Orchestration Scripts: A System for Encoding an Organization's Ways of Working to Support Situated Work

Kapil Garg
Northwestern University
Evanston, IL, USA
kgarg@u.northwestern.edu

Darren Gergle
Northwestern University
Evanston, IL, USA
dgergle@northwestern.edu

Haoqi Zhang
Northwestern University
Evanston, IL, USA
hq@northwestern.edu

ABSTRACT

Ill-structured problems demand that people adopt sophisticated strategies for planning, seeking support, and using available resources along their work process. These practices involve a challenging monitoring and strategizing process that existing tools cannot support since they largely lack an understanding of an organization's processes, social structures, venues, and tools. We introduce workplace programming for situationally-aware systems–an approach for encoding work situations using computational abstractions of an organization's ways of working and surfacing support strategies at appropriate times and settings. With this approach, we implement Orchestration Scripts, a system that supports various situated work activities in a socio-technical organization. Through a case study and field study, we show how our approach encodes different aspects of working effectively and helps people identify situations to enact effective strategies using the available support opportunities. Our results show how a programmable technology can provide situated support in today's workplaces.

CCS CONCEPTS

• **Human-centered computing → Interactive systems and tools**.

KEYWORDS

Socio-Technical Ecosystems; Situated Work; Ill-Structured Problems; Orchestration Scripts; Organizational Objects

ACM Reference Format:
Kapil Garg, Darren Gergle, and Haoqi Zhang. 2023. Orchestration Scripts: A System for Encoding an Organization's Ways of Working to Support Situated Work. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3544548.3581456

1 INTRODUCTION

Modern workplaces increasingly require their workers to solve complex, ill-structured problems with vaguely structured goals and constraints, and often multiple solutions [37]. Working on these problems involves continually assessing how needs and constraints are changing, and soliciting support from others in the organization [36]. In response, workplaces have adopted Agile work processes that encourage rapid iteration over waterfall models [19, 63, 65], social structures for collaboration [48] and dynamic teaming across an organization [27, 59], and a shared suite of tools that support collaboration activities [1, 5, 8]. Alongside, an increased emphasis is placed on learning and promoting effective ways of working (e.g., strategies for planning work; collaborating with peers) in these workplaces [36] and also in learning communities teaching students how to do complex work (e.g., for undergraduate research training [75] and design inquiry [42, 54]).

Being effective in these workplaces requires workers to be aware of their changing *work situations and needs* and the *situated strategies* they can enact themselves or with others across diverse venues in the workplace. For example, new workers may meet weekly with their onboarding mentor to ask questions about the organization's work practices as they onboard; team leaders can ask their team to bring up issues on deliverable scoping during their weekly planning meetings. However, workers struggle with monitoring for these work situations, strategizing on how to resolve them, and enacting those strategies across support opportunities in the workplace [31]. Coaching from experts can help, but it is infeasible as organizations grow since the distributed nature of the work prevents coaches from observing and scaffolding peoples' situated interactions across interactions in the ecosystem where they are not present.

Given the challenges in human-driven solutions for situated work, we may consider technology that supports workers in *identifying their current work needs and potential strategies they can enact in a work ecosystem.* However, existing systems for detecting work situations and enacting follow-up actions [4, 7, 10, 55] cannot encode situated work practices effectively because their programming models lack the constructs to encode practices that span the work processes, social relationships, and collaboration venues of a workplace. For example, a worker wants to raise a planning-related issue at their next team planning meeting but must currently encode this as a specific time (e.g., 3:00 PM this Wednesday) rather than a relevant *situation or venue* where it should come up (e.g., the team's planning meeting). Moreover, the automation-focused nature of these approaches can be overly prescriptive on how to resolve work situations when they lack a complete picture of the work situation [50, 61]. For example, a team may not have tested a prototype because users were unavailable or because they encountered programming bugs, each requiring different strategies to resolve. In short, we need systems that support situated work since it is too challenging to do with human-only approaches but require them to be flexible to the uncertainty present in day-to-day work.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CHI '23, April 23–28, 2023, Hamburg, Germany
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9421-5/23/04…$15.00
https://doi.org/10.1145/3544548.3581456

The core conceptual idea in this paper is *workplace programming for situationally-aware systems*, whereby providing systems with computational abstractions of an organization's ways of working allows for encoding strategies that inform workers about emerging work situations and specific work strategies to attempt along their work process. Unlike fully automated approaches, our approach provides workers with *signals* of emerging work situations and *suggested strategies* they can attempt on their own or with others as support opportunities become available in the broader work ecosystem. For example, team leads can use these constructs to inform them of when their team does not have planned deliverables during their weekly planning meeting, enabling them to discuss what the obstacles are (e.g., not knowing what to work on next; difficulty scoping) and suggest strategies, resource guides, or peers that can help. We hypothesize this form of human-machine orchestrated support—as opposed to full machine automation—is effective for ill-structured work because it lets workers complimentary use the benefits of machines for tracking known signals of when particular work needs require their attention and how to resolve them, but ultimately gives them complete control over how to proceed rather than having the machine automatically enact a follow-up action.

We implement this approach in *Orchestration Scripts*, a system for encoding and enacting *Situated Scripts* that detects everyday work situations in a workplace and recommends strategies to attempt during workers' interactions with each other. A key component of this system is *Organizational Objects* that provide programming abstractions for the work processes, social relationships, venues, and tools in a workplace. This allows us to encode strategies that reference the particular people for and the situations in which they should be enacted in a general manner (i.e., not encoding a specific person or time). These include suggesting newcomers discuss effective work processes with their onboarding mentor after weekly sprints, or providing reflection activities on how to prevent overworking to teams during planning meetings with a team lead. These objects are made executable through a *Studio API* that provides programmatic access to data for the abstractions. Finally, an *Orchestration Execution Engine* is responsible for executing general scripts for specific people by monitoring for work situations to emerge and presenting situated strategies at relevant venues for each person across the ecosystem. Through studies in a research learning community [75], we show how Orchestration Scripts can effectively encode situated work practices in a general manner that is less brittle to changes in people's relationships and venues than workflow automation tools while still being tailored to specific people and their circumstances; and how it creates opportunities to discuss effective strategies for emerging work situations across a work ecosystem. These findings show how programmable technologies can flexibly provide situated support in today's socio-technical workplaces.

2 BACKGROUND

We are interested in developing systems that support workers in enacting situated work practices in a socio-technical work ecosystem. We first describe what these work environments and practices look like, and then review the limitations of current technology in facilitating situated work.

2.1 Ill-Structured Work and How Socio-Technical Workplaces Support It

Modern work and learning communities require their members to tackle complex, ill-structured problems, such as in research, engineering, design, and entrepreneurship work. These problems are challenging due to the uncertainty involved when working on them, where work needs change as progress is made and resources from around the organization must be coordinated to resolve them [37, 58]. Working effectively on these problems requires developing *situated work practices* where workers determine a course of action by assessing their work needs and what forms of support are available, adapting their plans as their needs and the availability of resources change over time [62].

Emerging needs often require support from others when workers do not have the resources or skills to address them on their own [30, 36]. In response, workplaces have focused on providing increased access to support opportunities, including peers, collaboration venues, and resources [35, 41]. This includes shifting away from strict hierarchies and siloed teams towards more networked models of organization [52, 59]; adopting more flexible work processes (e.g., Agile practices that encourage rapid iteration versus waterfall ones [19, 63, 65]); and work and collaboration tools (e.g., Asana [1]; Jira [5]; Trello [8]) that support identifying work needs and collaborating across social structures in the organization [27].

Despite increased access to support, enacting effective situated work practices remains challenging [31]. Workers must continually monitor for emerging work situations; identify and plan relevant work strategies; and enact those strategies across interactions along the work process, re-planning and re-strategizing as progress is made. However, they struggle with monitoring for their needs [30]; determining relevant strategies in those situations [17, 42]; and knowing when and where support across the organization can be found [11, 12]. Personalized coaching from mentors can help [20, 70], but it becomes infeasible since people's work practices are distributed across situated interactions throughout the ecosystem. Given the challenges with human-only solutions, we must consider how technology can support situated work by understanding the *work situations* people encounter and surface *situated strategies* they can attempt at relevant times along the work process.

2.2 Tools to Support Common Work Activities

Unfortunately, many existing workplace tools leave the difficult task of actually orchestrating situated work practices across an ecosystem entirely to the user. While these tools support workers across different work activities (e.g., planning [1, 5, 8], help-seeking [13, 14, 34, 44, 45], and reflection [53]) and in particular venues (e.g., collaboration activities in the classroom [25, 28, 38, 47, 72]) by helping them strategize on how to resolve known work needs (e.g., guides for assessing project risks when struggling with planning [42]), they provide little support for monitoring emerging needs in the first place. Further, they do not help enact relevant strategies in the appropriate support opportunities across the work ecosystem (e.g., at a venue dedicated to planning). Instead, our work aims to develop systems that support these facets of orchestrating work and provide relevant strategies to practice.

2.3 Automated Approaches That Detect Work Needs and Facilitate Work Processes

Closer to this goal, significant work has studied workflow automation approaches for business processes in organizations [32, 46, 68, 73], and trigger-action programming systems where actions are contextually triggered based on a monitored condition (e.g., for online governance [74], context-aware computing [56, 66, 69], workplaces [7, 10, 55], and more [4, 67]). These approaches encode data-driven workflows where a sequence of actions is initiated across disparate systems when data is added, modified, or matches a set condition. For example, when a worker has too many tasks assigned to them on a Trello board during a work sprint, a workflow can message the team over Slack to discuss how to triage tasks at the start of their planning meeting stored in Google Calendar.

However, providing programming primitives at this level limits our ability to encode situated work practices. A key challenge is a difference in abstraction between what workers want to express about effective work practices and the programming model workflow tools use. For example, we may want to say, "when a worker is overcommitted based on the number of tasks, inform their team at the next planning meeting." Instead, we must express the specifics of how to enact the script to the system (e.g., the worker's relevant board and tasks, their planning meeting time, and their team's communication channel) *for each individual worker* rather than the general strategy that applies to all of them. Not only is this level of encoding removed from how we think about situated work practices, but it also makes the workflows brittle to changes (e.g., a different team manager or planning venue time) since a script author must manually revise all workflows that reference these data. Without technology that understands where and with whom work activities are happening (e.g., a planning or collaboration venue; teams and their leads), we will continue to struggle with developing software that lets us express the kinds of engagement with others that we want workers to have along their work process.

Finally, a general challenge with any trigger-action-based system is its philosophy of execution. When the trigger condition is realized, the desired system behavior is to execute any follow-up actions assigned to the trigger. However, work processes in practice–especially for the ill-structured problems found in the workplace–are not easy to codify due to the day-to-day complexities and uncertainties of the work [36, 50, 60, 61]. As a result, these approaches tend to be overly prescriptive in how to resolve work needs and can fail if systems lack essential tacit knowledge workers have about their needs [33].

In our work, we consider how to develop workplace systems that use computational abstractions of an organization's ways of working to support situated work. We develop programming constructs that provide access to a workplace's work processes, social relationships, collaboration structures, and data from its productivity tools. Through these constructs, we can encode models of situated work that inform us of when work situations arise and potential strategies to resolve them along our work processes but leave the locus of control on how to what strategies to enact to the workers (rather than automated systems).

3 ORCHESTRATION SCRIPTS

We present *Orchestration Scripts*, a system that supports situated work by enacting *Situated Scripts* which encode work situations and strategies using computational abstractions of an organization's ways of working. We first present design goals for this system and then detail how we realize these goals in its implementation.

3.1 Design Goals

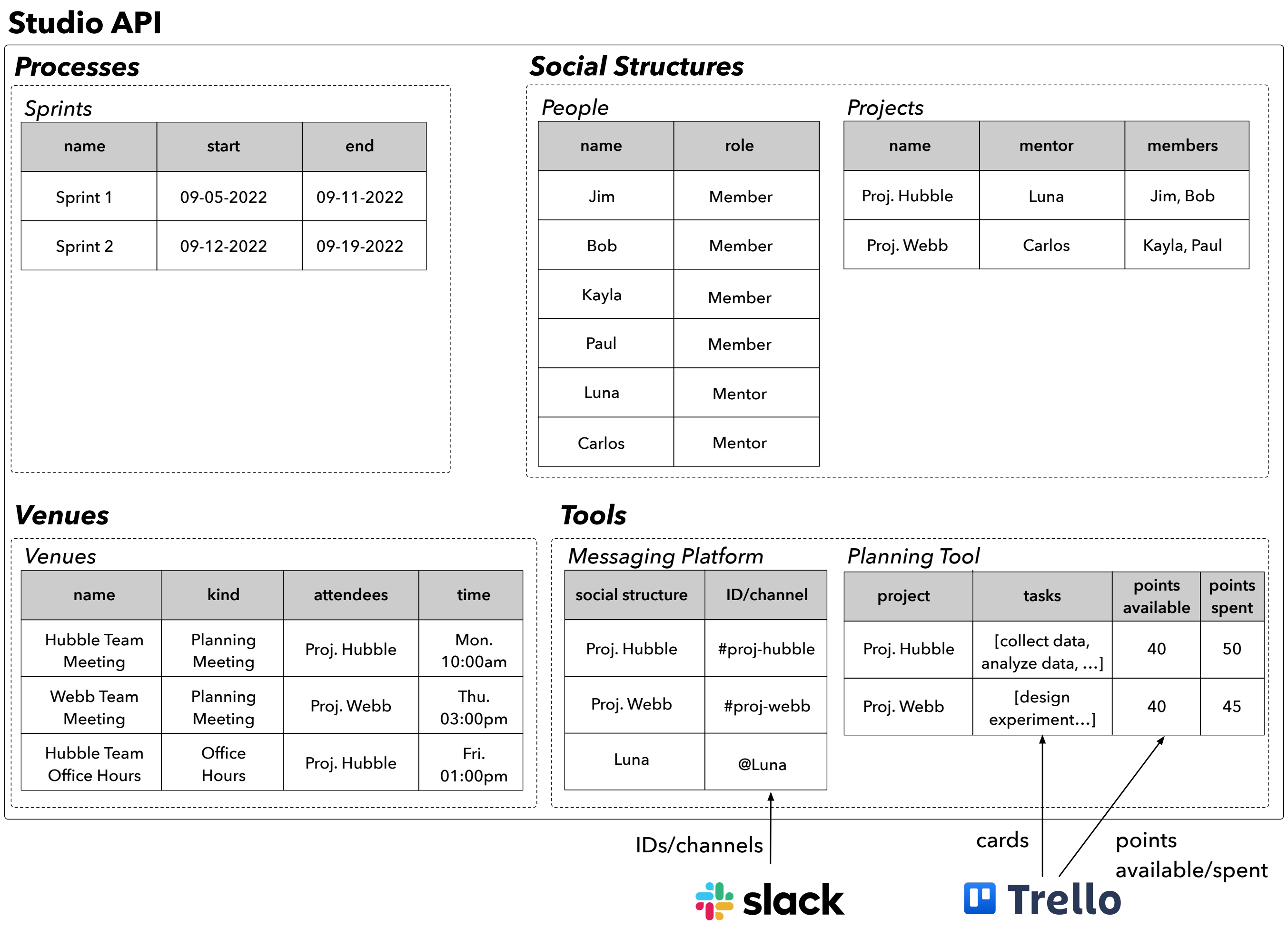
For systems to orchestrate situated activities to resolve work needs, they must model the work situations and relevant strategies to practice, and where to attempt them [31]. These activities occur along an organization's work processes, across social structures and collaboration venues, and are supported by various productivity tools. For example, a team may recognize a planning-related issue during a work session and bring that into a weekly planning meeting with their mentor later to address it. This suggests that modeling these situated activities requires that systems understand these ways of working. Therefore, our first design goal is to **enable the encoding of situated strategies using computational constructs of how work is done in an organization.**

Moreover, organizations often have shared practices and tools across people and teams, but the times and places where practices are enacted may differ. For example, two project teams follow the same planning practices where planning-related work situations and strategies should be raised at planning venues, but these venues occur with different team leads on different days. This suggests that scripts modeling general work situations and strategies relevant to the organization as a whole are useful but must be enacted *specifically* for each worker's individual context (i.e., present relevant planning strategies at *their* planning session to *their* team and lead). Therefore, our second design goal is to **allow for the expression of general situated work practices that become tailored to different people's specific situations.**

Finally, enacting situated activities requires systems to continually monitor for *work situations* to occur and present *situated strategies* at appropriate places and times (i.e., at specific venues along the work process). However, the moments when work situations arise and when situated strategies should be presented are often different. For instance, a team may want to discuss planning-related issues during their weekly planning session rather than at other times during the week. Moreover, these venues occur at different times for each worker or team in the organization. This suggests that an engine for supporting situated work needs to **track work situations and enact support strategies at relevant points of the work process separately for each worker or team**.

3.2 Encoding Situated Strategies in Terms of an Organization's Ways of Working

To encode programs that support situated work, we introduce *Organizational Objects* that provide computational abstractions of an organization's ways of working. To create this object, we included base-level constructs for work processes, social structures, venues, and productivity tools. These constructs can be used to generally model how work is done in an organization and customized to each specific organization. For example, an Agile workplace's



Sprints

name	start	end
Sprint 1	09-05-2022	09-11-2022
Sprint 2	09-12-2022	09-19-2022

People

name	role
Jim	Member
Bob	Member
Kayla	Member
Paul	Member
Luna	Mentor
Carlos	Mentor

Projects

name	mentor	members
Proj. Hubble	Luna	Jim, Bob
Proj. Webb	Carlos	Kayla, Paul

Venues

name	kind	attendees	time
Hubble Team Meeting	Planning Meeting	Proj. Hubble	Mon. 10:00am
Webb Team Meeting	Planning Meeting	Proj. Webb	Thu. 03:00pm
Hubble Team Office Hours	Office Hours	Proj. Hubble	Fri. 01:00pm

Messaging Platform

social structure	ID/channel
Proj. Hubble	#proj-hubble
Proj. Webb	#proj-webb
Luna	@Luna

Planning Tool

project	tasks	points available	points spent
Proj. Hubble	[collect data, analyze data, ...]	40	50
Proj. Webb	[design experiment...]	40	45

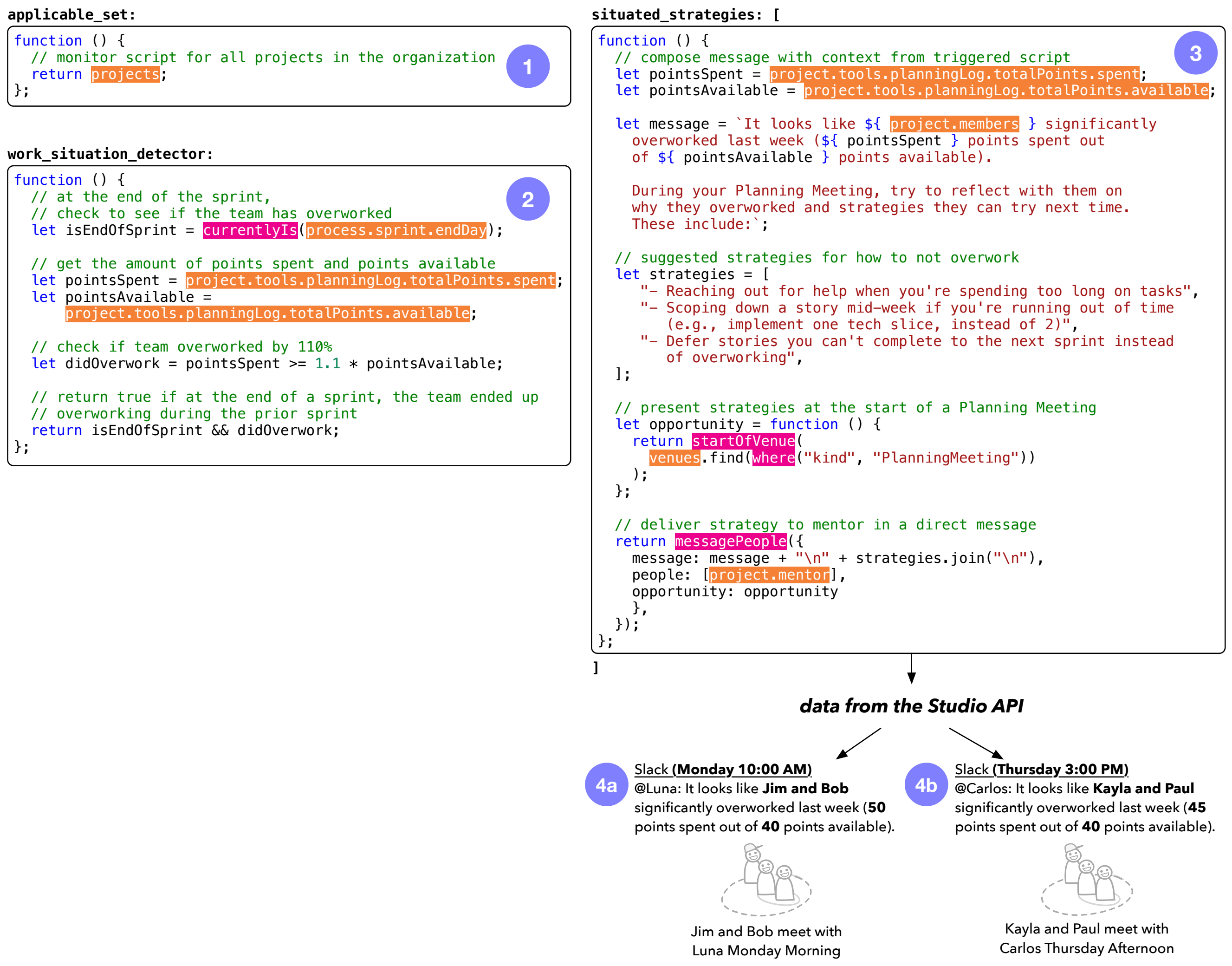
Figure 1: The Studio API provides programmatic access to data about an organization's processes, social structures, venues, and tools. It does this by storing information about their working processes, social relationships, and venues, and also collecting traces of traces of data from the organization's existing tools (e.g., Slack for communication; Trello for planning) that can be mapped to other information about the workplace (e.g., Trello cards or Slack channels for a project). In this way, the Studio API brings together all the data that corresponds to an organization's ways of working into a single space.

Organizational Objects may include sprint cycles for their work processes, teams and their leads for social structures, planning-specific and general office hours venues, and tools for Agile planning (e.g., Trello) and asynchronous communication (e.g., Slack) [19, 63]. In addition, we provide *Helper Functions* for common computations on the Organizational Objects, such as presenting a strategy the *morning before* a venue. Through these constructs, general situated work practices tied to an organization's ways of working can be expressed to software and surfaced at relevant times.

To operationalize Organizational Objects, we created a *Studio API* that provides a queryable instrumentation of the organization's ways of working; see Figure 1. It does this by collecting information about the organization's work processes, social relationships between members, collaboration venues, and traces of data from the organization's existing tools (e.g., Trello for planning purposes; Slack for asynchronous communication) into a single space. Moreover, these data can be mapped to each other (e.g., Trello board and Slack channel for a team), allowing us to query for different aspects of the relationships between people, their venues, and tools, such as when a project lead has their next office hours. This allows the general practices described using the Organizational Objects from earlier to become executable in organizations, allowing systems to coordinate situated work practices.

3.3 Expressing General Strategies That Become Tailored for Specific People

Using the Organizational Objects, *Situated Scripts* can be composed to externalize general *work situations* when support is needed and suggested *situated strategies* to resolve them. A Situated Script is defined by an object containing:



```
applicable_set:
function () {
  // monitor script for all projects in the organization
  return projects;
};
```

```
work_situation_detector:
function () {
  // at the end of the sprint,
  // check to see if the team has overworked
  let isEndOfSprint = currentlyIs(process.sprint.endDay);

  // get the amount of points spent and points available
  let pointsSpent = project.tools.planningLog.totalPoints.spent;
  let pointsAvailable =
    project.tools.planningLog.totalPoints.available;

  // check if team overworked by 110%
  let didOverwork = pointsSpent >= 1.1 * pointsAvailable;

  // return true if at the end of a sprint, the team ended up
  // overworking during the prior sprint
  return isEndOfSprint && didOverwork;
};
```

```
situated_strategies: [
function () {
  // compose message with context from triggered script
  let pointsSpent = project.tools.planningLog.totalPoints.spent;
  let pointsAvailable = project.tools.planningLog.totalPoints.available;

  let message = `It looks like ${ project.members } significantly
    overworked last week (${ pointsSpent } points spent out
    of ${ pointsAvailable } points available).

    During your Planning Meeting, try to reflect with them on
    why they overworked and strategies they can try next time.
    These include:`;

  // suggested strategies for how to not overwork
  let strategies = [
     "- Reaching out for help when you're spending too long on tasks",
     "- Scoping down a story mid-week if you're running out of time
        (e.g., implement one tech slice, instead of 2)",
     "- Defer stories you can't complete to the next sprint instead
        of overworking",
  ];

  // present strategies at the start of a Planning Meeting
  let opportunity = function () {
    return startOfVenue(
      venues.find(where("kind", "PlanningMeeting"))
    );
  };

  // deliver strategy to mentor in a direct message
  return messagePeople({
    message: message + "\n" + strategies.join("\n"),
    people: [project.mentor],
    opportunity: opportunity
    },
  });
};
]
```

1
2
3
data from the Studio API
4a
4b
Jim and Bob meet with Luna Monday Morning
Kayla and Paul meet with Carlos Thursday Afternoon

Slack (**Monday 10:00 AM**)
@Luna: It looks like **Jim and Bob** significantly overworked last week (**50** points spent out of **40** points available).

Slack (**Thursday 3:00 PM**)
@Carlos: It looks like **Kayla and Paul** significantly overworked last week (**45** points spent out of **40** points available).

Figure 2: Situated Scripts externalize work situations and relevant situated strategies using Organizational Objects and Helper Functions (highlighted in orange and magenta, respectively). The above script monitors for situations where any team (1) has overworked on the previous sprint (2). If so, the script delivers a situated strategy to the mentor that suggests planning strategies to discuss with their students at the start of the next Planning Meeting (3). When executed using the Studio API, we see how one team's mentor receives a message Monday morning (4a), whereas the other receives it Thursday afternoon (4b).

(1) an `applicable_set` that defines a list of people whom a script should be monitored for;
(2) a boolean `work_situation_detector` to be monitored for that may reveal an unmet need or ineffective practice; and
(3) a list of `situated_strategies` to enact, with each returning an object that includes whom to send the strategy to (e.g., a team or a person), a message with possible strategies to discuss, and the venue when it should be presented.

Unlike automated systems that enact follow-up interactions once the `work_situation_detector` is met, a key idea behind Situated Scripts is to surface relevant strategies to workers but ultimately let them decide whether to enact those strategies. This allows for situated support to be provided at relevant venues along the work process while still being flexible to the uncertainty in work needs and constraints present in ill-structured work.

As an illustrative example, consider a script that helps teams form and enact feasible work plans; see Figure 2. For all teams (1), this script encodes a `work_situation_detector` that checks if significantly more time was spent on the prior sprint than allocated, indicating that the time needed for the original plan was underestimated (2). When detected, the team's mentor is informed of this situation at the start of their Planning Meeting, along with some strategies they can discuss with their team (3). When executed using data from the Studio API, this script triggers differently for the team of Jim and Bob, who have their planning meeting on Mondays (4a), versus for Kayla and Paul, who have their planning meeting on Thursday (4b). This shows how we can express general work strategies using the Organizational Objects (e.g., connect with your project mentor at the next planning session before the sprint begins) but enacted in specific situations for different people using the Studio API (e.g., Jim and Bob meet with Luna Monday morning).

3.4 Enacting Situated Strategies at Relevant Points of the Work Process

To execute Situated Scripts, we designed the *Orchestration Execution Engine* that allows us to instantiate situated work practices for an arbitrary number of people and teams in an organization at different times; see Figure 3. It does this by maintaining two separate threads that simultaneously monitor for work situations to arise and for opportunities to present situated strategies. Monitoring for work situations begins by computing the `applicable_set` from a situated script (1), and then checking the `work_situation_detector` for each of the targets of the `applicable_set` using data from the Studio API (2). If any evaluates to true, a new `ActiveSituation` document is created (3), and monitoring for the rest of the targets and scripts continues. Separately, monitoring for opportunities to present `situated_strategies` begins by checking for the encoded strategies for each `ActiveSituation` using data from the Studio API (4). If the opportunity for any of these has arrived (5), the engine sends the encoded strategy to the relevant social structure (6). By monitoring these scripts and their executions, we can let the strategies unfold and be enacted in different venues where they are appropriate, rather than immediately when a work situation is detected, across people in the organization.

3.5 Technical Implementation

Orchestration Scripts is implemented using Node.js, Express.js, and MongoDB. Situated Scripts are written by defining an object with an `applicable_set`, a `work_situation_detector`, and a list of `situated_strategies` using the Organizational Objects and Helper Functions. The Orchestration Engine stores these scripts and any `ActiveSituations` generated from the monitoring process in a MongoDB database. Every 15 minutes, it checks to see if new work situations have emerged or if it is time to present a situated strategy for an `ActiveSituation`. Evaluation and presentation of the strategies are done using the Studio API, which has access to data about the organization and ways to send messages through communication tools (e.g., Slack). Our current implementation of the Studio API integrates with APIs provided by Google Drive [3] and Slack [6]; other workplace tools and their APIs can be similarly integrated (e.g., Trello [9]; Github [2]). Strategies from triggered scripts are currently sent to people using Slack; in the discussion, we expand on alternative methods for surfacing situated strategies that future work may explore.

4 CASE STUDY: HOW ORCHESTRATION SCRIPTS CAN SUPPORT SITUATED WORK

To demonstrate how Orchestration Scripts can support situated work, we present an example of a situated work practice solicited from an academic research community detailed below. Through this example, we show how our approach allows for monitoring work situations across an organization and strategizing how to resolve them, while still being flexible to people's changing needs.

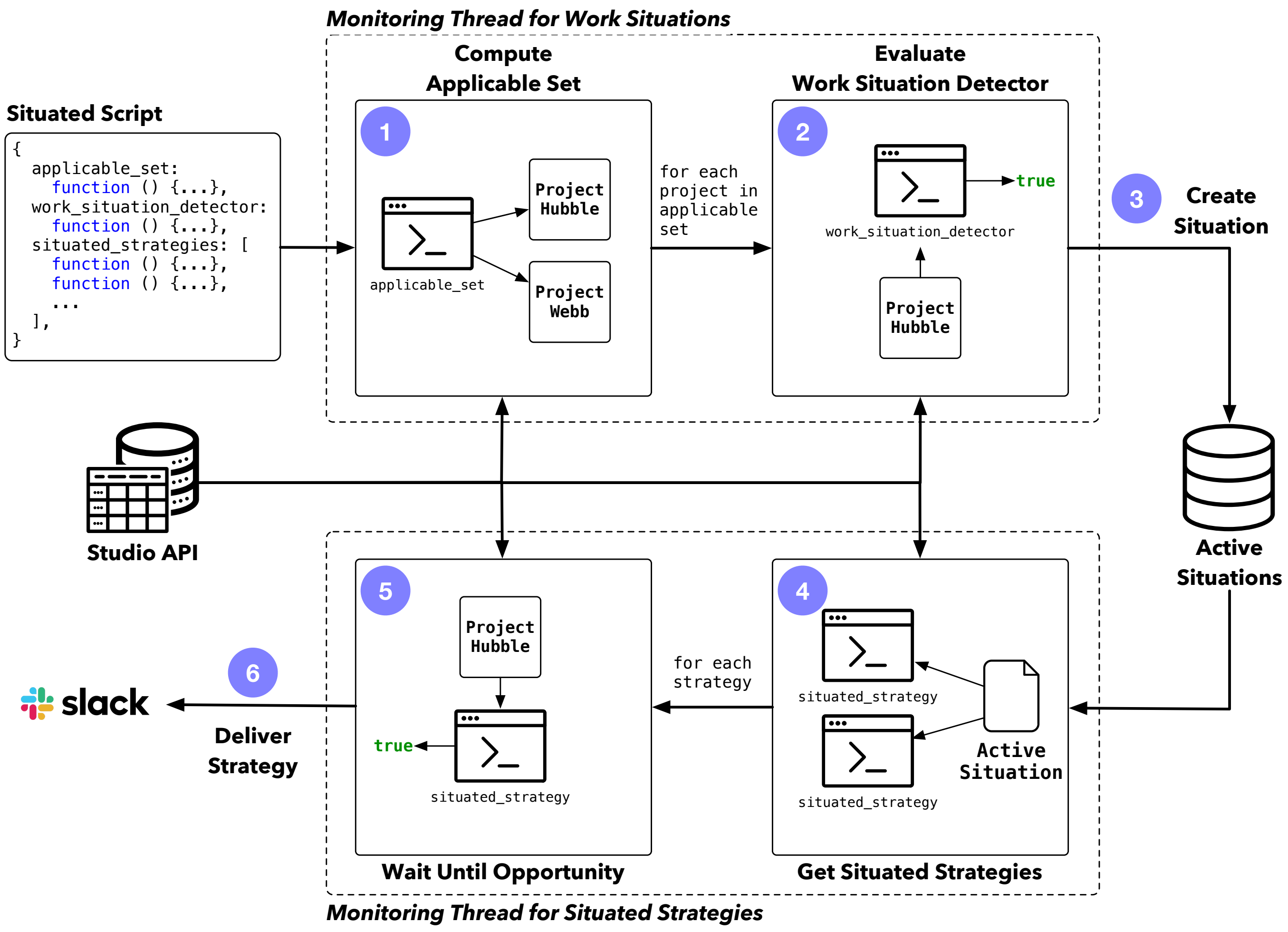
4.1 Research Setting

We situate our study within the Design, Technology, and Research (DTR)[1] academic research community at Northwestern University, which uses the Agile Research Studios (ARS) model to support students learning to independently lead research projects [75]. Work in DTR follows an Agile process [19, 63] where students plan research sprints that progress their research understanding each week. During this process, students are supported by *venues* where they can access peers and mentors from across the organization for help and coaching on their work needs (e.g., related to planning, feedback on research activities, prototyping or testing, and more), and *work and collaboration tools* that facilitate those interactions. Working effectively in this organization involves integrating these situated support opportunities into one's working process to progress work. In these ways, the DTR community and the ARS model resemble the kinds of socio-technical work ecosystems we want to develop systems for since they offer multiple support opportunities along established work processes in an organization.

In what follows, we show Orchestration Scripts can encode a strategy for helping students form scoped research plans and enacting those places across DTR's work ecosystem.

4.2 Encoding General Programs to Support Situated Work

Orchestration Scripts allow for encoding programs to support situated work in a more natural and less brittle manner than workflow tools due to the Organizational Object constructs our system provides. In Figure 4, we show a strategy where mentors want to know if any of their undergraduate students are overcommitted on their research plans, suggesting students strategies for re-scoping plans during the planning meeting if so. On the left, we show how Organizational Objects allow for a general representation of this strategy to be encoded while letting the author stay at a level of abstraction closer to how they think about situated work practices. In contrast, the right shows how individual workflows in a tool like Zapier must be authored for each team, along with hard-coded data about their tools, venues, and social relationships. Not only does this require a worker to focus on the details of the script implementation rather than the strategy (e.g, specific time for a venue versus saying the desired venue), any changes to these values (e.g., a planning venue gets moved) means editing *all* scripts that use those values, not just the one example shown here. Moreover, changes to the script, like adding new strategies or resources, means that *all* script instances would need to be revised. With our system, a worker only needs to update data in the Studio API, which automatically propagates to all active scripts since the Organizational Objects are evaluated using that API. Similarly, a revision to a situated script means that everyone the script is monitored for now receives the revised script. In this way, our use of abstractions of an organization's ways of working allows Situated Scripts to be more expressive, maintainable, and extensible than existing workflow systems.



```
{
  applicable_set:
    function () {...},
  work_situation_detector:
    function () {...},
  situated_strategies: [
    function () {...},
    function () {...},
    ...
  ],
}
```

Figure 3: The Orchestration Execution Engine allows for real-time monitoring of work situations and opportunities to present situated strategies as encoded in a Situated Script. Using data from the Studio API, the monitoring thread for work situations (top) continually computes who the script applies to using the `applicable_set` (1) and checks if work situations encoded by the `work_situation_detector` have occurred for each project (2), creating an `ActiveSituation` if so (3). Simultaneously, the monitoring thread for situated strategies (bottom) checks each `situated_strategy` for an `ActiveSituation` (4) to see if an opportunity to present strategies for a project has arrived (5), sending it to relevant project or person if so (6).

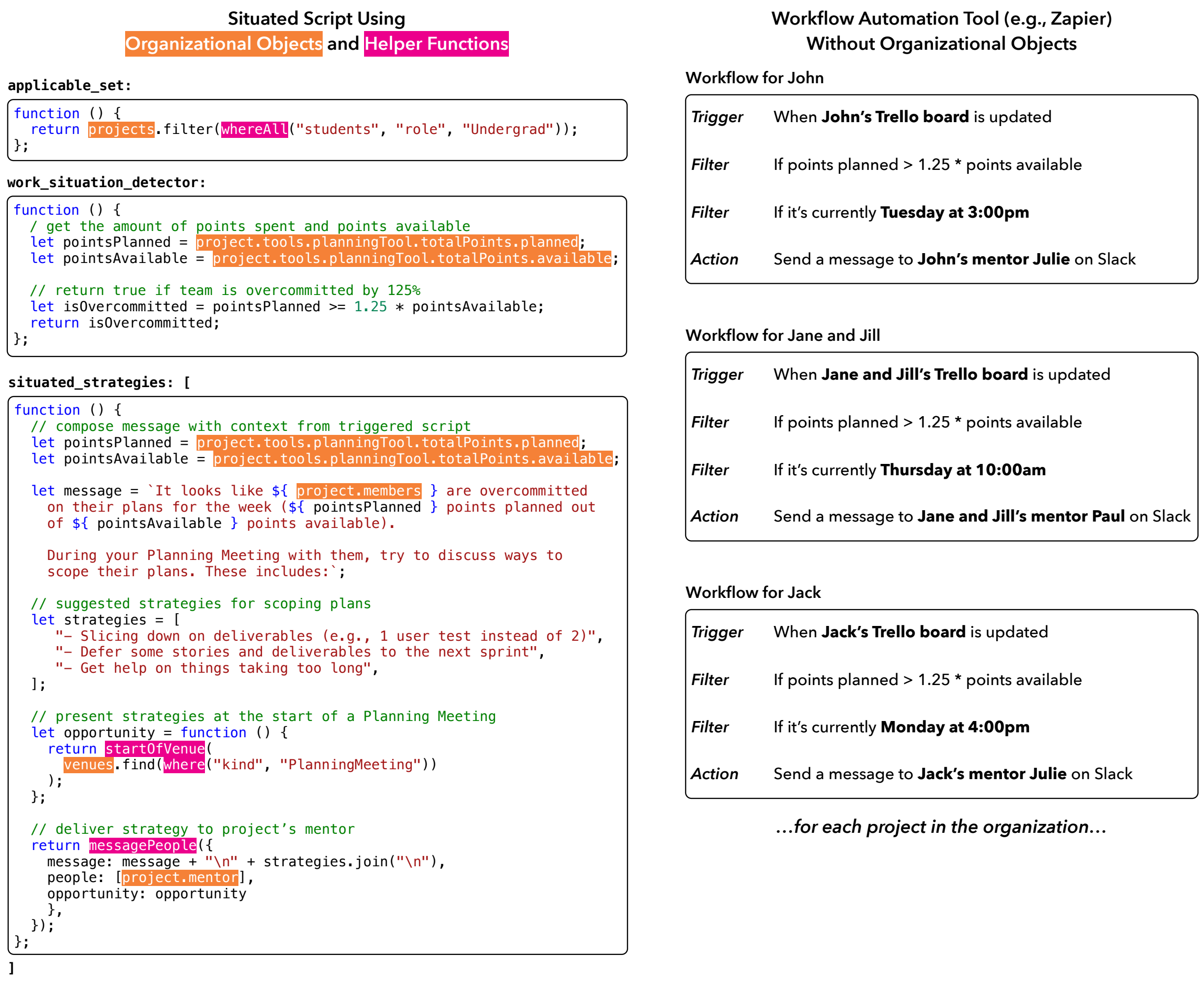
4.3 Tailoring General Strategies to Workers' Specific Social Relationships and Venues

Even further, these constructs allow us to encode general strategies that apply to different people and teams, but become tailored based on their relationships with others and their specific collaboration venues. In Figure 5, we extend the example from the previous section to have conditional strategies that are delivered based on which team the script was triggered for. The mentors are provided with general strategies for all teams, but recommended supporting teams with only one person to a greater extent since they do not have a partner to re-plan with. Furthermore, when an individual has an onboarding mentor, this strategy is also suggested to the mentor since their student could get support on replanning from them. Despite being specific to each project team when enacted, the conditional strategies can be expressed in a general manner with ease using our system, with the engine taking responsibility for presenting the relevant information. In this way, Orchestration Scripts can allow for even richer, general models of working and strategies to be encoded that are applicable across the organization but may execute in very different ways for each team.

4.4 Flexibly Supporting Work Activities Across a Work Ecosystem

Finally, Orchestration Scripts allows us to create spaces to discuss emerging work situations and some relevant strategies to attempt, but leaves the decision of how to address issues to the workers. Even though the higher-level work process might be well-structured, the

[1]https://dtr.northwestern.edu/


Situated Script Using
Organizational Objects and Helper Functions

applicable_set:
```
function () {
  return projects.filter(whereAll("students", "role", "Undergrad"));
};
```

work_situation_detector:
```
function () {
  / get the amount of points spent and points available
  let pointsPlanned = project.tools.planningTool.totalPoints.planned;
  let pointsAvailable = project.tools.planningTool.totalPoints.available;

  // return true if team is overcommitted by 125%
  let isOvercommitted = pointsPlanned >= 1.25 * pointsAvailable;
  return isOvercommitted;
};
```

situated_strategies: [
```
function () {
  // compose message with context from triggered script
  let pointsPlanned = project.tools.planningTool.totalPoints.planned;
  let pointsAvailable = project.tools.planningTool.totalPoints.available;

  let message = `It looks like ${ project.members } are overcommitted
    on their plans for the week (${ pointsPlanned } points planned out
    of ${ pointsAvailable } points available).

    During your Planning Meeting with them, try to discuss ways to
    scope their plans. These includes:`;

  // suggested strategies for scoping plans
  let strategies = [
     "- Slicing down on deliverables (e.g., 1 user test instead of 2)",
     "- Defer some stories and deliverables to the next sprint",
     "- Get help on things taking too long",
  ];

  // present strategies at the start of a Planning Meeting
  let opportunity = function () {
    return startOfVenue(
      venues.find(where("kind", "PlanningMeeting"))
    );
  };

  // deliver strategy to project's mentor
  return messagePeople({
    message: message + "\n" + strategies.join("\n"),
    people: [project.mentor],
    opportunity: opportunity
    },
  });
};
```
]

Workflow Automation Tool (e.g., Zapier)
Without Organizational Objects

Workflow for John

Trigger	When **John's Trello board** is updated
Filter	If points planned > 1.25 * points available
Filter	If it's currently **Tuesday at 3:00pm**
Action	Send a message to **John's mentor Julie** on Slack

Workflow for Jane and Jill

Trigger	When **Jane and Jill's Trello board** is updated
Filter	If points planned > 1.25 * points available
Filter	If it's currently **Thursday at 10:00am**
Action	Send a message to **Jane and Jill's mentor Paul** on Slack

Workflow for Jack

Trigger	When **Jack's Trello board** is updated
Filter	If points planned > 1.25 * points available
Filter	If it's currently **Monday at 4:00pm**
Action	Send a message to **Jack's mentor Julie** on Slack

...for each project in the organization...

Figure 4: Organizational Objects allow people to express general Situated Scripts that are closer in abstraction to how they think about situated work already, and are less brittle than workflow systems since they are encoded using Organizational Objects that refer to organization-specific data in a general manner versus individual workflows with hardcoded data for each project.

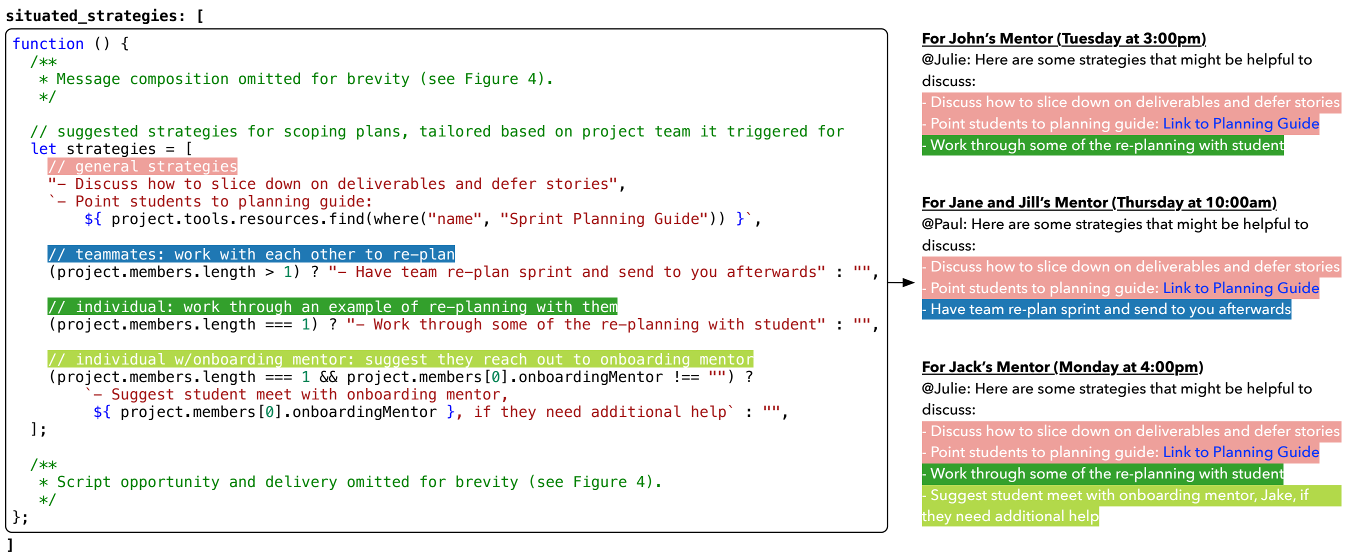
day-to-day work remains ill-structured, often causing unexpected needs to arise as progress is made. In Figure 6, we show how a student's mentor was notified that they were overcommitted for the week since they were finishing up a prototype for an upcoming study. While the mentor and system suggested that the student come to Office Hours to pair program (based on their prior interaction and the student's planned deliverables), the student raised a new issue about not being able to recruit users for their study that he and the mentor worked on instead. By leaving the locus of control on how to use support opportunities surface by the system to the people, Orchestration Scripts can flexibly provide situated support that fully automated approaches would prohibit.

5 FIELD STUDY: ORCHESTRATION SCRIPTS IN A RESEARCH LEARNING COMMUNITY

To understand how Orchestration Scripts support situated work within a work ecosystem, we ran a 1-week pilot study in a research learning community where we deployed situated scripts to orchestrate wrap-up activities for the term; see Table 1 for strategies. We asked: *how do situated orchestration scripts support people in enacting situated work practices across a socio-technical ecosystem?*

5.1 Participants

We situated our study in the DTR academic research community detailed in Section 4.1, from which we recruited 17 people during their Spring 2022 academic term. 12 people were undergraduate or masters students, 4 were Ph.D. students, and 1 was a faculty member. Undergraduate and masters students received situated strategies to support them as students when work situations were detected, while Ph.D. students received these as well as ones to discuss with the students they mentored; the faculty member only received strategies for the students they mentored. For privacy, we gave all participants pseudonyms. Two of the paper's authors are a faculty member and a Ph.D. student in the community.



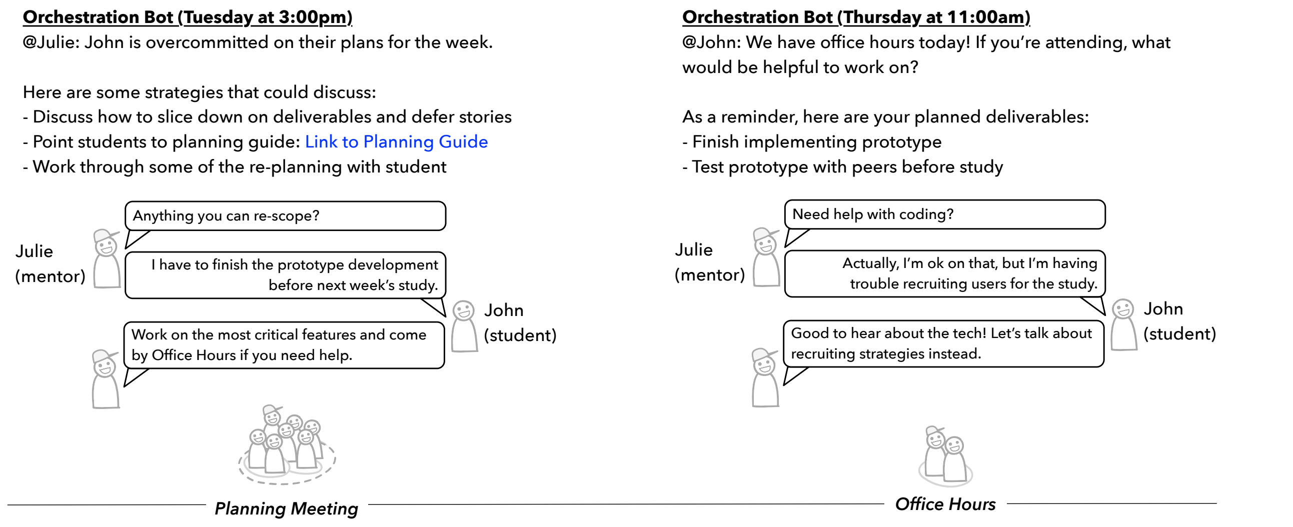
```
situated_strategies: [
function () {
  /**
   * Message composition omitted for brevity (see Figure 4).
   */

  // suggested strategies for scoping plans, tailored based on project team it triggered for
  let strategies = [
    // general strategies
    "- Discuss how to slice down on deliverables and defer stories",
    `- Point students to planning guide:
        ${ project.tools.resources.find(where("name", "Sprint Planning Guide")) }`,

    // teammates: work with each other to re-plan
    (project.members.length > 1) ? "- Have team re-plan sprint and send to you afterwards" : "",

    // individual: work through an example of re-planning with them
    (project.members.length === 1) ? "- Work through some of the re-planning with student" : "",

    // individual w/onboarding mentor: suggest they reach out to onboarding mentor
    (project.members.length === 1 && project.members[0].onboardingMentor !== "") ?
        `- Suggest student meet with onboarding mentor,
         ${ project.members[0].onboardingMentor }, if they need additional help` : "",
  ];

  /**
   * Script opportunity and delivery omitted for brevity (see Figure 4).
   */
};
]
```

For John's Mentor (Tuesday at 3:00pm)
@Julie: Here are some strategies that might be helpful to discuss:
- Discuss how to slice down on deliverables and defer stories
- Point students to planning guide: Link to Planning Guide
- Work through some of the re-planning with student

For Jane and Jill's Mentor (Thursday at 10:00am)
@Paul: Here are some strategies that might be helpful to discuss:
- Discuss how to slice down on deliverables and defer stories
- Point students to planning guide: Link to Planning Guide
- Have team re-plan sprint and send to you afterwards

For Jack's Mentor (Monday at 4:00pm)
@Julie: Here are some strategies that might be helpful to discuss:
- Discuss how to slice down on deliverables and defer stories
- Point students to planning guide: Link to Planning Guide
- Work through some of the re-planning with student
- Suggest student meet with onboarding mentor, Jake, if they need additional help

Figure 5: Situated Scripts can encode strategies that are sent conditionally, depending on who the script was triggered for. For example, this script suggests general replanning strategies to the mentor for all teams, but also specific strategies if the team has multiple people, if they are a single person, and if they are a new student with an onboarding mentor.



Orchestration Bot (Tuesday at 3:00pm)
@Julie: John is overcommitted on their plans for the week.

Here are some strategies that could discuss:
- Discuss how to slice down on deliverables and defer stories
- Point students to planning guide: Link to Planning Guide
- Work through some of the re-planning with student

Orchestration Bot (Thursday at 11:00am)
@John: We have office hours today! If you're attending, what would be helpful to work on?

As a reminder, here are your planned deliverables:
- Finish implementing prototype
- Test prototype with peers before study

Julie (mentor)
Anything you can re-scope?
I have to finish the prototype development before next week's study.
John (student)
Work on the most critical features and come by Office Hours if you need help.
Planning Meeting
Julie (mentor)
Need help with coding?
Actually, I'm ok on that, but I'm having trouble recruiting users for the study.
John (student)
Good to hear about the tech! Let's talk about recruiting strategies instead.
Office Hours

Figure 6: While Orchestration Scripts may suggest strategies based on its understanding of support opportunities and students' work needs, it leaves the decision of what and when to work on to the students and mentors. This provides them flexibility to handle new needs as they arise during interactions across the ecosystem.

5.2 Procedure

We deployed a variety of Situated Scripts to the DTR community modeled on existing effective work practices that students are encouraged to practice, such as forming feasible research plans and using support opportunities across the ecosystem to progress work; Table 1 includes summaries of these scripts. We instructed users to work as normal, interacting with the strategies surfaced by Orchestration Scripts over Slack when they thought it was helpful. Following the deployment, users participated in a 15-45 minutes long semi-structured interview, with the length depending on how many strategies were presented. During interviews, users were shown the strategies Orchestration Scripts presented and asked follow-up questions based on a semi-structured interview guide. For students, we asked if and how the presented strategies influenced their work practices, such as how they may have replanned, shared artifacts, or discussed strategies with their mentor. For mentors, we asked similar questions but focused on how the presented strategies influenced their mentoring with their students. Finally, we closed the interview with general questions about what users liked and disliked about the strategy prompts they had received.

Before the study, all users consented to participate and allowed their data to be used anonymously for research purposes. Before interviews, participants consented to audio, video, and screen recordings; audio recordings were transcribed for analysis. The lead author conducted all interviews over Zoom video calls.

5.3 Measures and Analysis

We used a thematic analysis approach [16] to analyze our interview transcripts. Our goal when coding the data was to identify how the Orchestration Scripts informed users about the work situations they were in and the strategies they could attempt. We did this by coding for moments where users said they became aware of these situations and how the surfaced strategies influenced their work practices. Once all data was coded, we reviewed the sub-themes generated from the analysis by identifying instances where our data supported or contradicted them. These sub-themes were then iteratively refined by splitting or combining them until they were distinct. We then reviewed all the themes and finalized the set we present below. The lead author conducted the coding of the interview transcripts and the generation and review of the initial themes; all authors collectively refined and finalized the themes.

5.4 Results

Orchestration Scripts presented 36 situated scripts to students and mentors during the study; see Table 1. Most strategies (22 out of 36) were delivered to project channels where students and mentors could see them; the rest were direct messages to mentors (12 out of 36) or to a channel for Ph.D. students and their faculty mentor (2 of 36). Strategies were delivered at venues for planning, getting help during office hours, and community-wide meetings, along with 1 presented mid-week. Most strategies were presented before the start of these venues so students could prepare for them–such as by sharing artifacts or progress–or to make mentors aware of potential strategies to discuss for a work situation.

In the rest of this section, we discuss how Orchestration Scripts supported situated work by: (1) monitoring for work situations that often get overlooked; (2) helping people strategize how to resolve work situations via suggested strategies; and (3) enacting check-ins between students and mentors across venues throughout the week.

5.4.1 Promoting Awareness of Emergent Work Situations. Orchestration Scripts informed mentors about situations in which their students were working ineffectively and raised those issues for discussion at appropriate venues. In one example, mentors were informed that their student's research plans were infeasible for the available time; see Figure 7. As Brady explained, it became an agenda item to talk with his students about: *"[Going into planning meeting], I was almost ready as a talking point like, 'hey it looks like you're over points,' and that was a nice thing to point to already have."* His student Melissa recalled how this led to them discussing strategies for rescoping their paper deliverable so that it and their user testing goals could both be accomplished: *"we were planning out how we can get the end-of-term deliverable in faster, looking at different parts of the paper that we need to update [based on a prior draft] instead of writing a new thing."*

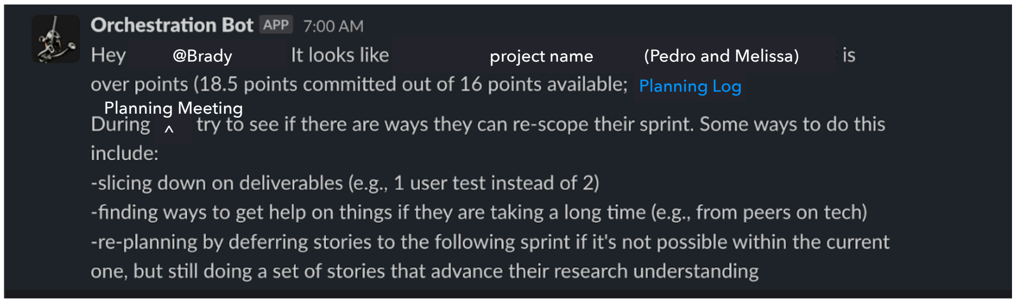
Before using our tool, mentors primarily monitored for ineffective practices–like being overcommitted on research plans or how widely students seek help from peers–by manually polling for any relevant data. In reality, mentors often forgot or chose not to check these things, focusing instead on issues they were already aware of and thought were important to discuss. However, when brought into focus by Orchestration Scripts, mentors, like Irene, made an effort to do so: *"I would [talk about planning] with new-ish students, so I didn't have the intention to do it with Amanda (her senior student), but when I saw that she was overcommitted, then I was like, 'Okay, we should actually talk about this, and like how many hours it's going to take to do each thing.'"* This discussion helped Amanda shift her goal from having a complete draft of findings and discussion for her latest study towards a more feasible outline that showed her new research understanding but involved substantially less work. In other words, these scripts provided an expanded awareness of work situations, particularly those that mentors may overlook due to heuristics they apply and that students may not bring up on their own (e.g., my experienced student knows how to plan within time constraints and will ask if they need help).

Orchestration Scripts also promoted timely discussion of "seasonal" situations that people have peripheral awareness of but only attend to when they get closer. For example, Lawrence had an upcoming Community-Wide presentation the following week, where he would get feedback from the entire community on his research. Lawrence shared that the prompt was, *"a good reminder to get the ball rolling on planning his [Community-Wide presentation],"* and to, *"potentially bring it up to Clark (his mentor), whether it be in [the planning meeting] or offline."* Clark also found prompt useful since the presentation is, *"one of the few times that [students] can get a ton of feedback very quickly."* Another example was for students preparing end-of-term deliverables during the final week of the academic term. Crystal found it helpful to be prompted the day before her last planning meeting of the term since she, *"usually preps for [planning] meetings like the day before. So it was nice to have [the script] there to think about while I was planning what I was going to say and talk about."* She also liked seeing what the deliverables involved, saying: *"[it was] nice to just make sure if I had any questions about anything, especially with things like the video or the write-up, that I could ask about that in [planning meeting] as well."* In this way, we see how Orchestration Scripts can have an awareness of situations–whether it be emergent ones based on people's working patterns or upcoming events–bubble up for discussion, but only for the things that are currently relevant to discuss and at the times where discussion is helpful (i.e., at a venue).

Finally, Orchestration Scripts surfaced opportune moments at venues that could help progress one's work; see Figure 8. While routine venues are generally known to people, having the system suggest them as support opportunities for *this week's* work needs prompted helpful discussions that would have otherwise been missed. For example, Amanda shared how the strategy for an upcoming Office Hour helped her reflect on what to work on with her mentor: *"I liked hearing more specifically about what Irene (her mentor) wanted me to practice in [office hours] before I went in, so I had more of a focus."* Similarly, Lawrence found it helpful to think about how he could work on his research with coaching from mentors and seek help from peers during a Community-Wide meeting:

Goal of Situated Script	Applicable Set	Work Situation Detector	Suggested Situated Strategy	Strategy Sent To	Times Sent
Planning end-of-term deliverables that show research understanding	All students	1 day before the final Planning Meeting	Planning end-of-term deliverables before the final Planning Meeting	Project team	14
		30 minutes before the final Planning Meeting	Discuss end-of-term deliverable plans during the final Planning Meeting	Mentor direct message	6
Using support venues throughout the week to progress work	Undergrad and Masters Students	Morning of a Planning Meeting	Share deliverables and research progress before today's Planning Meeting	Project team	2
		End of a Planning Meeting	Share takeaways from today's Planning Meeting discussion	Project team	2
		Morning of a Office Hours	Share plans for what you want to discuss at Office Hours today	Project team	1
		Morning of the Community-Wide Meeting	Share what you will work on during today's Community-Wide Meeting	Project team	2
Students form feasible plans (i.e., not overcommitted on plans)	Undergrad and Masters Students	Start of a Planning Meeting and project 110% of points committed than available	Scoping research sprint plans to be within time constraints	Mentor direct message	3
Students know project risks and next steps (i.e., not under-committed on plans)	Undergrad and Masters Students	Start of a Planning Meeting and project is 90% of points committed than available	Strategies for discussing planning-related obstacles	Mentor direct message	2
Plan an upcoming Community-Wide Presentation	Undergrad and Masters Students	1 week before a project's Community-Wide Presentation at the Community-Wide Meeting	Begin planning Community-Wide Presentation	Project Team	1
			Reminder to discuss Community-Wide Presentation plans at Planning Meeting	Mentor direct message	1
Support Ph.D. students in progressing research throughout the week	Ph.D. students	Afternoon 1 day before a Planning Meeting	Share deliverables and research progress before tomorrow's Planning Meeting	Ph.D. students channel	1
		Mid-week (2 days before a Planning Meeting)	Mid-week check-in on research progress	Ph.D. students channel	1
				Total	36

Table 1: Situated Strategies were delivered 36 times during the 1-week pilot study by the Orchestration Scripts system across venues for planning, getting help at office hours, and community-wide interactions, along with 1 being presented mid-week.



Orchestration Bot APP 7:00 AM

Hey @Brady It looks like project name (Pedro and Melissa) is over points (18.5 points committed out of 16 points available; Planning Log

During Planning Meeting ^ try to see if there are ways they can re-scope their sprint. Some ways to do this include:

-slicing down on deliverables (e.g., 1 user test instead of 2)

-finding ways to get help on things if they are taking a long time (e.g., from peers on tech)

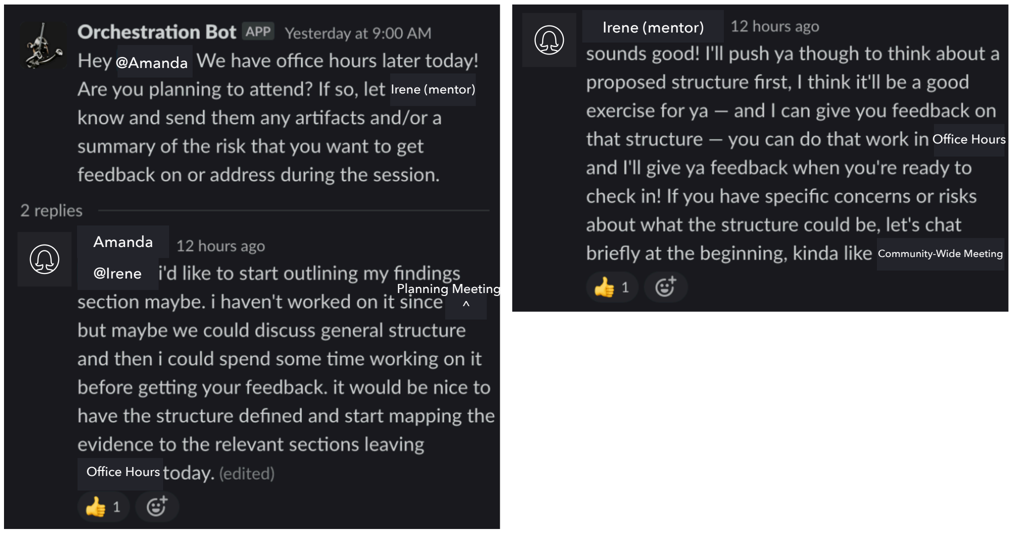
-re-planning by deferring stories to the following sprint if it's not possible within the current one, but still doing a set of stories that advance their research understanding

information referring to individuals, projects, groups, venues, and tools has been anonymized

blue text indicates a link to a tool in the message

Figure 7: Mentors were informed when their students had made infeasible research plans–a planning-related issue that they normally do not remember to discuss or check–when their students had planned more hours for an upcoming research sprint than they were allocated, and suggestions for how to help the students re-scope.

"I thought about what I wanted to do for [the peer help session]...I definitely tried to look at the [research activity scaffolds] spreadsheet [before the Community-Wide meeting], working out which one would fit and what I wanted to do." This shows how people benefit from being aware of even routine venues so they can strategize how those venues can help progress their work.


Orchestration Bot APP Yesterday at 9:00 AM
Hey @Amanda We have office hours later today! Are you planning to attend? If so, let Irene (mentor) know and send them any artifacts and/or a summary of the risk that you want to get feedback on or address during the session.

2 replies

Amanda 12 hours ago
@Irene i'd like to start outlining my findings section maybe. i haven't worked on it since Planning Meeting but maybe we could discuss general structure and then i could spend some time working on it before getting your feedback. it would be nice to have the structure defined and start mapping the evidence to the relevant sections leaving Office Hours today. (edited)

Irene (mentor) 12 hours ago
sounds good! I'll push ya though to think about a proposed structure first, I think it'll be a good exercise for ya — and I can give you feedback on that structure — you can do that work in Office Hours and I'll give ya feedback when you're ready to check in! If you have specific concerns or risks about what the structure could be, let's chat briefly at the beginning, kinda like Community-Wide Meeting

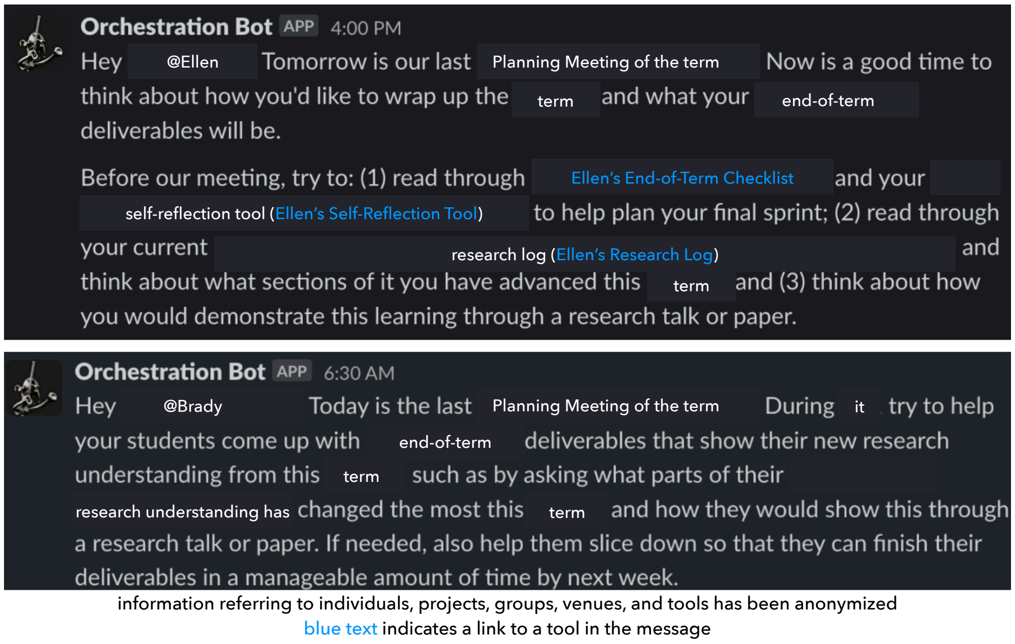
information referring to individuals, projects, groups, venues, and tools has been anonymized
blue text indicates a link to a tool in the message

Figure 8: Students and mentors were prompted when support opportunities in the ecosystem were approaching, creating a space for them to strategize how to use the opportunities if students were attending,

5.4.2 Suggesting Relevant Strategies for Work Needs at Appropriate Times. Orchestration Scripts surfaced situated strategies for detected work situations at opportune times to discuss them. Mentors found that the suggested strategies provided a useful scaffold when discussing the situation with their student. With the earlier sprint re-scoping example (see Figure 7), mentor Brady shared how he, *"really liked [how the system showed] the different ways [his students] can re-scope their sprint because they were all good strategies [for re-scoping research plans]."* As another example for deliverable planning (see Figure 9), Brady shared how the suggested strategies in the script became agenda items to discuss during the meeting, stating that *"I've been a mentor for a while, but it is quite helpful to have the explicit intentions for [discussing end-of-term deliverables]."* Robbie, a novice mentor, found the strategies targeted to the current situation, like the final planning meeting with students, particularly useful, stating: *"If this advice had been in some generic document, it wouldn't be as helpful as getting exactly the relevant thing right before that particular [planning] meeting."* Students similarly found the suggested strategies useful. Ellen shared that seeing the strategies helped her plan how she would accomplish her research work and deliverables for the end of the term: *"I kind of structured my Planning Log for the last week around what I need to get done for the end-of-term checklist."* In this way, Orchestration Scripts supported mentors and students in recognizing what strategies they could discuss or attempt at the relevant times and places.

5.4.3 Orchestrating Interactions Throughout an Ecosystem. Orchestration Scripts helped students adopt effective strategies to work towards their goals by providing a space for them and mentors to discuss work needs across ecosystem interactions during the week. For example, recall Amanda and her mentor Irene's interaction before their Office Hours; see Figure 8. Irene shared how this interaction helped her push Amanda to practice independent thinking rather than leaning on Irene to provide direction: *"I had a hunch that she wanted to check in because she was short on time, and this is the first time she can think about the structure [of the draft]. But rather than think about it with me, I'd rather she start thinking about it herself. So, I was glad we had that conversation because I think it reinforces this tone of trying [to get her] to do things independently."*

As another example, Orchestration Scripts enabled "course correcting" interactions between Steven, the faculty member, and his Ph.D. students when they felt stuck; see Figure 10. For example, Brady reflected on the feedback he got from his mid-week update, sharing how it helped him rethink the scope of his goals and progress his deliverables: *"...it gave me a way to think about the user study [that I was unsure about]. I was confused about whether that was a case study versus a user study, and this showed me that I could have a way to write a study that's pretty clear [regardless of the type]."* From Steven's perspective, this interaction was helpful because it created a check-in opportunity with Brady before the next planning meeting, allowing him to progress more effectively towards his goals: *"[Brady] was able to share how they didn't think there was a path forward and they were starting to like taking steps backward. And then I was basically like, couldn't you just do this and this? So it was me kind of holding them to a path or showing them a little bit more of a path which they probably had, but in their fear they couldn't see."* By facilitating check-ins across support opportunities in the ecosystem, practices discussed at earlier meetings can be re-emphasized at future working sessions people have, allowing for better practices to build up over time. In this way, we see how orchestration scripts can provide a sort of "continuity" of discussion that goes beyond the venues they originally happened in and potentially much further into the future.


Orchestration Bot APP 4:00 PM

Hey @Ellen Tomorrow is our last Planning Meeting of the term Now is a good time to think about how you'd like to wrap up the term and what your end-of-term deliverables will be.

Before our meeting, try to: (1) read through Ellen's End-of-Term Checklist and your self-reflection tool (Ellen's Self-Reflection Tool) to help plan your final sprint; (2) read through your current research log (Ellen's Research Log) and think about what sections of it you have advanced this term and (3) think about how you would demonstrate this learning through a research talk or paper.

Orchestration Bot APP 6:30 AM

Hey @Brady Today is the last Planning Meeting of the term During it try to help your students come up with end-of-term deliverables that show their new research understanding from this term such as by asking what parts of their research understanding has changed the most this term and how they would show this through a research talk or paper. If needed, also help them slice down so that they can finish their deliverables in a manageable amount of time by next week.

information referring to individuals, projects, groups, venues, and tools has been anonymized
blue text indicates a link to a tool in the message

Figure 9: Suggested strategies for preparing end-of-term deliverables as show to a student (top) so they can account for them in their planning the day before their final Planning Meeting, and to a mentor (bottom) shortly before the Planning Meeting so they are aware of strategies to discuss during the meeting.

6 DISCUSSION AND FUTURE WORK

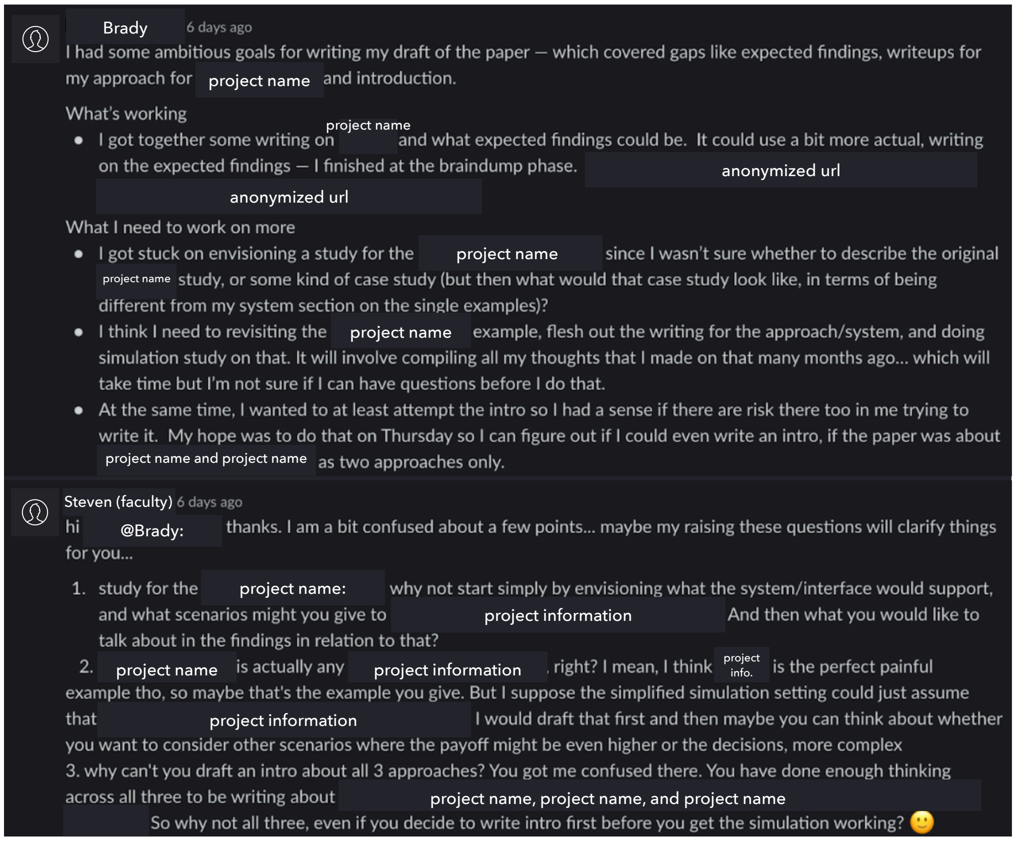
In summary, we demonstrate how Orchestration Scripts can support situated work by providing people with in-the-moment awareness of work situations that they may overlook, the appropriate strategies for work needs in situations where they can attempt or discuss them, and support for adopting effective work practices that extend across interactions in the ecosystem. We now revisit Orchestration Script's design elements and discuss how they may inform the design of future systems that support situated work.

6.1 Providing Programming Abstractions for an Organization's Ways of Working

Programming abstractions for an organization's ways of working allows us to tie effective practices to work situations along an established work process. Through our case study, we showed how Organizational Objects could model situated work practices that generalize across people and projects in an organization. Moreover, these general scripts can easily embed conditional strategies using our programming constructs, providing people and teams with tailored support at appropriate times. Not only are these scripts less brittle than existing workflow automation approaches to changes in people's relationships with each other or when venues occur, but also more closely resemble how we think about and discuss situated work practices in the workplace.

While the relationships captured by Organizational Objects are already useful for modeling situated work practices, they could be extended with additional relational ideas on where and with whom strategies should be attempted. When work needs arise, multiple venues can sometimes be appropriate to resolve them, and the best strategy is to attend the next venue that can provide support. For example, a programmer needing debugging support for ReactJS could go to a team lead's 1-1 venue or a pair programming session with a peer, whichever comes first and has expertise in ReactJS. Another extension could be adding constructs that sit at even higher levels of abstraction than what we provide. For example, instead of having to check data for how many people are on a project to determine if the project has teammates that can support each other (like in Figure 5), we could include a `hasTeammates` construct that is more natural and expressive for script writers to use to encode the same idea [49]. In this way, Organizational Objects can be expanded in the future to capture even more of the relational knowledge that workers already apply when working (e.g., attending the "next" relevant support opportunity; whether a person has a teammate or onboarding mentor; expertise of others [13, 45]), allowing for richer strategies based on these constructs to be encoded.

Organizational Objects are also helpful for work activities beyond supporting situated work. Because they capture people's relationships with each other (e.g., a person's mentor) and venues in an organization (e.g., their planning meeting), any tasks that are also relational in this way may benefit from using Organizational Objects. For example, this could be valuable for a human resources department doing quarterly performance reviews where each worker meets with their managers to discuss their growth and contributions; or to an administrator in an academic department for creating template emails for general tasks that would be routed to the appropriate people, such as confirming a student's funding source with their advisor each term [39, 51]. In short, having our



Brady 6 days ago

I had some ambitious goals for writing my draft of the paper — which covered gaps like expected findings, writeups for my approach for project name and introduction.

What's working

- I got together some writing on project name and what expected findings could be. It could use a bit more actual, writing on the expected findings — I finished at the braindump phase. anonymized url anonymized url

What I need to work on more

- I got stuck on envisioning a study for the project name since I wasn't sure whether to describe the original project name study, or some kind of case study (but then what would that case study look like, in terms of being different from my system section on the single examples)?
- I think I need to revisiting the project name example, flesh out the writing for the approach/system, and doing simulation study on that. It will involve compiling all my thoughts that I made on that many months ago... which will take time but I'm not sure if I can have questions before I do that.
- At the same time, I wanted to at least attempt the intro so I had a sense if there are risk there too in me trying to write it. My hope was to do that on Thursday so I can figure out if I could even write an intro, if the paper was about project name and project name as two approaches only.

Steven (faculty) 6 days ago

hi @Brady: thanks. I am a bit confused about a few points... maybe my raising these questions will clarify things for you...

1. study for the project name: why not start simply by envisioning what the system/interface would support, and what scenarios might you give to project information And then what you would like to talk about in the findings in relation to that?
2. project name is actually any project information, right? I mean, I think project info. is the perfect painful example tho, so maybe that's the example you give. But I suppose the simplified simulation setting could just assume that project information I would draft that first and then maybe you can think about whether you want to consider other scenarios where the payoff might be even higher or the decisions, more complex

3. why can't you draft an intro about all 3 approaches? You got me confused there. You have done enough thinking across all three to be writing about project name, project name, and project name So why not all three, even if you decide to write intro first before you get the simulation working? 🙂

information referring to individuals, projects, groups, venues, and tools has been anonymized
blue text indicates a link to a tool in the message

Figure 10: Orchestration Scripts provided touchpoints between students and mentors to check in on work progress throughout the week and created discussion space on how to course-correct when students were working ineffectively towards their goals.

constructs in future software may allow for various systems that benefit from relational knowledge in the workplace to be created.

6.2 Authoring Situated Scripts to Support Situated Work Practices

Situated Scripts enacted by Orchestration Scripts helped people practice effective situated work practices. Users in our field study found that having these strategies delivered at venues along their work process helped them realize effective ways of working and coaching at opportune moments. Because these strategies were suggested to users (rather than an automatically enacted workflow), students and mentors could choose to use the strategies they felt were relevant. In short, our approach provides the necessary flexibility for ill-structured work, while still surfacing the necessary context (i.e., work situation and strategies) to workers at the opportune times to support situated work practices.

More broadly, the usefulness of Situated Scripts likely depends on the kinds of situations raised and peoples' familiarity with the suggested strategies. While we would expect scripts that surface emergent situations that are easy to miss but straightforward to detect to be the most valuable (e.g., from our study, when students were overcommitted on research plans), scripts surfacing routine collaboration opportunities were well-liked for the discussion space they created. Similarly, while those less familiar with effective strategies benefited the most from our system's prompts, more experienced mentors still found value in being reminded of strategies to discuss with their students. Put another way, supporting situated work ultimately means that *everyone*–mentors and students (or managers and employees)–know what work situations they are in and the strategies they can use. As long as Situated Scripts continue to provide this awareness, we argue that they will be helpful for many use cases and experience levels.

When encoding Situated Scripts, we expect most scripts to be written based on authors' prior experience with situated work practices. While not a prerequisite, it is easier to encode programs for common challenges people encounter and the general approaches to resolve them (e.g., re-planning strategies when someone is overworking), or for activities along the existing work process that script authors are familiar with (e.g., planning weekly deliverables; onboarding newcomers; how to use collaboration opportunities). We can also consider a community-based approach to script authoring where people from across the organization can create customized forks of scripts others have written that can be applied to themselves or teams they manage [15, 40]. This would be possible and easy to do since the scripts are all written using the Organizational Object constructs, meaning that strategies can apply to people other than those whom the script author originally intended.

Related is how we can support the end-user authoring process of Situated Scripts. Our work focused on developing the necessary computational tools for supporting situated work, with the authors handling writing scripts based on practices used by the community. For systems to be usable by programmers and non-programmers alike, future work should consider developing authoring tools built on prior work for end-user programming tools for trigger-action programming [21–24, 43, 71, 76]. Moreover, future work can consider studying how to support potential challenges in the authoring process, including: debugging work detectors triggering incorrectly, when it is unclear what strategies to suggest for a work situation, or when the situation authors want to model is several steps removed from the available programming constructs (e.g., when someone is struggling to make progress) [43]. Through this work, we can help people express the kinds of situated support they want a system like Orchestration Scripts to facilitate in their workplaces.

6.3 Towards Systems That Integrate With Our Ways of Working

Our work shows that supporting situated work requires our systems to be embedded with ecosystem-level intelligence about where and how people work. As workplaces grow in complexity, it is increasingly crucial for tools to support us in taking effective situated action [62], which will be challenging for tools to do if they do not understand our situated ways of working. Orchestration Scripts moves towards realizing this vision by explicitly modeling the ways of working that underpin a workplace and allowing people to write scripts that facilitate situated work using those constructs.

To fully realize this vision, future work will need to consider how organizations develop and maintain the Studio APIs and Organizational Objects that underpin systems like Orchestration Scripts. These components are necessary for any system that supports situated work since these constructs are what provide the necessary ecosystem-level intelligence to orchestrate situated work practices. As a first step, we can consider how Studio APIs–which are simplified knowledge bases of how people work in an organization–can be developed and maintained using inspiration from prior work that studied organizational knowledge bases in workplaces [12, 44].

With the newly available ecosystem-level intelligence our work provides, we can consider how to imbue our existing tools with this understanding. For example, we could extend Slack's standard workflow and reminder system to use our constructs, allowing the team lead to ask things like, "remind my project team to discuss progress on their study design deliverable during Office Hours," rather than specifically saying who and what time a reminder should be sent. This would allow for us to continue using the tools we find effective for our work needs, but add in an additional layer of intelligence for how we want them to support us in doing situated work.

Crucially, we can start moving workplace systems towards ones that support us in communicating and strategizing about work activities in a way that considers interactions across an ecosystem as part of their designs. For example, consider routine meetings between teams and their leads. Prior to these meetings teams may generate meeting agendas to surface work needs they want to address, and then generate follow-up tasks during the meeting based on their discussions [18, 57]. However, these tools lack the ability to pull in relevant context about how work is progressing across the broader work ecosystems (e.g., whether people are getting help from each other), relying solely on workers' self-report for what to discuss. Moreover, the action items generated from the meetings cannot be contextualized to the venues or situations in which they should be enacted (e.g., introduce potential helpers to a person at the next inter-team meeting). To that end, we envision future *ecosystem-wide meeting support tools* built upon Orchestration Scripts that can provide awareness of tracked work situations and any relevant context, a collaborative space to discuss them along with any other work needs a team identifies, and encode situated follow-ups *at other venues* that Orchestration Scripts would enact. In short, for future workplace tools to effectively support situated work, we need them to understand and surface the relevant context for the situations that the tools are designed for and help us enact any follow-up actions throughout our work process.

6.4 Limitations

Our work has two limitations that future work should consider. First, our user study focused on understanding how Orchestration Scripts can support situated work practices by using a Slack bot interface to surface strategies since the studied community used Slack across venues. Given the study's short length, we do not know how behaviors may change over an extended period or what potential issues are with this kind of intervention. For example, people may suffer from alert fatigue from being over-notified as time goes on. As a result, they may ignore helpful prompts about routine situations like collaboration venues or–more critically–ignore important prompts for emergent work situations that they are less aware of. As such, future work should conduct longer-term studies to understand how usage patterns with Orchestration Scripts change when using a bot-based interface [26, 29, 64].

Second, Orchestration Scripts was studied in the context of the ARS academic organization where goals and motivations for using the system may differ from a workplace. While ARS provides students and mentors with work processes and socio-technical support similar to those in a workplace, its primary goal is helping students learn effective work practices for conducting research. Learning effective work practices is also of emphasis in the workplace, but so is the increased importance of meeting work outcomes. While these motivations might differ, we expect that the kind of situated support our approach provides will still be valuable in the workplace. Given these differences, future work should consider studying our approach in workplaces to understand what aspects still work well and potential changes to make due to domain differences between workplaces and academic institutions.

7 CONCLUSION

This paper introduces workplace programming for situationally-aware systems, an approach to building workplace systems using computational abstractions of an organization's ways of working (i.e., its processes, social structures, and venues). We implemented this approach in Orchestration Scripts, a system for encoding effective work practices in terms of these abstractions. Through a case study and field study, we showed how our approach can encode different aspects of working effectively, and help people identify situations and enact effective strategies using support opportunities available across a work environment. These results show promise for how future technologies that use computational abstractions of how workplaces are organized can be developed to support situated work, for which technological supports are critically important.

ACKNOWLEDGMENTS

We thank the Design, Technology, and Research program at Northwestern University for accommodating our study; Neha Sharma and Fardeem Munir for advice on technical implementations; Mike Massimi, Christina Janzer, and Lucas Puente of Slack and the Future Forum for helpful discussions; and Ryan Louie, Noshir Contractor, members of the Delta Lab, and members of the CollabLab for feedback. Funding for this research was provided by the National Science Foundation under Grant No. IIS-1623635.

REFERENCES

[1] 2022. Asana. https://asana.com/product.

[2] 2022. GitHub REST API. https://ghdocs-prod.azurewebsites.net/en/rest.

[3] 2022. Google Drive API. https://developers.google.com/drive.

[4] 2022. IFTTT. https://ifttt.com/.

[5] 2022. Jira. https://www.atlassian.com/software/jira.

[6] 2022. Slack API. https://slack.com/.

[7] 2022. Slack Workflow Builder. https://slack.com/features/workflow-automation.

[8] 2022. Trello. https://trello.com/.

[9] 2022. The Trello REST API. https://developer.atlassian.com/cloud/trello/.

[10] 2022. Zapier. https://zapier.com/.

[11] Mark S. Ackerman, Juri Dachtera, Volkmar Pipek, and Volker Wulf. 2013. Sharing Knowledge and Expertise: The CSCW View of Knowledge Management. *Computer Supported Cooperative Work (CSCW)* 22, 4 (Aug. 2013), 531–573. https://doi.org/10.1007/s10606-013-9192-8

[12] Mark S. Ackerman and Christine Halverson. 2003. Sharing Expertise: The next Step for Knowledge Management. In *In Social Capital and Information.* MIT Press, 273–300. https://doi.org/10.7551/mitpress/6289.003.0015

[13] Mark S. Ackerman and Thomas W. Malone. 1990. Answer Garden: A Tool for Growing Organizational Memory. In *Proceedings of the ACM SIGOIS and IEEE CS TC-OA Conference on Office Information Systems (COCS '90).* ACM, New York, NY, USA, 31–39. https://doi.org/10.1145/91474.91485

[14] Mark S. Ackerman and David W. McDonald. 1996. Answer Garden 2: Merging Organizational Memory with Collaborative Help. In *Proceedings of the 1996 ACM Conference on Computer Supported Cooperative Work - CSCW '96.* ACM Press, Boston, Massachusetts, United States, 97–105. https://doi.org/10.1145/240080.240203

[15] Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R. Klemmer. 2010. Example-Centric Programming: Integrating Web Search into the Development Environment. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10).* Association for Computing Machinery, New York, NY, USA, 513–522. https://doi.org/10.1145/1753326.1753402

[16] Virginia Braun and Victoria Clarke. 2006. Using Thematic Analysis in Psychology. *Qualitative Research in Psychology* 3, 2 (Jan. 2006), 77–101. https://doi.org/10.1191/1478088706qp063oa

[17] Spencer Evan Carlson, Leesha V. Maliakal, Daniel Rees Lewis, Jamie Gorson, Elizabeth Gerber, and Matthew Easterday. 2018. Defining and Assessing Risk Analysis: The Key to Strategic Iteration in Real-World Problem Solving. (July 2018).

[18] Patrick Chiu, John Boreczky, Andreas Girgensohn, and Don Kimber. 2001. LiteMinutes: An Internet-based System for Multimedia Meeting Minutes. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01).* Association for Computing Machinery, New York, NY, USA, 140–149. https://doi.org/10.1145/371920.371971

[19] Alistair Cockburn. 2002. *Agile Software Development.* Addison-Wesley.

[20] Allan Collins, John Seely Brown, and Susan E. Newman. 2018. *Cognitive Apprenticeship: Teaching the Crafts of Reading, Writing, and Mathematics.* Routledge. 453–494 pages. https://doi.org/10.4324/9781315044408-14

[21] Luca Corcella, Marco Manca, Fabio Paternò, and Carmen Santoro. 2019. A Visual Tool for Analysing IoT Trigger/Action Programming. In *Human-Centered Software Engineering (Lecture Notes in Computer Science),* Cristian Bogdan, Kati Kuusinen, Marta Kristín Lárusdóttir, Philippe Palanque, and Marco Winckler (Eds.). Springer International Publishing, Cham, 189–206. https://doi.org/10.1007/978-3-030-05909-5_11

[22] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2019. Empowering End Users in Debugging Trigger-Action Rules. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19).* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300618

[23] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. 2004. A CAPpella: Programming by Demonstration of Context-Aware Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04).* Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/985692.985697

[24] Anind K. Dey, Timothy Sohn, Sara Streng, and Justin Kodama. 2006. iCAP: Interactive Prototyping of Context-Aware Applications. In *Pervasive Computing (Lecture Notes in Computer Science),* Kenneth P. Fishkin, Bernt Schiele, Paddy Nixon, and Aaron Quigley (Eds.). Springer, Berlin, Heidelberg, 254–271. https://doi.org/10.1007/11748625_16

[25] P. Dillenbourg and P. Tchounikine. 2007. Flexibility in Macro-Scripts for Computer-Supported Collaborative Learning. *Journal of Computer Assisted Learning* 23, 1 (Feb. 2007), 1–13. https://doi.org/10.1111/j.1365-2729.2007.00191.x

[26] Hyo Jin Do, Ha-Kyung Kong, Jaewook Lee, and Brian P. Bailey. 2022. How Should the Agent Communicate to the Group? Communication Strategies of a Conversational Agent in Group Chat Discussions. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (Nov. 2022), 387:1–387:23. https://doi.org/10.1145/3555112

[27] Amy C. Edmondson. 2012. *Teaming: How Organizations Learn, Innovate, and Compete in the Knowledge Economy.* John Wiley & Sons.

[28] Frank Fischer, Ingo Kollar, Heinz Mandl, and Jörg M. Haake (Eds.). 2007. *Scripting Computer-Supported Collaborative Learning: Cognitive, Computational and Educational Perspectives.* Springer US. https://doi.org/10.1007/978-0-387-36949-5

[29] Asbjørn Følstad, Theo Araujo, Effie Lai-Chong Law, Petter Bae Brandtzaeg, Symeon Papadopoulos, Lea Reis, Marcos Baez, Guy Laban, Patrick McAllister, Carolin Ischen, Rebecca Wald, Fabio Catania, Raphael Meyer von Wolff, Sebastian Hobert, and Ewa Luger. 2021. Future Directions for Chatbot Research: An Interdisciplinary Research Agenda. *Computing* 103, 12 (Dec. 2021), 2915–2942. https://doi.org/10.1007/s00607-021-01016-7

[30] Sharon Nelson-Le Gall. 1981. Help-Seeking: An Understudied Problem-Solving Skill in Children. *Developmental Review* 1, 3 (Sept. 1981), 224–246. https://doi.org/10.1016/0273-2297(81)90019-8

[31] Kapil Garg, Darren Gergle, and Haoqi Zhang. 2022. Understanding the Practices and Challenges of Networked Orchestration in Research Communities of Practice. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW (Nov. 2022), 344:2–344–28. https://doi.org/10.1145/3555764

[32] Diimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. 1995. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases* 3, 2 (April 1995), 119–153. https://doi.org/10.1007/BF01277643

[33] Eric Horvitz. 1999. Principles of Mixed-Initiative User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99).* Association for Computing Machinery, New York, NY, USA, 159–166. https://doi.org/10.1145/302979.303030

[34] Julie S. Hui, Darren Gergle, and Elizabeth M. Gerber. 2018. IntroAssist: A Tool to Support Writing Introductory Help Requests. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems.* ACM, 22. https://doi.org/10.1145/3173574.3173596

[35] Christopher M Johnson. 2001. A Survey of Current Research on Online Communities of Practice. *The Internet and Higher Education* 4, 1 (Jan. 2001), 45–60. https://doi.org/10.1016/S1096-7516(01)00047-1

[36] David Jonassen, Johannes Strobel, and Chwee Beng Lee. 2006. Everyday Problem Solving in Engineering: Lessons for Engineering Educators. *Journal of Engineering Education* 95, 2 (2006), 139–151. https://doi.org/10.1002/j.2168-9830.2006.tb00885.x

[37] David H. Jonassen. 1997. Instructional Design Models for Well-Structured and III-structured Problem-Solving Learning Outcomes. *Educational Technology Research and Development* 45, 1 (March 1997), 65–94. https://doi.org/10.1007/BF02299613

[38] Lars Kobbe, Armin Weinberger, Pierre Dillenbourg, Andreas Harrer, Raija Hämäläinen, Päivi Häkkinen, and Frank Fischer. 2007. Specifying Computer-Supported Collaboration Scripts. *International Journal of Computer-Supported Collaborative Learning* 2, 2 (Sept. 2007), 211–224. https://doi.org/10.1007/s11412-007-9014-4

[39] Nicolas Kokkalis, Chengdiao Fan, Johannes Roith, Michael S. Bernstein, and Scott Klemmer. 2017. MyriadHub: Efficiently Scaling Personalized Email Conversations with Valet Crowdsourcing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17).* Association for Computing Machinery, New York, NY, USA, 73–84. https://doi.org/10.1145/3025453.3025954

[40] Sam Lau, Sruti Srinivasa Srinivasa Ragavan, Ken Milne, Titus Barik, and Advait Sarkar. 2021. TweakIt: Supporting End-User Programmers Who Transmogrify Code. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3411764.3445265

[41] Jean Lave and Etienne Wenger. 1991. *Situated Learning: Legitimate Peripheral Participation.* Cambridge University Press.

[42] Daniel G. Rees Lewis, Jamie Gorson, Leesha V. Maliakal, Spencer E. Carlson, Elizabeth M. Gerber, Christopher K. Riesbeck, and Matthew Wayne Easterday. 2018. Planning to iterate: Supporting iterative practices for real-world ill-structured problem-solving. In *Proceedings of International Conference of the Learning Sciences, ICLS*, Vol. 1. International Society of the Learning Sciences, 9–16.

[43] Ryan Louie, Darren Gergle, and Haoqi Zhang. 2022. Affinder: Expressing Concepts of Situations That Afford Activities Using Context-Detectors. In *CHI Conference on Human Factors in Computing Systems (CHI '22)*. Association for Computing Machinery, New York, NY, USA, 1–18. https://doi.org/10.1145/3491102.3501902

[44] Wayne Lutters, Mark Ackerman, James Boster, and David McDonald. 2000. Mapping Knowledge Networks in Organizations: Creating a Knowledge Mapping Instrument. *AMCIS 2000 Proceedings* (Jan. 2000).

[45] David W. McDonald and Mark S. Ackerman. 2000. Expertise Recommender: A Flexible Recommendation System and Architecture. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*. ACM Press, Philadelphia, Pennsylvania, United States, 231–240. https://doi.org/10.1145/358916.358994

[46] Raul Medina-Mora, Terry Winograd, Rodrigo Flores, and Fernando Flores. 1992. The Action Workflow Approach to Workflow Management Technology. In *Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work (CSCW '92)*. Association for Computing Machinery, New York, NY, USA, 281–288. https://doi.org/10.1145/143457.143530

[47] Mariel Miller and Allyson Hadwin. 2015. Scripting and Awareness Tools for Regulating Collaborative Learning: Changing the Landscape of Support in CSCL. *Computers in Human Behavior* 52 (Nov. 2015), 573–588. https://doi.org/10.1016/j.chb.2015.01.050

[48] Robert C. Miller, Haoqi Zhang, Eric Gilbert, and Elizabeth Gerber. 2014. Pair Research: Matching People for Collaboration, Learning, and Productivity. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*. ACM, New York, NY, USA, 1043–1048. https://doi.org/10.1145/2531602.2531703

[49] Brad A. Myers, John F. Pane, and Amy J. Ko. 2004. Natural Programming Languages and Environments. *Commun. ACM* 47, 9 (Sept. 2004), 47–52. https://doi.org/10.1145/1015864.1015888

[50] Gary J. Nutt. 1996. The Evolution towards Flexible Workflow Systems. *Distributed Systems Engineering* 3, 4 (Dec. 1996), 276. https://doi.org/10.1088/0967-1846/3/4/007

[51] Soya Park, Amy X. Zhang, Luke S. Murray, and David R. Karger. 2019. Opportunities for Automating Email Processing: A Need-Finding Study. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, Glasgow, Scotland Uk, 1–12. https://doi.org/10.1145/3290605.3300604

[52] Walter Powell. 1990. Neither Market Nor Hierarchy: Network Forms of Organization. *Research in Organizational Behaviour* 12 (Jan. 1990), 295–336.

[53] Molly Pribble, Neha Sharma, Haoqi Zhang, and Leesha Maliakal Shah. 2022. MindYoga: Scaffolding the Metacognitive Reflection Process within Learning Ecosystems. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3491101.3519751

[54] Sadhana Puntambekar and Janet L. Kolodner. 2005. Toward Implementing Distributed Scaffolding: Helping Students Learn Science from Design. *Journal of Research in Science Teaching* 42, 2 (2005), 185–217. https://doi.org/10.1002/tea.20048

[55] Fazle Rabbi and Wendy MacCaull. 2012. T-Square: A Domain Specific Language for Rapid Workflow Development. In *Proceedings of the 15th International Conference on Model Driven Engineering Languages and Systems (MODELS'12)*. Springer-Verlag, Berlin, Heidelberg, 36–52. https://doi.org/10.1007/978-3-642-33666-9_4

[56] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. 1999. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*. Association for Computing Machinery, New York, NY, USA, 434–441. https://doi.org/10.1145/302979.303126

[57] Till Schümmer, Hilda Tellioglu, and Jörg M. Haake. 2009. Towards Living Agendas — Shaping the next Generation of Business Meetings. In *ECSCW 2009*, Ina Wagner, Hilda Tellioğlu, Ellen Balka, Carla Simone, and Luigina Ciolfi (Eds.). Springer, London, 263–282. https://doi.org/10.1007/978-1-84882-854-4_16

[58] Herbert A. Simon. 1973. The Structure of Ill Structured Problems. *Artificial Intelligence* 4, 3 (Dec. 1973), 181–201. https://doi.org/10.1016/0004-3702(73)90011-8

[59] Linn C. Stuckenbruck. 1979. The Matrix Organization. *Project Management Quarterly* 10, 3 (1979), 21–33.

[60] Lucy Suchman. 1993. Do Categories Have Politics? The Language/Action Perspective Reconsidered. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work 13–17 September 1993, Milan, Italy ECSCW '93*, Giorgio de Michelis, Carla Simone, and Kjeld Schmidt (Eds.). Springer Netherlands, Dordrecht, 1–14. https://doi.org/10.1007/978-94-011-2094-4_1

[61] Lucy A. Suchman. 1983. Office Procedure as Practical Action: Models of Work and System Design. *ACM Transactions on Information Systems* 1, 4 (Oct. 1983), 320–328. https://doi.org/10.1145/357442.357445

[62] Lucy A. Suchman. 1987. *Plans and Situated Actions: The Problem of Human-Machine Communication.* Cambridge University Press.

[63] Jeff Sutherland. 2014. *Scrum: A Revolutionary Approach to Building Teams, Beating Deadlines, and Boosting Productivity.* Penguin Random House.

[64] Carlos Toxtli, Andrés Monroy-Hernández, and Justin Cranshaw. 2018. Understanding Chatbot-mediated Task Management. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3173574.3173632

[65] Duane P. Truex, Richard Baskerville, and Heinz Klein. 1999. Growing Systems in Emergent Organizations. *Commun. ACM* 42, 8 (Aug. 1999), 117–123. https://doi.org/10.1145/310930.310984

[66] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. 2014. Practical Trigger-Action Programming in the Smart Home. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. Association for Computing Machinery, New York, NY, USA, 803–812. https://doi.org/10.1145/2556288.2557420

[67] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L. Littman. 2016. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 3227–3231. https://doi.org/10.1145/2858036.2858556

[68] Wil MP Van der Aalst. 2013. Business Process Management: A Comprehensive Survey. *International Scholarly Research Notices* 2013 (2013).

[69] Max Van Kleek, Brennan Moore, David R. Karger, Paul André, and m.c. schraefel. 2010. Atomate It! End-user Context-Sensitive Automation Using Heterogeneous Information Sources on the Web. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. Association for Computing Machinery, New York, NY, USA, 951–960. https://doi.org/10.1145/1772690.1772787

[70] Simone E. Volet. 1991. Modelling and Coaching of Relevant Metacognitive Strategies for Enhancing University Students' Learning. *Learning and Instruction* 1, 4 (Jan. 1991), 319–336. https://doi.org/10.1016/0959-4752(91)90012-W

[71] Marcel Walch, Michael Rietzler, Julia Greim, Florian Schaub, Björn Wiedersheim, and Michael Weber. 2013. homeBLOX: Making Home Automation Usable. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication (UbiComp '13 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 295–298. https://doi.org/10.1145/2494091.2494182

[72] Armin Weinberger, Ingo Kollar, Yannis Dimitriadis, Kati Mäkitalo-Siegl, and Frank Fischer. 2009. Computer-Supported Collaboration Scripts. In *Technology-Enhanced Learning: Principles and Products*, Nicolas Balacheff, Sten Ludvigsen, Ton de Jong, Ard Lazonder, and Sally Barnes (Eds.). Springer Netherlands, Dordrecht, 155–173. https://doi.org/10.1007/978-1-4020-9827-7_10

[73] Terry Winograd. 1987. A Language/Action Perspective on the Design of Cooperative Work. *Human-Computer Interaction* 3, 1 (March 1987), 3–30. https://doi.org/10.1207/s15327051hci0301_2

[74] Amy X. Zhang, Grant Hugh, and Michael S. Bernstein. 2020. PolicyKit: Building Governance in Online Communities. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)*. Association for Computing Machinery, New York, NY, USA, 365–378. https://doi.org/10.1145/3379337.3415858

[75] Haoqi Zhang, Matthew W. Easterday, Elizabeth M. Gerber, Daniel Rees Lewis, and Leesha Maliakal. 2017. Agile Research Studios: Orchestrating Communities of Practice to Advance Research Training. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW '17)*. ACM, New York, NY, USA, 220–232. https://doi.org/10.1145/2998181.2998199

[76] Valerie Zhao, Lefan Zhang, Bo Wang, Michael L. Littman, Shan Lu, and Blase Ur. 2021. Understanding Trigger-Action Programs Through Novel Visualizations of Program Differences. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–17. https://doi.org/10.1145/3411764.3445567