# Orchestration Scripting Environment: A System for Expressing Learning Situations in Networked Learning Communities

JASON FRIEDMAN, Northwestern University, USA
HANG YIN, Northwestern University, USA

Working well in networked communities requires that its members learn to access the support opportunities available for their needs, but this can be challenging for novices who have not worked in this way before. Mentors play an important role in teaching their mentees effective ways to get support from the network, but have limited awareness of how their mentees are accessing support across the network. Existing tools cannot help because they are designed to track task status or suggest predetermined actions rather than the effective ways of working in the community. We introduce Orchestration Scripting Environment, a block-based authoring environment that supports the encoding of learning situations into machine representations using available data points across a networked learning community. Results from a pilot study show that affordances such as intermediate concept variables and a visual workspace enable mentors to describe a learning situation for a networked community.

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

Additional Key Words and Phrases: authoring environments, networked learning community, block-based programming

**ACM Reference Format:**
Jason Friedman and Hang Yin. 2022. Orchestration Scripting Environment: A System for Expressing Learning Situations in Networked Learning Communities. In *CHI '22, April 30 - May 6, 2022, New Orleans, LA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/xxxxxxx.xxxxxxx

## 1 INTRODUCTION

Modern work and learning communities have become networked as they adopt agile working practices [12], project tracking and collaboration tools [1, 5, 7], and flexible organization structures [11] that enable them to respond to their members' work and learning needs. These are crucial to supporting people working on complex, ill-structured problems in which a person's needs and what support they need from the community may rapidly change as progress is made [10]. In other words, members in these networked communities need to access support throughout a network to succeed, but new members do not know how. Mentors can teach these network access skills, but they lack awareness of how mentees are getting help because mentees access the network in ways the mentors cannot see [8]. This results in many support opportunities for novices to learn and practice effective ways of working in a community being missed.

Despite the many project tracking tools that provide visibility into task status (e.g., Asana [1], Jira [5], Trello [7], Github [3]) and tools for orchestrating follow-up actions based on data triggers (e.g., IFTTT and Github issues [4]; Slack Workflows [6]), they cannot model situations in how people are accessing support in the network, or how they should be accessing support when they are working ineffectively. Instead, we require tools which can enable mentors to translate their models of the effective ways of working in a networked community into machine-understandable


Manuscript submitted to ACM

representations in terms of the structure of a community, when venues occur, which activities are appropriate in those venues, and the access strategies people use across the network.

In this paper, we introduce the *Orchestration Scripting Environment*: an authoring environment for expressing *learning situations* – opportunities to teach novices how to access the network for support – by enabling network access strategies to be modeled in terms of low level data points on venues, tools, and social structures in a networked community. Informed by needfinding with the members of a networked learning community for undergraduate research [14], our authoring environment provides data on the venues, tools, and social structures in the community, and guides mentors in identifying learning situations and expressing them in terms of those data. This advances existing tools by providing a structured environment designed to help mentors represent these ways of working in machine executable form.

To evaluate the efficacy of our system, we recruited four mentors from an undergraduate research community to use our system to construct programs for networked access strategies that they already practice with their students. We found that mentors were able to use scaffolding questions to identify and define a learning situation, concept variables to link high level concepts to intermediate concepts, an exhaustive list of blocks to find relevant data, and the block based visual environment to organize the components of a learning situation. These findings show that script construction environments with interface affordances for expressing learning situations can help people translate their understanding of human situations into machine understandable representations.

## 2 BACKGROUND

Mentors in networked communities can coach students on effective ways of accessing support in the community for their needs, but have limited awareness of how their mentees are working across the entire network that current tools cannot provide [8]. Project tracking tools including Trello [7] and Asana [1]) that are often used in the workplace only track task progress, and not how people are actually using the network (i.e., the venues, tools, and social structures in the organization) in the process. Some tools like Microsoft MyAnalytics (a.k.a. Viva Insights) collect data from sources in a networked community of employees (Outlook Email; Teams, Outlook Calendar; Microsoft 365; etc.) and suggest pre-determined actions based on those data (e.g., scheduling group meetings based on availability) [13], but are limited in the kinds of situations that can be expressed to the system. Rather, we require tools which allow people to express learning situations which are unique to the venues, tools, and social structures in their networked community.

Closer to this need, recent work on Affinder provided an authoring environment that supports expressing an author's concepts of situations in the physical world that support specific activities (e.g., consuming a hot drink on a cold day) using a machine's available data-points (e.g., location and nearby places; weather) [9]. It does this by providing several cognitive supports for helping authors articulate the concepts for situations they want to detect and linking them to data points a system can track. Our work builds upon these ideas by exploring the cognitive supports needed for mentors in networked communities for encoding learning situations. Our system uses interface features from Affinder including concept variables and a block-based programming environment, while adding scaffolding questions and a list of available data points to support mentors in expressing learning situations in terms of venues, tools, and social structures of the networked working community.

## 3 ORCHESTRATION SCRIPTING ENVIRONMENT

To support mentors in encoding their learning strategies as programs in terms of a community's venues, tools, and social structures, we designed the Orchestration Scripting Environment for constructing programs that represent learning situations and strategies for students to attempt; see Figure 1. Because students interact across many forms of support

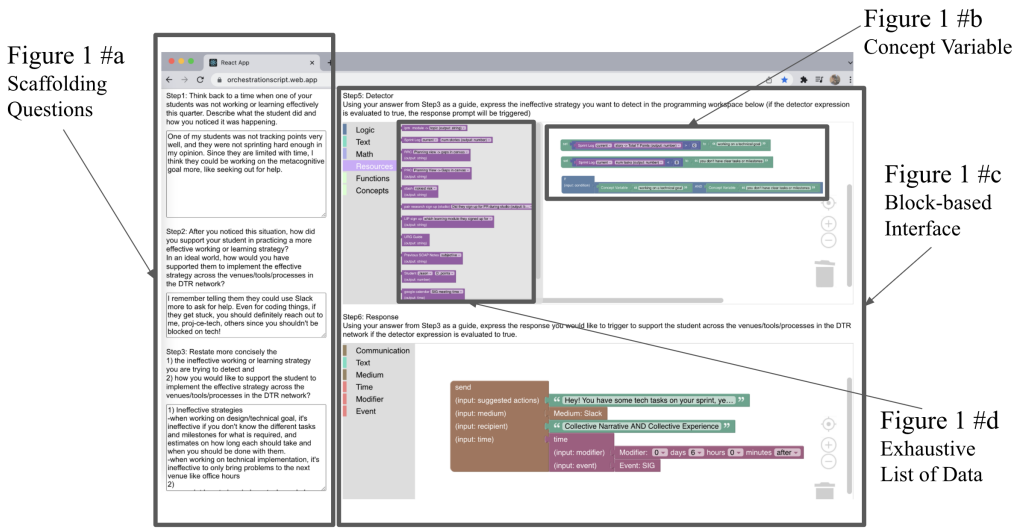


Fig. 1. The Orchestration Scripting Environment provides: (a) scaffolding questions to identify and define a learning situation; (b) concept variables to model intermediate situations; (c) a block-based interface to visually construct detectors and response; and (d) an exhaustive list of data to forage for relevant context detectors.

in the network each week, it makes it challenging for mentors to identify which data points from students' interactions with these tools are most relevant to modeling a learning situation. To that end, the *Orchestration Scripting Environment* supports: (1) scaffolding mentors in selecting a specific learning situation they want to model; and (2) a block-based workspace with concept variables for linking high level thoughts with intermediate concepts, affordances for visual organization and alignment of intermediate concepts and expressions, and an exhaustive list of data points to show mentors which data is available and relevant for their situation.

### 3.1 Scaffolds for Constructing Specific Learning Situations

For mentors to compose scripts for specific learning situations, they need to identify the situation itself and describe it concretely enough such that it can be modeled with the available data. During needfinding, we gave mentors a document of available data points and asked them to write pseudocode to detect a learning situation using the available data. Initially, their learning situations were too broad, which made it difficult for them to find relevant data points to construct the detection condition. To address this obstacle, we implemented a set of scaffolding questions that guide mentors in describing a learning situation in our prototype; see Figure 1 a. It does this by (1) having mentors reflect on when their students were being ineffective; (2) what strategies they recommended the students try; and (3) having them summarize their responses to (1) and (2) in preparation for composing the script.

### 3.2 Block-Based Workspace for Constructing Detectors and Responses

From prior needfinding, we learned that mentors struggled to write scripts with flexibility in a text-based language since there are no visual affordances to help mentors organize their thoughts. To tackle this problem, our system provides a block-based workspace using Google's Blockly library [2] which provides visual affordances for mentors to organize

their thoughts and easily combine data and logic; see Figure 1 c. We implement two features through this interface that supports the translation process: (1) intermediate concept variables; and (2) available list of data.

*3.2.1 Intermediate Concept Variables.* After identifying a specific learning situation, mentors need to translate their high-level idea of that situation into computer-executable scripts. A learning situation often consists of multiple components or conditions, which make it hard for mentors to encode at once. Instead, our system supports the creation of concept variables that resemble intermediate concepts with which mentors can break down high-level ideas into manageable chunks. For example, in Figure 1 b mentors can set an expression to a concept variable named "student is working hard." To combine this variable with other expressions and variables, mentors can simply create an instance of the concept variable with the same name and use it elsewhere.

*3.2.2 Browsable Display of Available Data.* While constructing a script for a specific situation, mentors need to include data points that are relevant to the situation. However, it's hard for mentors to remember or brainstorm all the required data points since there are many tools, venues, and social structures available across a networked community. To tackle this problem, our system includes an exhaustive list of data points that contains all the possible sources of data across a community; see Figure 1 d. By allowing mentors to scan through this list, they can discover data points they have forgotten or never thought of and include them in their scripts.

## 4 USER STUDY

To evaluate how the Orchestration Script environment enables mentors to express learning situations, we performed a pilot study in which we ask: *How will providing an authoring environment for determining learning situations and expressing them in terms of a community's venues, tools, and social structures help mentors compose programs for orchestrating activities in a networked learning community?*

### 4.1 Participants

We situate our study in an undergraduate research learning community at a mid-sized U.S. university that implements the Agile Research Studios [14] model. We recruited 1 faculty mentor and 3 Ph.D. students who mentor undergraduate student researchers for a 1 hour lab study. The faculty mentor has been with the community since its inception in Spring 2014, while the 3 Ph.D. students have mentored students between 1-4 years. In the results, we will refer to the participants as P1-P4.

### 4.2 Procedure and Analysis

During each user test, mentors would use our interface to create scripts for identifying learning situations and recommending activities to practice for those situations. During the first 10 minutes, mentors would write responses to scaffolding questions in our scripting environment interface based on mentoring interactions they recently had with their students. In the next 20 minutes, mentors would write a script to detect and respond to a learning situation detailed in the scaffolding questions using the block based programming interface. In the final 30 minutes, we interviewed mentors to understand how the scaffolding questions helped them identify a learning situation, how the programming interface helped them compose a script to detect and respond to the situation, what challenges there were, and to what extent they felt the system allowed them to express themselves.

Before each user test, each mentor was given informed consent and asked permission to take screen and audio recordings that were later transcribed for analysis. During each test, we screen recorded participants using our system

and collected audio recordings of them using the system and of the post-test interview. User tests were conducted in-person or over Zoom if the mentor was not available in-person.

Afterwards, we rewatched the screen recordings to identify actions and quotes which showed how our interface features helped or did not help mentors identify, define, and represent learning situations. Then, we extracted common themes from the observations. In what follows, we detail how different parts of the interface helped mentors in the process of expressing learning situations and strategies using our interface.

## 5 RESULTS

### 5.1 Scaffolding Questions Promote Brainstorming, Defining, and Narrowing of Learning Situations

All 4 participants used the scaffolding questions to brainstorm, define, and narrow down a learning situation to detect and respond to. For example, P2 used step1 to remember details of the learning situation he wanted to detect: *"[step 1] just made me think of a lot of these different details and a journey of when I first noticed it, how I saw it during a meeting, and how I saw it again coming up"*. P2 also described how the prompt *"primed my mind for all the ways I wanted to help them,"* such as giving their students examples, and talking about milestoning and help-seeking strategies. These prompts also helped with narrowing down learning situations. P4 says, *"wordings like 'one of your students' and 'what the student did',"* helped them choose specific venues and resources for their script.

### 5.2 Intermediate Concept Variables Link Scaffolding Prompts to Programming Interface

Concept variables enabled mentors to translate their written responses to scaffolding questions into intermediate concepts during the construction process for which they could then find low level data points to represent.[1] Writing down these higher-level concepts helped P1 to be, *"really organized in thinking,"* before constructing their final script. P2 wrote two concept variables that were closely related to the text he wrote in step 3, which allowed him to have representations for the things like, *"working on a technical goal and they're ineffectively doing the task."* P3 shared a similar sentiment, stating: *"It felt like a tidy way to package an idea that can be mapped to the stuff I wrote in the scaffolding questions. It gave me something that felt like it was linked."*

### 5.3 Block-Based Interface Enables Visual and Mental Organization of Learning Situations

Three of four participants mentioned that the block-based visual workspace helped them visually organize and think about the scripts they were writing. P1 used the response section to flesh out the ways he could help his students and think about his own strategies for helping, stating: *"In this tool, it's not just the notion of creating individual scripts but the notion of space...for me to think about the kind of things I need to think about and the kinds of strategies because these strategies are somewhere in [my mind]"* P2 shared similar thoughts, where they would move data blocks relevant to a concept variable they created close together during the authoring process: *"I ... visually organized it. I was like 'working on a technical goal'. [Blocks] knew which were relevant to technical goals, I brought them next to each other."*

### 5.4 Browsing Available Data Promotes Foraging for Relevant Context Detectors

As the construction process progressed, all four participants scanned through the resource blocks to identify relevant data points to use for writing the detection condition that they may not have considered before. P2, for example, came across helpful low level data points for a different concept when looking at data blocks for the "working on a technical

[1]While our test with P1 did not have the feature implemented at the time, we found that they replicated the behavior on their own using text boxes to note when they were creating these intermediate expressions. We added this feature for subsequent tests due to P1's extensive use of it.

goal" concept: *"[I] started realizing through just scanning that some of these [data points] would be helpful for both of those concepts"*. P4 said, *"resources blocks were super helpful because it really drills down into one of the structures students should be working with"*, which allowed them to think about how the student's planning log (a resource in the studied community that all students use) could provide the data necessary for their script.

## 6 DISCUSSION AND FUTURE WORK

Our paper contributes to the literature on bridging a person's understanding of a learning situation into machine executable code by providing mentors with a tool that is able to access data on venues, tools, and social structures in a community and construct learning situations with that data. We found that a structured series of questions (i.e. scaffolding questions) helps people identify and narrow down how they define situations they want to represent to machines. After defining a learning situation at a high-level, we found that the affordances of our authoring environment (i.e., concept variables; visual workspace; foraging through data blocks) helped our users link and refine their high-level learning situation from the scaffolding questions into machine executable programs. By providing people with authoring environments that include affordances for expressing situations in terms of the venues, tools, and social structures in the community, we enable people to translate their understanding of a human situation in which mentoring might be useful into machine executable code that other systems can monitor for. A limitation of our current work is that the scripts themselves were not executed. In future work, we hope to integrate our system with an execution engine to run the composed scripts in a networked learning community to understand their efficacy in helping novices learn effective ways of working. Further, we envision making the authoring process faster and less effortful by exploring how higher-level pre-built functions and data structures may enable scripts to be constructed more easily.

## REFERENCES

[1] 2022. Asana. https://asana.com/product.
[2] 2022. Blockly. https://developers.google.com/blockly.
[3] 2022. GitHub Issues. https://github.com/features/issues.
[4] 2022. IFTTT and Github Issues. https://ifttt.com/github.
[5] 2022. Jira. https://www.atlassian.com/software/jira.
[6] 2022. Slack Workflow Builder. https://slack.com/features/workflow-automation.
[7] 2022. Trello. https://trello.com/.
[8] Anonymized. 2022. Anonymized. (2022).
[9] 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, 18. https://doi.org/10.1145/3491102.3501902
[10] 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
[11] Linn C. Stuckenbruck. 1979. The Matrix Organization. *Project Management Quarterly* 10, 3 (1979), 21–33.
[12] 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
[13] Michael Winikoff, Jocelyn Cranefield, Jane Li, Alexander Richter, and Cathal Doyle. 2021. The Advent of Digital Productivity Assistants: The Case of Microsoft MyAnalytics. https://doi.org/10.24251/HICSS.2021.040
[14] 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