Foundry: Managing Teams of Experts Online

Alexandra To
Stanford University, Symbolic Systems Program
Stanford, CA 94305
ato1120@stanford.edu

Advisor: Michael Bernstein
Second Reader: Daniela Retelny

To the Directors of the Program on Symbolic Systems:
I certify that I have read the thesis of Alexandra To in its final form for submission and have found it to be satisfactory for the degree of Bachelor of Science with Honors.

Michael Bernstein
Computer Science

To the Directors of the Program on Symbolic Systems:
I certify that I have read the thesis of Alexandra To in its final form for submission and have found it to be satisfactory for the degree of Bachelor of Science with Honors

Daniela Retelny
Management Science & Engineering

ABSTRACT

The crowd provides significant power to complete micro tasks on-demand, but fails to provide the expertise necessary to complete complex, interdependent projects. Flash teams, a framework for coordinating teams of expert crowds, have been successful in providing efficient and consistent outcomes for project requesters, but are not currently accessible to the average person. We present Foundry, an authoring and runtime environment for flash teams. Foundry enables non-expert project requesters to use flash teams by guiding the requester through creating their own teams of expert crowd workers using decomposed team roles and task-based events in the authoring environment. On the platform teams can be created computationally - they can be entirely replicated for another requester, or their various parts can be broken apart for reuse in other related projects. In this project, we run studies to evaluate the use of the authoring interface to determine if Foundry is accessible for the general population, whether the concept of flash teams is made salient with the help of Foundry, and how comprehensive and realistic their teams would be. We found that participants were able to create teams within Foundry, particularly after we implemented the event library where they could view and/or use previously authored events. Our findings also helped us identify opportunities and areas for future work, such as data-driven creation of teams and Foundry-mediated teams that can "follow the sun." Foundry enables the everyday population to take advantage of the considerable power of the expert crowd.

ACKNOWLEDGEMENTS

I would like to thank the Symbolic Systems Program and in particular, Todd Davies, who has encouraged me to do good research and supported my work for the past couple of years. I could not have done with without that push you gave me.

I'd like to recognize that for *most* professors and graduate students, there is little motivation to take on undergraduate researchers. I'd like to thank Michael Bernstein, Daniela Retelny, and Sebastien Robaszkiewicz for taking on a very last minute request to join their amazing Flash Teams research group and for always encouraging me to be an active participant in the conversation and direction of the work. Foundry would not have come about without all of your hard work and mentorship. Additionally, thank you to everyone in the Stanford HCI Group for being so warm, welcoming, and encouraging - your wisdom and witticisms have made this past year of work deeply engaging and exciting.

Foundry is a major code undertaking. It in no way could have existed with any semblance of stability without the hard work of Jay Patel, Negar Rahmati, Virginia Tong, Tulsee Doshi, Roger Chen, and Lilith Wu. Thank you all so much for believing in and contributing to this project!

To my parents, John and Joanne, and my sisters, Robyn, Rebecca, and Caroline: thank you for all of your support. Additional thanks is due to my grandparents, Reiko and Michio Yoshizu and Stanley and Rosemary To. Despite my failures to explain what it is exactly that I do at this university, thank you for your never-wavering support and enthusiasm. Thank you especially to Stanley To for encouraging me to pursue CS and SymSys, and whose foresight and love have allowed me to attend this school.

Thanks to my friends, Patrick, Vannida, Xiao Song, Roger, and Melissa, and to my partner, Joe for your constant emotional support. You guys are the best.

Work on Foundry has been supported by a UAR Minor Grant. The studies in this paper have been authorized by Stanford's IRB board, protocol 28440. Some text from this paper comes from a submission to UIST 2014.

TABLE OF CONTENTS

I. INTRODUCTION

Crowdsourcing systems coordinate large groups of people to solve problems that a single individual could not achieve at the same scale. Microtasking systems, such as Amazon Mechanical Turk, typically use highly-controlled workflows to manage paid, non-expert workers toward expert-level results [4, 21, 7, 17, 29]. 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 that anyone can complete [19, 33, 31].

Paid experts from the crowd may be able to solve these complex, interdependent problems. Because they are online, these crowds can be accessed on-demand and the experts have a wide range of specializations (video producers, software developers, designers, etc.). Experts from the crowd, however, tend to work as isolated contractors, and microtask crowdsourcing techniques (e.g., [2, 23, 17, 34]) cannot coordinate these experts because they do not effectively leverage participants' diverse skills and expertise.

On-demand teams of paid experts from the crowd succeed in completing complex and interdisciplinary projects when coordinated properly. *Flash teams*, a framework for dynamically assembling and managing these teams solve many challenges to coordinating on-demand, online teams [35]. Flash teams assign members of a team to roles based on their expertise and the tasks they must complete and guide them through a timeline of task-based events. Flash teams are modular and thus create replicable structures that provide reliable outcomes. We envision that end users can call upon the crowd to complete something as complex as a prototype of a web application, from team creation to hiring to implementation, in a matter of a couple days.

The requester of a flash team typically doesn't have the expertise or resources to complete some project on their own. Because of these limitations, they struggle in coordinating these teams of expert crowd workers. Non-experts may not know who they need to hire (e.g., hiring an 'artist' vs. 'sketch artist') and once they have hired, they may not know how much guidance the team needs. Once finding that on-demand workers need an actual work timeline, another issue is that while people tend to be good at decomposing tasks, for some project (e.g. a team that creates an eBook), a non-expert may struggle to know what they are decomposing in the first place. For example, in the "create an interactive eBook team", some non-expert may be able to decompose artistic tasks (e.g., "character design," "storyboarding," "illustrations"), but may not know that the programming of the book can be decomposed, and may just list one large "programming" task. This decomposed workflow is critical for workers who are online, and thus (presumably) unfamiliar with each other to work interdependently.

Without the flash teams methodology, these teams of expert crowd workers fall into many different inefficient and inappropriate work behaviors. Amongst other things, in a controlled study the work became very late, workers fill all spare time with meaningless work, and fail to communicate with their teammates leading to redundant work [35] - all of which leads to expensive projects with inconsistent outcomes.

To enable a vision in which these groups of expert crowd workers can work as an interdependent team and avoid those behaviors, there is need for structured workflows that dictate how and when workers might work individually, as well as how and when the workers should collaborate and communicate with each other. As mentioned above, however, project requesters may not have the managerial knowledge to create this structure on their own. The objective of this research, therefore, is to provide support for creating these teams in a way that structures their collaboration and take away managerial overhead from the requesters through an online platform - Foundry. This effort might usually be handled by a project manager, but the goal is to enable the requester to cut out this middle man who may not fully understand the vision for the project that the requester wants.

Distributed work faces a number of known coordination issues [30, 13, 8, 14]. In our experience, the major issues faced by the on-demand teams of expert workers are: busy working and lack of communication/visibility. We solve these problems by implementing a task-based timeline where at least one team member is assigned to each event. Team members are aware of where the team is on this timeline and keep open channels of communication in order to coordinate their work, collaborate when told to, and to be notified when tasks are either on-time, late, or early. These approaches are embedded into Foundry.

This thesis presents Foundry, a platform for assembling flash team structures and coordinating live flash teams. Foundry guides the requester through decomposing a team into conceptual pieces. These project requesters can create flash teams from scratch or by replicating or recombining existing teams from the flash teams library. The platform grounds the user in the flash teams methodology by dividing the authoring interface into two major sections - “team roles” and “timeline.”

In the team roles section, users add role-based members to the team. On the timeline users add task-based events with specifications including: input, output, team roles working on the task, a directly-responsible individual (DRI), and duration. Finally, on the timeline users can draw interactions between events on the timeline which can take the form of either handoffs or collaborations. Users can draw a handoff to indicate that one event on the timeline is “handing-off” some output as an input to the subsequent event (ex. event 1 may produce a low-fi prototype, which event 2 receives and then evaluates) via an arrow from one event to the next. A collaboration between two events indicates that the members of the two events keep in communication throughout the duration of their work, displayed as shading connecting the two events.

Once the team has been authored and all of the expert workers have been hired, Foundry supports the team’s collaboration. When the requester hits “start,” the team members are invited to log onto the system runtime interface where they can communicate with each other via a chat, upload files to a common Google Drive folder, and monitor the timeline. They indicate when they are finished with the tasks they have been assigned to via a “complete” button on their individual tasks. 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 [11]. Foundry abstracts away low-level management effort while allowing for high-level oversight and guidance. The runtime mode of Foundry is critical to the end-to-end process for the requester, but the focus this paper's contribution is the authoring mode.

To test whether non-experts can successfully create flash teams, this research empirically tests whether users of Foundry can author team structures and workflows. Specifically, we are use binary measures to test whether individuals can decompose team roles and tasks, reason about the duration of tasks, assign team roles to tasks, and utilize task inputs and outputs. Our initial results suggested that non-experts struggled with reasoning about the duration of tasks as well as using the inputs and outputs. This led us to implement the event library, which contains a searchable list of events that have been known to work under their specified inputs, outputs, and duration. We predicted that this would introduce enough structure for the user to create tasks with accurate durations by either directly taking events from the library or by framing their understanding of similar tasks. Our results suggested that some users drew inspiration for potential tasks from the event library, but were still unable to create tasks with accurate durations.

We also evaluated the breadth of flash teams by asking users to choose another team from a list of 20 (e.g., add a dish to your restaurant, write a song, etc.) to create from scratch. Users were enthusiastic and more invested in the outcome of these alternative teams.

In order to create the timeline, participants used intuitions from other calendaring interfaces to draw, drag, resize, and edit the details of events on the timeline. They create the timeline in a relatively linear fashion. They will have added most of the team roles they intend to use first, often leaving out the details, and will return to add more as they become relevant on the timeline.

Chapter 2 discusses the work that has lead to the flash teams methodology as well as previous authoring platforms for crowd work. Chapter 3 discusses the Foundry system by describing a scenario in which a potential user might use the system. Next I describe what flash teams are and the aspects of flash teams that can be computationalized in Foundry. I expand upon the details of how system usage operates when a requester wants to create a team and when the team logs on to use the system as they work. We evaluate how non-expert users create flash teams using Foundry (Ch. 4) and discuss the results of this study and its implications (Ch. 5). I conclude with the future work surrounding Foundry (Ch. 6).

Flash teams structure expert crowd work so that teams are efficient, communicate, and provide consistent outcomes. By combining organizational behavior research with human-computer interaction research, flash teams enable complex, on-demand project. But to date, flash teams have only been created by domain experts (or at least domain journeymen). In contrast, the microtasking crowd is easily accessible, but has very little utility for the average person. With Foundry, we can use our knowledge of the crowd and flash teams to enable the general population to use crowd work for these complex projects such as web design or creating a short film. Foundry opens up access to the crowd in ways that more and more people can take advantage of.

Foundry empowers non-experts to take advantage of the power of expert crowds through the flash teams methodology. By using computation to aid the process of creating a team and workflow, non-experts in both project management as well as the actual purpose of the particular project can run and oversee on-demand teams completing interdependent and complex work. Through Foundry's pipelining and runtime management, these teams can complete the work on an efficient timeline, cost effectively, and with consistent outcomes.

II. RELATED WORK

In this section, we connect two threads of research: crowdsourcing and organizational behavior, particularly team effectiveness research, very briefly to set the context for the flash teams research. This is followed by a more exhaustive discussion of existing authoring platforms for crowd work.

Crowdsourcing expertise

To date, crowdsourcing has largely focused on goals that any individual can support: many crowdsourcing platforms are built to accomplish tasks that require little training (e.g., Amazon Mechanical Turk) and recruit amateurs (e.g. FoldIt) [7]. Consequently, most crowdsourcing workflows and algorithms [17] aim to structure non-expert contributions to produce expert-level performance. For example, MapReduce frameworks can guide crowds to write simple encyclopedia entries [16], and clustering workflows can produce expert-level taxonomies [5]. These workflows can even be authored by crowd members themselves [20]. Artificial intelligence techniques optimize these workflows [10]. Prior work that has gone beyond microtasks has still focused on recreating the performance of a single expert [22].

Prior work suggests that expertise might play a role in crowdsourcing [18]; flash teams instantiate a first step toward that goal. Other crowdsourcing campaigns already recruit experts, for example using workflow support tools [26] or helping solve planning problems [34]. Often these campaigns focus on a single expertise such as math [9]. 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. As with microtask crowds, expert crowd work could then be integrated into crowd-powered systems (e.g., [19]).

Organizational behavior

Research from the field of organizational behavior (OB) is also relevant to this research because it identifies both the obstacles to effective team coordination, and also the conditions and structures that enable effective team coordination.

Prior work has shown that the obstacles to effective team coordination include geographic dispersion, technology-mediated communication, and fluid (or changing) team membership [30, 13, 8, 14]. These conditions impede team coordination because team members lack a shared understanding of their overall task, and each person's responsibilities within the task [27]. Because their communication is all electronically-mediated, they also struggle to engage in the rich communication necessary to build a shared understanding [8]. We suspect that these obstacles will be exacerbated in teams composed of experts pulled at random from the crowd.

To respond to the extremely challenging working conditions that crowdsourced expert teams will necessarily confront, flash teams draw on OB studies that demonstrate how to help teams coordinate effectively even when working in challenging conditions. One key insight is that team structures enable coordination by encoding who is responsible for what work and which team members are interdependent [5]. Team structures provide shared understanding of each person's role on the team, but usually develop over time through repeated interactions working together, which is clearly not possible for crowdsourced expert teams. Still, recent studies show that 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 [32].

In this research, we integrate these insights with expert crowdsourcing. We argue that this previous literature reveals that for teams of crowdsourced experts to be most successful, they need to quickly understand their shared work, their interdependencies, and their respective roles in completing that work [28]. Our aim is to provide the value of team structures (as identified in OB research), but to do so in ways that leverage automation, computation, and the scale and flexibility of the crowd.

Authoring Interfaces

This work raises opportunities for platform design in expert crowdsourcing. Foundry draws inspiration from visual workflow tools [16, 20] and management tools such as Gantt charts to structure its teams using a visual timeline language. It is possible to design for worker interest, honesty, and motivation (e.g., [3, 12]); Foundry thus aims to make collaborators visible and make the larger goal clear. Finally, Foundry opens opportunities to construct new platform affordances for expert workers, which has traditionally been difficult.

Other authoring platforms support users creating and managing workflows for complex crowd work. Platforms can model the work of the crowd for the requester [16]. Alternatively, the crowd can create their own workflows for complex work [20]. Others focus on the empowerment of the microtask crowd to cooperate and seamlessly contribute [15]. Rather than using experts, other platforms seek to use microtask workers to solve complex problems through extreme decomposition and later recombination of tasks [17]. However, none of these systems encourage the crowd or expert crowds to collaborate as an interdependent team. For complex work, micro-task workers who are non experts are unable to complete the projects without interference and guidance from a requester. In the decomposed workflow platforms, communication is not facilitated between members, limiting their ability to collaborate.

III. THE FOUNDRY SYSTEM

Foundry is a flash team authoring and runtime environment. Using Foundry, non-experts can form and execute complex projects rapidly and with consistent outcomes. The requester is guided through creating a team that suits their specific needs and can then hire experts to carry out the cost efficient project. On the system, members can be edited using categories and necessary skills that come from oDesk [1] - a commercial website for hiring freelance workers. While the team works, Foundry facilitates their workflow progression and communication and allows the requester to check in on the team. As site usage increases, Foundry can offer more information to requesters on tasks and provide more end-to-end teams. We discuss Foundry's core functionality in more detail at the end of this section.

Foundry is backed by a Ruby on Rails server and PostgreSQL database which stores flash teams in JSON format, allowing extremely quick data storage and retrieval. Through automatic saves via Javascript to the backend, Foundry allows requesters to experiment with various team structures in the authoring environment without having to manually click a 'Save' button. This makes a very robust interface that requesters can "play" with without worrying about losing any data. The interface is highly interactive with such capabilities as using the mouse to draw an event and dragging events around, made possible by using the well-known Javascript libraries D3.js, jQuery and Twitter Bootstrap and taking advantages of user intuitions from calendaring interfaces. Foundry is seamless, with each part of the interface being automatically updated to reflect a change in data. For example, if the color used to represent a member is changed, every event to which that member is assigned to sees a change in that color immediately.

Scenario

The following scenario provides an example of how and why someone would use Foundry. Imagine an employee of a large company, Joanne, has an exciting new idea for a web-based project (e.g., a photo album interface). She has an important meeting coming up and wants three interactive prototypes of the web application. Joanne has done simple prototypes in the past, but she doesn't have the expertise to create three prototypes on her own in a short timeframe. She goes online, where she knows she can hire experts on-demand to help complete the work with the budget for her project. Joanne goes to Foundry to create her own team from scratch, given her knowledge of the design process and her desire to guide the team.
When deciding to create the team, Joanne sees that the Foundry team-authoring interface has three major panes - team roles, timeline, and event library. She begins by adding team roles to the team - a user interface (UI) designer and a web programmer. Next she moves to the timeline to add a "Low-fi Prototype V1" event (Figure 1). She specifies details, adding that this event takes her sketches as an input and outputs a low-fi prototype, checks the UI designer as the worker and the DRI, and changes the duration of this task to 2.5 hours.

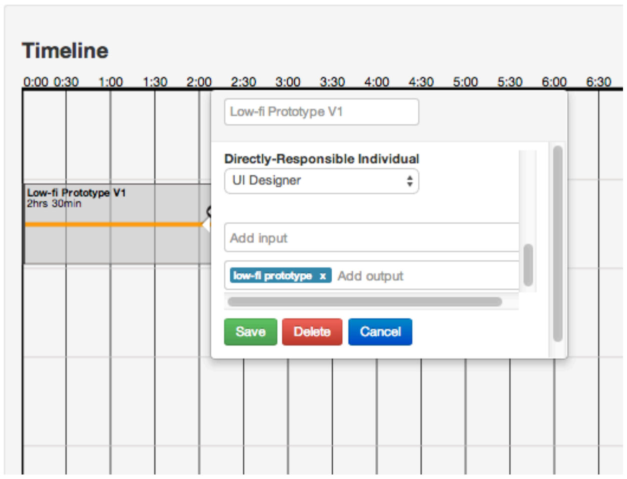


Figure 1. Joanne edits the low-fi prototype with a pop-up form

To reaffirm her intuitions about the next events, Joanne goes to the event library and searches “low-fi prototype” to see what other known events might take this as an input. She finds three events that she wants to add: “web programming V1,” low-fi prototype v2,” and “heuristic evaluation” and drags them from the event library on to the timeline. She arranges all three events drawing handoffs from the low-fi prototype output to input. While adding team members to the event, Joanne realizes she does not have a role in her team that can do a heuristic evaluation and goes back to team roles to add a user experience (UX) researcher. Following this process, she eventually creates the entire timeline - shown in Figure 1. After completing the team, she can duplicate the team two more times for her three web app prototypes.

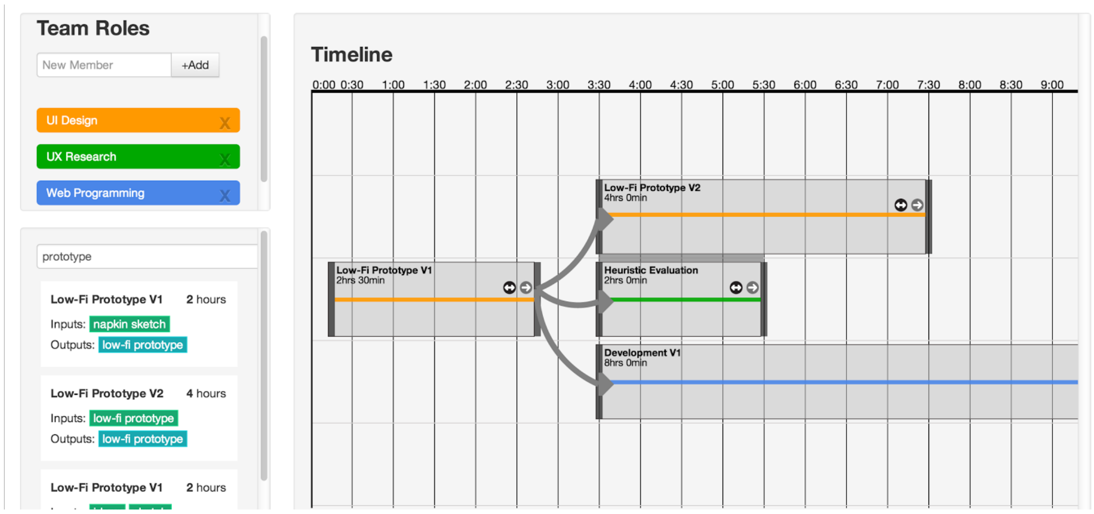


Figure 2. A partial view of the completed web app prototype team

Now Joanne begins to hire for the project - perhaps starting from within her company (who may have more familiarity with how her project ties into the company) looking for people who are available and then looking to outside vendors like oDesk. She gives potential candidates the estimated timeline and interviews based on their availability and ability to complete the tasks. Because the timeline is flexible, she could even take feedback from her hires into account and make adjustments. After hiring is complete, she uses the system to send login details to the team members for each of her three projects and can click begin to start the ticker on the timeline that displays the progression of the project.

The Foundry interface now switches to team runtime mode, hiding the event library and including a chat for the team and Joanne and links on each event to a Google Drive account for sharing files and collaborative work. The team members can each see when their tasks are upcoming, and when they complete a task, they mark it as completed. Foundry sends notifications when a task is coming up, and notifications when tasks are either early or delayed. This allows the teams to be autonomous, but Joanne can check in on their progression and utilize their group chat. She logs in right as the low-fi prototype v1 is scheduled to end. She sees that the UI designer has completed the event 15 minutes early and successfully handed-off the low-fi prototype to the programmer, and are in discussion with the UX researcher about their impending collaborative work. Joanne continues to periodically check in, but the team is able to follow the team workflow with little to no managerial effort on her part. In the end, they are able to create her working web app prototype in under 24 hours.

Flash Teams

Here we reintroduce flash teams, which are lightweight structures that allow an interactive computational system to guide a team of paid experts from the crowd reproducibly and at scale. 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 produce meaningful visualizations of progress. We take a position that expert crowd work can succeed by using 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 flash teams can grow and adapt.

Flash Team Composition

To structure expert crowd work, flash teams focus on small atomic actions called *blocks*. Each block represents one or more experts performing a task. It requires an input, instructions for how one or more members of the crowd should work, and specifies an output (Figure 3). 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 audio files`), and any blocks that interface along the same input or output use the same tag. For example, 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.

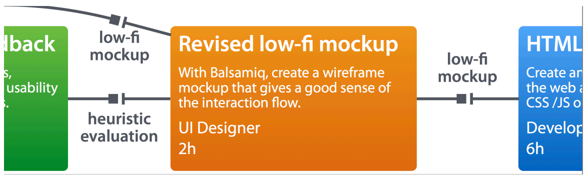


Figure 3. A heuristic evaluation and low-fi mockup are the inputs and the output is another low-fi

Because each block may have a different member of the crowd, flash teams exhibit distributed leadership [26] where the user (requester) may be coordinating with multiple workers in each block. To avoid a diffusion of responsibility, each block specifies a *directly responsible individual* [24], 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 completes 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.

Blocks can be linked via handoffs and collaborations - representations of team members from each block's interactions with each other. Handoffs occur from the end of one block to the beginning of another - representing a handoff of an output from block one as an input to block two. Collaborations can happen between overlapping events, indicating that the workers on each block are in constant communication with each other while working on their different tasks.

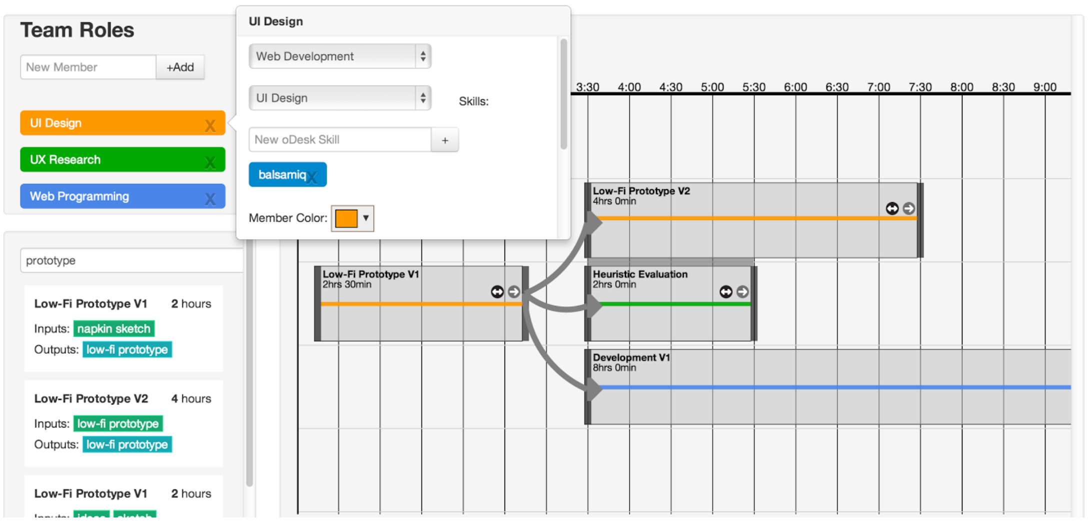
Computationally-enhanced flash teams
Flash teams are represented in a machine-readable way. One major strength of flash teams is that computation can leverage these representations. For example, we use a block's particular output tag to generate a list of blocks that use the same tag as an input. In this section, we introduce mechanisms for Foundry to react, optimize and author flash teams on demand.

Elasticity - Simple workflows cannot react to changing requirements: for example, an application may require additional help or a new skill. Because crowdsourcing platforms allow us to recruit quickly, however, 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. 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 - So far, flash teams have been described as blocking operations: downstream tasks must wait for upstream tasks to finish and upload their final output 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 waiting for the entire task to complete. 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 many pipelining scenarios allow team members to work productively in parallel. In addition, instead of communicating through a single brief handoff, pipelining encourages more synchronous feedback between team members, reducing the risk of communication failures or misunderstandings.

Team Authoring and Runtime

Team roles are represented as pills in the upper left panel of Foundry. After adding a new team role, the user can categorize the roles using two-tiered categories from the oDesk website (e.g., Web Development -> Web Design).   Requesters can also add desired skills based on the offerings of the oDesk website to team roles. Team roles are distinguished from each other by their role-based title and customizable color (Figure 4).



Team Roles

New Member +Add

UI Design x

UX Research x

Web Programming x

UI Design

Web Development

UI Design

Skills:

New oDesk Skill +

balsamiq x

Member Color:

prototype

Low-Fi Prototype V1 2 hours
Inputs: napkin sketch
Outputs: low-fi prototype

Low-Fi Prototype V2 4 hours
Inputs: low-fi prototype
Outputs: low-fi prototype

Low-Fi Prototype V1 2 hours

3:30 4:00 4:30 5:00 5:30 6:00 6:30 7:00 7:30 8:00 8:30 9:00

Low-Fi Prototype V2 4hrs 0min

Low-Fi Prototype V1 2hrs 30min

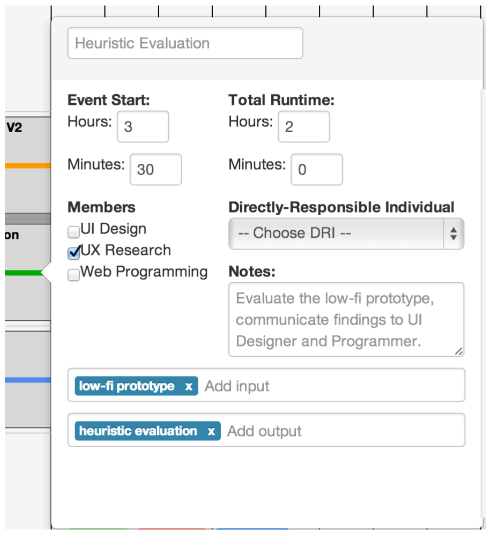
Heuristic Evaluation 2hrs 0min

Development V1 8hrs 0min

Figure 4. Add and edit a UI Designer

Users add blocks to the timeline as they would on a calendaring interface. Each block requires a title, a target length of time, the input and output tags, a description of the task (ideally with an example), and team member(s) who will complete the block. Inputs and outputs are represented as user interface pills, which the requester can use to pipe different events together. A DRI is also specified from amongst the team members on the event (Figure 5).

Foundry also makes available a library of all blocks that previous teams have used for drag-and-drop. Events in this event library are shown as blocks that display task name, duration,



Heuristic Evaluation

Event Start:
Hours: 3
Minutes: 30

Total Runtime:
Hours: 2
Minutes: 0

Members
☐ UI Design
☑ UX Research
☐ Web Programming

Directly-Responsible Individual
-- Choose DRI --

Notes:
Evaluate the low-fi prototype, communicate findings to UI Designer and Programmer.

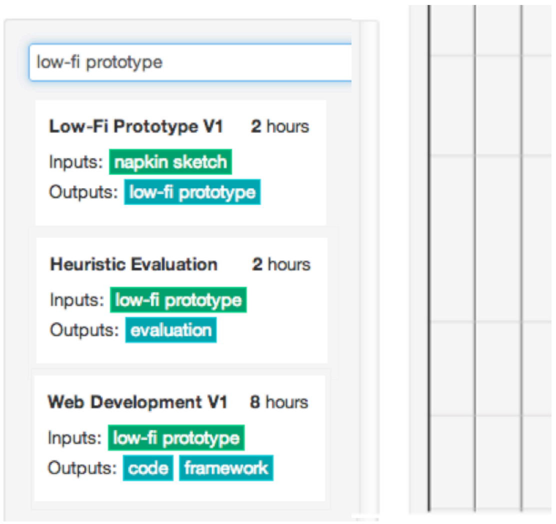
low-fi prototype x Add input

heuristic evaluation x Add output

Figure 5. Editing the details of a "heuristic evaluation"

inputs, and outputs. A requester may search over the name, inputs, and outputs to use directly or just to get a sense of the space (e.g., what is accomplishable in a given time frame, what direction can the team head in next) (Figure 6).

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. Requesters click to draw handoffs between non-overlapping tasks and collaborations between overlapping tasks. When the team is complete, the requester can begin to recruit and hire experts to fulfill the various team roles. In the hiring, the requester can use the timeline and team role descriptions



low-fi prototype

Low-Fi Prototype V1 2 hours
Inputs: napkin sketch
Outputs: low-fi prototype

Heuristic Evaluation 2 hours
Inputs: low-fi prototype
Outputs: evaluation

Web Development V1 8 hours
Inputs: low-fi prototype
Outputs: code framework

Figure 6. Search the event library for "low-fi prototype"

During runtime, the event library is hidden and replaced with a chat interface for the team and the requester. After the requester has completed hiring they send a link to the team to each team member so that they can log in. After the requester clicks "Start", progress is tracked through a red ticker; tasks that are currently live are highlighted as are upcoming tasks for any given user. 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 and hit complete on the task. Foundry indicates when a block is completed early, moving up the start time of downstream blocks and vise versa for delayed tasks (Figure 7).

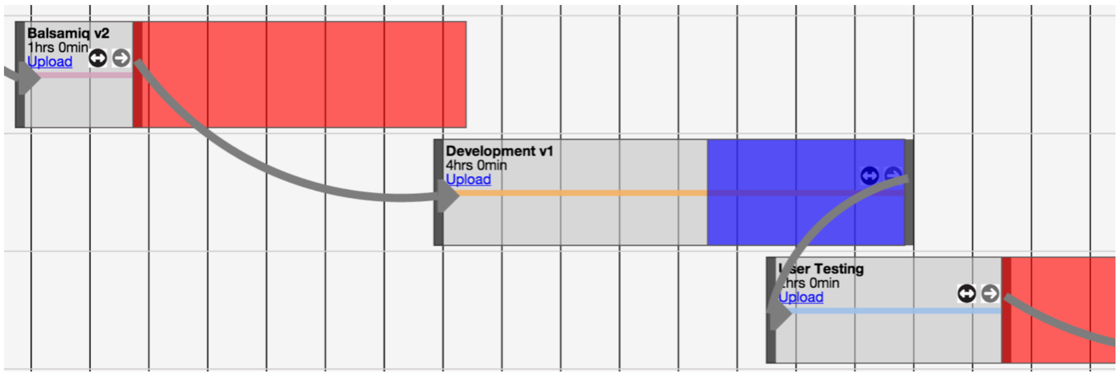


Figure 7. Task 1 is delayed, indicated by red, shifting back task 2. Task 2 completed early, indicated by blue, and shifted up the start time of Task 3, which also became delayed.

IV. EVALUATION

Methods

The purpose of the Foundry study was to evaluate whether the authoring interface is easy and accessible for the general population. In particular, we were interested in seeing how quickly people were able to pick up on the concept of a flash team with the help of the Foundry authoring interface and how comprehensive and realistic their teams would turn out to be.

To evaluate Foundry's authoring interface, we conducted a study of 23 participants (15 female, ages 18-43, mean age 23.6). The study was broken into two rounds to evaluate Foundry's use early and implement any necessary improvements. Round one included 13 subjects (7 female, mean age 22.7 years); round two included 10 subjects (8 female, mean age 24.6). The version of Foundry in round one did not include the event library - that feature was introduced in round two of the study to improve user behavior. Subjects were recruited on a university campus primarily through email lists and were given $15 monetary compensation for the hour-long study (although most participants finished in under 40 minutes).

Users were introduced to the concept of flash teams and the Foundry authoring system through an onboarding experience that encompassed a walkthrough of the web application prototype team (Figure 2) as well as a guided completion of a partially-finished team. During the walkthrough, participants were shown team roles and their specified skill details as well as how to interface with the timeline (create events, add inputs/outputs to events, change durations of events, add handoffs to events, and create collaborations between two or more events). To verify that participants understood the functionality of the Foundry authoring system, participants were asked to complete a partially-finished team for creating a lesson for a massively open online course (MOOC) (Figure 8). Participants had to add an additional team role to the team, create an event block, draw a handoff from one event block to another event block, and draw a collaboration between two events blocks.

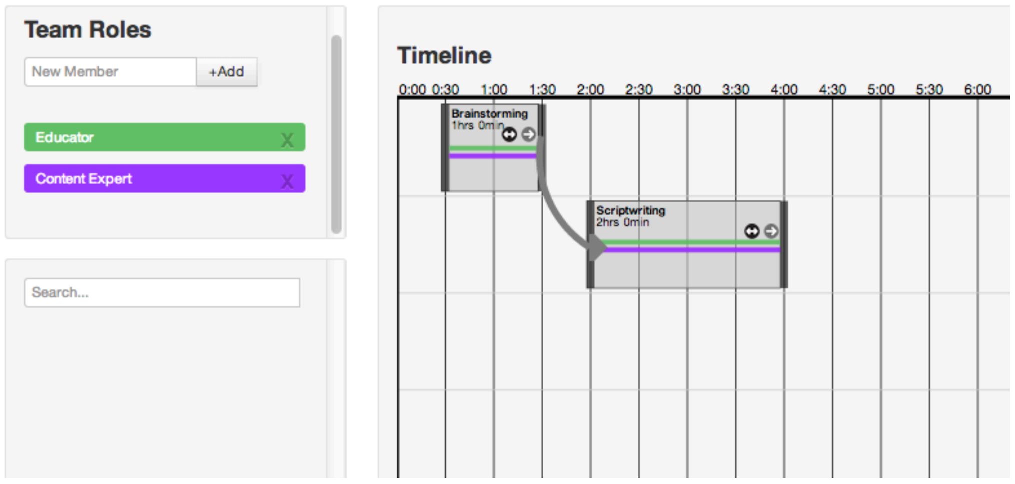


Figure 7. A partially complete MOOC team

After the onboarding experience, participants were asked to create a team that could create an interactive children's eBook. Users were reminded to add team roles, task-based events, and that they could utilize handoffs and collaborations. While working on the team, users were encouraged to think aloud.

In order to evaluate the information that we gathered from our participant studies, we identified several criteria that are critical to creating flash teams. In early studies of flash teams, these were identified as controlling variables that, when not used, result in the poor outcomes discussed when teams of expert crowd workers are not coordinated. Participants were given a score of 1 if they met the requirements of the parameter and received a score of 0 otherwise (boolean evaluation). The final set of criteria consisted of:

1. *Meaningful roles* - roles names are distinct (not counting cases where there were multiple *instances* of the same role, e.g. two photographers) and more descriptive than just "worker 1", and reflect on the role that these experts play in the flash team.

2. *Decomposed roles* - no roles such as a pan-functional member or a member that function as whole teams in and of themselves (e.g., "marketing person" vs. "social media specialist," "marketing director," and "marketing analyst")

3. *Decomposed events* - no events encompassing tasks that would be better represented as separate, more specific events. The threshold for this metric is whether the event decomposition is sufficient to indicate what must happen within the event. (e.g., "create eBook" vs. "add text to eBook," "add interactive widgets," "add illustrations to eBook")

4. *No use of default duration* - no group of events are all of the same duration without any reasoning behind it. This may be clear in the structure of the team itself, where every task is the default duration of one hour, or determined via researcher observation and participant verbalization.

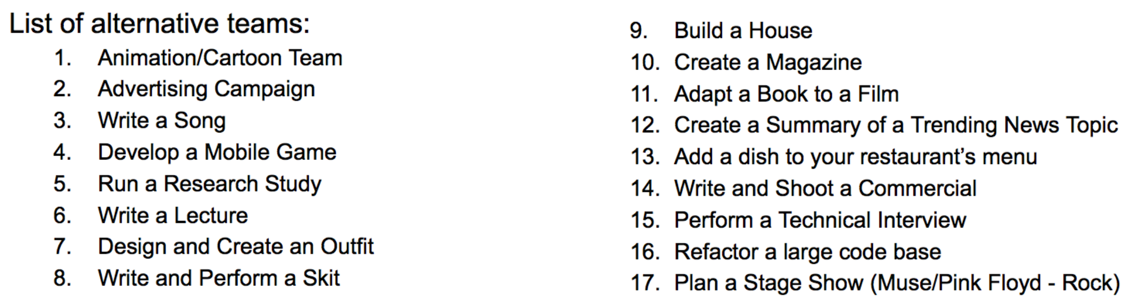
5. *Assigned members to tasks* - all intentional tasks have members assigned to them. Intentional tasks are defined as those which have been configured in some way and not a default "New Event."

6. *Presence of inputs/outputs or handoffs* - all intentional tasks have an input/incoming handoff and/or an output/outgoing handoff where appropriate; there are no unconnected workflows.

7. *Use of the event library* - the participant used the event library in such a way that demonstrated understanding of not only the existence of the event library, but also its function in some way. At minimum, the user typed a query into the search bar. (This metric was only used for the second round, TBD)

We hypothesized that users would be able to successfully creating meaningful and decomposed roles, decomposed events, and assign members to tasks. We expected users to not use the default duration, but we did expect them to have difficulty when coming up with duration. We did not expect to see much use of the inputs/outputs or handoffs.

For their second flash team, participants chose from a selection of twenty different objectives that ranged over a wider variety of potential projects and interests (Figure 8). Participants were once again reminded about the functionality that was provided by the Foundry authoring system and encouraged to verbally explain their thought process while going along.


List of alternative teams:

1. Animation/Cartoon Team
2. Advertising Campaign
3. Write a Song
4. Develop a Mobile Game
5. Run a Research Study
6. Write a Lecture
7. Design and Create an Outfit
8. Write and Perform a Skit
9. Build a House
10. Create a Magazine
11. Adapt a Book to a Film
12. Create a Summary of a Trending News Topic
13. Add a dish to your restaurant's menu
14. Write and Shoot a Commercial
15. Perform a Technical Interview
16. Refactor a large code base
17. Plan a Stage Show (Muse/Pink Floyd - Rock)

Figure 8. The list of alternative teams for users to choose from

The purpose of the alternative teams was to examine the breadth of flash teams and explore different uses of the teams. Additionally we wanted to observe behavior in participants when they were able to choose a team that was closer to their own interests.

Results

The results from our evaluation suggest that Foundry enabled users to create decomposed and meaningful team roles. Furthermore, after an addition to the system, Foundry successfully guided users to create decomposed and pipelined workflows.

In our initial results, we found that, as expected, most users were able to create meaningful and decomposed roles and assign members to tasks. Unexpectedly, users struggled with decomposing events as well as avoiding the default duration (Table 1). Users also used the handoffs more than expected.

Parameter	Round 1 (n=13)	Round 2 (n=10)
Meaningful roles	1.000	1.000
Decomposed roles	0.923	0.900
Decomposed events	0.615	0.900
No use of default duration for some stretch	0.769	1.000
Assigned members to tasks	0.923	1.000
Inputs/outputs or handoffs	0.923	0.900
Use of event library	N/A	0.500
Average Time to Create Team	9.9 min	14.24 min

Table 1: Aggregate participant scores across the seven above parameters for rounds one and two. In round one, participants struggled primarily with decomposing events and avoiding the default task duration. This was improved on in round two, but was not a statistically significant change.

After around half of the participants had completed the study, we paused to evaluate the trend we had seen above. In particular, we were surprised at the difficulty a number of participants had with decomposing events - something that we believed to be critical to a flash team. Without decomposed events, the workers will not have the necessary structure and we could imagine that they would fall into those coordination problems we have seen in teams of expert crowd workers who do not use flash teams.

Thus, in what we refer to as Round Two of the study, we implemented the event library. The event library allowed users to search over inputs, outputs, and titles of events from teams either we had used in the past or events that we created with the help of interviewed online experts.

The introduction of the event library seems to have had a positive impact on the decomposition of events. It is not captured in this table, but another noteworthy behavior we saw was that in round one, most users used handoffs only, but in round two, there was a mix of either handoffs or input/outputs, but rarely both. Additionally, users took longer in creating teams in round two. We had not originally planned to measure how long it took to create a team, but the change in time was noticeable.

Authoring Process & Types of Teams

In general, most participants first defined members of their team, then created events that these members could be assigned to. It was rare for a participant to add new members as they realized that a new member was required after creating an event. Participants often started on a new row for each new event, even though events can be put one after another in a linear fashion (Figure 10). A total of five rows were available on the timeline for events. Some participants attempted to scroll in a vertical direction to find additional event rows, then were told that such rows did not exist.

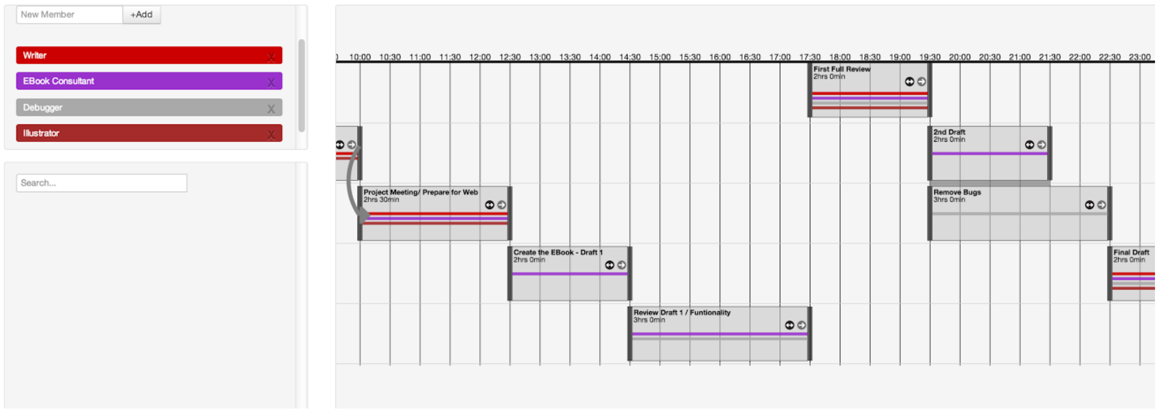


Figure 10. Example of a user's cascading event organization, although there is no meaning attributed to the rows on the Foundry timeline, this cascading style was seen across many participants.

Most understood that handoffs represented a transfer of information from one event block to another, and made sure that event blocks were logically linked to each other. However, in round one, no one used the fields for inputs and outputs and in round two, inputs/outputs were used by three of ten participants. Although all participants were introduced to the concept of collaborations between event blocks, less than one fourth of participants in both rounds did not utilize the collaboration functionality, even when they had events that occurred in parallel.

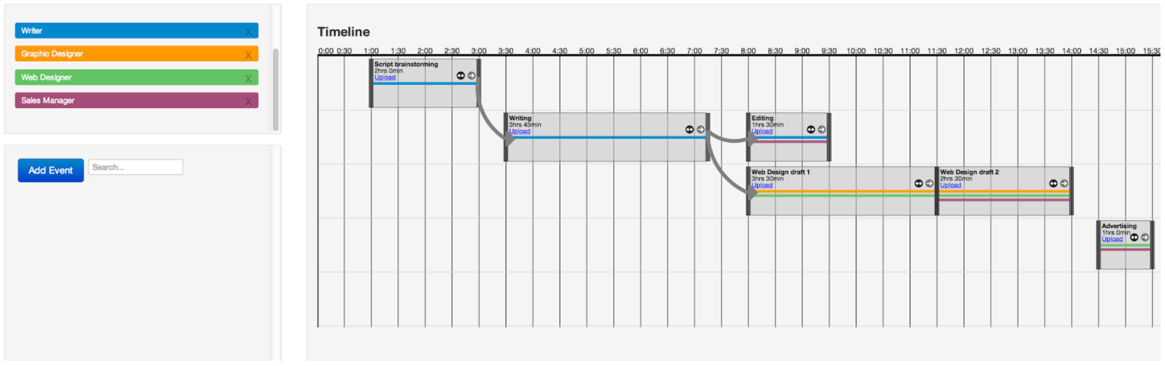


Figure 11. An example of 30-minute breaks purely for visibility, while none of the participants indicated that they wanted the team to have a 30-minute grace period, over half of them added similar spacing - indicating that Foundry does not currently handle spacing/visibility well.

The most common duration between events was thirty minutes. Many participants reported putting this gap in between events for readability purposes, and not necessarily because a break of thirty minutes was necessary between events (Figure 10).

In round two, half of the participants showed awareness of the event library's functionality, at minimum typing something in the search bar. One subject made extensive use of the event library, searching the library first for any event she wished to create, and only creating one from scratch if there was no match in the library.

Participants generally expressed more enthusiasm and excitement while creating their alternative team, as the objective was often more aligned to their personal interests, and they were often able to go into more detail with the event blocks than with the children's eBook team.

Usage

Think aloud feedback and behavior during studies gave us insight into the way participants use Foundry. Feedback was generally positive that the system was new and its use understandable and interesting. However, we did see some outlier behavior that indicated that interactivity should expand and there were a few requests for different features.

Notable outlier behavior included one participant placing member names in event names (e.g. "Writer and Illustrator generate content draft") in addition to assigning them to the event. Another participant, in round two, always used events from the event library unless it did not contain the event she wanted, at which point she created a new event. She expressed unfamiliarity with eBook creation as well as the usefulness of the event library multiple times. Another participant from round two created only two handoffs for a total of 8 events, but created collaborations for every in-parallel event. This participant expressed frustration because she believed that overlapping events made for smoother transitions, but then was not able to create handoffs, which Foundry allows only for non-overlapping events. Similarly, a few other participants express frustration that they could not handoff to the middle of an event, though asking for new functionality was a relatively rare event. Other requests included the ability to author cyclical or iterative workflows, and a more obvious way to represent groups of workers who serve the same purpose.

Many users gave positive feedback about the system during or after the study, and at least two participants asked if we would be releasing the platform soon for public use. In the older half of participants in particular (graduate students and other) there was particular enthusiasm for the system and how it might bolster their own work without eating up their time by requiring them to constantly manage the team.

Users were also much more excited when choosing their own alternative team. Many users expressed the sentiment, "oh this team looks fun," and were more engaged with the system on a more creative project.

Runtime Evaluation

We have also evaluated flash teams using Foundry's runtime mode in a controlled study. The details are available in an anonymized submission to UIST 2014, but for proof-of-concept a summary of the method and results are below.

Method

We recruited six napkin sketch design teams from oDesk by requesting workers with expertise in UI design, UX research, and web application development. We accepted applicants with high rating and conducting brief filtering interviews. We randomized these workers into groups and tasked them with creating a simple task manager for party planning that would allow the end user 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 teams were assigned to either the flash team condition (i.e., Foundry with a fully decomposed workflow with coordinations support via handoffs and notifications) or the control condition (i.e., Foundry but no decomposed workflow or coordination support), resulting in a total of three teams in each condition.

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.

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. The control teams took longer than the flash teams to fulfill the tasks for each role, as well. The control teams spent 2.4x the time on UI Design, 1.89x the time on UX Research and 1.42x the time on Development.

By following the sequence of modular tasks on Foundry, flash teams followed the iterative design process more closely than the teams in the control condition. 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.

V. DISCUSSION

Despite overall support for Foundry, we also observed trends in behavior that indicate areas of potential improvement. In round one, we noted that several participants exhibited defaulting behavior regarding event duration, where a series of events have the same duration, usually the one hour events are assigned upon creation, apparently for no meaningful reason. This may indicate that during the authoring process in isolation (i.e. when the running of the team is not actively considered) the timeline is not viewed as a meaningful measure of time, only a relative ordering. There was another behavior we noted which supports this hypothesis: users did not appear to consider the gaps between events meaningful, for example accidentally dragging an event horizontally and thus changing the between-events gap, but making no move to rectify the change unless it altered the overlap or event ordering. Without considering the runtime feasibility of event durations, we had at least hoped to see more evidence of planning and intentional duration assignment.

We note that data from round two indicates no incidence of the defaulting behavior, which was unanticipated. Given that the increase in default-duration avoidance was just shy of significant (p=0.0536) and that only half of participants looked at the event library, it is not clearly related to change in behavior. It may be that the additional mention of duration in the training (mentioning the fact that events in the library having specified duration) impacts the saliency of duration to the subjects, but at present that is merely speculation.

While overall we see that very few subjects did not add either input and output, or handoffs between all events where applicable, we also note that often creating handoffs are chosen over specifying input and output explicitly. This may be because subjects view them as fulfilling the same capacity and therefore an either-or choice. However, unless at least one of a) *output of the first event involved in a handoff* or b) *input of the second event* are specified, the handoff provides no useful information about what the members on one event must produce and what the members on the other team may expect to receive. Furthermore when an event hands off to more than one other event, output subsets may be specific to each handoff; only creating handoffs is in this case ambiguous. This disconnect is bad, especially for projects where the handoffs are not strictly linear. This indicates that the inputs/outputs and handoffs need to be better linked visually within our system.

The visual nature of the handoff as opposed to text specification of inputs and outputs may also be a factor in choosing the former over the latter--particularly in such a visually based authoring environment. It is also worth noting that the inputs and outputs are located in the event popover, so users must first click the event, scroll to the bottom of the popover, type inputs and outputs, pressing the enter key after each one, and then hit save. By contrast, handoffs are created by clicking an arrow visible on the originating event in the timeline, and then clicking the body of the other event.

All subjects decomposed member roles, and all except one subject assigned members to each event. This is promising, although some subjects expressed uncertainty about what certain roles actually did, despite being familiar with the role's name. The selection of role specifics for hiring purposes happens through category and sub-category drop-down menus, so this "name recognition" ability may be sufficient to author a functional team. However, it is difficult to determine this from our studies because those menus are not yet comprehensive and so we cannot draw any conclusions from the cases in which they were or were not used.

An individual's domain knowledge likely contributed to variance across behavior, and in particular, event decomposition. Event decomposition varied over participants but improved in the second round; five out of thirteen participants in the first round and one out of ten in the second round failed to do so (Table 1). There were some instances of events such as "coding stuff" and "make eBook" which, while decomposed, also represented instances where they might have been either split into sequences of events or better specified. Without domain knowledge, it is difficult to decompose to a level that is informative for the workers - leading to potentially bad teams. Here, the event library may help by providing access to the event space, although a user must at least know enough to specify a partial search term. We postulate that the dearth of collaborations created also indicates a type of suboptimal event decomposition where synchronous tasks were grouped into a single event instead of multiple events in parallel, and all members assigned to that single event.

The scarcity of participants who showed awareness of the event library's functionality may be due to the event library's appearance as an empty search bar until something is typed in. The one participant who almost exclusively used the library may be a point in its favor, but also highlights potential problems. The user did not alter any of the events taken from the library except to assign members, and used two events which may not have been suitable out-of-the-box. If this is symptomatic of general user behavior with the event library, the library may need to be more specific about the contexts for which events are and are not suitable, or have strict curation to minimize negative effect if users do not customize. Otherwise it could introduce a new set of "defaulting" behavior issues.

Apart from considering how participants scored on binary measures, we also looked at how they compare to one another. Event decomposition varied widely although member decomposition was generally similar. Every team had a writer or author and a programmer or developer, but some participants considered marketing or deployment as well and decomposed events and added members accordingly. There was also variation in the level of member decomposition (e.g. "Programmer" versus "Backend Developer", "iOS Developer", and "UI Designer"), and often more in a certain area (writing, art, software). This is likely related to aforementioned domain familiarity differences.

Limitations

During the latter portion of the second round, we were more aggressive about recruitment, soliciting participants more directly in dormitories and public locations to participate in the study (vs. broad email lists). All participants come from the same pool and there does not appear to be any significant difference between users who were passively recruited and those who were actively recruited.

Another factor we could not control for was how well individuals understood the context of the study and concept of flash teams. Our goal was to evaluate how non-experts in the task of creating an eBook would create a team for said task. We met this goal in that none of the participants expressed that they had experience creating an eBook in the past, although they had varying familiarity with relevant components (e.g., creative writing, programming). On the other hand, while most were familiar with crowdsourcing, the difference in familiarity between those who were and those who weren't was quite large.

We realize that the study conditions cannot perfectly replicate a user. All participants were recruited from one university, and therefore are not representative of all potential project requesters. Participants are recruited and given set tasks, and the fact that participants expressed more excitement for the alternative tasks where they had more freedom highlights the fact that conditions would be different when the user is someone who comes to the platform with their own goal, rather than one who is shown the platform and given a goal. Given that we are interested in authorship, internal motivation and investment is important to consider. Moreover, we recognize that authorship of a real team that is intended to be run has an additional set of concerns such as cost, time, and quality.

Like most studies, there is also a question of observer effect, especially since our studies were conducted in person, which may cause the participants to behave differently than in a natural situation. Participants may try to please the researcher or creating teams in a way they think is expected of them. They may also feel pressure to avoid appearing ignorant and either give indication during training that they understand when they do not, or be cautious in using the interface and simply avoid functions they are not sure of. Some participants apologized for using the interface "wrong," or asked for researcher input on authoring decisions, signalling the former. Researchers strived to remain neutral, instructing subjects to use their judgment. Almost all participants made use of all of Foundry's authoring capabilities, with the exception of the event library and collaborations, so there is not strong evidence for the latter case.

VI. FUTURE WORK

Although non-experts seem enabled to create flash teams with Foundry, it is clear that they will have more stable outcomes if they use pre-existing teams. Accordingly, in our future work we plan to build out the existing library of complete teams for users. This will simultaneously build out the event library as well, giving users more guidance and options when creating their own teams.
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 oDesk to pre-clear a set of workers who are high quality and available at the desired time. However, platforms such as oDesk 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.

Relatedly, we also envision a future where the hiring process can take advantage of the global workforce in automated ways. Although flash teams use pipelined workflows, work is constrained by the typical 8-hour workday. Because Foundry handles most managerial effort, we could envision handoffs happening around the globe, "following the sun," so projects can be completed even faster.

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.

VIII. CONCLUSION

Foundry enables end-users to utilize the flash teams methodology for coordinating teams of expert crowd workers through an interactive online platform for team authoring and project runtime. Specifically, Foundry provides an interface for flash team creation, offering the abilities to define members and events, assign members to events, situate events on a timeline, and relate events to one another through inputs and outputs, handoffs, and collaboration, all in a way that is possible to grasp with a brief introduction. Non-experts displayed the ability to complete these tasks in line with flash teams methodology and expressed interest in the future of the platform. The addition of the event library had a moderate impact in encouraging more decomposition and computation, however we believe that further work in this direction could provide even more stability in team creation.

While our study showed that the system can improve in various ways (including visual spacing of events, encouraging use of inputs/outputs/handoffs, and assisting requesters with task duration), we have shown that anyone, even without management skills, can lead a team towards a complex and interdependent goal. While the micro task crowd is easily accessible, its utility is very low for the everyday population. Expert crowds provide easy access to a large body of diverse skill sets, and with the organization and computation of Foundry, non-experts can take advantage of this access to complete complex and interdependent projects. Where the average person would have needed to either contract with a company or hire project managers to complete their projects with potentially highly varying outcomes, we have opened the door for anyone to take advantage of the crowd. In the future, project requesters may be able to organize around a project idea or perhaps even crisis situation in a manner of hours and get people working together interdependently instantaneously.

REFERENCES

1. oDesk. URL www.odesk.com.
2. Ahmad, S., Battle, A., Malkani, Z., and Kamvar, S. The jabberwocky programming environment for structured social computing. *Proc. UIST '11*, 2011.
3. Antin, J. and Shaw, A. Social desirability bias and self-reports of motivation: a cross-cultural study of Amazon Mechanical Turk in the US and India. *Proc. CHI '12*, 2012.
4. Bernstein, M.S., et al. Soylent: a word processor with a crowd inside. *Proc. UIST '10*, 2010.
5. Bunderson, J.S. and Boumgarden, P. Structure and Learning in Self-Managed Teams: Why Bureaucratic Teams Can Be Better Learners. *Organization Science*, 21(3):609–624, 2010.
6. Chilton, L.B., et al. Cascade: crowdsourcing taxonomy creation. *Proc. CHI '13*, 2013.
7. Cooper, S., et al. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.
8. Cramton, C.D. The mutual knowledge problem and its consequences for dispersed collaboration. *Organization Science*, 12(3):346–371, 2001.
9. Cranshaw, J. and Kittur, A. The polymath project: lessons from a successful online collaboration in mathematics. *Proc. CHI '11*, 2011.
10. Dai, P., Mausam, and Weld, D. Decision-theoretic control of crowd-sourced workflows. *Proc. AAAI '10*, 2010.
11. Dourish, P. and Bellotti, V. Awareness and coordination in shared workspaces. *Proc. CSCW '92*, 1992.
12. Dow, S., Kulkarni, A., Klemmer, S., and Hartmann, B. Shepherding the crowd yields better work. *Proc. CSCW '12*, 2012.
13. Hinds, P., Liu, L., and Lyon, J. 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.
14. Huckman, R.S., Staats, B.R., and Upton, D.M. Team Familiarity, Role Experience, and Performance: Evidence from Indian Software Services. *Management Science*, 55(1):85–100, 2009.
15. Karduck, A. TeamBuilder: a CSCW tool for identifying expertise and team formation. *Computer Communications*, 17(11): 777-787*,* 1994.
16. Kittur, A., Khamkar, S., Andre ́, P., and Kraut, R.E. CrowdWeaver: visually managing complex crowd work. *Proc. CSCW '12*, 2012.
17. Kittur, A., Smus, B., Khamkar, S., and Kraut, R.E. Crowdforge: Crowdsourcing complex work. *Proc. UIST '11*, 2011.
18. Kittur, A., et al. The future of crowd work. *Proc. CSCW '13*, 2013.
19. Kokkalis, N., et al. Email Valet: Managing email overload through private, accountable crowdsourcing. *Proc. CSCW '13*, 2013.
20. Kulkarni, A., Can, M., and Hartmann, B. Collaboratively crowdsourcing workflows with turkomatic. *Proc. CSCW '12*, 2011.
21. Lasecki, W., et al. Real-time captioning by groups of non-experts. *Proc. UIST '12*, 2012.

22. Lasecki, W.S., et al. Real-time crowd control of existing interfaces. *Proc. UIST '11*, 2011.
23. Lasecki, W.S., et al. Chorus: A Crowd-Powered Conversational Assistant. *Proc. UIST '13*, 2013.
24. Lashinsky, A. *Inside Apple*. Hachette Book Group, New York, 2012.
25. Little, G., Chilton, L., Goldman, M., and Miller, R.C. Exploring iterative and parallel human computation processes. *Proc. HCOMP '10*, 2010.
26. Luther, K., Fiesler, C., and Bruckman, A. Redistributing leadership in online creative collaboration. *Proc. CSCW '13*, 2013.
27. Mathieu, J.E., et al. The influence of shared mental models on team process and performance. *Journal of applied psychology*, 85(2):273–83, 2000.
28. Nardi, B.A. and Whittaker, S. The Place of Face-to-Face Communication in Distributed Work. In P.J. Hinds and S. Kiesler (editors), *Distributed Work*, pp. 83–113. MIT Press, Cambridge, 2002.
29. Noronha, J., Hysen, E., Zhang, H., and Gajos, K.Z. Platemate: crowdsourcing nutritional analysis from food photographs. *Proc. UIST '11*, 2011.
30. Olson, G. and Olson, J. Distance Matters. *Human-Computer Interaction*, 15(2):139–178, 2000.
31. Stol, K.J. and Fitzgerald, B. Researching Crowdsourcing Software Development: Perspectives and Concerns. *ICSE '14 Workshop on Crowdsourcing in Software Engineering*, 2014.
32. Valentine, M. Team Scaffolds: How Minimal In-group Structures Support Fast-paced Teaming. *Academy of Management Proceedings*, 2012.
33. Yu, L. and Nickerson, J.V. Cooks or cobblers? Crowd creativity through combination. *Proc. CHI '11*, 2011.
34. Zhang, H., et al. Human computation tasks with global constraints. *Proc. CHI '12*, 2012.
35. Expert Crowdsourcing with Flash Teams. *Anonymized Submission to UIST '14,* 2014.