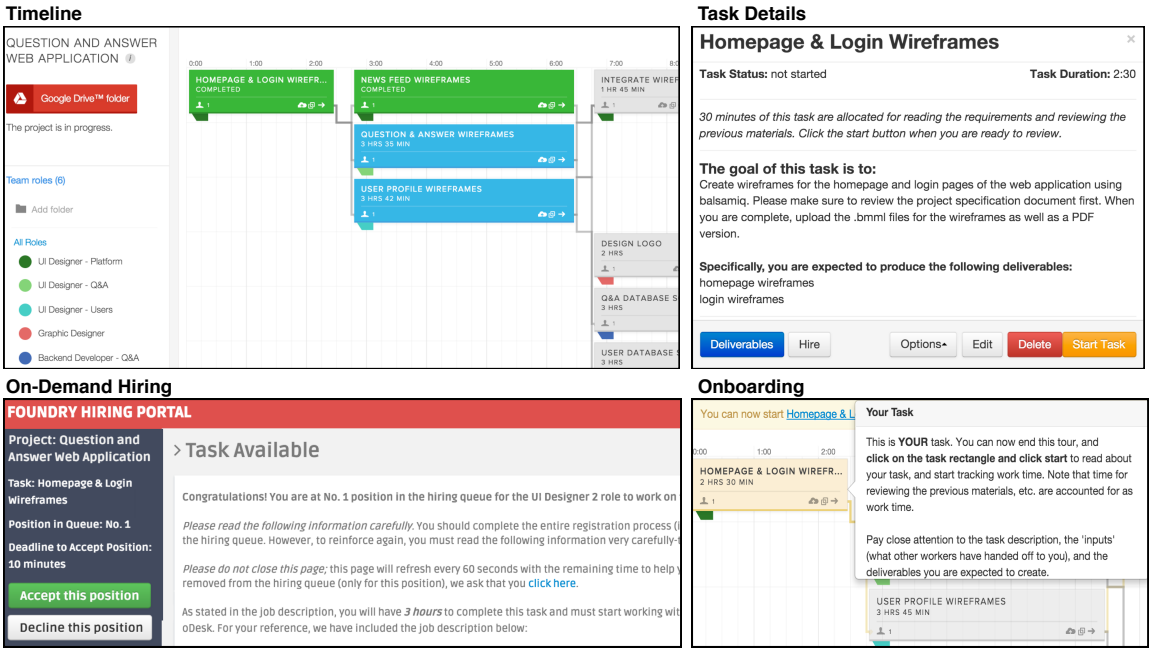



**Figure 5.2: Foundry supports the authoring of flash organizations. The Foundry timeline (top left) displays all roles and current tasks. The task description (top right) displays task requirements and allows the leader to hire a worker to fill the role. When hiring for a new role (bottom left), Foundry notifies qualified workers; those who respond enter a hiring queue. Once hired, they are oriented to the purpose of the organization and upstream and downstream tasks (bottom right).**

To address asset specificity, we draw on research on temporary organizations such as disaster response teams and movie crews. These temporary organizations coordinate successfully even without asset specificity [12, 21] by relying heavily on *role structures*, which are activity-based positions that can be assumed by anyone with necessary training [12, 169]. Example role structures for a film crew include main grip and director; example roles for a disaster response team include strike team leader and firefighter. Role structures encode responsibilities and interdependencies; boom operators and grips know how to coordinate by virtue of their roles.

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

A flash organization coordinates these roles by arranging them into a hierarchy, which encodes authority and decisions rights [175]. Hierarchies enable flash organizations to take action with centralized, coordinated purpose, much like a director has executive authority on a movie set. Hierarchy

means that flash organizations can be formally represented as a tree of nested role structures. The role structure encodes interdependencies, and the nesting encodes hierarchy and decision rights. Leaf nodes are roles representing workers, which can be nested into teams. Teams can optionally be led by a team lead. For example, a team might include a set of workers with expertise in interface design, nested under a UI Design Lead. At the top of the tree is the organization's leader.

The flash organization's hierarchy (Figure 5.1 top) determines the actions that each worker can perform. The goal is for information to flow up to the central leader so that decisions can be made with awareness of the state of the entire organization. When a worker submits a task in the system, the hierarchy one level above is alerted, and then reviews and accepts it or returns it with feedback for revision. The leader has full formal decision rights, or delegates rights to team leads.

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

Once a role is created, it can be assigned *tasks*, the basic unit of work in Foundry (Figure 5.2). Foundry tasks are parametrized by desired duration, description, required inputs from other tasks, required outputs to other tasks, and the individual directly responsible for ensuring the task is completed [148]. A task timeline visualizes sequencing. Workers can start, pause, and complete tasks using the Foundry interface. For each role, Foundry highlights their upcoming tasks and shows the remaining time for any active tasks. When a worker submits a task, they upload the file to Foundry, and answer documentation questions to record decisions or other information to the organizational record. Foundry visualizes the role hierarchy by organizing tasks into rows by team (Figure 5.1).

Finally, Foundry draws on CSCW best practice for virtual organizations. A timeline interface supports shared awareness of progress; text and video chat, availability indicators (present, offline, idle), and Slack integration support coordination (Figure 5.2). Foundry automatically publishes major notifications such as tasks started or completed, and pull requests issued.

### 5.3.2 Reconfigurable Organizational Structures

In contrast to crowdsourcing efforts using workflows, flash organizations are designed to adapt throughout the work process. To enable this adaptation, the organizational structures must be modified as new opportunities or challenges arise. For example, when a movie crew encounters unexpected weather, they change which scene they are shooting, quickly activate new parts of the organization, or redeploy experts to fill needed roles [12]. Likewise, when a disaster response crew finds unexpected materials at the scene of a fire, a firefighting team must quickly reconfigure itself to mount a hazardous materials response [21]. Flash organizations likewise must be able to adapt

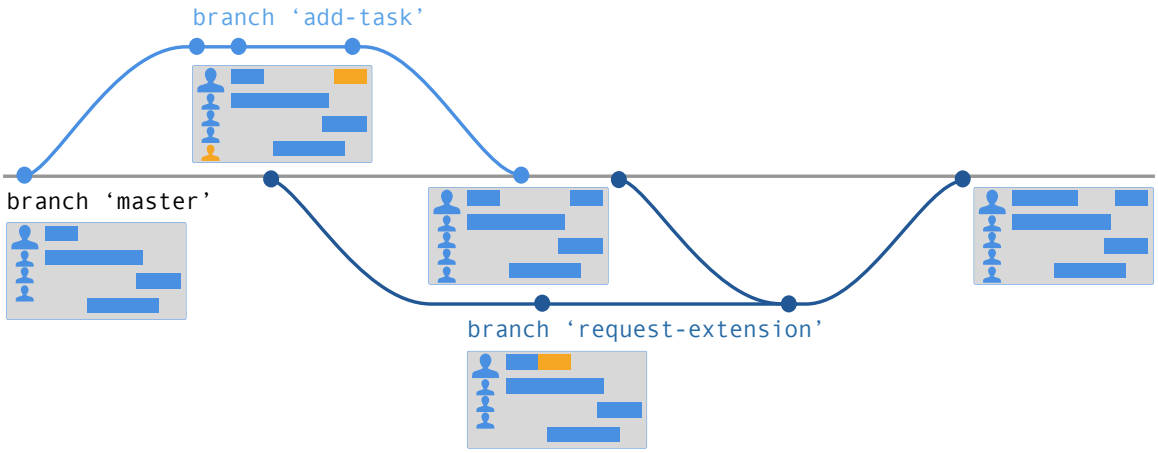



**Figure 5.3: Workers can branch the current organizational structures, make any desired edits, and then issue a pull request for review. This mechanism enables the organization to continuously adapt.**

to changing conditions by reconfiguring their structures.

However, for adaptation to be feasible, flash organizations need to enable distributed workers from across the organization to update the organizational structures in real-time. Members of an engineering team, for example, might be adjusting deliverables at the same time as a quality assurance lead is changing tasks' dependencies. This creates pressure to adapt rapidly, but flash organizations must also not lurch without centralized purpose in reaction to each new adaptation. A single shared, multi-author organizational structure (e.g., Google Docs) would be reactive but susceptible to uncoordinated changes; distributed structures ensure consistency but require significant effort to recombine. Likewise, a globally-writeable organizational structure (e.g., a wiki) enables changes to come from any worker, but can cause organizational chaos if workers disagree and engage in edit wars [105]; a locally-writeable structure accessible only to the leader centralizes control but silences good ideas from lower in the hierarchy. Flash organizations ideally require an approach that allows workers to explore changes quickly in a sandbox, and then ask people up the hierarchy to quickly review and merge them.

Flash organizations enable reconfiguration through a technique inspired by version control, enabling workers to branch, merge, and issue pull requests for the organizational structures that define the organization (Figure 5.3). There are many flavors of version control, for example distributed (git) vs. centralized (Subversion), and changes pushed directly (git and Subversion) or through a review process (GitHub pull requests and Subversion with patch files). We chose our model based on flash organizations' design requirements. There is only one instance of each flash organization, so a decentralized model with multiple copies is unnecessary. However, to coordinate changes, flash organizations require a model that supports review and automatic merging. So, at a high level, flash organizations enable a contributor to *branch* (copy) the organization's current state and edit

it while the system tracks the differences from the *master* (original), then merge any changes when the branch is ready via a *pull request* where other team members review the changes and decide to accept or reject them.

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

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

As in software version control, implementing this approach requires merging the branch and the master organizational structures. Typical version control operates using a three-way merge on unstructured text such as program code [131]. Three-way merges require tracking history of each version, so that the algorithm can identify a common ancestor of the master and the branch in order to perform the merge. So, Foundry maintains ancestry history for organizational structures: as in source code, the parent is the version of the organizational structure that was branched or edited. However, three-way merges require diffs, and most diffs are designed for text instead of hierarchically structured objects like Foundry's organizational structures. So, we use a diff algorithm designed for structured objects (e.g., arrays) and hierarchical objects (e.g., JSON, which Foundry uses) [150]. With this infrastructure, the three-way merge algorithm can return insertions, deletions, and edits, which Foundry manages automatically, as well as any conflicts, which Foundry returns to the user to resolve. Following this branch-and-merge process, Foundry hires newly required workers and notifies members of changes.

### 5.3.3 On-Demand Hiring of Expert Crowd Workers

Foundry populates the organizational structures using *on-demand hiring* (Figure 5.2). On-demand hiring is entirely automated and enables flash organizations to hire relevant experts from expert crowdsourcing marketplaces such as Upwork within about fifteen minutes on average. Foundry enables on-demand hiring through worker panels that have been pre-vetted via a skill-based qualification task. These panels are a retainer pool of high-quality workers [16]. Example Foundry panels include Android application development, graphic design, quality assurance testing, and video animation.

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

We formed panels by posting skill-based qualification tasks [87, 132] for each panel. Upwork workers completed a 1–3 hour task (e.g., simple Android development, logo design, QA on an existing website) to apply for the panel. Their submissions were reviewed by an expert reviewer such as a highly rated domain expert on Upwork. If the worker's submission was of sufficiently high quality, the worker was added to the panel. We envision that as the reputation systems on platforms such as Upwork improve [55, 75], the panel role could eventually be played by the platform itself.

Arriving in the middle of a fully functional organization requires workers to quickly learn their specific responsibilities and interdependecies, and organizational goals. Foundry onboards new workers by orienting them to their role responsibilities and their position in the organization through a guided walkthrough (Figure 5.2 bottom right). Foundry then calls out relevant inputs, upcoming tasks, and the description of the worker's first task. This process takes under five minutes.

## 5.4 Evaluation

Do flash organizations enable crowds to mount large-scale coordinated efforts toward complex and open-ended goals? In this section, we explore this question by reporting results from a system deployment where three flash organizations pursued goals that have remained open challenges for crowds: open-ended product design, software development, and game design. These projects represent more open-ended and complex goals than past targeted successes such as crowdsourced interface prototyping [148], code debugging [27, 108], and ideation [7]. Our evaluation strategy is inspired by prior work in crowdsourcing, which has likewise demonstrated proof of concept goals via systems

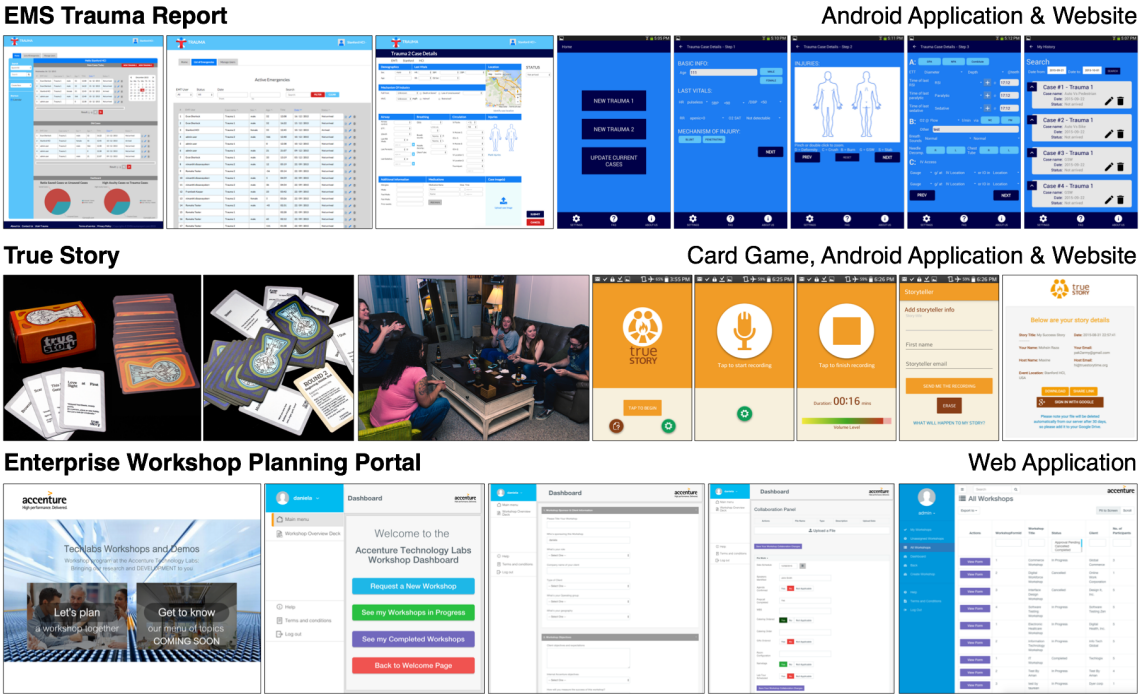



**Figure 5.4: Three flash organizations successfully developed: (top) a tablet application and web portal for emergency medical responders; (middle) art, content, and a supporting application for a storytelling card game; and (bottom) an enterprise IT portal for consultant workshop planning.**

operating on real tasks (e.g., [17, 96, 106]).

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

### 5.4.1 Method

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

The three resulting organizations spanned software, product, and game design. The first organization, EMS Trauma Report, designed and led by a medical student, used the crowd to create a prototype Android mobile and web application for emergency medical technicians (EMTs) to report

trauma injuries from an ambulance en route to the hospital. The second organization, True Story, designed and led by a team of crowdfunded card game makers, used the crowd to design, manufacture, and playtest a storytelling card game and an accompanying mobile application. The third organization, Enterprise Workshop Planning Portal, designed and led by a member of a technology lab at the Accenture software consulting firm, used the crowd to create an enterprise web portal to administer client workshops.

All decisions were made by the organizational leaders, including the creation and execution of roles, teams, and tasks. Leaders could use Foundry to delegate decisions to team leads and workers. Their customized organizational structures were automatically populated with diverse crowd workers including graphic designers, poets, and programmers. Table A.7 in the Appendix reports the panels accessible to the leaders, and the number of applicants in each panel hiring process. We paid Upwork workers their posted profile hourly wage. When necessary, we aided the leaders in using Foundry, but did not make any organizational decisions on their behalf.

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

## 5.5 Results

All three leaders spun up and led an organization to complete their goals within six weeks, convening workers on-demand in 14 minutes on average. Each organization successfully completed its goal (Figure 5.4) to the satisfaction of the leader and received an acceptable quality rating by the three expert reviewers. The organizations collectively comprised 93 crowd workers, including 22 team leads and 24 teams (Figure 5.5). These workers completed 639 tasks across 3,261 person-hours of work time. Altogether, the organizations wrote 52,000 lines of software code, including two mobile applications and three full-stack web applications, and created two illustrated 80-card decks. The median task across organizations lasted 3.05 hours, and the median daily concurrent work time was 14.1 hours by a median five workers. The organizations spent $46,191 (EMS Report), $6650 (True Story), and $36,604 (Enterprise Workshop Planning Portal), with the interquartile range of workers' profile wages $15–$30 per hour. We first present a case analysis of the adaptation and hiring involved and then present quantitative measures of these activities.

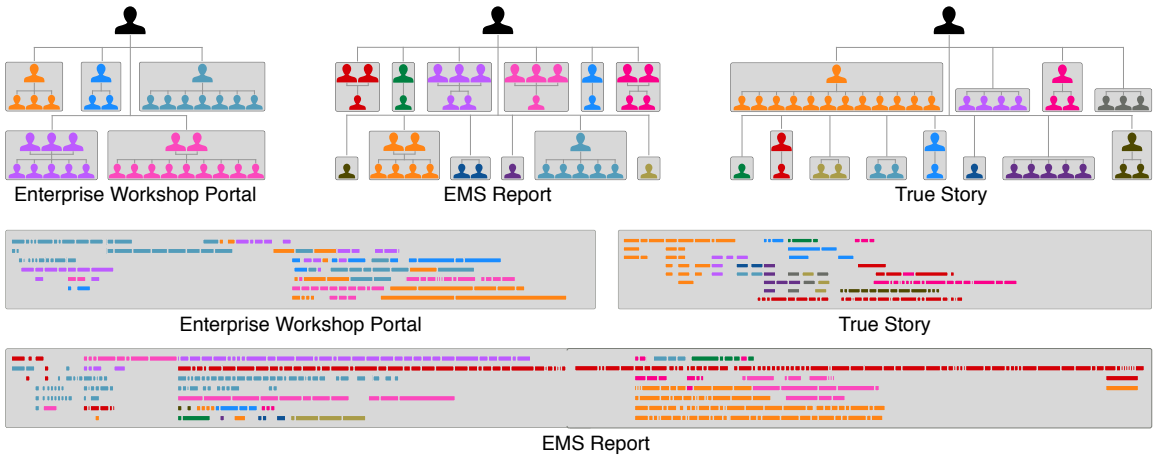



**Figure 5.5: Final organizational structures and task timelines for the three flash organizations. Colors indicate different roles and corresponding tasks (details in Appendix). Structures varied from flat to nested hierarchies, and included 24 teams and 639 tasks across 3,261 person-hours of work time.**

### EMS Trauma Report

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

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

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

The team produced an early prototype, prompting the EMS leader to spin up a user testing team. He took the prototype to local users and sent their feedback to this team. User feedback

indicated that the system needed to be redesigned to accommodate high-acuity trauma cases. The EMS leader sketched out wireframes for the revised design, which team leads used to create pull requests for 15 additional tasks to cover high-fidelity mocks, engineering, testing and debugging.

During this same period, the EMS leader wondered about compliance with federal Health Insurance Portability and Accountability Act (HIPAA) regulations. He wanted to hire someone to oversee HIPAA compliance, but Upwork did not have workers with that expertise. He hired a web security engineer in Egypt to train himself on HIPAA policy and connected him with the local university compliance officer.

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

**True Story**

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

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

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

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

### Enterprise Workshop Planning Portal

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

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

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

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

### Results: Reconfiguring Organizational Structures

Each organization continuously adapted to changing conditions by adding people, tasks, teams, and time, and by revising groupings, hierarchy, and task requirements. These on-demand adaptations, which included 113 pull request changes in True Story, 118 in Enterprise and 335 in EMS Report (Table 5.2), resulted in 566 pull request changes. They came from leaders (9.4%), team leads (68.2%), and team members (22.4%). Adaptation was continuous over the duration of each organization, with

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

**Table 5.2: Automated hiring and organizational reconfigurations.**

a median of four changes per day per organization. Changes were both top-down per the leaders' directives, and bottom-up per workers' initiative.

Top-down adaptations allowed quick realignment whenever the leader needed to react to new information or unexpected feedback. Table 5.3 summarizes examples, including adding new roles and reconfiguring hierarchies (e.g., True Story spinning up a mobile application development team with one week until the deadline). Such top-down adaptations are more reminiscent of traditional requester-driven crowdsourcing models. Flash organizations also allowed crowd workers to initiate reconfigurations bottom-up. Examples in Table 5.3 include adding tasks that had not been anticipated (e.g., EMS workers proposing revised content for the application's web page).

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

**Bottom-up change: add new tasks** *EMS Trauma Report*
One of the marketers realized that the application's hiring page would be better off as a feature overview. Other workers agreed, so he created four pull requests adding tasks to redesign the page and its content.

**Bottom-up change: add new tasks** *Enterprise Workshop Portal*
As the engineering teams completed their milestones, the quality assurance team took the initiative to create pull requests with tasks that would coordinate the upcoming engineering team's testing process.

**Top-down change: reconfigure hierarchy** *EMS Trauma Report*
A back-end engineer showed particular initiative and skill. The EMS leader reconfigured his role in the team hierarchy, making him the team lead.

**Top-down change: add new role** *True Story*
The True Story leaders sourced card content from team of poets. Upon review, they decided to create a new role on the team to integrate content.

**Table 5.3: The flash organizations used top-down pull requests per leaders' directives, and bottom-up pull requests per workers' initiative.**

**Results: On-demand hiring**

Seventy five workers were hired automatically across each organization (Table A.1 in the Appendix). Leaders used this on-demand hiring process to fill new organizational structures in a median of 13.7 minutes. In contrast, (manual) warm hires took much longer, a median 15 hours. These hiring processes unfold on significantly different scales: automated hires in 14 *minutes*, warm hires in 15 *hours*, and traditional organization hiring processes in 14–25 *days* [37].

The organizations used on-demand hiring to quickly hire needed expertise and to source diverse ideas. Examples are summarized in Table 5.4, including True Story hiring a team of 12 poets and a "Chief Poetry Officer" to create game content, and EMS hiring a security engineer to train himself on American HIPAA privacy laws.

Although flash organizations could in theory recruit new crowd workers for each new task, the organizations in practice accreted members over time. Leaders used Foundry's hiring functionality to rehire members for new tasks. Rehiring minimized the necessary onboarding, and meant that organization members inhabited several different roles in the organizational structure at different points in time. On-demand hiring produced other challenges. There was a somewhat unpredictable skill fit of the hired crowd experts. Reliable reputation signals remain an issue for online labor markets [75, 147]. Future research can aim to improve crowd platforms' reputation systems, and in particular to identify professionals with similar styles or skills in a domain.

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

**Table 5.4: Organizations used on-demand hiring to spin up expertise.**

## 5.6 Discussion

Like many other social structures, the nature of work is being reshaped by the internet, computation and algorithms. Responsibilities that used to be the domain of human managers are becoming the domain of computational systems. To date these systems have focused on distributed and independent work [19]. In this paper, we envision a future in which computational systems instead orchestrate organizations that achieve complex and open-ended goals requiring diverse expertise. We present a field deployment of our system in which flash organizations automatically hired experts

from the crowd into role-based organizational structures and reconfigured the structures as work progressed.

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

Reciprocally, flash organizations also advance research on organizational design in three main ways. First, flash organizations draw on the principles of role-based coordination that enable temporary organizations, but represent the first example of a computational system that encodes and reconfigures these role structures. Future research can explore the social and behavioral dynamics introduced by this approach. Second, flash organizations convene expertise near-instantaneously rather than reconfiguring fixed groups of employees like traditional organizations. This property means these temporary organizations can fluidly assemble participants from online labor markets, a novel capability compared to traditional organizations. But it introduces a trade-off between rapid hiring and workers' familiarity with each other and with the organizational context. Future research can explore this trade-off, and develop approaches for supporting familiarity [155] and helping with relevant context. Third, because flash organizations log all organizational activity, they offer an unprecedented opportunity to conduct data science on organizational structures and processes. Organizational structures are a powerful coordination mode in their own right, but coupled with large-scale data on tasks, roles, hierarchies, and customized workflows, they offer a powerful tool that can usher in an era of flexible networked collaboration.

Flash organizations advance a future of work that is increasingly mediated by computation and algorithms. As new such techniques are developed, continued conversation around desired social outcomes is necessary [58, 157]. If the sociotechnical ecosystem is not designed carefully, flash organizations (like other labor shifts through history) may drive down wages and impose rigid working conditions [85, 127]. Instead, we hope that flash organizations will offer crowd workers, who are currently predominantly piecework employees, the opportunity to join longer-term and more fulfilling projects. More broadly, flash organizations help envision a world that enables crowd workers to pursue long-term careers [95], including skill growth [162], access to labor collectives [154], and guarantees of stable income.

### 5.6.1 Limitations

Flash organizations may not be appropriate for every kind of goal, and more research is needed to explore this design space and its failure points. For example, because flash organizations are nimble, they are likely to prove a good fit for early-stage organizations, or organizations that want to prototype new project-level efforts. However, efforts that require high asset specificity and rely on years of expertise, for example familiarity with Google's massive code base, may benefit from

more traditional formats. Hybrid opportunities exist: large stable organizations could view their own employees as a crowd, engaging workers in primary projects while rotating them onto secondary on-demand projects to enable secondary teams to grow rapidly if needed.

Other tradeoffs and limitations exist: flash organizations reach a worldwide labor pool, but must therefore contend with cultural and timezone differences [71, 140, 146]; flash organizations enable near-strangers to coordinate effectively through roles, but strangers are less effective than familiar teams [80]; crowd platforms' hiring can be noisy due to over-inflated reputation systems [55, 75]. No organizational form is perfect for all situations [111], and flash organizations are better suited for situations where these tradeoffs matter less. Future designs will iterate on these issues.

We chose a field deployment as our evaluation strategy, which allowed us to demonstrate the feasibility, strengths, and limitations of the approach. Field study deployments are less able to isolate specific causal mechanisms than a randomized field experiment with a matched counterfactual. However, they are a strong fit when the phenomenon is nascent [46], and they provide a strong method for "thick" [114] descriptive accounts, including strengths and limitations.

## 5.7 Conclusion

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

## 5.8 Acknowledgments

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

# Chapter 6

# Discussion

This thesis advances a vision of expert crowd work that supports complex and highly interdependent goals. In particular, the question that motivates much of this research asks: can we use computational techniques and organizational structures to bring together crowd collectives that can achieve complex and interdependent goals? To achieve this vision, we shift away from crowd work that relies on homogeneous microtask workers or even independent expert workers and focus on interdependent teams and reconfigurable organizations composed of expert crowd workers that are gathered on-demand and are computationally managed.

While expert crowd work has evolved since the introduction of flash teams, there are still many opportunities to further expand the capabilities and effectiveness of expert crowds. In this chapter, we lay out these opportunities and tradeoffs and reflect on how this work can help shape future research directions in crowdsourcing, human-computer interaction and organizational behavior.

## 6.1 Expert Crowd Work Tradeoffs

This thesis introduces and evaluates two generalizable approaches for achieving complex goals with expert crowds. By combining computation, organizational design principles and expert crowd workers, we demonstrate that crowds can achieve goals ranging from web development to animated videos to large scale enterprise applications. In this section, we reflect on the tradeoffs of the flash team and organization approaches as well as expert crowd work more broadly.

### 6.1.1 Hiring

Despite the fact that online labor marketplaces, such as Upwork, have resulted in millions of successful matches, the hiring process remains burdensome and inefficient [142]. Many of the difficulties

arise from the challenges and transaction costs caused by the influx of potential options and asymmetric information, both of which hinder the marketplace's ability to facilitate successful matches. These challenges, which are known as search frictions [9], impact agents on both sides of the market (e.g., requesters and workers). Two of the most prevalent search frictions in online labor marketplaces, such as Upwork, are application costs and screening costs. For example, an increase in applications can result in congestion, which can make it difficult for requesters to screen every application, thereby causing some applications to never get screened. Furthermore, due to the lack of information and transparency in many online marketplaces, a requester may screen and select a promising applicant only to find out that the applicant is no longer available [9, 53].

The instant hiring process introduced in the flash organizations served as a workaround for these hiring challenges. By screening workers in advance for specific roles and expertise, this approach has the potential to reduce transaction costs and search frictions and ultimately make the marketplaces more efficient. The challenge with this approach is that it requires a significant upfront cost of building panels of experts. This upfront cost, however, is similar to what workers and requesters face each time they apply to or post a job. The benefit is that once a worker has been screened and added to a panel, they no longer need to apply for jobs or be screened, making the entire hiring process much faster later on. Ideally, if this approach proves to be successful, online marketplaces could create the panels and make them available as a feature to workers and requesters, thereby reducing the upfront cost.

This approach to instant hiring is best suited for jobs requiring skills that are popular and well represented among workers in the marketplace. Furthermore, by allowing requesters to hire workers right when they are needed, instant hiring helps resolve time zone and scheduling challenges, especially for shorter and well defined tasks. Finally, through our interactions with panel members during the application and panel onboarding process, it became clear that applicants are readily available, interested in the instant hiring model and eager to get started. For example, panel members frequently reached out to inquire about when the next task would become available and to emphasize that they were eager to participate. Similarly, many applicants indicated in their job applications that they were excited about the instant hiring model. Therefore, while further testing is needed to fully understand this approach, the results from the three flash organization deployments along with our experience creating and maintaining the panels suggest that this approach has the potential to improve the pain points and inefficiencies in online hiring processes.

While the instant hiring approach resolves many of the hiring challenges and marketplace inefficiencies, there are still challenges and limitations to the approach that need to be resolved in future work. First, there is a large time investment and start up cost to creating the panels. This requires posting jobs for each panel, reviewing applicants and screening each one to confirm that they are qualified for the panel. Second, the first come first served queuing approach used to assign the tasks currently assumes that all panel members are equally as qualified for the tasks sent out, which is not

always the case. Future iterations of the instant hiring approach could prioritize the matching based on worker's past experiences and outcomes, timezone, availability, among other criteria. Finally, one other challenge with the instant hiring approach is that worker's are unable to predict when the task available notifications would be sent out. In the flash organizations we deployed, this resulted in workers waiting around their computer in case a notification was sent out. This issue would likely be solved with scale. Specifically, if more tasks are being sent out to the panels, similar to platforms like Uber and Lyft, then there would be a constant flow of opportunities to workers.

### 6.1.2 Motivation and Incentives

Paid expert crowds are typically extrinsically motivated to complete their tasks. While money is a big motivator, expert crowd workers are also motivated by their reputation scores, which are often highly inflated [75].

The payment incentive structures used in online labor marketplaces, such as Upwork, have tradeoffs. Currently, workers on Upwork can be paid hourly or a fixed price set by the requester in advance. Hourly pay structures encourage workers to work slower to accumulate more hours and income. In contrast, fixed price models encourage workers to get tasks done as quickly as possible in order to move on to the next paid opportunity. Furthermore, the guarantees and protections provided by Upwork, which rely on their work diary and screenshot technology, do not apply to fixed price contracts. This makes it even easier for worker's to work less or spend time on other tasks and activities. Therefore, for these reasons, it is important to consider these tradeoffs and the implications they have on worker's motivations and outcomes when deciding how to compensate workers.

In addition to money, another big motivator for paid expert crowd workers is their reputation. On Upwork, reputation is indicated through an "job success score", which Upwork defines as the ratio of successful to unsuccessful contract outcomes[1]. For many years, however, including when we ran the flash team and flash organization projects, reputation scores consisted of a five-star average feedback score from a worker's prior requesters. This was problematic because it led to awkward interactions and negotiations between workers and requesters, resulting in highly inflated rating scores. Given that reputation systems are intended to reduce transaction costs by improving search and information filtering, distorted reputation scores make the entire marketplaces less efficient.

By replacing the 5-star average feedback store with the more holistic job success score, Upwork aimed to incorporate additional criteria in addition to the numerical 5-star ratings provided by the requesters. While this new score might result in less inflated scores, our experiences with the flash teams and flash organizations imply that many of the same challenges regarding incentives and reputation ambiguity will remain. Below we summarize several of these challenges.

First, the worker's profile displays both the Job Satisfaction Score as well as their work history, which contains all previous projects completed on Upwork along with the rating and review provided

[1] https://support.upwork.com/hc/en-us/articles/211068358-My-Job-Success-Score

by the requester. As a result, like before, there is still an awkward interaction and negotiation between workers and requesters with regard to this score. Furthermore, even though requesters can submit private scores that are not shown to workers or displayed on their profile, the public scores and reviews, which are displayed to future requesters will likely still be inflated. Finally and most importantly, the new scores make it unclear to requesters and workers which factors and performance outcomes are influencing their score. Given that the criteria used to calculate the job success score vary across workers depending on their history, the significance and reliability of the score becomes much harder for requesters to assess.

### 6.1.3 Speed and Scale of Expert Crowds

By combining the range of expertise readily available in crowdsourcing marketplaces with the scale and interactivity of computation, this thesis demonstrates the speed benefits of expert crowd work. For example, flash teams achieved web applications in as little as 11 hours and completed even more complex goals, such as a MOOC platform with content in under 20 hours. Flash organizations, moreover, automatically hired workers in under 14 minutes on average, and completed three highly complex, large scale, open-ended goals in 46 days or less.

Compared to traditional teams and organizations, therefore, a big benefit of expert crowd work is the speed and scale in which workers can be convened and goals can be completed. These benefits introduce the possibility of teams and organizations with unique affordances and capabilities. For example, teams could operate 24 hours a day or be composed of workers representing multiple countries, cultures or languages. Future work should explore the implications of these affordances as well as ways in which requesters and crowd workers can both benefit from them.

### 6.1.4 Quality Control

Similar to microtask crowdsourcing, expert crowd work needs to deal with issues regarding quality control. Unfortunately, hiring expert workers from online labor marketplaces does not guarantee high quality outcomes. This is in part due to the: 1) wide range of quality of workers available on the marketplace; 2) unverifiable, distorted or ambiguous signals provided on workers' profiles; 3) lack of coordination tools and quality control procedures embedded into the marketplaces.

Prior work has relied on various techniques for quality control. Examples of these techniques include crowds vetting the output of other crowd workers [17], aggregating crowd outcomes, votes and independent agreement measures [116, 173], providing workers with crowd-generated feedback [43, 120, 179] and comparing outputs to the ground truth [83]. Most of these techniques, however, are designed for microtask crowds, which typically complete much smaller tasks compared to expert crowds. Furthermore, microtask vetting techniques don't apply to expert crowds since most expert work is not categorical or orchestrated through worker agreement measures.

Flash teams and flash organizations take inspiration from prior approaches but rely more heavily on alternative quality control techniques better suited for expert work. For example, flash teams used role structures, which were filled with workers pre-screened for specific skills. We also provided the teams with example deliverables to better communicate expectations. Additionally, in both the flash teams and flash organizations, Foundry decomposed tasks, specified inputs and outputs and identified dependencies. By defining the work and embedding constraints directly into the work structures, Foundry helped prevent certain pitfalls, such as tasks starting before prior tasks were completed, thereby reducing the chance of lower quality outputs.

In addition to the quality control techniques used by the flash teams, the flash organizations managed the quality of outcomes in several other ways. First, workers admitted to each of the role panels were pre-screened. Specifically, after submitting the application on Upwork, qualified workers were invited to complete a starter task, which was designed to be similar to the types of tasks they would be completing. After completing the tasks, we admitted the workers whose deliverables were of our good quality and satisfied our criteria. Second, the tasks on Foundry required workers to document important details and decisions they made, which were then visible to future workers. Furthermore, the task completion process also required confirmation that all outputs were completed and uploaded to the shared google drive folder, preventing problems due to missing deliverables later on. More importantly, when there were quality issues or potential issues, the flash organizations could be edited to resolve the problems. For example, a low quality task could be completed again or revised as needed. Finally, by introducing hierarchy into the work structures, the flash organizations could maintain a global view, which helped ensure consistent and well integrated outcomes.

### 6.1.5 Goal Decomposition and Work Structures

Finally, there are also tradeoffs with regard to how goals are decomposed as well as how work structures are designed. Typically, crowdsourcing relies on tasks that are decomposed into independent subunits that can then be recombined to create much larger deliverables. These tasks are coordinated using crowdsourcing workflows, also known as human computation algorithms. The challenge with workflow-based crowdsourcing approaches is that they struggle to scale to larger, open-ended goals. As we demonstrated in chapter 4, goals of this nature are difficult to decompose a priori and require adaptation as the work proceeds, which workflows don't inherently support.

Flash organizations introduced reconfigurable work structures, which enabled both crowds and requesters to adapt the organization's structure using distributed version control techniques. By overcoming the need to completely define goals in advance, flash organizations open up new types of goals to crowdsourcing. The challenge with reconfigurable work structures, however, is that work structures are much more emergent and therefore no longer predictable or standardized, making them more difficult to computationally manage. As more organizations are deployed, however, computational systems can learn from the resulting structures and ultimately improve their ability

to support workers and guide their efforts.

While reconfigurable workflows support open-ended goals by allowing for the work structures to be adapted, they don't fully solve the goal decomposition problem. In particular, the reconfigurable workflows do not help decompose the goals, leaving it up to the requesters and workers to figure out on their own. For example, if a worker is completing wireframes and needs to adapt the workflow, the adaptation they propose will be based on their prior experiences and knowledge about the goal. Similarly, even though the team leader or requester needs to approve the change, this assumes that they know the best way to decompose and complete the goal. Future research should explore techniques for automatically suggesting the best work structures and reconfigurations to workers and requesters.

## 6.2 Uses of Expert Crowd Work

Given the tradeoffs and implications described, it is important to recognize what types of goals are ideal for expert crowd work. In particular, when does it make sense to use expert crowds instead of traditional teams and organizations? While future work is needed to fully understand the conditions and practices that facilitate fluid coordination in dynamic teams and organizations [169] and interdependent crowds [95], this section starts to reflect on this question.

### 6.2.1 Bounded, Decomposable and Routine Goals

Our experiences with the flash teams and flash organizations, along with prior work [25], suggest that attributes of the task or goal are one of the most important factors to consider when deciding to use expert crowd work. As will be described in this section, the benefits of expert crowd work are most effective for goals that are bounded, decomposable and routine.

First, expert crowd work is best suited for goals that are bounded in scope and duration. By bounded, we are referring to goals that have a specific starting point, such as an input or well defined specification, along with a specific final output or defined end point. For example, the True Story game completed by the flash organization had a defined end point, which was the card game and supporting mobile application. Other goals, such as creating a social network like Facebook, have no defined end point, making them much harder to source to temporary expert crowds. Furthermore, while expert crowds are capable of achieving goals that range in duration, at a certain point, it becomes more effective to create a traditional team or organization composed of long-term contributors (e.g., employees). This is because the amount of knowledge accumulates within the team or organization as work occurs over longer periods of time, making it increasingly more difficult to transfer to new workers or preserve over time.

In addition to bounded goals, expert crowd work is ideal for goals that can be decomposed. Specifically, the flash teams and flash organizations succeeded by decomposing much larger goals

into goals that could be achieved by distributed workers that were computationally managed on Foundry. When goals cannot be decomposed or have high integration costs, they become much harder to distribute to expert crowds and are best achieved by individual workers or traditional teams and organizations. For example, creating a strategy plan or product roadmap, which cannot be easily decomposed into subtasks, might be more difficult for expert crowds to achieve. Goals of this nature, which require a global view by multiple stakeholders and a lot of interaction and iteration, benefit from more traditional organizational forms and methods.

Furthermore, expert crowd work is an effective approach for routine tasks or goals. In particular, certain goals, such as web development and video production, can be completed through structured, routine processes, making them easier to distribute to expert crowds. Even though goals of this nature might require iteration as the work proceeds, the types of tasks that need to be completed to achieve the final deliverable are well-defined and routine. As a result, assuming workers have the skills required to complete their tasks, well-defined goals and routine work processes can be achieved by workers with little context or prior knowledge. For example, the flash teams were able to achieve several different types of mobile web applications using the same routine work structure.

In contrast, when goals, task requirements or work processes are unique and non-routine, they are likely better suited for traditional teams and organizations. These types of goals and work processes benefit from asset specificity and prior knowledge and experiences. For example, organizations like Google or Salesforce have developed many of their own requirements and work processes. These organizations have also amassed massive amounts of internal knowledge and infrastructure, such as design standards, large and complex code bases and specialized tools and systems. While organizations could use expert crowd work for certain goals, such as idea generation or quality assurance, traditional methods for coordination, management and organizational design are best for most of their goals and projects.

Finally, expert crowd work is ideal for goals that benefit from the scale and speed of online labor marketplaces. When goals require niche skills or specialized knowledge, they likely do not benefit from the scale of the crowd. In particular, given that the "scale" of the crowd relies on the availability of qualified workers in the marketplace, skills that are underrepresented among workers in the crowd may be harder or more expensive to recruit. For example, one of the flash organizations tried to recruit a HIPAA expert to ensure the privacy of patient information submitted to the platform. Experience with HIPAA was not a skill readily available on the Upwork marketplace. While we recruited an expert who offered to train himself in the HIPAA laws, this ended up being too difficult and we had to remove the HIPAA requirement from the initial version of the application.

### 6.2.2 Aligning Incentives, Available Resources and Risk Factors

In addition to the attributes of tasks and goals, it is also important to evaluate managerial constraints and requirements when deciding to use expert crowd work. Specifically, in this section we reflect on

three factors, including work incentives, available resources and level of risk.

Our experiences along with prior research suggest that crowds are often, although not always [152, 164], motivated by extrinsic factors, such as money and reputation [19, 95] whereas employee motivations are often both intrinsic and extrinsic factors [38, 63]. Therefore, expert crowd work is an ideal approach when incentives and expectations can be clearly communicated and aligned (e.g., the task and time requirements are mutually agreed upon). In contrast, when incentives are difficult to align or expectations are unclear or evolving, salaried workers and traditional organizational forms are more appropriate.

In our experiences, crowd workers were motivated to do whatever it took to ensure a five star rating. This often times led to less risky or creative ideas as workers didn't want to risk producing a deliverable that did not meet our expectations. Furthermore, we noticed a tradeoff between hourly and fixed price compensation. When we hired workers hourly, which we almost always did, they often gave us push back on the deadlines we set. While some research has started to investigate the tradeoffs between different pricing mechanisms [128, 158, 168], our experiences in the flash teams and flash organizations suggest that hourly incentives encourage workers to work slower and longer, leading to higher costs and longer completion times. In contrast, fixed price work had the opposite problem, such that workers were incentivized to complete their tasks as quickly as possible to get paid and move on.

Another factor to consider when deciding to use expert crowd work has to do with the management overhead and requirements. If a goal can be effectively decomposed, defined and managed through crowdsourcing workflows and computational systems, expert crowd work is ideal. In contrast, when goals require a lot of managerial oversight and guidance, especially over long periods of time, traditional organizational structures and hierarchies are more efficient.

While approaches like flash teams and flash organizations and systems like Foundry reduce the amount of management overhead, expert crowd work still requires a certain amount of oversight and commitment by requesters. Given that expert work requires diverse skills and knowledge, requesters must have a certain level of competence and skills in order to author the work structures and manage the workers. Depending on the task or goal, however, the commitment and involvement of requesters can be reduced. For example, by replicating prior team structures for a goal or algorithmically generating team structures based on inputs and outputs, flash teams allow requesters to author a task with little effort. Flash organizations, moreover, introduced hierarchical structures with team leads from the crowd who handled most of the administrative and managerial tasks. Despite the computational and organizational affordances of these techniques, however, requesters will still need to give feedback, help guide the project, pay workers and resolve problems when they occur. This requires some knowledge and skills related to the goal and can be time consuming for the requester.

Finally, using expert crowd work over traditional organizations requires some consideration of risk factors. In particular, expert crowd work is optimal for goals in which the benefits of multiple

contributions from diverse expert crowd workers are of greater value than consequences of potential risk factors. In contrast, when privacy, intellectual property and other legal issues are a top priority, using traditional organizations instead of distributed expert crowd work is more appropriate.

Online labor marketplaces provide some protection and guidance on legal issues, such as labor laws and taxes. They don't currently, however, provide protection or guidance on intellectual property and privacy. Therefore, reducing these risk factors requires requesters taking preventive measures, such as having workers sign non-disclosure agreements, which can be extremely time consuming and difficult to manage when there is a lot of turnover. As a result, confidential or highly sensitive projects and goals requiring contributions for multiple workers are better achieved through organizations with permanent employees and contractual agreements whereas less sensitive goals, such as prototyping, idea generation and more transparent projects, are better suited for expert crowd work.

# Chapter 7

# Conclusion and Future Work

This dissertation envisions a future of work in which teams and organizations are created on-demand by gathering experts from the crowd on-demand and using computational structures to orchestrate their efforts. By introducing and evaluating two generalizable techniques, we demonstrate how crowd workers can achieve complex and interdependent goals, such as large-scale software applications, interactive games and animated videos. We also created and deployed, Foundry, an advanced authoring and coordination platform for expert crowd work.

## 7.1 Summary of Contributions

This research makes contributions to crowd computing, system design and organizational design.

- *Crowd Computing.* We introduce and evaluate two generalizable techniques that expand the capabilities of paid crowd work and crowd computing. Specifically, flash teams and flash organizations introduce computational techniques that bring together *expert* crowd workers on demand and enable them to achieve goals that have previously been out of reach to crowds. Flash teams contribute modular structures that can be replicated and recombined to achieve interdependent goals. Our mixed-method case comparison sheds light on the limitations of workflow-based approaches and identifies some of the key affordances missing from crowd computing techniques. Finally, flash organizations introduce techniques for overcoming prior limitations of crowd work approaches and enable crowds to achieve large scale, open-ended goals that cannot be decomposed in advance. To do this, flash organizations bring together the structure and coordination techniques of traditional organizations and the scale and computational abilities of crowdsourcing.

- *System Design.* This thesis brings together literature, theories and methods from organizational behavior, human-computer interaction, and computer science to inform the design and

understanding of computational crowdsourcing systems and techniques. Furthermore, our computational platform, Foundry, contributes a new crowdsourcing system capable of authoring and orchestrating a range of complex crowdsourced goals. To our knowledge, Foundry represents the first computational authoring and coordination platform for interdependent expert crowd work. In this thesis, we developed and deployed two versions of Foundry. Taken together, the computational and organizational techniques embedded in both versions of Foundry enable crowds to achieve a range of complex and interdependent goals. More importantly, Foundry overcomes many of the limitations and challenges of existing coordination and crowd work tools, which were designed with other types of users and goals in mind.

- *Organizational Design.* In addition to drawing on organizational design research to inform the team and organizational structures as well as Foundry's design and features, this dissertation also advances research on organizational design. Specifically, flash teams and organizations represent the first examples of computational systems and frameworks that encode and reconfigure role-based coordination structures capable of orchestrating interdependent efforts in temporary teams and organizations. This research also highlights generalizable and scalable approaches for bringing together experts into team and organizational structures on-demand. While the expertise in flash teams and organizations is sourced from crowdsourcing marketplaces, traditional organizations can apply similar techniques with their existing employees. Therefore, by combining computation, organizational design and expert crowds, this dissertation helps envision the future of traditional and online work and advances crowdsourcing and organizational design research.

## 7.2 Impact and Recent Developments

Flash teams introduced the first approach for gathering interdependent teams of expert crowd workers and helped establish expert crowd work as a research field. Since then, there has been growing set of literature on expert crowd work that builds on this work. Recent contributions include expert crowd work systems and frameworks for complex goals, such as software development [27, 49], mentorship and skill development [162], writing [134] and physical world tasks [164]. Database researchers have also explored using expert crowd work for data crowdsourcing [56] as well as data management in organizations [125].

This work has also garnered attention from a range of organizations around the globe who have expressed using the flash team and flash organization approaches as well as the Foundry platform internally to author and manage their workers. Other organizations have also expressed using these techniques to incorporate expert crowd workers into their teams and organizations. We discuss some of these opportunities in the future work section below.

## 7.3 Future Work

This thesis departs from traditional crowdsourcing approaches that combine independent microtasks completed by homogeneous workers. Instead, we introduce teams and organizations composed of diverse experts as a core component of crowdsourcing systems. In doing so, this research opens a set of complex, interdependent and open-ended goals previously out of reach for crowdsourcing.

We envision a world in which work can occur online by experts who have never worked together before. Flash teams and organizations serve as initial examples of this vision. Below we propose future research directions and goals for the nascent field of expert crowd work as well as more established teams and organizations.

### 7.3.1 Flexible Work Structures and Coordination Approaches

Flash teams and organizations introduce new work structures and coordination approaches. In the case of flash teams, these structures were defined a priori, were fairly rigid, and could only be adapted in ways the system could understand (e.g., adding a new member, updating start times when prior tasks were delayed, etc.). While these types of modular work structures provide coordination benefits, they prevent crowds from achieving goals that can't be fully decomposed in advance.

Given that most complex goals cannot be completely pre-specified in advance and that it is impossible to predict all of the contingencies that might occur, future work needs to explore approaches for supporting flexible work structures in crowdsourcing systems. The reconfigurable organizational structures enacted by the flash organizations offer one approach but may not be best suited for every type of goal and environment. Furthermore, coordination and organizational design research has long emphasized the need for both formal and informal coordination [8, 48, 137]. As crowds try to take on more complex goals, supporting formal and informal coordination while maintaining the scale and computational affordances of crowd work will be critical.

To overcome the limitations of the reconfigurable work structures and achieve the formal and informal coordination needs of complex expert crowd work, we propose two alternatives for future research to explore.

First, given that crowdsourcing work structures and systems already support formal coordination, future work should explore ways for these existing systems to support informal coordination and interactions. For example, is there a way to create and embed a virtual water cooler directly into crowdsourcing systems? Systems like Slack and Hipchat, which provide multiple channels, effectively support these water cooler-like interactions. Specifically, by allowing teams and organizations to create formal goal-related channels as well as more informal, creative and social channels, these systems provide workers with multiple "virtual" spaces in which they can formally and informally interact with other workers. In addition to channels, many existing coordination platforms support integrations with other tools, such as video conferencing tools, task managers and virtual white boards.

These integrations make it easy for workers to satisfy their different coordination needs and informal interactions without losing the formal coordination affordances provided by the primary platform. Future research should explore incorporating similar tools and features into existing crowdsourcing systems to see if they supplement the formal coordination with the informal coordination capabilities that are currently lacking.

Second, to overcome the limitations of reconfigurable workflows, we propose exploring automated and intelligent approaches to orienteering crowd work structures. For example, is there a way to create a "Waze" for crowdsourcing work structures? Systems like Waze, use crowdsourced data and machine learning to propose alternative directions and navigation patterns based on real-time information about the surrounding and future environment. Rather than just telling drivers that there is an accident ahead, these intelligent systems also provide them with the best alternative routes to avoid the accident and still reach their destination as quickly as possible. Crowdsourcing workflows and computational systems should explore similar automated solutions to generating and adapting crowdsourcing workflows. For example, if tasks are delayed or deliverables require iteration, the system could suggest alternative workflows to the requester or team lead based on the constraints of the project, the task outcomes and the contingencies encountered. Similar to how systems like Waze allow drivers to contribute contingencies or other information about their surroundings (e.g., speed traps, accidents, etc.), these intelligent crowdsourcing systems could allow workers to contribute information about their tasks and other project-related information they encounter.

### 7.3.2 Expert Crowd Work Tools, Platforms and Marketplaces

While online labor marketplaces, such as Mechanical Turk, CrowdFlower, Upwork and Freelancer, provide access to millions of workers, these marketplaces are limited in many ways. For example, Mechanical Turk and CrowdFlower assume workers are homogeneous and provide very basic filtering capabilities. Therefore, even though crowd workers on these platforms may have additional skills to contribute [58, 95], there is no way for them to showcase these expertise and experiences on microtask platforms. In contrast, platforms like Upwork and Freelancer, aim to match project requesters with workers with specific expertise. These platforms, however, were not designed around interdependent expert crowd work and have critical limitations, such as hiring transaction costs and search frictions [9, 142]. While recent platforms, such as Huddler [155] have started to explore these challenges, there continues to be a big opportunity for researchers to design new types of platforms centered around interdependent and on-demand expert crowd work.

Online labor marketplaces also overlook the coordination needs of requesters and workers. In particular, most online marketplaces focus their attention and resources on the matching process rather than the tools needed to support coordination once workers are hired. For example, Upwork's tools are limited to the work diary and the team room, neither of which support the coordination and collaboration needs of interdependent experts. As a result, most crowd workers and requesters

attempt to use other coordination and project management tools. The challenge with existing tools, however, is that they do not capitalize on the unique affordances and scale of distributed expert crowd work. For example, tools like Asana, Basecamp, Skype and Slack require individual accounts, making it burdensome to frequently add and remove crowd workers. These tools also don't integrate with crowdsourcing marketplaces and are not designed to support short term, yet complex, work. Although platforms like Foundry begin to fill this gap, additional research and coordination tools for interdependent expert crowd work are needed.

To overcome the challenges and limitations of existing coordination tools, future research should evaluate which tools are most common among crowd workers and where there are gaps and unfilled needs unique to the work practices of crowd workers. Furthermore, researchers should conduct in depth field studies of crowd workers in their work environments to understand how they appropriate and adapt existing tools as they do their jobs. This understanding would then enable researchers and system designers to create crowd-centered coordination and project management tools designed around the unique challenges and practices of modern day expert crowd workers.

### 7.3.3 Internal Crowds and Hybrid Organizations

Finally, flash teams and organizations suggest important research directions for traditional teams and organizations. They also introduce the opportunity for new types of teams and organizational forms, which future research should explore.

Traditional organizations are composed of individuals, typically referred to as employees, with a range of expertise. Some organizations, such as startups, have only a handful of employees whereas other organizations, such as global corporations like IBM, Accenture and General Electric, have hundreds of thousands of employees. Many organizations, especially those that are large or geographically distributed, could view their employee pool as a crowd and take advantage of computational organizational structures and expert crowd work approaches to convene and coordinate internal teams of employees on-demand. Future research should explore the effectiveness, benefits and limitations of internal flash teams and organizations.

In addition to exploring these approaches in traditional teams and organizations, future research should explore the possibility of creating hybrid teams and organizations. Specifically, there is an opportunity to create new types of teams and organizations composed of both internal employees and crowd workers. For example, a startup might supplement their founding team with additional expertise as the needs of the organizations evolve. Larger organizations might supplement project teams with crowd workers that can provide expertise that are not currently available internally. There are many potential use cases and implications of hybrid crowds. Researchers have the opportunity to define and evaluate these new organizational forms as well as the goals and environments for which they are best suited.

## 7.4 Looking Ahead

Taken together, this thesis re-imagines a future of work in which teams and organizations are no longer anchored in traditional labor models but are instead fluidly assembled and re-assembled from globally networked labor markets. The result of this vision is a system that empowers anyone with an internet connection to create entire teams and organizations composed of workers from paid crowdsourcing marketplaces and then lead those workers in pursuit of complex, open-ended goals.

# Appendix A

# Flash Organizations

The main paper presented in chapter 5 introduced *flash organizations:* rapidly assembled and reconfigurable organizations composed of online crowd workers. In this appendix we report details on the: 1) results from our deployment of the three flash organizations; 2) on-demand hiring; 3) reconfigurable organizational structures; 4) final deliverables produced by the three flash organizations.

## A.1 Tables and Figures

| | EMS Report | True Story | Enterprise | All Projects |
|---|---|---|---|---|
| Number of automated hires | 25 | 29 | 21 | 75 |
| Median hiring time (automated hires)* | 0:13:40 | 0:12:40 | 0:15:13 | 0:13:40 |
| Hiring time IQR (automated hires)* | Q1: 0:06:46<br>Q3: 2:37:07 | Q1: 0:05:42<br>Q3: 1:19:10 | Q1: 0:07:06<br>Q3: 0:39:15 | Q1: 0:06:46<br>Q3: 0:59:13 |

**Table A.1: Hiring time for the automated hires in the three flash organization deployments. *Hiring time was defined as the time lapsed between when the automated hiring email was sent and first engagement occurred, defined as the earliest of logging into the Foundry timeline view or joining the organization's Slack collaboration tool.**

| | EMS Report | True Story | Enterprise | All Projects |
|---|---|---|---|---|
| Number of warm hires | 6 | 8 | 6 | 20 |
| Median hiring time (warm hires)* | 25:14:19 | 19:28:41 | 5:55:51 | 14:48:45 |
| Hiring time IQR (warm hires)* | Q1: 5:58:25<br>Q3: 44:45:40 | Q1: 12:39:36<br>Q3: 31:36:13 | Q1: 0:51:14<br>Q3: 21:32:38 | Q1: 2:29:26<br>Q3: 35:58:26 |

**Table A.2: Hiring time for the warm hires in the three flash organization deployments. *Hiring time was defined as the time lapsed between when the hiring request was made and first engagement occurred, defined as the earliest of logging into the Foundry timeline view or joining the organization's Slack collaboration tool.**

| | EMS Report | True Story | Enterprise | All Projects |
|---|---|---|---|---|
| Number of Pull Requests | 335 | 113 | 118 | 566 |
| Mean Changes Per Day* | 7.3 | 2.8 | 3.4 | 4.6 |
| Percent Breakdown by Pull Request Author | | | | |
| % Workers | 7.2 | 21.2 | 66.9 | 22.4 |
| % Team Leads | 92.8 | 47.8 | 17.8 | 68.2 |
| % Organizational Leader | 0 | 31.0 | 15.3 | 9.4 |

**Table A.3: The number and source of the organizational structure pull requests for the three flash organizations. *Calculated by dividing the total number of requests by the number of active days for the organization.**

| **Organizational Unit** | **# of Tasks** | **# of Workers** | **# of Team Leads** | **# of Total Members** |
|---|---|---|---|---|
| Android Development | 120 | 1 | 2 | 3 |
| Application Name | 2 | 1 | 0 | 1 |
| Back End Development | 36 | 2 | 3 | 5 |
| Front End Development | 36 | 1 | 3 | 4 |
| HIPAA | 1 | 1 | 0 | 1 |
| Market Research | 3 | 2 | 0 | 2 |
| Marketing Website | 13 | 2 | 2 | 4 |
| Pitch Deck and User Documentation | 7 | 1 | 1 | 2 |
| Quality Assurance | 82 | 4 | 2 | 6 |
| User Interface Design | 82 | 5 | 1 | 6 |
| User Testing | 4 | 1 | 1 | 2 |
| Video | 4 | 1 | 0 | 1 |
| **Total: 12** | **Total: 390** | **Total: 20*** | **Total: 11*** | **Total: 31*** |

**Table A.4: An overview of the number of tasks, workers, leads and total team members for each of the nine teams and three individuals in the EMS Report organization. *These numbers represent the total number of unique workers, leads and team members, respectively. Some people filled these positions for more than one team.**

| **Organizational Unit** | **# of Tasks** | **# of Workers** | **# of Team Leads** | **# of Total Members** |
|---|---|---|---|---|
| Android Development | 33 | 1 | 1 | 2 |
| Content Creation | 24 | 12 | 1 | 13 |
| Graphic Design: Card Back | 6 | 4 | 0 | 4 |
| Graphic Design: Card Front | 2 | 2 | 0 | 2 |
| Graphic Design: Logo | 7 | 5 | 0 | 5 |
| Graphic Design: Packaging | 2 | 1 | 0 | 1 |
| High Fidelity Mockups | 12 | 2 | 1 | 3 |
| User Testing | 7 | 1 | 1 | 2 |
| User Testing: Photo | 3 | 1 | 0 | 1 |
| User Testing: Video | 3 | 2 | 0 | 2 |
| Video Transcription | 4 | 3 | 0 | 3 |
| Website Development | 19 | 2 | 1 | 3 |
| **Total: 12** | **Total: 122** | **Total: 34*** | **Total: 3*** | **Total: 37*** |

**Table A.5: An overview of the number of tasks, workers, leads and total team members for each of the ten teams and two individuals in the True Story organization. *These numbers represent the total number of unique workers, leads and team members, respectively. Some people filled these positions for more than one team.**

| **Organizational Unit** | **# of Tasks** | **# of Workers** | **# of Team Leads** | **# of Total Members** |
|---|---|---|---|---|
| Back End Development | 23 | 5 | 3 | 8 |
| Front End Development | 29 | 9 | 2 | 11 |
| Quality Assurance | 20 | 3 | 1 | 4 |
| User Interface Design | 44 | 7 | 1 | 8 |
| User Testing | 11 | 2 | 1 | 3 |
| **Total: 5** | **Total: 127** | **Total: 22*** | **Total: 8*** | **Total: 27*** |

**Table A.6: An overview of the number of tasks, workers, leads and total team members for each of the five teams in the Enterprise Workshop Portal organization. *These numbers represent the total number of unique workers, leads and team members, respectively. Some people filled these positions for more than one team.**

| **Panel Name** | **Open-call applicants** | **Completed starter task** | **Selected for panel** | **Auto-hired from panel** | **Warm-hired from panel** |
|---|---|---|---|---|---|
| Android Development | 19 | 15 | 14 | 4 | 1 |
| Back End Development | 41 | 27 | 22 | 11 | 2 |
| Content Creation | 56 | 46 | 16 | 13 | 2 |
| Front End Development | 55 | 33 | 23 | 12 | 2 |
| Photography | 6 | 3 | 3 | 1 | 0 |
| Graphic Design | 36 | 28 | 14 | 3 | 4 |
| High Fidelity Mockups | 9 | 8 | 9 | 4 | 0 |
| HIPAA | 21 | 4 | 3 | 0 | 1 |
| Market Research | 8 | 4 | 2 | 2 | 0 |
| PowerPoint | 18 | 8 | 1 | 1 | 0 |
| Quality Assurance | 71 | 53 | 35 | 10 | 1 |
| User Experience Research | 12 | 9 | 2 | 0 | 2 |
| User Interface Design | 37 | 25 | 22 | 8 | 3 |
| User Testing | 7 | 6 | 6 | 0 | 2 |
| Video Editing | 18 | 13 | 3 | 2 | 0 |
| Video Transcription | 16 | 9 | 6 | 3 | 0 |
| Video Animation | 6 | 5 | 1 | 1 | 0 |

**Table A.7: The full set of panels utilized by the three flash organizations, as well as the number of workers in each stage of the panel creation pipeline.**

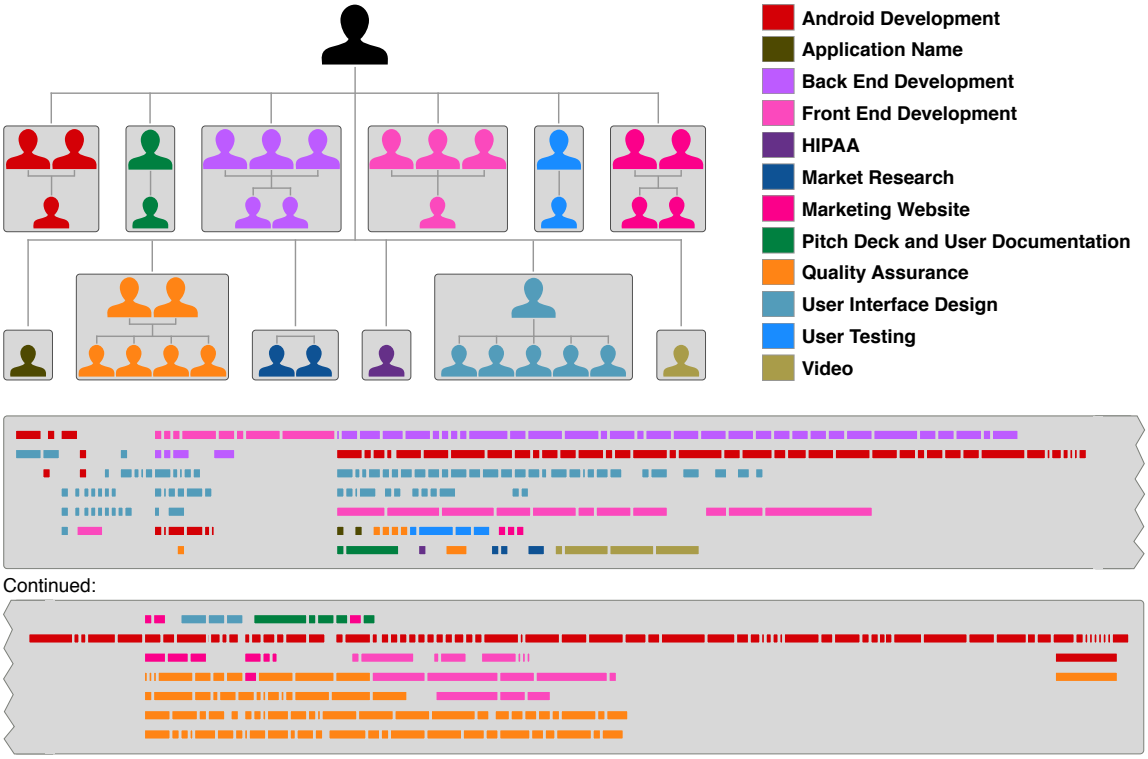



**Figure A.1: Organizational structure (top) and workflow (bottom) for the EMS Report organization. The organization was composed of the CEO, nine teams, and three individual workers. The workflow shows the timing of the 390 completed tasks. Colors in the organizational structure and workflow indicate the respective organizational units, listed top right.**

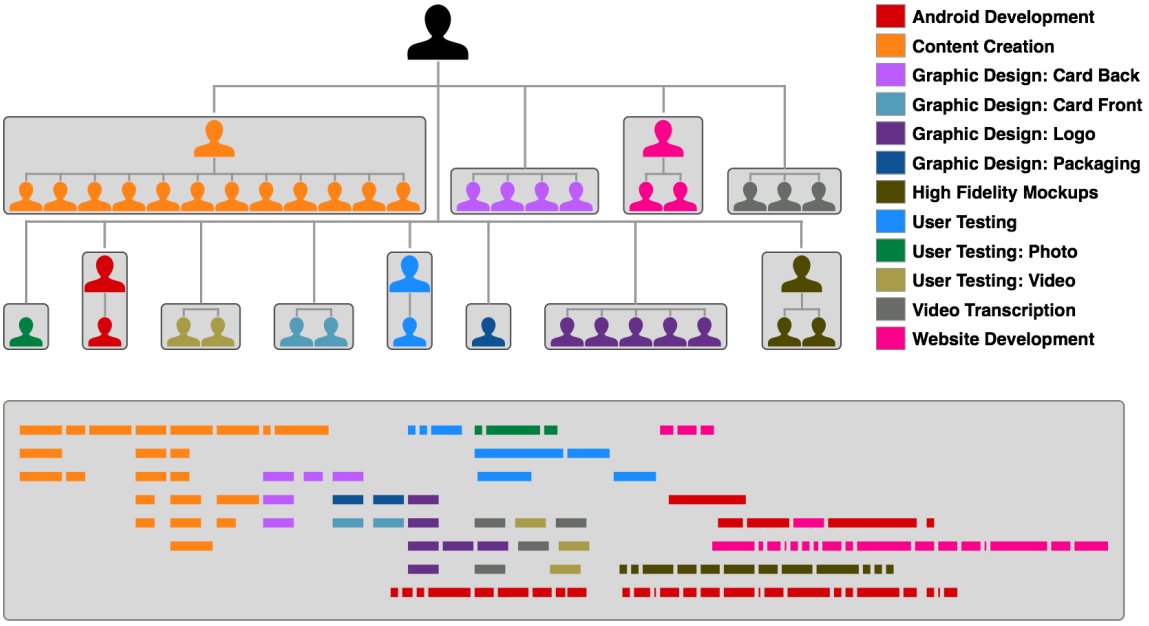



**Figure A.2: Organizational structure (top) and workflow (bottom) for the True Story organization. The organization was composed of the CEO, 10 teams, and two individual workers. The workflow shows the timing of the 122 completed tasks. Colors in the organizational structure and the workflow indicate the respective organizational units, listed top right.**

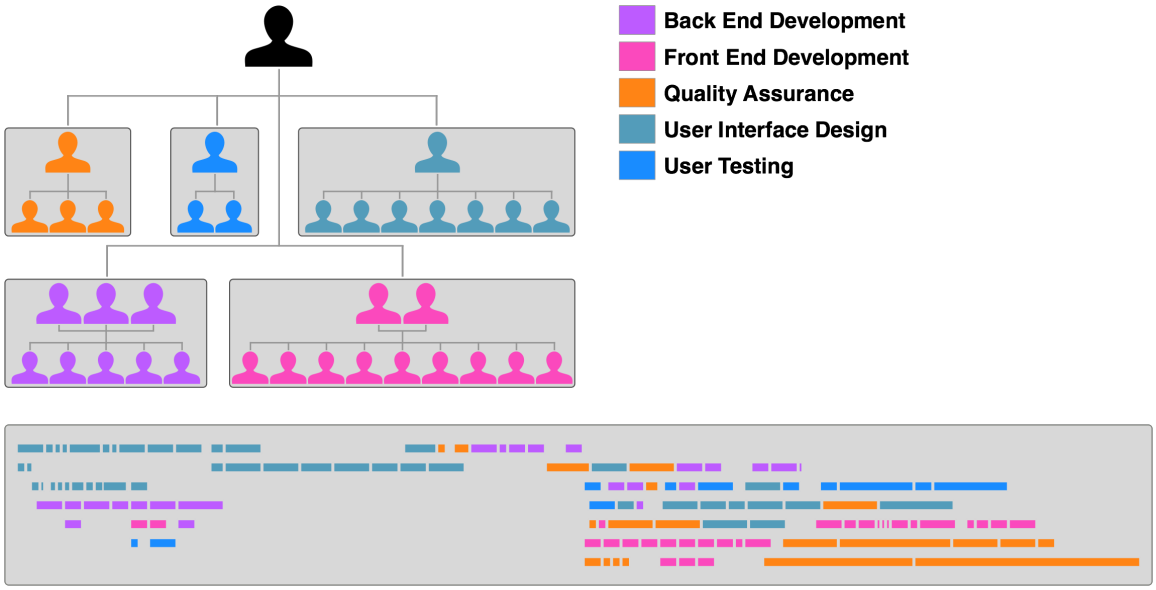



**Figure A.3: Organizational structure (top) and workflow (bottom) for the Enterprise Workshop Portal organization. The organization was composed of the CEO and five teams. The workflow shows the timing of the 127 completed tasks. Colors in the organizational structure and workflow indicate the respective organizational units, listed top right.**


**Figure A.4: Screenshots from the EMS Trauma Report Android Application. After opening the application, users can create an account or login with an existing account. After logging in, the application allows users to either create a new trauma case or update a current case. When adding a new Trauma case, the user will be presented with a different set of screens depending on whether it is a Trauma 1 or 2 case. The application automatically identifies the location of the case and continues to track the location of the device until the patient reaches the hospital.**

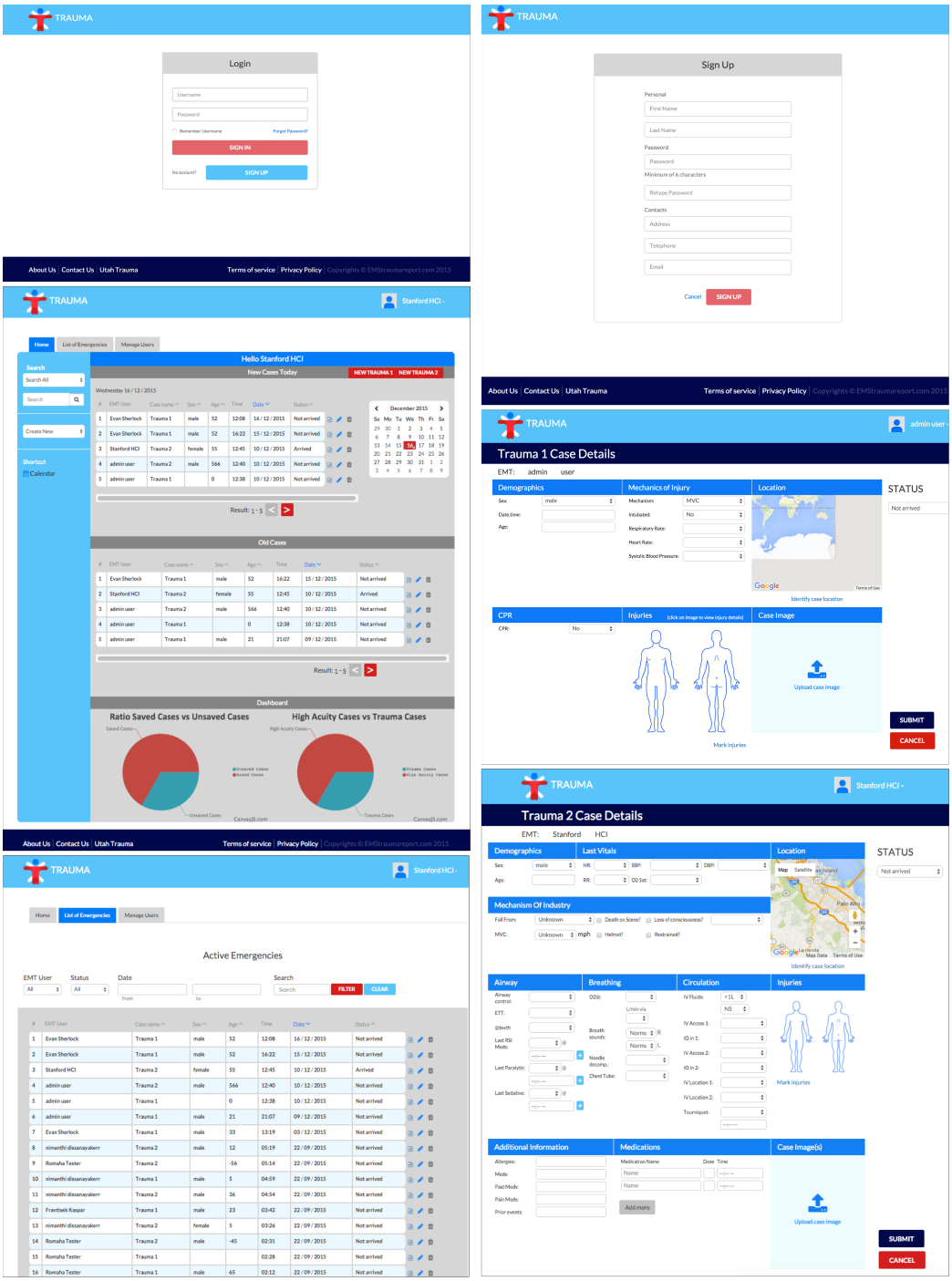



**Figure A.5: Screenshots from the EMS Trauma Report website, which is monitored by staff at the hospital. After logging in, users are shown a dashboard with new and old cases as well as relevant statistics. Similar to the mobile application, users can create new Trauma 1 and 2 cases and update existing cases. Users with administrative privileges have the ability to manage and update all user accounts.**

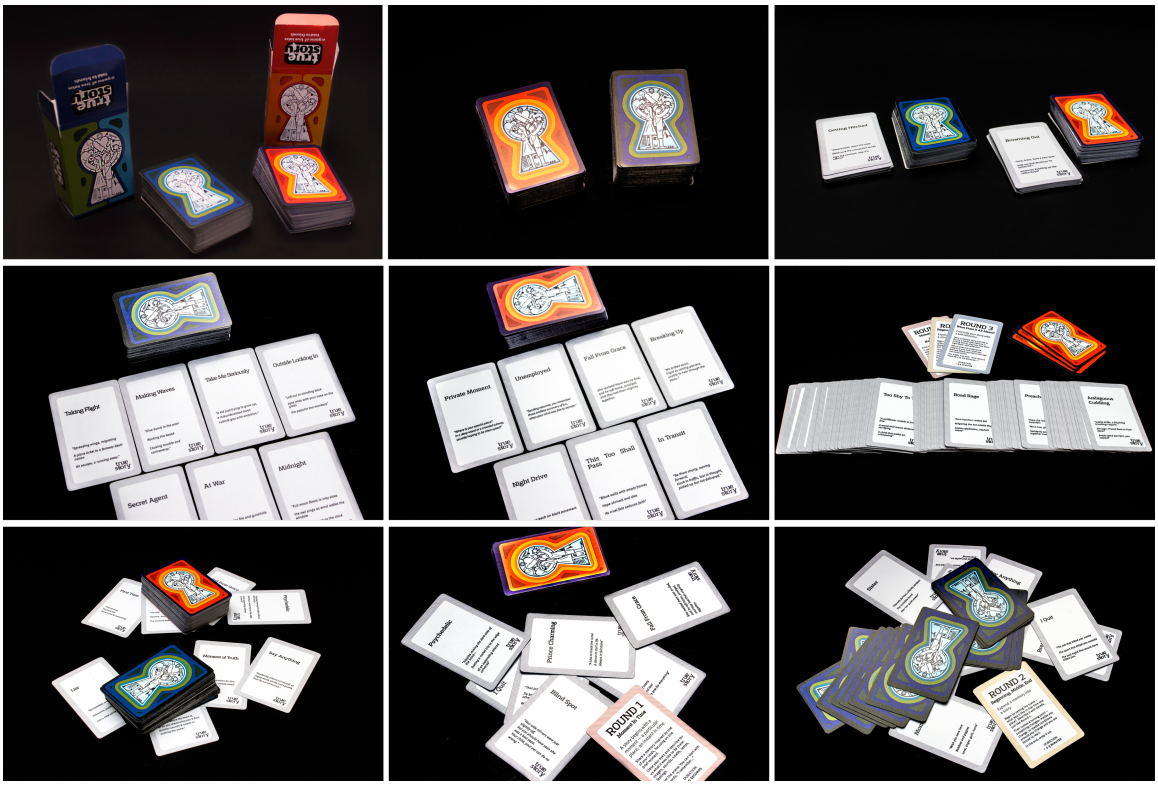
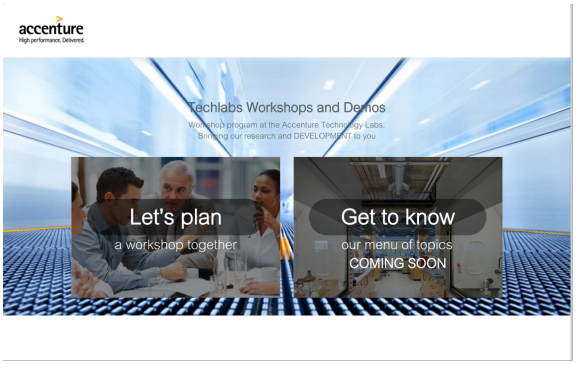
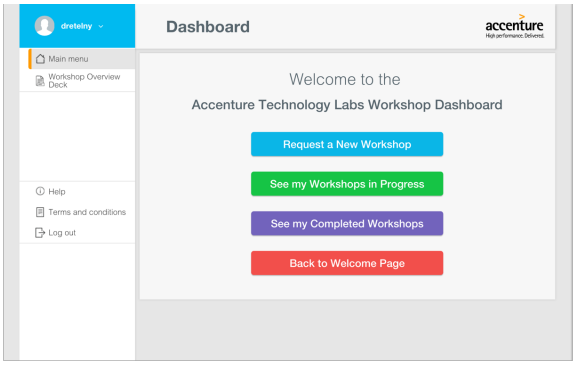
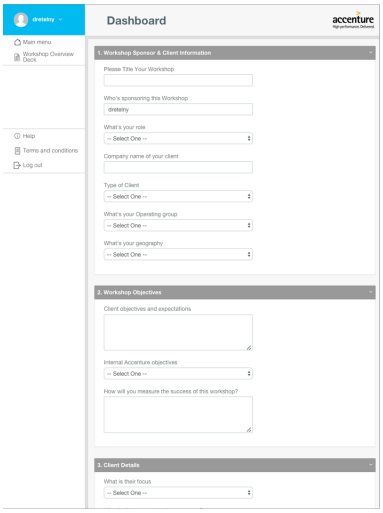


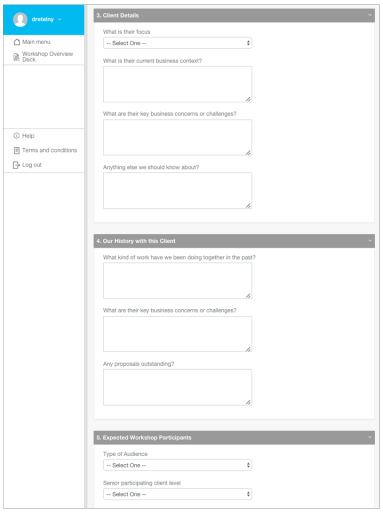

**Figure A.6: Photographs of the True Story game. The organization created two separate card decks, which included a family friendly version (the blue deck) and an adult only version (the red deck). Each card deck consisted of 80 story prompt cards and an instruction card for each of the three rounds in the game (e.g., 3 instruction cards total). Each story prompt card featured a topic and a descriptive quote aimed at helping game players recall a relevant story related to the prompt. The back of all cards in the deck had the same graphic, which was either blue or red depending on if it was the family friendly or adult only deck.**

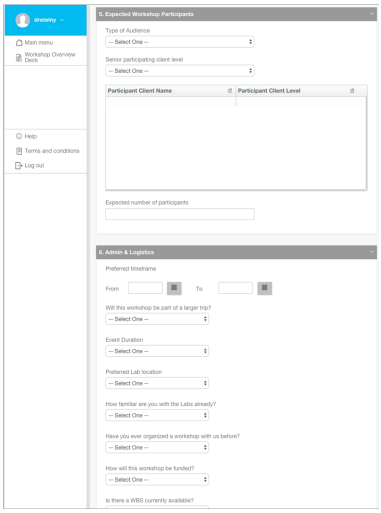

**Figure A.7: Screenshots of the True Story mobile application and website. When opening the application for the first time, the user is presented with a brief four-screen tutorial. They can always view the tutorial again by clicking on the question mark icon on the main screen. The application allows users to record the audio of their stories and upload it to the True Story web server. Once users upload their story, they receive an email with the link to access their story on the True Story website. From there, users have the option to download the story, save the story to their Google Drive account or share the link to their story with others. Users can also access their previous recordings from the last 30 days.**

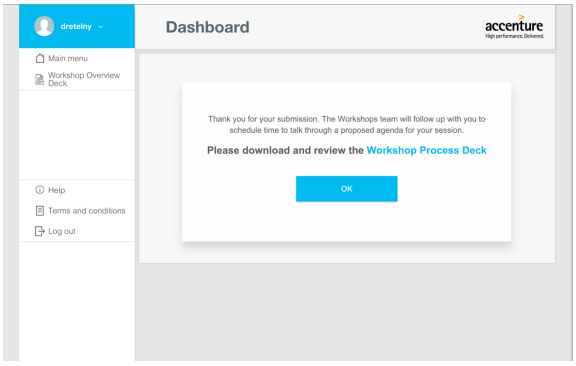

accenture
High performance. Delivered.
Techlabs Workshops and Demos
Let's plan
a workshop together
Get to know
our menu of topics
COMING SOON

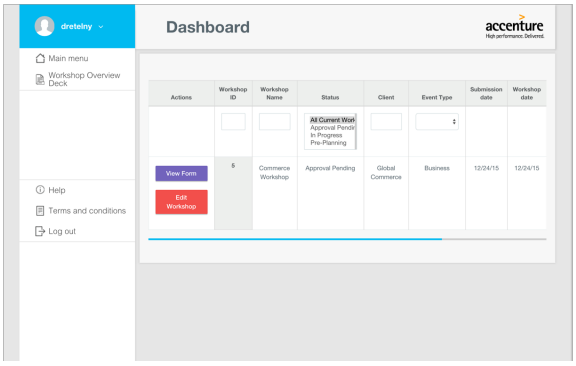

dretelny
Dashboard
accenture
High performance. Delivered.
Main menu
Workshop Overview Deck
Help
Terms and conditions
Log out
Welcome to the
Accenture Technology Labs Workshop Dashboard
Request a New Workshop
See my Workshops in Progress
See my Completed Workshops
Back to Welcome Page


dretelny
Dashboard
accenture
Main menu
Workshop Overview Deck
1. Workshop Sponsor & Client Information
Please Title Your Workshop
Who's sponsoring this Workshop
What's your role
Company name of your client
Type of Client
What's your Operating group
What's your geography
2. Workshop Objectives
Client objectives and expectations
Internal Accenture objectives
How will you measure the success of this workshop?
3. Client Details
What is their focus
Help
Terms and conditions
Log out


dretelny
3. Client Details
What is their focus
What is their current business context?
What are their key business concerns or challenges?
Anything else we should know about?
4. Our History with this Client
What kind of work have we been doing together in the past?
What are their key business concerns or challenges?
Any proposals outstanding?
5. Expected Workshop Participants
Type of Audience
Senior participating client level
Main menu
Workshop Overview Deck
Help
Terms and conditions
Log out


dretelny
5. Expected Workshop Participants
Type of Audience
Senior participating client level
Participant Client Name
Participant Client Level
Expected number of participants
6. Admin & Logistics
Preferred timeframe
From
To
Will this workshop be part of a larger trip?
Event Duration
Preferred Lab location
How familiar are you with the Labs already?
Have you ever organized a workshop with us before?
How will this workshop be funded?
Is there a WBS currently available?
Main menu
Workshop Overview Deck
Help
Terms and conditions
Log out


dretelny
Dashboard
accenture
High performance. Delivered.
Main menu
Workshop Overview Deck
Thank you for your submission. The Workshops team will follow up with you to schedule time to talk through a proposed agenda for your session.
Please download and review the Workshop Process Deck
OK
Help
Terms and conditions
Log out


dretelny
Dashboard
accenture
High performance. Delivered.
Main menu
Workshop Overview Deck
Actions
Workshop ID
Workshop Name
Status
Client
Event Type
Submission date
Workshop date
View Form
Edit Workshop
5
Commerce Workshop
Approval Pending
Global Commerce
Business
12/24/15
12/24/15
Help
Terms and conditions
Log out

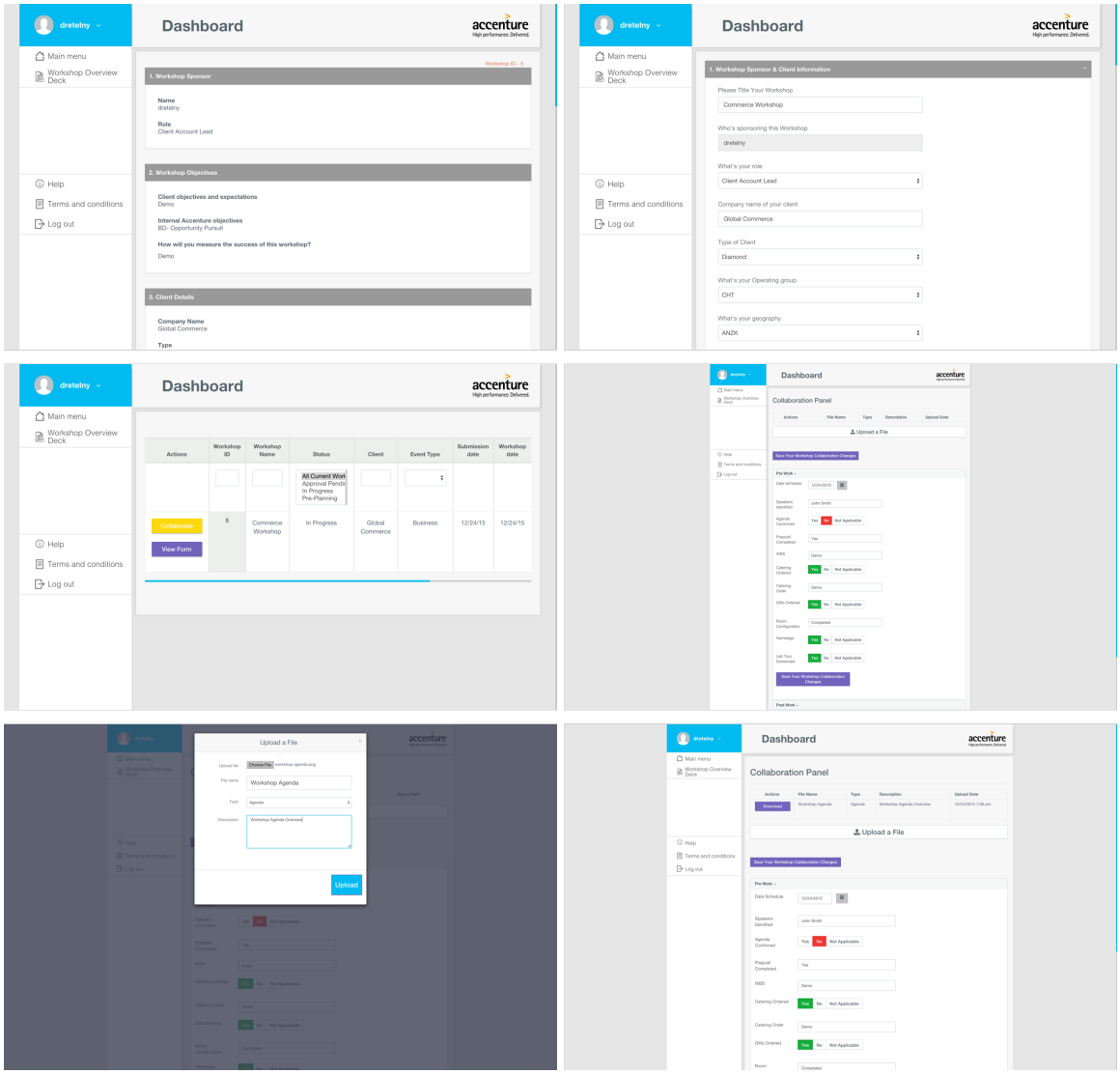



**Figure A.8: Screenshots of the enterprise workshop portal worker view. After logging in, users are taken to the portal welcome page. From there, they can access the workshop dashboard screen. On the dashboard page, users can request a new workshop and see their in progress and completed workshops. When requesting a new workshop, users are presented with a form with six different sections. In order to submit the form, they must answer all of the required questions across all sections. Once the user has submitted the workshop request form, the status of the request is "pending" until it has been approved and assigned by an administrator. During this time, users can view and edit their request forms. Once the request has been approved, the status updates to "in progress" and the user is given access to the "collaboration panel" for the workshop, which allows them to update the status of specific tasks, such as whether the catering has been ordered, as well as upload relevant files, such as an agenda or menu.**

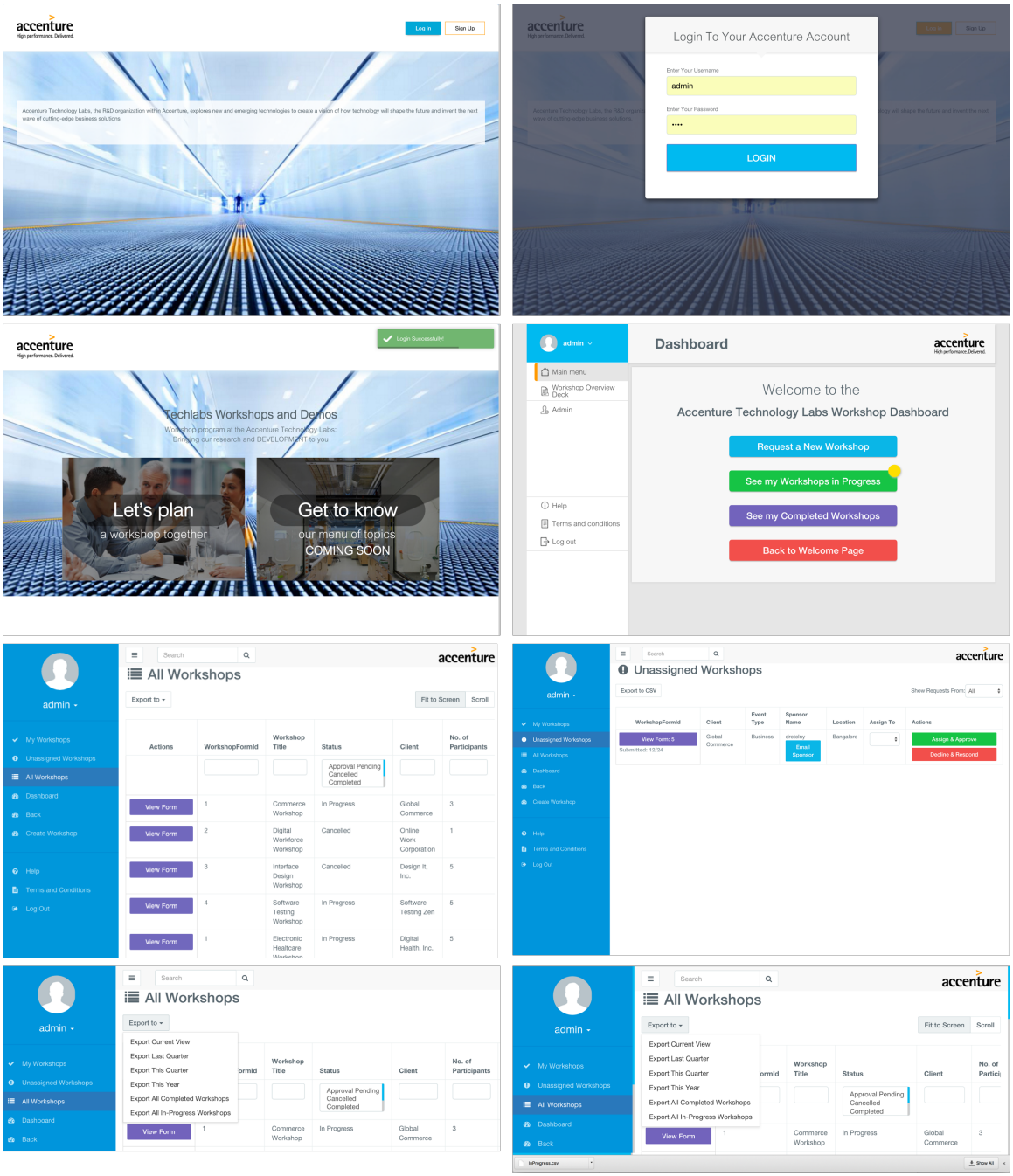



**Figure A.9: Screenshots of the enterprise workshop portal administrative view. After logging in, administrative users are taken to the portal welcome page. From there, they can access the workshop dashboard screen. On the dashboard page, administrative users can request a new workshop, see their in progress and completed workshops and access the content management system. The CMS allows administrative users to view and assign unassigned workshops as well as access, edit, filter and export any of the other workshops in the system.**

# Bibliography


[1] 99Designs. https://99designs.com/.

[2] Paul S. Adler and Bryan Borys. Two Types of Bureaucracy: Coercive and Enabling. *Administrative Science Quarterly*, 41(1):61–89, 1996.

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

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

[5] Paul André, Haoqi Zhang, Juho Kim, Lydia Chilton, Steven P. Dow, and Robert C. Miller. Community Clustering: Leveraging an Academic Crowd to Form Coherent Conference Sessions. In *First AAAI Conference on Human Computation and Crowdsourcing*, HCOMP '13, pages 9–16. AAAI Press, 2013.

[6] Judd Antin and Aaron Shaw. Social Desirability Bias and Self-Reports of Motivation: A Study of Amazon Mechanical Turk in the US and India. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2925–2934, New York, NY, USA, 2012. ACM.

[7] Ricardo Matsumura Araujo. 99designs: An Analysis of Creative Competition in Crowdsourced Design. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing*, HCOMP '13, pages 17–24. AAAI Press, 2013.

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

[9] Nick Arnosti, Ramesh Johari, and Yash Kanoria. Managing Congestion in Dynamic Matching Markets. 2014.

[10] Carliss Y. Baldwin and Kim B. Clark. Managing in an Age of Modularity. *Harvard Business Review*, 75(5):84–93, 1997.

[11] Stephen R. Barley and Gideon Kunda. Bringing work back in. *Organization Science*, 12(1):76–95, 2001.

[12] Beth A. Bechky. Gaffers, Gofers, and Grips: Role-Based Coordination in Temporary Organizations. *Organization Science*, 17(1):3–21, 2006.

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

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

[15] Yochai Benkler, Aaron Shaw, and Benjamin Mako Hill. Peer Production: A Modality of Collective Intelligence. In Thomas W. Malone and Michael S. Bernstein, editors, *Handbook of Collective Intelligence*, pages 175–204. MIT Press, 2015.

[16] Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. 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, pages 33–42, New York, NY, 2011. ACM.

[17] Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: A Word Processor with a Crowd Inside. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 313–322, New York, NY, USA, 2010. ACM.

[18] Katerina Bezrukova, Karen A. Jehn, Elaine L. Zanutto, and Sherry M. B. Thatcher. Do Workgroup Faultlines Help or Hurt? A Moderated Model of Faultlines, Team Identification, and Group Performance. *Organization Science*, 20(1):35–50, 2009.

[19] Jeffrey P. Bigham, Michael S. Bernstein, and Eytan Adar. Human-Computer Interaction and Collective Intelligence. In *Handbook of Collective Intelligence*, pages 57–84. MIT Press, 2015.

[20] 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. VizWiz: Nearly Real-Time Answers to Visual Questions. In *Proceedings of the 23nd Annual ACM Symposum on User Interface Software and Technology*, UIST '10, pages 333–342, New York, NY, USA, 2010. ACM.

[21] Gregory A. Bigley and Karlene H. Roberts. The Incident Command System: High-Reliability Organizing for Complex and Volatile Task Environments. *Academy of Management Journal*, 44(6):1281–1299, 2001.

[22] Shona L. Brown and Kathleen M. Eisenhardt. The Art of Continuous Change: Linking Complexity Theory and Time-Paced Evolution in Relentlessly Shifting Organizations. *Administrative Science Quarterly*, 42(1):1–34, 1997.

[23] J. Stuart Bunderson and Peter Boumgarden. Structure and Learning in Self-Managed Teams: Why "Bureaucratic" Teams Can Be Better Learners. *Organization Science*, 21(3):609–624, jun 2010.

[24] John M. Carroll, Dennis C. Neale, Philip L. Isenhour, Mary Beth Rosson, and D. Scott McCrickard. Notification and awareness: Synchronizing task-oriented collaborative activity. *International Journal of Human Computer Studies*, 58(5):605–632, 2003.

[25] L. Elisa Celis, Koustuv Dasgupta, and Vaibhav Rajan. Adaptive Crowdsourcing for Temporal Crowds Categories and Subject Descriptors. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 1093–1100, New York, NY, USA, 2013. ACM.

[26] Joel Chan, Steven Dang, and Steven P. Dow. Improving Crowd Innovation with Expert Facilitation. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, CSCW '16, pages 1221–1233, New York, NY, USA, 2016. ACM.

[27] Yan Chen, Steve Oney, and Walter S. Lasecki. Towards Providing On-Demand Expert Support for Software Developers. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 3192–3203, New York, NY, USA, 2016. ACM.

[28] Lydia B. Chilton, Juho Kim, Paul André, Felicia Cordeiro, James A. Landay, Daniel S. Weld, Steven P. Dow, Robert C. Miller, and Haoqi Zhang. Frenzy: Collaborative Data Organization for Creating Conference Sessions Lydia. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 1255–1264, New York, NY, USA, 2014. ACM.

[29] Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. Cascade: Crowdsourcing Taxonomy Creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1999–2008, New York, NY, USA, 2013.

[30] Susan G. Cohen. What Makes Teams Work: Group Effectiveness Research from the Shop Floor to the Executive Suite. *Journal of Management*, 23:239–290, 1997.

[31] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, and Foldit-Players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.

[32] Catherine Durnell Cramton. The Mutual Knowledge Problem and Its Consequences for Dispersed Collaboration. *Organization Science*, 12(3):346–371, jun 2001.

[33] Justin Cranshaw and Aniket Kittur. The Polymath Project: Lessons from a Successful Online Collaboration in Mathematics Justin. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1865–1874, New York, NY, USA, 2011. ACM.

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

[35] Richard L. Daft and Robert H. Lenger. Organizational Information Requirements, Media Richness and Structural Design. *Management Science*, 32(5):554–571, 1986.

[36] Peng Dai, Mausam, and Daniel S. Weld. Decision-Theoretic Control of Crowd-Sourced Workflows. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI '10, pages 1168–1174, 2010.

[37] Steven J. Davis, R. Jason Faberman, and John C. Haltiwanger. The Establishment-Level Behavior of Vacancies and Hiring. *The Quarterly Journal of Economics*, 128(2):581–622, may 2013.

[38] Edward L. Deci, Richard Koestner, and Richard M. Ryan. A Meta-Analytic Review of Experiments Examining the Effects of Extrinsic Rewards on Intrinsic Motivation. *Psychological Bulletin*, 125(6):627–668, 1999.

[39] Pete Deemer, Gabrielle Benefield, Craig Larman, and Bas Vodde. The Scrum Primer: A Lightweight Guide to the Theory and Practice of Scrum, 2012.

[40] Jia Deng Jia Deng, Wei Dong Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2–9, 2009.

[41] Paul Dourish and Victoria Bellotti. Awareness and Coordination in Shared Workspaces. In *Proceedings of the 1992 ACM Conference on Computer Supported Cooperative Work*, CSCW '92, pages 107–114, New York, NY, USA, 1992. ACM.

[42] Steven P. Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L. Schwartz, and Scott R. Klemmer. Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. *ACM Transactions on Computer-Human Interaction*, 17(4):1–24, 2010.

[43] Steven P. Dow, Anand Kulkarni, Scott R. Klemmer, and Björn Hartmann. Shepherding the Crowd Yields Better Work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1013–1022, New York, NY, USA, 2012. ACM.

[44] Edelman Intelligence (Commissioned by Upwork and Freelancers Union). Freelancing in America: 2016, 2016.

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

[46] Amy C. Edmondson, Harvard Business School, and Stacy E. Mcmanus. Methodological Fit in Management Field Research. *Academy of Management Review*, 32(4):1155–1179, 2007.

[47] Kathleen M. Eisenhardt. Building Theories from Case Study Research. *Academy of Management Review*, 14(4):532–550, 1989.

[48] Samer Faraj and Yan Xiao. Coordination in Fast-Response Organizations. *Management Science*, 52(8):1155–1169, 2006.

[49] Ethan Fast and Michael S. Bernstein. 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, New York, NY, USA, 2016. ACM.

[50] Will Felps, Terence R. Mitchell, and Eliza Byington. How, When, and Why Bad Apples Spoil the Barrel: Negative Group Members and Dysfunctional Groups. *Research in Organizational Behavior*, 27(06):175–222, jan 2006.

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

[52] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1972.

[53] Andrey Fradkin. Search Frictions and the Design of Online Marketplaces. 2014.

[54] Freelancer. https://www.freelancer.com/.

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

[56] Hector Garcia-Molina, Manas Joglekar, Adam Marcus, Aditya Parameswaran, and Vasilis Verroios. Challenges in Data Crowdsourcing. *IEEE Transactions on Knowledge and Data Engineering*, 28(4):901–911, 2016.

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

[58] Mary L. Gray, Siddharth Suri, Syed Shoaib Ali, and Deepti Kulkarni. The Crowd is a Collaborative Network. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, CSCW '16, pages 134–147, New York, NY, USA, 2016. ACM.

[59] Jonathan Grudin. Groupware and Social Dynamics: Eight Challenges for Developers. *Communications of the ACM*, 37(1):92–105, 1994.

[60] Carl Gutwin and Saul Greenberg. A descriptive framework of workspace awareness for real-time groupware. *Computer Supported Cooperative Work*, 11(3-4):411–446, 2002.

[61] Daniel Haas, Jason Ansel, Lydia Gu, and Adam Marcus. Argonaut: Macrotask Crowdsourcing for Complex Data Processing. In *Proceedings of the VLDB Endowment*, volume 8, pages 1642–1653, 2015.

[62] J. Richard Hackman. *The Design of Work Teams*. Prentice Hall, Englewood Cliffs, NJ, 1987.

[63] J. Richard Hackman and Greg R. Oldham. Motivation through the Design of Work: Test of a Theory. *Organizational Behavior and Human Performance*, 16(2):250–279, 1976.

[64] Nathan Hahn, Joseph Chang, Ji Eun Kim, and Aniket Kittur. The Knowledge Accelerator: Big Picture Thinking in Small Pieces. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2258–2270, New York, NY, USA, 2016. ACM.

[65] Kieran Healy and Alan Schussman. The ecology of open-source software development. *Unpublished manuscript*, 2003.

[66] Chip Heath and Nancy Staudenmayer. Coordination neglect: How lay theories of organizing complicate coordination in organizations. *Research in Organizational Behavior*, 22:153–191, 2000.

[67] Kurtis Heimerl, Brian Gawalt, and Kuang Chen. CommunitySourcing: Engaging Local Crowds to Perform Expert Work via Physical Kiosks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1539–1548, New York, NY, USA, 2012. ACM.

[68] James D. Herbsleb and Rebecca E. Grinter. Splitting the Organization and Integrating the Code: Conway's Law Revisited. In *Proceedings of the 1999 International Conference on Software Engineering*, ICSE '99, pages 85–95, New York, NY, USA, 1999. ACM.

[69] Benjamin Mako Hill. *Essays on volunteer mobilization in peer production*. PhD thesis, Massachusetts Institute of Technology, 2013.

[70] Benjamin Mako Hill and Andres Monroy-Hernandez. The Cost of Collaboration for Code and Art: Evidence from a Remixing Community. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, pages 1035–1045, New York, NY, USA, 2013. ACM.

[71] Pamela Hinds, Lei Liu, and Joachim Lyon. Putting the Global in Global Work: An Intercultural Lens on the Practice of Cross-National Collaboration. *The Academy of Management Annals*, 5(1):135–188, 2011.

[72] Pamela J. Hinds and Cathleen McGrath. Structures that Work: Social Structure, Work Structure and Coordination Ease in Geographically Distributed Teams. In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*, CSCW '06, pages 343–352, New York, NY, USA, 2006. ACM.

[73] Pamela J. Hinds and Mark Mortensen. Understanding Conflict in Geographically Distributed Teams: The Moderating Effects of Shared Identity, Shared Context, and Spontaneous Communication. *Organization Science*, 16(3):290–307, 2005.

[74] Jim Hollan and Scott Stornetta. Beyond Being There. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 119–125, New York, NY, USA, 1992. ACM.

[75] John J. Horton, Leonard N. Stern, and Joseph M. Golden. Reputation Inflation: Evidence from an Online Labor Market. 2015.

[76] Jeff Howe. http://crowdsourcing.typepad.com/.

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

[78] Jeff Howe. *Crowdsourcing: How the power of the crowd is driving the future of business*. Century, 2008.

[79] Chang Hu, Benjamin B. Bederson, Philip Resnik, and Yakov Kronrod. MonoTrans2: A New Human Computation System to Support Monolingual Translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1133–1136, New York, NY, USA, 2011. ACM.

[80] Robert S. Huckman, Bradley R. Staats, and David M. Upton. Team Familiarity, Role Experience, and Performance: Evidence from Indian Software Services. *Management Science*, 55(1):85–100, 2009.

[81] Panagiotis G. Ipeirotis. Analyzing the Amazon Mechanical Turk Marketplace. *XRDS: Crossroads*, 17(2):16–21, 2010.

[82] Panagiotis G. Ipeirotis. Mechanical Turk, Low Wages, and the Market for Lemons, 2010.

[83] Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality Management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '10, page 64, New York, NY, 2010. ACM.

[84] Lilly Irani. The cultural work of microwork. *New Media & Society*, 17(5):720–739, 2015.

[85] Lilly C. Irani and M. Six Silberman. Turkopticon: Interrupting Worker Invisibility in Amazon Mechanical Turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 611–620, New York, NY, USA, 2013. ACM.

[86] Sirkka L. Jarvenpaa and Dorothy E. Leidner. Communication and Trust in Global Virtual Teams. *Organization Science*, 10(6):791–815, 1999.

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

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

[89] Henry A. Kautz and Bart Selman. Planning as Satisfiability. In *Proc. ECAI '92*, 1992.

[90] Katherine C. Kellogg, Wanda J. Orlikowski, and JoAnne Yates. Life in the Trading Zone: Structuring Coordination Across Boundaries in Postbureaucratic Organizations. *Organization Science*, 17(1):22–44, 2006.

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

[92] Kinshuman Kinshumann, Kirk Glerum, Steve Greenberg, Gabriel Aul, Vince Orgovan, Greg Nichols, David Grant, Gretchen Loihle, and Galen Hunt. Debugging in the (very) large: Ten years of implementation and experience. *Communications of the ACM*, 54(7):111 – 116, 2011.

[93] Aniket Kittur, Susheel Khamkar, Paul André, and Robert E. Kraut. CrowdWeaver: Visually Managing Complex Crowd Work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1033–1036, New York, NY, USA, 2012. ACM.

[94] Aniket Kittur, Bryant Lee, and Robert E. Kraut. Coordination in Collective Intelligence: The Role of Team Structure and Task Interdependence. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1495–1504, New York, NY, USA, 2009. ACM.

[95] Aniket Kittur, Jeffrey V. Nickerson, Michael S. Bernstein, Elizabeth M. Gerber, Aaron Shaw, John Zimmerman, Matthew Lease, and John J. Horton. The Future of Crowd Work. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1301–1318, New York, NY, USA, 2013. ACM.

[96] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. CrowdForge: Crowdsourcing Complex Work. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, New York, NY, USA, 2011. ACM.

[97] Katherine J. Klein, Jonathan C. Ziegert, Andrew P. Knight, and Yan Xiao. Dynamic Delegation: Shared, Hierarchical, and Deindividualized Leadership in Extreme Action Teams. *Administrative Science Quarterly*, 51(4):590–621, dec 2006.

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

[99] Robert Kraut and Lynn Streeter. Coordination in Software Development. *Communications of the ACM*, 38(3):69–81, 1995.

[100] Anand Kulkarni, Matthew Can, and Björn Hartmann. Collaboratively Crowdsourcing Workflows with Turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1003–1012, New York, NY, USA, 2012. ACM.

[101] Anand Kulkarni, Philipp Gutheim, Prayag Narula, David Rolnitzky, Tapan S. Parikh, and Björn Hartmann. MobileWorks: Designing for Quality in a Managed Crowdsourcing Architecture. *IEEE Internet Computing Magazine*, 16(5):28, 2012.

[102] Walter S. Lasecki, Christopher Homan, and Jeffrey P. Bigham. Architecting Real-Time Crowd-Powered Systems. *Human Computation*, 1(1):67–93, 2014.

[103] Walter S. Lasecki, Juho Kim, Nick Rafter, Onkur Sen, Jeffrey P. Bigham, and Michael S. Bernstein. 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, New York, NY, USA, 2015. ACM.

[104] Walter S. Lasecki, Christopher Miller, Adam Sadilek, Andrew Abumoussa, Donato Borrello, Raja Kushalnagar, and Jeffrey P. Bigham. Real-Time Captioning by Groups of Non-Experts. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '15, pages 23–33, New York, NY, USA, 2012. ACM.

[105] Walter S. Lasecki, Kyle I. Murray, Samuel White, Robert C. Miller, and Jeffrey P. Bigham. Real-time Crowd Control of Existing Interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, New York, NY, USA, 2011. ACM.

[106] Walter S. Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F. Allen, and Jeffrey P. Bigham. Chorus: A Crowd-Powered Conversational Assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, New York, NY, USA, 2013. ACM.

[107] Adam Lashinsky. *Inside Apple*. Hachette Book Group, New York, 2012.

[108] Thomas D. LaToza, W. Ben Towne, Christian M. Adriano, and Andre van der Hoek. Microtask Programming: Building Software with a Crowd. In *Proceedings of the 27th Annual ACM symposium on User Interface Software and Technology*, UIST '14, pages 43–54, New York, NY, USA, 2014. ACM.

[109] Edith Law and Luis Von Ahn. Human Computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5:1–121, 2011.

[110] Edith Law and Haoqi Zhang. Towards Large-Scale Collaborative Planning: Answering High-Level Search Queries Using Human Computation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI '11, pages 1210–1215. AAAI Press, 2011.

[111] Paul R. Lawrence and Jay W. Lorsch. Differentiation and Integration in Complex Organizations. *Administrative Science Quarterly*, 12(1):1–47, 1967.

[112] Min Kyung Lee, Daniel Kusbit, Evan Metsky, and Laura Dabbish. Working with Machines: The Impact of Algorithmic and Data-Driven Management on Human Workers. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1603–1612, New York, NY, USA, 2015. ACM.

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

[114] Yvonna S. Lincoln and Egon G. Guba. *Naturalistic inquiry*, volume 75. Sage, 1985.

[115] Chris J. Lintott, Kevin Schawinski, Anže Slosar, Kate Land, Steven Bamford, Daniel Thomas, M Jordan Raddick, Robert C. Nichol, Alex Szalay, Dan Andreescu, Phil Murray, and Jan Vandenberg. Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, sep 2008.

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

[117] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. TurKit: Human Computation Algorithms on Mechanical Turk. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, page 57, New York, NY, USA, 2010. ACM.

[118] Kurt Luther, Kelly Caine, Kevin Ziegler, and Amy Bruckman. Why it works (when it works): success factors in online creative collaboration. In *Proceedings of the 16th ACM International Conference on Supporting Group Work*, GROUP '10, New York, NY, USA, 2010. ACM.

[119] Kurt Luther, Casey Fiesler, and Amy Bruckman. Redistributing leadership in online creative collaboration. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, New York, NY, USA, 2013. ACM.

[120] Kurt Luther, Jari-lee Tolentino, Wei Wu, Amy Pavel, Brian P. Bailey, Maneesh Agrawala, Björn Hartmann, and Steven P. Dow. Structuring, Aggregating, and Evaluating Crowdsourced Design Critique. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 473–485, New York, NY, USA, 2015. ACM.

[121] Ioanna Lykourentzou, Angeliki Antoniou, Yannick Naudet, and Steven P. Dow. 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*, pages 260–273, New York, NY, USA, 2016. ACM.

[122] Thomas W. Malone and Kevin Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, 1994.

[123] Thomas W. Malone, Kevin Crowston, and George A. Herman. *Organizing Business Knowledge: The MIT Process Handbook*. MIT Press, Cambridge, MA, USA, 2003.

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

[125] Adam Marcus and Aditya Parameswaran. Crowdsourced Data Management: Industry and Academic Perspectives. *Foundations and Trends in Databases*, 6(1-2):1–161, 2015.

[126] Gloria Mark. Extreme Collaboration. *Communications of the ACM*, 45(6):89–93, 2002.

[127] David Martin, Benjamin V. Hanrahan, Jacki O'Neill, and Neha Gupta. Being A Turker. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 224–235, New York, NY, USA, feb 2014. ACM.

[128] Winter Mason and Duncan J Watts. Financial incentives and the performance of crowds. *Proceedings of the ACM SIGKDD Workshop on Human Computation*, 11(2):77–85, 2009.

[129] John E. Mathieu, Tonia S. Heffner, Gerald F. Goodwin, Eduardo Salas, and Janis A. Cannon-Bowers. The Influence of Shared Mental Models on Team Process and Performance. *Journal of Applied Psychology*, 85(2):273–83, apr 2000.

[130] Martha L. Maznevski and Katherine M. Chudoba. Bridging space over time: Global virtual team dynamics and effectiveness. *Organization Science*, 11(5):473–492, 2000.

[131] Tom Mens. A state-of-the-art survey on software merging. *IEEE Transactions on Software Engineering*, 28(5):449–462, 2002.

[132] Tanushree Mitra, C.J. Hutto, and Eric Gilbert. Comparing Person- and Process-centric Strategies for Obtaining Quality Data on Amazon Mechanical Turk. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1345–1354, New York, NY, USA, 2015. ACM.

[133] Bonnie A. Nardi and Steve Whittaker. The Place of Face-to-Face Communication in Distributed Work. In Pamela J. Hinds and Sara Kiesler, editors, *Distributed Work*, pages 83–113. MIT Press, Cambridge, 2002.

[134] Michael Nebeling, Alexandra To, Anhong Guo, Adrian A. de Freitas, Jaime Teevan, Steven P. Dow, and Jeffrey P. Bigham. WearWrite: Crowd-Assisted Writing from Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 3834–3846, 2016.

[135] Tsedal B. Neeley, Pamela J. Hinds, and Catherine D. Cramton. The (Un)Hidden Turmoil of Language in Global Collaboration. *Organizational Dynamics*, 41(3):236–244, 2012.

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

[137] Gerardo A. Okhuysen and Beth A. Bechky. Coordination in Organizations: An Integrative Perspective. *Academy of Management Annals*, 3(1):463–502, jan 2009.

[138] Michael Boyer O'Leary and Jonathon N. Cummings. *The Spatial, Temporal, and Configurational Characteristics of Geographic Dispersion in Teams*, volume 31. 2007.

[139] Gary Olson and Judith Olson. Distance Matters. *Human-Computer Interaction*, 15(2):139–178, 2000.

[140] Judith S. Olson and Gary M. Olson. How to make distance work work. *Interactions*, 21:28–35, 2014.

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

[142] Amanda Pallais. Inefficient hiring in entry-level labor markets. *American Economic Review*, 104(11):3565–3599, 2014.

[143] Charles Perrow. A framework for the comparative analysis of organizations. *American Sociological Review*, pages 194–208, 1967.

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

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

[146] Katharina Reinecke, Minh Khoa Nguyen, Abraham Bernstein, Michael Näf, and Krzysztof Z. Gajos. Doodle Around the World: Online Scheduling Behavior Reflects Cultural Differences in Time Perception and Group Decision-Making. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 45–54, New York, NY, USA, 2013. ACM.

[147] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[148] Daniela Retelny, Sebastien Robaszkiewicz, Alexandra To, Walter S. Lasecki, Jay Patel, Negar Rahmati, Tulsee Doshi, Melissa Valentine, and Michael S. Bernstein. Expert Crowdsourcing with Flash Teams. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, New York, NY, USA, 2014. ACM.

[149] Horst W. J. Rittel and Melvin M. Webber. Dilemmas in a General Theory of Planning. *Policy Sciences*, 4(2):155–169, 1973.

[150] Philip J. Roberts. No Json-delta: a diff/patch pair for json-serialized data structures, 2017.

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

[152] Joel Ross, Andrew Zaldivar, Lilly Irani, and Bill Tomlinson. Who are the Turkers? Worker Demographics in Amazon Mechanical Turk. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pages 2863–2872, New York, NY, USA, 2010. ACM.

[153] Richard M. Ryan and Edward L. Deci. Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being. *American Psychologist*, 55(1):68–78, 2000.

[154] Niloufar Salehi, Lilly C. Irani, Michael S. Bernstein, Ali Alkhatib, Eva Ogbe, Kristy Milland, and Clickhappier. We Are Dynamo: Overcoming Stalling and Friction in Collective Action for Crowd Workers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '15, pages 1621–1630, New York, NY, USA, 2015. ACM.

[155] Niloufar Salehi, Andrew McCabe, Melissa Valentine, and Michael S. Bernstein. 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, New York, NY, USA, 2017. ACM.

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

[157] Ben Shneiderman. Science 2.0. *Science*, 319(5868):1349–1350, 2008.

[158] Yaron Singer and Manas Mittal. Pricing Mechanisms for Online Labor Markets. In *Proceedings of the AAAI Human Computation Workshop*, HCOMP '11. AAAI Press, 2011.

[159] Alexander Sorokin and David Forsyth. Utility data annotaton with Amazon Mechanical Turk. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '08, 2008.

[160] Klaas-Jan Stol and Brian Fitzgerald. Two's company, three's a crowd: a case study of crowdsourcing software development. In *Proc. ICSE '14*, 2014.

[161] Lucy Suchman. *Human-Machine Reconfigurations: Plans and Situated Actions*. Cambridge University Press, Cambridge, 2nd edition, 2007.

[162] Ryo Suzuki, Niloufar Salehi, Michelle S. Lam, Juan C. Marroquin, and Michael S. Bernstein. Atelier: Repurposing Expert Crowdsourcing Tasks as Micro-internships. In *Proceedings of the*

*2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, New York, NY, USA, 2016. ACM.

[163] Jaime Teevan, Shamsi T. Iqbal, and Curtis Von Veh. Supporting Collaborative Writing with Microtasks. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, New York, NY, USA, 2016. ACM.

[164] Rannie Teodoro, Pinar Ozturk, Mor Naaman, Winter Mason, and Janne Lindqvist. 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, pages 236–247, New York, NY, USA, 2014. ACM.

[165] James D. Thompson. *Organizations in Action: Social Science Bases of Administrative Theory.* McGraw-Hill, New York, 1967.

[166] Upwork. http://upwork.com/.

[167] Upwork. Online Work Report: Global, 2014 Full Year Data, 2014.

[168] Donna Vakharia and Matthew Lease. Beyond mechanical turk: An analysis of paid crowd work platforms. In *Proceedings of the iConference*, iConference '15, pages 1–17, 2015.

[169] Melissa A. Valentine and Amy C. Edmondson. Team Scaffolds: How Mesolevel Structures Enable Role- Based Coordination in Temporary Groups. *Organization Science*, 26(2):405–422, 2015.

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

[171] Andrew H. Van de Ven, Andre L. Delbecq, and Richard Koenig. Determinants of Coordination Modes within Organizations. *American Sociological Review*, 41(2):322, 1976.

[172] Vasilis Verroios and Michael S. Bernstein. Context Trees: Crowdsourcing Global Understanding from Local Views. In *Second AAAI Conference on Human Computation and Crowdsourcing*, number 1351131 in HCOMP '14, pages 210–219. AAAI Press, 2014.

[173] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM.

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

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

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

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

[178] Terry Winograd. A Language/Action Perspective on the Design of Cooperative Work. *Human–Computer Interaction*, 3(1), 1987.

[179] Anbang Xu, Shih-Wen Huang, and Brian Bailey. Voyant: Generating Structured Feedback on Visual Designs Using a Crowd of Non-experts. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 1433–1444, New York, NY, USA, 2014. ACM.

[180] Lixiu Yu, Aniket Kittur, and Robert E. Kraut. Distributed Analogical Idea Generation with Multiple Constraints Lixiu. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, CSCW '16, pages 1236–1245, New York, NY, USA, 2016. ACM.

[181] Lixiu Yu, Aniket Kittur, and Robert E. Kraut. 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, pages 1214–1222, New York, NY, USA, 2016. ACM.

[182] Lixiu Yu and Jeffrey V. Nickerson. Cooks or Cobblers? Crowd Creativity through Combination. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1393–1402, New York, NY, USA, 2011. ACM.

[183] Haoqi Zhang, Edith Law, Robert C. Miller, Krzysztof Z. Gajos, David C. Parkes, and Eric Horvitz. Human Computation Tasks with Global Constraints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 217–226, New York, NY, USA, 2012. ACM.

[184] Mengyao Zhao and Andre van der Hoek. A Brief Perspective on Microtask Crowdsourcing Workflows for Interface Design. In *2nd International Workshop on Crowdsourcing in Software Engineering*, pages 45–46, 2015.

[185] Yue Maggie Zhou. Designing for Complexity: Using Divisions and Hierarchy to Manage Complex Tasks. *Organization Science*, 24(2):339–355, 2013.