Collaborative Conversational Support
Across Contexts

Sreecharan Sankaranarayanan
Thesis Proposal
May 1, 2019

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA - 15213

Thesis Committee:
Carolyn Penstein Rosé, Language Technologies Institute and Human-Computer Interaction Institute, Carnegie Mellon University (Chair)
Alan W. Black, Language Technologies Institute, Carnegie Mellon University
Sharon Carver, Department of Psychology, Carnegie Mellon University
Graham Neubig, Language Technologies Institute, Carnegie Mellon University
Stian Håklev, École polytechnique fédérale de Lausanne
Pierre Dillenbourg, École polytechnique fédérale de Lausanne

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2019 Sreecharan Sankaranarayanan
Thesis Proposal
Abstract

Collaboration is a fundamentally human activity spanning almost every sphere of interaction including the school, the workplace, the home, and the social spheres. In order to support collaboration in the contexts associated with these spheres, it becomes important to understand what effective collaboration means in them, and what idiosyncrasies could come in the way of offering this support. We start with a construct called transactivity, the notion of original reasoning expressed in an utterance operating on the reasoning expressed in another. Transactivity is known to be associated with improved learning, knowledge integration, knowledge transfer, and effective collaborative problem-solving. However, prior work providing conversational support for transactive exchange in a classroom context provides a clue that factors unique to that context such as difficulty of the materials, and the skill level of the learner interacts with these positive effects. Related concepts in the workplace and organizational contexts also produce positive effects but point to context-specific factors interacting with them. In this thesis, we stress-test transactivity across more strikingly different contexts in the four different spheres of human collaboration identified above. In doing so, we expect to identify context-specific factors that might prevent its benefits from being realized, and infer a more nuanced set of best practices for generalizing the insights from one context to a broader set of contexts.

First, in an online classroom context, we use a transactivity-based team formation strategy to provide team recommendations to students. We find that student preferences encourage them to undermine these recommendations undercutting the ability of transactivity to have a positive impact. Second, we investigate an instructional scaffold for collaboration programming called Online Mob Programming (OMP), that involves learners being supported by a conversational agent in taking up interdependent roles. We see that the instructional materials not being well adapted to learner prior knowledge potentially interferes with the benefits of transactivity. Further, some functions of the scaffold mirroring the expected benefits from transactivity could wash out benefits from it.

In proposed work, we intend to use the data collected in the online classroom, to learn reinforcement learning-informed dialogue strategies for better agent-support, and for generating code explanations that can provide additional shared context in support of knowledge co-construction. Further, we propose to extend this work to the community college context, allowing us to test the generality of injecting workplace-relevant collaborative experiences into university learning environments in another context where employment is the salient goal. Finally, in the home sphere, we propose the use of the conversational agent to engage patients and their caregivers in discussions about living healthfully together post a life-altering medical procedure and in the social sphere, we propose to use the conversational agent to support transactive exchange between individuals with opposing political inclinations in order to achieve the related notion of civil political discourse.
# Contents

1 Introduction

1.1 Transactivity in the Other Spheres .................................................. 2
1.2 Conversational Support for Transactivity Across Contexts .................. 2
1.3 Overview of Completed and Ongoing Work ...................................... 3
1.4 Overview of Proposed Work .......................................................... 3
1.5 Proposed Timeline ........................................................................... 5

2 Transactivity-Based Team Formation as a Choice

2.1 Motivating Transactivity-Based Team Formation ............................. 8
2.2 Prior Work on Automated Team Formation ..................................... 9
2.3 Course Context and Intervention .................................................... 10
  2.3.1 Reflection-Feedback Setup ....................................................... 11
  2.3.2 Transactivity-Based Team Formation ....................................... 12
  2.3.3 Team Recommendation .......................................................... 14
2.4 Methods ......................................................................................... 15
2.5 Analysis and Results ....................................................................... 15
  2.5.1 Online Discussion Quality ....................................................... 15
  2.5.2 Team Formation Processes ....................................................... 16
  2.5.3 Team Work ............................................................................ 16
2.6 Follow-Up Study ............................................................................ 17
2.7 Analysis and Results ....................................................................... 19
  2.7.1 Online Discussion Quality ....................................................... 19
  2.7.2 Team Formation Processes ....................................................... 19
2.8 Discussion ....................................................................................... 19

3 In Process Support for Teams in Project-Based Courses .................... 21

3.1 Related Work ................................................................................. 22
3.2 The Design of Online Mob Programming ....................................... 23
  3.2.1 Mob Programming ................................................................. 23
  3.2.2 Adapting Mob Programming for Pedagogy ............................... 24
  3.2.3 Bringing Mob Programming Online ....................................... 24
3.3 Experimental Setup - Study 1 ......................................................... 26
  3.3.1 Course Context ....................................................................... 26
  3.3.2 Course Timeline ..................................................................... 26
Chapter 1

Introduction

“If you want to go fast, go alone. If you want to go far, go together.”

- Unknown

Collaboration is a fundamentally human activity spanning almost every sphere where humans interact with each other, including the school, the workplace, the home and the social spheres. Understanding how to support collaboration in the contexts associated with these spheres involves understanding what effective collaboration means in that context and discovering the idiosyncrasies that might come in the way of offering support including the participants, their motivations, goals and prior preparedness, and the setting for the support among other factors. We start therefore, with the socio-linguistic notion of transactivity, which emerges from a Piagetian learning paradigm that proposes a link between transactive reasoning and collaborative knowledge integration.

A transactive pair of utterances can be understood as one conversational utterance that contains original reasoning that builds off of, or operates on the reasoning of another utterance [12]. In this learning paradigm, learners operating off of each other’s reasoning provides opportunities for making internal beliefs explicit, allowing for the emergence of cognitive conflict as a result of the potential dissonance between their ideas and that of their partner. Providing support for the emergence of these opportunities therefore, can result in cognitive restructuring and learning [20]. Further, even outside of the Piagetian learning paradigm, Schwartz argues from the Vygotskian perspective for instance that learners offer each other a kind of mental scaffolding with one learner’s articulated reasoning serving as an initiator for another’s ideas [93]. Transactivity is known to be higher in teams where there is mutual respect [6] and a desire to build common ground [35]. Further, higher transactivity teams are associated with higher learning [51 [102], higher knowledge integration [20], higher knowledge transfer [36], and effective collaborative problem solving [6]. In more recent work, transactivity was used as an indicator of collaboration potential to form groups with significantly improved group product quality [113 [114].
1.1 Transactivity in the Other Spheres

Although transactivity, and related notions such as productive agency [93], knowledge co-construction [111], common ground, intersubjectivity, and cognitive convergence [102], have been investigated primarily in learning contexts, there are several notions in the contexts associated with the other spheres of investigation in this thesis that are connected to transactivity.

The organizational psychology literature for instance, points to the notion of information elaboration [106] which is described as a “complex form of communication that involves the exchange of information and perspectives, the process of feeding back the results of this individual level processing into the group, and discussion and integration of its implications” [106]. These processes extend beyond information sharing to capture the extent to which team members elaborate on their ideas, and spend time constructively discussing each others perspectives, integrating information, and determining how to apply their knowledge resources to the problem at hand [45]. Information elaboration has further been linked to team performance [80] which is a related metric that becomes important in the work sphere apart from individual learning. Similar to transactivity’s use as an indicator for collaboration potential, the notion of Collective Intelligence [116] is also meant to predict the general ability of a group to work together across a range of tasks. It is defined and measured using collections of individual and group characteristics that have been demonstrated to correlate with properties of group interactions, and even more importantly, outcomes of group interactions [87].

In the social sphere, the notion of civil productivity [73] can be related to the ideas of mutual respect and common ground that are known to exist in highly transactive groups. Transactivity can therefore serve as the theoretical underpinning of effective collaboration in all of the contexts associated with the spheres of support that this thesis is targeting.

1.2 Conversational Support for Transactivity Across Contexts

Although several benefits of encouraging transactive discussions have been outlined above, some prior work gives us a clue that the idiosyncrasies of contexts could prevent these benefits from being realized. Adamson et al. [11] in their work, used academically-productive talk [67] prompts to encourage students to elaborate on their own reasoning while challenging and extending the reasoning of their teammates. They found that this form of support can significantly increase learning, but that the effect was context specific. In particular, the effectiveness of each strategy depended on factors such as difficulty of the material (new versus review) and the skill level of the learner (public high school versus private university). In this thesis, we stress-test conversational support for transactivity even harder, by comparing across contexts that are more strikingly different. In the school sphere, we first investigate transactivity as the basis for team formation and offer transactivity-based team recommendations to students in an online course offered to university students. We see in this context that student preferences drive them to undermine the recommendations preventing transactivity from having the chance to produce benefits. We then investigate a scaffold for collaborative programming online offered to both working professionals and university students. We see here that the scaffold might serve a
purpose similar to transactivity and wash out its benefits. In proposed work, we investigate transactivity across the other two spheres, the home, where, as in the school sphere, the lack of willingness to do an activity might interfere and the social sphere, where emotions might interfere. Across all of these spheres, we expect to verify the benefit of transactivity for learning but uncover factors that might interfere with realizing this benefit. We will also be able to infer nuanced set of best practices about the parts of the insights from studies in one context that can generalize to a broader set of contexts.

1.3 Overview of Completed and Ongoing Work

We first take a transactivity-based team formation strategy that has been shown in prior work to be better than randomly formed teams both in a synthetic lab context and a real-world online course context [113, 114] and apply it to the classroom context. In this context, the transactively formed teams are provided as recommendations to students rather than forced on them. We find that student preferences in this context encourage them to undermine these recommendations undercutting the ability of transactivity to have a positive impact. This work is outlined in Chapter 2 and is situated in the school sphere shown in 1.1.

We then investigate an instructional scaffold for collaborative programming online called Online Mob Programming (OMP), that involves learners being supported by a conversational agent in taking up different roles that are designed to be interdependent. Learner uptake of this scaffold was studied first in an online course offered to working professionals. In the subsequent classroom study, we compare this scaffold between individuals, transactively-formed and random teams and see that the instructional materials not being well adapted to learner prior knowledge potentially interferes with the benefits of transactivity playing out. Further, some of the functions of the scaffold itself mirrors the expected benefits from transactivity and could therefore wash out further benefits from it. In current work, we are investigating the potential learning gains from this scaffold alone and intend to conversationally support the emergence of transactive discussions in future studies. We also intend to extend our learnings from the university classroom context to the community college context where employment is a much more salient goal. This work is outlined in Chapter 3 and is situated at the intersection of the school and work spheres shown in 1.1 because it is an industry-inspired paradigm that was first investigated in the work sphere before being imported into the classroom context.

1.4 Overview of Proposed Work

In proposed work, we intend to use the in-process code and conversational data generated in parallel in the OMP studies to produce explanations of the current state of code. These code explanations can be used to provide additional shared context for idea co-construction especially for learners in roles that aren’t directly involved in writing the code. We also intend to use the data from several runs of the OMP study to learn reinforcement learning based strategies for offering adaptive conversational agent-support to learners in future studies. This proposed work is outlined in Chapter 4.
Finally, we intend to investigate conversational agent-support for transactivity in two other spheres, the home sphere and the social sphere. In the home sphere, the conversational agent will be used to engage patients and their caregivers in discussions about how they can live healthfully together post the patient undergoing a lifestyle-changing medical procedure. A card-sorting activity described in Chapter 5, is designed to reduce the influence of emotions in discussions about especially sensitive topics that might prevent the benefits of transactivity from playing out.

In the social sphere, the conversational agent will be used to prompt participants with opposing political inclinations to build off of each other’s reasoning towards achieving mutual respect. The associated notion of civil political discourse is measured to see if conversationally supporting transactive exchange can support civil political discourse also. This is described in Chapter 6.
## 1.5 Proposed Timeline

<table>
<thead>
<tr>
<th>May 2019</th>
<th>ijAIED Article Submission</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 2019</td>
<td>SIGDIAL Article Submission</td>
</tr>
<tr>
<td>May 2019</td>
<td>Prep for work with Community Colleges</td>
</tr>
<tr>
<td>May 2019 - Aug 2019</td>
<td>Internship at Facebook Inc.</td>
</tr>
<tr>
<td>Aug 2019 - Dec 2019</td>
<td>Code Explanations and Reinforcement Learning Experiments</td>
</tr>
<tr>
<td>Aug 2019 - Dec 2019</td>
<td>Community College and CMU Mob Programming Study</td>
</tr>
<tr>
<td>Jan 2019 - Apr - 2020</td>
<td>Community College and CMU Mob Programming Study 2</td>
</tr>
<tr>
<td>Apr 2020 - May 2020</td>
<td>Dissertation Writing</td>
</tr>
<tr>
<td>May 2020</td>
<td>Dissertation Defense</td>
</tr>
</tbody>
</table>
Chapter 2

Transactivity-Based Team Formation as a Choice

The importance of teamwork in the workplace has increased dramatically over the last few decades as organizations increasingly move towards using workplace teams for projects. Reports from the United States [5], Canada [40], the United Kingdom [71], Australia [32], Eastern Europe [96], and China [120] express the view that teamwork and related interpersonal skills are as important, if not more important than technical skills for a graduating student. Teamwork has thus moved from being simply "desirable" to "essential" [81]. This demand is especially salient in the software engineering workplace where practices like Agile software development necessitate collaboration not just within a team but across cross-functional teams in order to deliver products of value to the end user [11]. In response to these rising demands, team-project based courses have become an important staple in the higher education curricula valued by both faculty and students alike not just for the opportunities they provide to integrate and apply knowledge while refining skills learned in more basic courses, but also for their relevance to industry-standard practices.

While team projects could provide a realistic experience in terms of group processes such as group decision making, communication, and leadership, while helping students deliver projects larger than they could have completed alone and learn from each other in the process [64, 66, 79], they are plagued with several issues preventing this ideal from being met. First, in many cases, students are simply placed into teams without adequate preparation resulting in ineffective performance while working in those teams [7, 25, 39]. Second, there are group process issues that can arise once students are placed into teams including mismanagement resulting from lack of structure of group work, unequal distribution of labor manifesting in both free-riding and few individuals doing a lion’s share of the work, and the possibility of conflict as a result [17, 65]. In fact, the anecdotal experience of many is that the inherent reward structure resulting from grades based on the quality of the end-project in team-project based courses fosters a performance orientation, where the most capable students take on the lion’s share of the work, doing tasks they already know how to do well, while undercutting the opportunity for other students to practice and for the group to reflect on underlying concepts. This problem is also salient in the workplace where a productivity orientation when faced with impending deadlines undercuts learning opportunities.
In an effort towards investigating technology support that can promote a balance of productivity and learning, we take a two-pronged approach. We first investigate, in a series of studies, intelligent transactivity-based team formation methods that better prepare students for teamwork. We then introduce a role-taking paradigm for structuring team-projects in software engineering courses called Online Mob Programming with the aim of addressing the group process issues outlined earlier including lack of structure of group process and uneven distribution of work. This chapter describes a series of studies of transactivity-based team formation in an online cloud computing course offered at Carnegie Mellon University and its various campuses worldwide (Rwanda, Adelaide, Qatar and Silicon Valley). As opposed to prior studies where students were placed into the teams formed by the algorithm, this study provides them as recommendations instead. We see that student preferences encourage them to undermine the recommendations undercutting the ability of transactivity to have a positive impact. On the other hand, students that formed teams based on observed activity on the platform, a mechanism similar to that used by the algorithm, performed just as well as the transactively formed teams.

2.1 Motivating Transactivity-Based Team Formation

Theoretical frameworks like social constructivism [107] and connectivism [94] outline the many benefits of group collaborations. When designed well, collaborative learning experiences in the offline context have been shown to provide many cognitive, meta-cognitive and social benefits to learners [54, 110]. Students when learning with peers have shown improved levels of conceptual understanding and engagement producing downstream effects on course performance [18, 95] and course completion [77]. These positive effects however, result from intentional pedagogy and support for peer interaction without which peer learning has been found to fail [16]. Efforts toward replicating this success of peer learning from offline contexts in Massive Open Online Course (MOOC) contexts has been met with mixed results. Attempts to encourage unstructured discussions using real-time chat for instance, did not find improvements in students’ retention rate or academic achievement [15] whereas chat facilitated by an intelligent agent led of an approximately 50% reduction in dropout [26].

Early work on MOOCs further revealed that although online students call for more social interaction, peer learning opportunities fail without support specifically designed to cater to the needs of online learners [55]. Students may fail to provide feedback on the work of their peers; they may not show up for a collaboration session they signed up for; or they may quit working with their team altogether as they drop out of the course. Learners have reported facing more frustration with groups formed online than in face-to-face learning environments [95]. Simply providing communication technology has also proven insufficient. For instance, an early MOOC that offered optional learning groups found that only 300 out of a total of 7350 participants in the course signed up for one of the 12 learning groups [56]. One explanation is that students don’t fully recognize the role that social interaction is meant to play in their learning suggesting a need for much more intentional and tighter coupling of social interaction opportunities with the course content and pedagogy [98]. The importance of offering effective support for social interaction in online learning contexts as a research problem has thus
been established. In these series of studies, we investigate a particularly challenging aspect of this broader agenda, namely supporting team-based project learning at scale in online learning contexts.

While much of the research on collaborative learning experiences in online contexts focuses on informal discussion forums or short-term chat activities, some recent work tries to tackle the challenge of importing team-based projects into MOOCs [109, 112, 113, 114]. Crafting effective teams can be split into two phases - Team formation, and support for coordination and communication once teams are formed. Forming well-functioning teams has been a particularly challenging problem which has been further compounded by the need for any team-formation method to be scalable [28, 61, 121]. Prior work has thus investigated evidence from the behavior traces of students that can be used to do the team formation in an automated fashion [30, 72, 74, 112, 121] providing evidence of some success as well as many failures.

In this work, we take one such approach for automated team formation that has been successful both in a large controlled lab study [113] and in a 4-week MOOC [114] and apply it to two successive semester-long offerings of a large online for-credit project-based course context. While it is clear that the empirically grounded transactivity-based team formation approach we used, worked well in a context where the collaboration was short-lived and the stakes were low, it becomes imperative to investigate the impact that a much longer-term, higher stakes collaboration will have on the design of the paradigm. The first study which investigated the application of the paradigm to the fall 2017 offering of a 16 week large online project course on cloud computing that students took as a part of degree-granting programs at Carnegie Mellon University found evidence of some success as well as several design insights on how this paradigm might need to adapted to this new context [86]. Specifically, working with friends turned out to detrimental to students both in terms of student grade and in terms of relationship between team members. Following the iterative design-based research approach, the second study, conducted in the spring 2018 offering of the course, presented students with this finding and investigated the team formation strategies they used in the face of this new information. Results once again provide evidence of some success of the paradigm but more importantly, shed light on decision making processes that students use when provided the agency to choose their own teams.

2.2 Prior Work on Automated Team Formation

Algorithmic team formation approaches that have emerged have seen both successes and failures. Attempts at providing support for team formation have sparked research on criteria that leads to better teams and algorithms that can then optimize over those criteria. Many automated approaches to team formation base their team assignments on characteristics of individual learners such as learning style, personality or demographic information [21]. This information however needs to first be assessed or discovered before it can be provided to the algorithm for optimization. Therefore, these approaches are often not feasible in typical online course environments. Further, forming teams based on typical demographic features such as gender and time zone has not been shown to significantly improve team engagement or success in MOOCs [121].
Approaches to automated team formation that have succeeded have focused on inducing buy-in among the participants. Opportunistic group formation for instance, triggers a negotiation process between learners to form groups once it detects that the learners can move from the individual learning phase to the group learning phase. The negotiation process allows learners to be assigned to roles based on their learning goals and the goals for the whole group thus creating buy-in. The approach showed that learners using the framework performed as well as students in face-to-face situations. Another such successful strategy has made use of a collaborative process measure called transactivity, which can be defined as the reasoning of one utterance building off or operating on the reasoning of another utterance. The construct of transactivity stems from a Piagetian learning paradigm where it is believed to flourish in social settings that have a balance of respect and a desire to build common ground. Groups that exhibited high transactivity were shown in previous studies to be associated with higher learning, higher knowledge transfer and better problem solving.

In the case of team formation, it is the social underpinnings, the signal of mutual interest and respect that a transactive exchange entails that renders this construct an estimate of collaboration potential between students. The automated strategy makes use of evidence of transactive exchanges in a whole course online discussion process that happens prior to the team activity. The resultant teams were shown to perform better than random teams first in a synthetic environment on Amazon Mechanical Turk and then externally validated in a Team-Based Massive Open Online Project-Based (TB-MOOP) course offered on edX.

2.3 Course Context and Intervention

The Cloud Computing course offered on Carnegie Mellon University’s (CMU) Open Learning Initiative (OLI) platform is a semester-long completely online project-based course offered to university students from CMU’s various campuses. The studies described in this paper were conducted in two subsequent semester-long offerings of the course. The first study was conducted in the Fall 2017 offering which saw participation from the Pittsburgh, Silicon Valley and Rwanda campuses of CMU. The general course practice followed in 9 prior offerings of the course required students to complete 10 individual projects and then self-select into teams for a 7-week team project. In each unit, students receive conceptual instruction and assessment on the OLI platform before proceeding to a homegrown platform called TheProject.Zone to complete their individual and team programming projects. Students do not have an opportunity to meet face-to-face as a part of the course because there is no in-person lecture. Instead, they get an opportunity to interact with other students and the course staff on the class Piazza, an online question-answer forum. Students who are co-located on the same campus can of course encounter each other outside of the course and based on the expectation set by prior offerings of the course, students often started to form their own teams from the very beginning of the course.

Two changes to these established course practices needed to be made in order to allow us to

1 http://oli.cmu.edu/
2 http://www.cs.cmu.edu/ msakr/15619-f17/
3 https://theproject.zone/
provide team-member recommendations using Transactivity-Based Team Formation. The first change introduced a reflection and feedback component to each of the 10 individual projects that students had done in preparation for the team project. After each individual project, students were required to reflect on their experience in the individual project and respond to specific prompts. These responses would be posted to the discussion forum where they were also required to read and provide feedback to at least three other student reflections. This Reflection-Feedback setup first of all engendered a lot more social interaction between students but additionally provided us with the opportunity to observe these social interactions and mine them for evidence of collaboration potential in the form of transactive feedback contributions. An automated measure of transactivity exchange between students in this context was first used to estimate pairwise collaboration potential, and then a constraint satisfaction algorithm was used to assign teams in such a way that students were more likely to be part of teams with the other students they have interacted with transactively than those they have not interacted transactively with. The second change took the teams formed using the Transactivity-Based Team Formation paradigm and offered them as suggestions to the students rather than asking them to self-select into teams as had been done in the past. In an effort not to upend established course practices completely and also preserve student agency in the process of selecting teams, we did not force students to take our recommendations. Instead, we simply provided them as suggestions allowing us to observe the extent to which receiving recommendations would be viewed as attractive by the students.

2.3.1 Reflection-Feedback Setup

After each individual project, students responded to a series of prompts like the ones shown below -

- Pick a task you found most challenging. Why was it challenging and how did you end up solving it?
- Pick a task and choose among different solutions paths for this task. What were the trade-offs you ended up making?
- Describe how you tested one of the tasks. How did you design your test? Was your initial test sufficient? If not, how did you improve it?

Their responses were then shared as a reflection post to the discussion forum that the entire class could access. Students were then encouraged to provide constructive feedback to at least 3 of their peers‘ feedback posts. An example reflection, prompt and feedback post can be seen in Fig. 2.1. Substantive discussions resulted from this reflection-feedback setup as shown in the example feedback post. These posts showed evidence of students synthesizing knowledge from several posts, achieving common ground and providing encouragement to each other. An example of a transactive and non-transactive exchange between students in the Reflection-Feedback setup is shown below -

- **Transactive Exchange**
  - **Student 1**: “...I used ‘f.readlines()’ to read the wiki log file. It worked well on my own computer, but it caused a ‘MemoryError’ when I tested it on AWS…”
  - **Student 2**: “The file object itself is a iterator. So if you ‘for x in file’, you get lines as x. This
I really spent a lot of time in this project (more than 40 hours).
In the first 15 hours, I was all considering the workflow of this project. I read all the relative primers of this project, and I even created a
S3 bucket and installed AWS SDK and AWS eclipse plug-in. Most of the things I tried proved to be no use. Waiting much time here really
reflects my lack of background knowledge.
When I do the filter job, the most difficult part is how to keep it encoding in UTF-8 everywhere. I have few experiences programming in
multiple environments. It really cost much time for me to find a way to make it consist in UTF-8.
For the questions, to be honest, I have never used awk and grep before, and I am even not familiar with the Linux basic commands. I
finished those questions while learning from them, and I found awk and grep very powerful.
I was mainly using JAVA in the project, while in q8 I used Python. Pandas is really an amazing lib to solve such problems, so I chose to
switch to Python even I almost have no previous Python programming experience. It works very well, and I felt it crazy that the problem
is solved in only few lines.
Another thing I felt interesting is the regex. I spent a long time working on it, especially in q8. Previously I have seen someone use things
like `\s` to represent zero or more occurrences of any characters, and now I know how did they come up with such a rule.

Arrow
CloudComputingBot
Please provide constructive feedback to your peer’s post and substantiate your criticisms/suggestions with reasons. Evaluate the
arguments presented, be clear with your arguments and use supporting evidence.

My situation is a little similar to yours, but don’t worry about spending lots of time on primers. I think you’ll save a lot of time in the
following projects because of doing this. I’ve never used python before, either.
Besides, according to classmates summaries of this project, I think python is really efficient when solving certain problems compared to
java. Congratulations that you have learned to use it in q8!

Figure 2.1: Examples of a reflection post, a prompt soliciting constructive feedback and a feedback post. Feedback post highlights instances of common ground, synthesis and encouragement.

is a more pythonic way than using `readline()`:

- Non-Transactive Exchange
  Student 1: “...I approached the problem by breaking it out into different modules and functions which made it possible to test different cases really fast. ...”
  Student 2: “Well done!”

In the first case, the second student is referring explicitly to the reasoning of the first student and building on that reasoning further with their own reasoning. The interaction is therefore transactive. In the second case, the second student is referring to the reasoning of the first student but is not contributing original reasoning of their own and the interaction is therefore non-transactive. Evidence of transactive exchanges can thus be mined from these interactions to automatically inform our team formation algorithm.

2.3.2 Transactivity-Based Team Formation

Data from the feedback exchange in the Reflection-Feedback setup was used as input to the transactivity-based team-formation algorithm. Before an estimate of pairwise transactivity ex-
change can be computed, posts from the discussion forum must first be annotated as transactive or not. In our work, this was accomplished using a text classification approach developed in prior work on automated collaborative learning process analysis [3, 51, 113]. This approach requires training data including a validated and reliable coding of transactivity [35]. For our work, we used a previously validated coding manual [35] and coded 200 feedback exchanges by hand. Using this training data, we trained a model to perform the transactivity analysis over the whole set automatically.

Team assignment was based on behavior traces for the first five weeks of the course. By that point, students had completed 3 individual assignments and had written a total of 1007 discussion forum posts. For each pair of students, we computed the total number of threads where either they both contributed a transactive post to the discussion or one of them started the thread and the other contributed a transactive post. We refer to this quantity henceforth as the Pairwise Transactivity Score for this pair of students. Once the Pairwise Transactivity Score is computed for each pair of students, a team score can be computed by averaging the Pairwise Transactivity Score for each pair within the group. A score for the resulting teams across the whole class can be computed by averaging across the team scores. The goal of the automated team-matching algorithm is to assign students to teams in such a way that the score over the whole class is maximized. An exhaustive search would take inordinately long. Thus, a constraint satisfaction algorithm is used to find an approximate solution that comes close to the optimal assignment that maximizes the score across the class without having to do an exhaustive search. The specific constraint satisfaction algorithm we used is called the minimal cost max network flow constraint satisfaction algorithm [2]. The algorithm generally tackles the resource allocation problem with constraints and in prior work, role assignments such as the roles of a Jigsaw condition were used as constraints [113]. In this paper, location was used as the constraint in addition to maximizing average transactivity across teams i.e., all members of the team are to be located on the same CMU campus. This was because, based on past runs of the course, it was observed by the course staff that co-located teams worked better together and co-location was an expressed desire of students who had taken the course in the past also. The algorithm finds an optimal grouping within $O(N^3)$ time complexity where $N$ is the number of students. A brute force approach would have $O(N!)$ time complexity and would be infeasible in practice.

The algorithm is capable of forming teams of arbitrary size and approximates the solution in admissible time by maximizing the transactivity post count between two adjacent pairs of users instead of the total accumulated transactivity post count. A discussion network which is a directed weighted graph of the student’s discussion in the reflection-feedback phase weighted by the transactivity score is built and the successive shortest paths algorithm shown in Algorithm 1 greedily finds the minimum cost flow until there is no remaining flow in the network.

The algorithm can be extended to accommodate more than one constraint but it is important to note that adding additional constraints could mean that an optimal team assignment ceases to exist.
Algorithm 1 Successive Shortest Paths for Minimum Cost Max Flow

1: \( f(v_1, v_2) \leftarrow 0 \ \forall (v_1, v_2) \in E \)
2: \( E' \leftarrow \forall (v_1, v_2) \in E \)
3: \textbf{while} \( \exists \Pi \in G' = (V, E') \text{ s.t. } \Pi \text{, a minimum cost path from source to destination} \ \textbf{do} \)
4: \textbf{for each} \( (v_1, v_2) \in \Pi \) \textbf{do}
5: \textbf{if} \( f(v_1, v_2) > 0 \) \textbf{then}
6: \( f(v_1, v_2) \leftarrow 0 \)
7: \text{remove} \(-a(v_2, v_1)\) \text{from} \( E' \)
8: \text{add} \( a(v_1, v_2) \) \text{to} \( E' \)
9: \textbf{else}
10: \( f(v_1, v_2) \leftarrow 1 \)
11: \text{remove} \( a(v_1, v_2) \) \text{from} \( E' \)
12: \text{add} \(-a(v_2, v_1)\) \text{to} \( E' \)
13: \textbf{end if}
14: \textbf{end for}
15: \textbf{end while}

2.3.3 Team Recommendation

At the end of Week 6, automated team assignments were formed using data from the reflection-feedback setup through Week 5 and then sent by the course instructor to the students who belonged to the same team over email. The email provided information to the students about how the teams were formed, what empirical evidence was used to form the teams and also served as their introduction to their teammates. Importantly, the email highlighted that the team assignments were only a recommendation and not a prescription from the course staff. An excerpt from the email is given below -

“At this point in the semester, it is time to get organized into teams for the course team project work. Up until now, your work has been entirely individual, though you have shared it with the class community, and have offered each other feedback. Research in online team-based learning suggests that some aspects of interactions in a public class space signal collaboration potential. In particular, these aspects relate to expression of ideas and ways of evaluating the ideas and perspectives of others. Based on observation of your participation in the online feedback activities in this class, that prior research suggests you would work well together. Please consider this recommendation as you make your official declaration of team commitment for this course.”

Students were then asked to deliberate over these recommendations and their final team assignments were to be submitted on the TheProject.Zone. 35 total teams were formed with 23 teams in the Pittsburgh campus, 11 teams in the Silicon Valley campus and 1 team in the Rwanda campus. All suggested teams had 3 members each except the Rwanda team which had 2 members. Out of these 35 teams formed algorithmically, 5 teams took up our recommendation at least partially. A total of 27 teams survived till the end of the course with 7 teams of 2 members each and 20 teams of 3 members each.

At the end of the course, students filled out a post-course survey where they discussed their reasons for taking the recommendation or not.
2.4 Methods

A case-study methodology is adopted in this work allowing us to investigate how the Transactivity-Based Team Formation paradigm plays out differently in a 16-week long for-credit project-based course as opposed to a 4-week MOOC where it was evaluated in prior work. The avenues for the data used for our analysis include the Reflection-Feedback Setup, the teams recommended by the team formation paradigm, the teams that the student self-selected into, their performance in the course and the post-course survey. The following measures were quantified in preparation for the analysis presented in the following section.

- **Auto and Manual Project Grade** - We measure the success of the teamwork in terms of the grade each team received on the final project. There were two aspects of the project - an autograded portion based on their code and a manually graded portion based on their report.
- **Post-Course Survey** - Students were given a post-course survey at the end of the semester. The survey contained three open response questions including:
  - What was most valuable to you and worked best in the team experience?
  - What was least valuable to you and worked least well about the team experience?
  - What criteria did you use to choose team members, and when did you begin that process?

From this data we coded three variables at the level of each student, namely **SelectionProcess**, **RelationalIssues**, and **DivisionOfLaborIssues**. **SelectionProcess** was coded as a nominal variable having the values **Know** if the students mentioned knowing their teammates prior to selection, **Recommendation** if the students indicated taking our recommendation, **Observation** if students indicated making their choice based on activity on the course platform or **Generic** if they didn’t indicate how they found their teammates. **RelationalIssues** was coded as a numeric variable with a score of 1 if a student mentioned something positive about their teammates, -1 if they mentioned something negative and 0, otherwise. Similarly, **DivisionOfLaborIssues** was coded as 1, -1 or 0 for students mentioning something positive, negative or nothing about issues relating to the division of labor in their teams.

- **Transactivity Score** - Transactivity score was assigned at the level of a team, which was the average of pairwise transactivity scores across each pair of students within that team. The team assignment used for this analysis was the final team that students worked in for their projects.

2.5 Analysis and Results

2.5.1 Online Discussion Quality

The automated team formation paradigm relies on data from the online reflective discussion that was requested of students after each individual project. If the students were not engaged in this process, the paradigm would have broken down from the beginning. One of the big successes
we observed was student engagement in these discussions. In total, the students contributed approximately 200 posts to the discussion forum after each individual project. Up through the third individual project when team recommendations were computed, a total of 1007 posts had been contributed. Out of these, 438 (43.5% altogether) were labeled as transactive in the automated analysis. Since transactivity is typically low in discussion forums of online courses, this finding suggests that uptake of the intervention was strong at the initial stage. The reflection-feedback setup led to substantive discussions between students. Discussion about the methods they used in their individual projects led to fruitful interactions between the students. Even though the teams were co-located and the students were also meeting outside of class, participation in the reflection-feedback setup remained high throughout the course. This was presumably because students found the feedback and interaction they had to be useful. Survey responses highlighted the value of different perspectives, approaches and suggestions that students were able to obtain from their interactions on the reflection-feedback setup. Students also reacted positively to the team suggestions.

2.5.2 Team Formation Processes

As mentioned in the previous section, most team recommendations were not taken up in this study. Thus, uptake of the intervention was low at this point. We found that if one team chooses not to take the team recommendation, it has a ripple effect where the students they choose to work with instead must then leave the teams they were assigned, and then their team-mates must also find alternative arrangements. The ripple effect in conjunction with some students desiring to pick their own team meant that the structure overall broke down. In the end, only 5 team recommendations were partially preserved in the teams that were eventually finalized. The lack of uptake of the recommendations afforded the opportunity to test whether students of their own accord would choose team-mates that inadvertently maximized our estimate of collaboration potential. For this analysis we measured for each student the average pairwise transactivity score of the team they were assigned to as well as the average pairwise transactivity score of the team they eventually ended up on. The score for teams that were assigned was 1.14, while that of the final teams was .27. The pooled standard deviation was .89. We used a 2-tailed pairwise t-test to test the difference in scores, t(77) = 7.3, p < .0001 and the effect size was .98 s.d., thus indicating a large effect. For 63.38% of students who did not take the team assignment recommendation, the average transactivity of the self-selected teams was lesser than that of the teams we assigned them to. In 9.86% of the cases, the average transactivity was the greater and in 26.67% of the cases, the transactivity of the assigned and self-selected teams was the same.

2.5.3 Team Work

The final project grade, both the manual and autograded portions, provide an indication of how effectively teams were able to work together. Here the team is the unit of analysis. SelectionProcess is a quasi-experimental variable which we can use to obtain correlational evidence to evaluate the intervention. In this analysis, we investigate the role of Transactivity as an influence on group processes that affect how well teams produce joint work. For both team
performance measures, we built an ANOVA model with project grade as the dependent variable, SelectionProcess as the independent variable, and Transactivity score as a covariate nested within the independent variable. We nested Transactivity score because it has different implications for team process if it was used in order to select teams or just happened to be the case. As a covariate, we also included the average grade the students per team scored on an individual assignment they did prior to the teamwork activity. There were no significant effects on the manual portion of the grade, which focused on the written report. However, there was a trend on the autograded portion for the SelectionProcess variable, which targets the actual software they produced, and a significant positive effect of the transactivity variable, $F(4, 7) = 3.3, p < .05$. The transactivity variable accounts for an additional 30% of total variance in team performance accounted for by the model. As for SelectionProcess, teams where selection was based on prior friendship performed worse than the other 3 approaches. The two highest scoring categories were Observation of behavior on the platform and Recommendation.

The post-course survey provides an indication of the subjective perception of teamwork within projects. Here the individual is the unit of analysis. In this analysis, we investigate the role of Transactivity as a criterion for team selection as well as implications of student response to the recommendations. We measure perception of teamwork experience using RelationalIssues and DivisionOfLaborIssues as outcome variables. First, we built an ANOVA model with DivisionOfLaborIssues as the dependent variable, SelectionProcess as the independent variable, and Transactivity score as a covariate nested within the independent variable. There was a significant effect of SelectionProcess, $F(3, 69) = 3.4, p < .05$. A student-t posthoc analysis indicated that students with SelectionProcess had significantly lower scores than all other students with an effect size of .83 s.d. The TransactivityScore variable showed a moderate positive correlation with DivisionOfLabor issues within the set of students coded as Recommendation suggesting that the recommendations may have been more effective to the extent that the algorithm was able to find a high criterion solution. Next, we built an ANOVA model with RelationalIssues as the dependent variable, SelectionProcess as the independent variable, and Transactivity score as a covariate nested within the independent variable. We nested Transactivity score because it has different implications for team process if it was used in order to select teams or just happened to be the case. In this case, there was no significant effect. However, comparing those students with SelectionProcess coded as Know with all other students showed a marginal negative effect, $F(1, 75) = 1.69, p < .1$, effect size of .52 s.d. Overall, this suggests that student tendency to select team-mates they were friends with worked out poorly for them. On the other hand, students who based their choice on their observation of other students’ behavior, did not suffer the same fate.

### 2.6 Follow-Up Study

The second study which was conducted in the subsequent Spring 2018 offering saw participation from the Adelaide campus in addition to the Pittsburgh, Silicon Valley and Rwanda campuses. The changes to course practices that were made for the first study were preserved i.e., the Reflection-Feedback Setup where students posted reflections and provided feedback to

\[\text{http://www.cs.cmu.edu/~msakr/15619-s18/}\]
each other after each individual project and the Team Recommendation process that provided team member recommendations to students over email prior to them finalizing their teams in TheProject.Zone. There were however, two important changes informed by the observations made in the previous study. The first change was informed by the observation that students were forming teams from the very beginning of the course based on the expectations set by prior offerings. In the previous study, students did not know that they’d receive team recommendations until Week 5 of the course when most of them had at least partially selected into teams. In order to lessen the possible effect of this on the uptake of the recommendations, students in this study were informed in the beginning of the course itself that team member recommendations would be offered to them a week before they were meant to make the final decision. The prompt that was shown to the students was as follows -

“Students in 15-619 will participate in a team project in the second half of the semester. Based on your interaction with the course tools, we will suggest effective team members in week 5. Refrain from forming teams until you receive our team suggestions”.

Importantly, students were not informed of the mechanism that would be used to inform the team member recommendations which would have possibly changed their behavior in the Reflection-Feedback Setup. The second change was informed by the results of the first study that indicated that working with friends turned out to be counter-productive in terms of the eventual project grade and indicated relationship issues between team members. This meant a change in the messaging of the email which in addition to the mechanism that was used to inform team member recommendations now included a summary of the results from the previous study also. The motivation behind the change was to observe if student behavior would change when they were informed of some common practices from prior cohorts that led to detrimental effects.

The Reflection-Feedback Setup remained identical. As mentioned before, the messaging used for the team member recommendations was updated with the results from the first study. An excerpt from the email sent to students is given below -

“At this point in the semester, it is time to get organized into teams for the team project work. Up until now, your work has been entirely individual, though you have shared it with the class community, and have offered each other feedback. Research conducted in previous cloud computing courses on online team-based learning suggests that friends don’t make good partners. Specifically, our analysis showed that teams formed of members who are friends tend to have problems in division of labor and in coordination. This typically leads to lower performance when compared to teams who observe and evaluate the behavior and skill sets of other students before they form teams or if they follow our recommendation. We are sending this email to you as a group because based on our observation of your participation in this class, that prior research suggests you would work well together. Please consider this recommendation as you make your official declaration of team commitment for this course.”

49 total teams were formed this time around with 36 teams in the Pittsburgh campus, 8 from the Silicon Valley campus, 4 from the Adelaide campus and 1 from the Rwanda campus. All suggested teams had 3 members except for one Silicon Valley team and one Pittsburgh team which had 2 members. Out of the 49 teams formed algorithmically, 9 teams took up our recommendation partially. A total of 47 teams survived until the end of the course with 45 teams of 3 members each, 1 team with 2 members and 1 team with a single member.
As before, students filled out a post-course survey where they discussed their reasons for taking the recommendation or not.

2.7 Analysis and Results

2.7.1 Online Discussion Quality

As in the case of the previous study, student engagement in the Reflection-Feedback setup was quite high. Out of a total of 2784 posts contributed by students after each individual project up to Week 5 of the course, 771 (27.69%) were labeled as transactive in the automated analysis. This meant again that the initial uptake of the intervention was high.

2.7.2 Team Formation Processes

Even after telling students about the eventual team recommendations coming in Week 5 of the course at the very beginning, we found that the uptake of the intervention was not very high. As mentioned previously, out of a total of 49 teams recommended, 9 took up our recommendation partially. The ripple effect from the prior study once again manifested in this iteration leading to a breakdown of the overall structure.

Investigating whether teams this time around, armed with a little more knowledge about detrimental team formation strategies would of their own accord inadvertently maximize transactivity, our estimate of collaboration potential showed that the average pairwise transactivity of the teams that students finalized was 0.28 as opposed to those we assigned them to which was 1.13. The pooled standard deviation was 0.76. Using a 2-tailed pairwise t-test to test the difference in scores, \( t(138) = 9.2, p < .0001 \) and the effective size was 1.1 s.d., indicating a large effect. In 60% of the cases where our suggestion was not taken, the transactivity was lesser than what we would have suggested, in 11.67% of the cases it was higher and in 28.33% of the cases it was the same as what we would have suggested.

2.8 Discussion

Even though strong claims cannot be made from these studies because they did not include an experimental manipulation, they do illustrate how factors that were not present in the earlier evaluations of the approach impact the success of the transactivity-based team formation. In this study, the transactivity-based matching broke down once some students chose to form their own teams. In order to be successful, the recommendations must be taken by all. However, the fact that the students chose to ignore the recommendations in most cases suggests that forcing students to take a recommendation made without their involvement would not be appreciated by students. However, a policy of recommendations taken by all would not need to require students to be passive recipients of the recommendations. If we can actively engage their preferences in the constraint satisfaction process, we may be able to achieve the success observed in past evaluations of the approach. Consistent with prior work[69], the data
from this study suggests that allowing students to choose to work with their friends is counter-productive. In this course, students appear to be as successful in selection based on goal-directed observation as the algorithm is in transactivity-based assignment. Together, these two observations suggest that in engaging the preferences of students in the constraint satisfaction process of team assignment, we should encourage application of wise criteria observed from behavior within the course rather than selecting friends.
Chapter 3

In Process Support for Teams in Project-Based Courses

Concurrent with ensuring that students are well prepared for teamwork by intelligently forming teams, we also need to ensure that the teamwork itself is conducted without falling prey to problems outlined earlier such as lack of structure, uneven distribution of work and other process issues. As team-project based courses become frequently positioned at critical junctures within computer science and engineering curricula, there are a number of persistent problems that this form of structuring seeks to solve. First is the challenge of figuring out how to balance the tendency of students to over-emphasize the end-product above learning from the project. Although instructors make it their goal to offer pedagogically valuable learning experiences to students, the reward structures in these activities often undercut the learning goals. Since the assessment is based on the end-product, students are motivated to distribute work in such a way that they work on tasks they are already good at rather than challenging themselves to learn new ones. In order to allow for assessment that focuses not on the end-product but on the process, the collaboration needs to be instrumented and captured. Ju and Fox, for instance, highlight in their work, the need to instrument learning environments to capture student behaviors during collaboration rather than the end product alone [52]. Another issue is that learning frequently requires feedback and scaffolding. However, in order for instructors to offer in-process feedback and scaffolding, they need to be able to formulate an accurate assessment of group processes and activities. It is known that instructors are largely unable to make such assessments accurately since they are not present during most of the group work [34], and are also known to fall prey to specific cognitive biases when they form impressions from what they do observe [36]. Finally, even though collaborative project based learning offers the opportunity for students to foster needed teamwork skills, it opens a host of other difficulties in terms of management overhead and conflict, to name just a few. These challenges are exacerbated online and at scale, two contexts that have become more and more prominent in computer science education. Against this backdrop, we offer a path towards a solution.

We therefore advocate for a new paradigm for project based learning for computer science instruction that draws inspiration and a conceptual foundation from industry trends while motivating potential solutions to these issues faced in project based learning. The challenge is to create a context in which learning and productivity can be jointly optimized within group
work [13]. The emerging trend from industry we build on is called Mob Programming [115][122].

In an instructional context, we cast the Mob Programming paradigm as a form of collaborative learning. Mob Programming grows out of Pair Programming, and is a group activity in which 3-6 participants assume different roles to collectively contribute a solution to a programming challenge. In this way, cognition is distributed, and group members with differing abilities are able to contribute in different roles while benefiting from the support of the group. Further, Mob Programming is becoming increasingly popular in the industry and bringing it to the class allows students to learn an industry-relevant skill while learning to work more efficiently with their peers.

We thus advocate for the Online Mob Programming (OMP) paradigm in which group work is conducted online where it can be instrumented. The instrumentation would enable instructors to check on group processes and progress, but also allows for automated forms of support for group learning such as Intelligent Conversational Agents which have been used in prior intervention studies to offer dynamic, context-sensitive support for collaboration in the midst of text-chat based interactions thus allowing for process feedback and scaffolding [57][58][103][104]. These forms of support have been shown to improve students’ learning, knowledge integration and team performance in conversational agent-facilitated collaborative activities [1][24][109]. OMP also provides the benefit of a structured collaboration that manages group processes for relatively large groups of 4-6 students. The roles imposed and switching between roles allow students to contribute in roles most suited for them and learn from peers when cast in other roles.

In this first study, the OMP paradigm is investigated in the context of a 6-week free online Cloud Computing course offered to working professionals in the summer of 2018. We first see if the industry-inspired paradigm can be cast in a pedagogical setting to simultaneously prioritize productivity and learning. To that end we investigate the following research questions -

RQ1: To what extent do students follow the behavior prescribed by Online Mob Programming?

RQ2: How does student behavior in OMP differ based on mob size, problem difficulty, and whether students were successful in solving the problem in the allotted time?

Results show evidence of success with students following the structure of OMP and the mob setup scaling to groups having 3 to 6 participants. Investigating mob behavior provides pointers for ways to better support students in the future. Further, subjective feedback from students indicate that they are teaching and learning from their peers and shifting from focusing solely on productivity to a combination of productivity and learning [44][85][88]. The success of the paradigm in this context has prompted us to further investigate OMP in the undergraduate computer science context where it is offered as a part of a semester-long project-based Cloud Computing course.

3.1 Related Work

Mob Programming We present here Online Mob Programming (OMP), a new pedagogical construct for collaborative project-based CS education. Buchan and Pearl [14] report on a software company team’s experiences over 18 months of in-person Mob Programming. They report
benefits, including that team members gained a broader knowledge of their entire code base. Additionally, they also present open questions where more research into Mob Programming is needed. Kattan et al. [53] present three case studies on Mob Programming. Each study consists of a team in an academic setting working on open source code. Overall, they report that there was unanimous approval of mob programming during team retrospectives. Ulrika Malmgren [63] describes in a blog post about how she participates in “Remote Mob Programming”, and Sal Freudenberg [27] writes about how she found remote mob programming to be inclusive after being diagnosed with autism.

**Pair Programming** One way that Mob Programming has been described is as an extension of pair programming. There has been extensive research into pair programming. Some pair programming work relevant to the context of OMP is included below. Harsley et al. [41] use an intelligent tutor to assist with pair programming. Rodriguez et al. [82] found that more active participation from the driver may be important to improve learning which is consistent with the active participation of the driver in OMP. Other researchers have studied distributed pair programming [8, 19, 100, 105] where the pairs are not physically co-located. Additionally, researchers have investigated the idea of pair programming with more than two participants. Saros [84] is a platform that supports up to five participants in distributed party programming. However, Saros does not provide guidance for how the participants should interact during a session. Nguyen et al. [68] present EduCo, a Web-based collaborative learning environment. They report supporting 40 students participating in a single session.

### 3.2 The Design of Online Mob Programming

#### 3.2.1 Mob Programming

Online Mob Programming is adapted from the industrial practice of Mob Programming, where participants rotate through the following roles - **Driver**: A single participant who converts high-level instructions from the Navigator into code. **Navigator**: A single participant who makes decisions from discussing with the Mob and communicates that to the Driver to be implemented into code. **Mob**: A group of participants who consider and deliberate between multiple alternative implementations ultimately informing the decision of the Navigator. **Facilitator**: A single participant (optional) who observes and intervenes when necessary, such as to indicate when roles are to switch and to keep the activity progressing.

The rotation of mob role affords participants the opportunity to experience how group processes change when leadership changes within a group. Each participant will experience all the roles throughout a single mob programming session getting an opportunity to contribute as well as observe different perspectives and approaches to solve problems in the same session.

In the Navigator role, a participant is responsible for making decisions about the next step in the implementation. They will solicit input from the mob, decide on the next action, and provide directions to the driver. In the Driver role, responsibilities include taking directions from the navigator, and translating those into code. The Driver role allows a participant to focus on writing syntactically correct code, while the navigator worries about the overall direction. In the Mob role, a participant is responsible for considering alternative directions, understanding
the current state of the code, and providing input to the navigator on how the code can be improved.

When applied in a pedagogical context, this synchronous learning supports students at different skill levels or those with complementary skills to scaffold each other. The group context creates pressure to perform and yet balances that pressure with support from the team. Students are exposed to different perspectives in solving problems, building solutions, experimenting, debugging and writing readable code. Students are forced to externalize their thinking, which provides the opportunity for knowledge gaps to be revealed and addressed. They also have the opportunity to observe knowledge and expertise in action as they are learning.

3.2.2 Adapting Mob Programming for Pedagogy

The concern with group collaborative activities always is that group processes come in the way of learning and productivity, and without structure, it can descend into chaos. Moreover increasing group size can exacerbate instances of social loa/fing (freeloading) and losses due to increased difficulty in group coordination. Without guidance, students can also end up choosing sub-optimal strategies such as working on tasks or roles that they are already good at, instead of taking the opportunity to learn. Additionally, group work can be dominated by the most outspoken members.

Mob Programming is pedagogically positioned to solve these issues. First, providing a structure around the collaboration and helping students understand how this structure contributes to their learning and success on the problem, can stem group coordination difficulties at the outset. Second, assigning students to roles and periodic rotation of the role assignments can prevent the adoption of sub-optimal strategies. Moreover, the role assignments can solve the problem of freeloading by introducing an element of social pressure but at the same time balancing that pressure with support from team members in other roles.

RQ1 poses the question of whether the intended structure of mob programming which was designed to solve these issues was followed in practice. In order to better design mob programming for supporting students in the future, RQ2 poses the question of how mob behavior changes with group size, problem difficulty, and whether successful and unsuccessful mobs behave differently. The understanding in industry is that mob programming is suitable for more complex problems, but in the pedagogical setting, it is important to strike the right balance between achieving a solution in the time frame and providing a good learning opportunity. Investigating the difference in behavior between successful and unsuccessful mobs can help us improve OMP to better support students.

3.2.3 Bringing Mob Programming Online

The primary technological infrastructure needed for bringing in-person mob programming online include a collaborative code editor, a terminal that can run the collaboratively composed code and a means of communicating around the code. However, as has been observed in the context of collaborative learning in online courses in the past, merely providing the technology without intentional design that can support the collaboration leads to failure. Students have reported facing more frustration with online groups than with in-person groups.
simply providing a means for communication is also insufficient because of low adoption [56]. This is possibly because of the lack of awareness among students about the role that social interaction plays in the process of learning necessitating a much tighter coupling of the social interaction opportunities with the course content and pedagogy [98]. We therefore adopted an iterative design process over a series of pilot studies pausing after each one to reflect on what worked and what didn’t work and using the insights from investigating this space of possibilities to design the final OMP setup for this study.

One locus of investigation was the communication channel. While the audio channel can lead to a much smoother interaction and coordination experience, it comes at two important costs. First, the audio channel is much harder and much more resource-intensive to capture and instrument for the purpose of analyzing the interaction between students. The text channel is readily instrumented because it completely captures the communication among students around the code. Second, the audio channel could inherently disadvantage certain kinds of students.

Prior work has found that audio channels place a higher cognitive load on students when compared to text channels therefore disadvantaging students with low or intermediate speaking proficiency [83, 99]. Audio channels have further been tied to more anxiety levels among students than text channels [38, 91]. It has also been observed that while audio channels provide a heightened sense of social presence to online students and minimize the chances of misinterpretations because of the use of voice and tonal cues, text channels may be entirely more appropriate in tasks that require students to explore dissonances among ideas and negotiate opinions, which is the case in mob programming. These tasks typically require that students challenge each others’ opinions and ideas [59] which is better supported by the text channel as it provides more time to structure responses [43].

The second locus of investigation was the collaborative coding environment. While tools like Coderpad provide support for collaborative coding and log code edits for retrieval and analysis later, they do not have a built-in communication channel. This necessitates the use of another application like Google Hangouts to supplement this need. Therefore students would have to switch between the Hangouts text-chat and Coderpad, leading to coordination issues and a suboptimal interaction experience. This further had the unintended consequence of slowing down the entire group, resulting in the student in the driver role having to wait for the group to come to a consensus before translating that into code. Coupled with the relatively frequent switching of roles, this meant that not much code was written in the amount of time allotted for the exercise. The learning goals weren’t fulfilled because of the sub-optimal interaction experience and the productivity goals weren’t fulfilled because of the coordination issues. This meant that the ideal collaborative environment would provide for both collaborative coding and support for communicating around the code in the same environment without the need for context switching.

The Cloud9 IDE which includes the editor, terminal, text-chat and the file navigation all on one screen was therefore chosen. The collaboration pane above the chat includes some social presence tools allowing users, at any given time to determine which files and location

---

1https://coderpad.io/
2https://aws.amazon.com/cloud9/
within files other users are looking at. Cloud9 being a part of the Amazon Web Services (AWS) infrastructure allows the setup to scale to a large class and to service many simultaneous mobs. Further, contributions to the code and to the chat could be logged along with user timestamps making it conducive for analysis.

The final locus of investigation was the use of a conversational agent to support the collaboration between students participating in the mob. Specifically, the role of the facilitator, which if assigned to a student would mean either that the student cannot fully concentrate on the task at hand or would be idly occupied by time-keeping and other coordination tasks providing no pedagogical benefit. Thus, a conversational agent-facilitator, based on the open source Bazaar framework[3][4] was integrated into Cloud9 to take care of timing and sequencing the activity by informing students when they would need to switch and what their new roles would be. The introduction of the agent also allowed for instructions about the use of the Cloud9 IDE and the activity to be introduced more naturally as a part of the conversation rather than as a part of an instructions file, and opened up the possibility of more dynamic context-sensitive conversational support for students and their roles in the future.

### 3.3 Experimental Setup - Study 1

#### 3.3.1 Course Context

The OMP study was done in the context of a free, 6-week online course on Cloud Computing offered by a US University. The CloudCourse[4] was offered to working professionals with prior programming experience in Java, Python and Bash. Since mob programming as a paradigm originated in industry and has not been empirically evaluated for pedagogy in the past, we conducted the study in a learning context provided to working professionals. This study could thus serve as an intermediary for importing a paradigm from industry into undergraduate computer science classes. Once the efficacy of the paradigm has been tested, we can use our findings from this study to better inform the design of activities around OMP for undergraduate students.

#### 3.3.2 Course Timeline

A series of mob activities were added to the course to offer students the opportunity to collaborate with classmates to extend the knowledge they had gained in their individual project. The course consisted of a total of four mob sessions - Mob 0 through Mob 3. Grouped by their availability, up to 7 students were assigned to each mob in order to account for students who wouldn’t show up and those who would drop out of the course entirely. Assigning 4 to 7 students to each group allows for each mob role to be satisfied and allows for 2 to 5 students to be in the mob at any given time providing room for sufficient deliberation.

Prior to the first mob session, students were provided a primer introducing the idea of online mob programming, the roles involved, and the responsibilities associated with each role. A walk-through provided as a part of the primer gave a step-by-step example using Test Driven

---


Development (TDD) in the context of OMP allowing them to understand what it constituted at a high level. The students then crystallized their understanding of OMP and were introduced to the Cloud9 environment by participating in Mob 0 around a trivial Fizz-Buzz problem[^5]. Three more mob sessions were conducted in the weeks following Mob 0. Each mob session was held after an individual project so that students would have the background needed to productively participate in the mob. Special consideration needed to be given to the problem as it needed to be difficult enough that an individual couldn’t solve it on their own and easy enough that a group can solve it in 80 minutes while building on what students had learned in their individual projects. A grade was assigned both participating in the mob thus incentivizing participation[^29] and for successfully completing the assigned problem.

The individual projects in each week of the course leading up to the mob sessions were as follows -

**Mob 0** - In the first week, students explored the offerings of each cloud service provider (Amazon Web Services, Microsoft Azure and Google Cloud Platform) used in the course. The mob that followed was an introductory mob on the FizzBuzz problem.

**Mob 1** - The project for Mob 1 introduced students to the Map Reduce paradigm and required them to design a Map Reduce job to filter and aggregate Wikipedia page view statistics. During the mob session, students built on this knowledge by implementing more complex data processing rules in a parallelized fashion.

**Mob 2** - In the project leading up to Mob 2, students had to develop a software solution to autoscale virtual machines in response to changing HTTP traffic. After developing a custom solution, students were tasked with using cloud load balancers and scaling polices to maintain a given resource usage limit. The task chosen for the mob session expanded on the load balancing concepts introduced in the project by having the students implement the Token Bucket algorithm for HTTP traffic rate limiting.

**Mob 3** - In the project leading up to Mob 3, students develop a video processing pipeline that integrated machine learning services and search engines to build a video indexing service. The mob programming task required them to develop a new AWS Lambda function that will identify faces within thumbnail images and to blur all faces in the image, using the Pillow imaging library[^6].

### 3.3.3 Data Collection

The following were the data sources that informed our analyses.

**Pre-Course Survey** - Students filled out a pre-survey at the beginning of the course answering questions about programming proficiency, familiarity with collaborative learning paradigms like pair and mob programming, English language proficiency, familiarity with AWS or other cloud services and their course goals.

**Post-Mob Surveys** - After each mob session, students filled out a post-mob survey which asked about their OMP experience including questions like whether they thought it would help their learning, whether they liked the activity, what worked, what didn’t work and what could

be done to make OMP better.

**Post-Course Survey** - At the end of the course, students filled out a post-course survey providing feedback about the course as a whole but also answering questions reflecting on their mob experience as a whole including how it will be relevant to their learning and work in the future.

**Code Revisions** - The code revisions during each mob session were logged by timestamp.

**Chat Contributions** - The chat contributions during each mob session were also logged by timestamp. Table 3.1 gives an overview of the data collected across all four mobs.

### 3.4 Analysis and Results

#### 3.4.1 Structured Collaboration

To answer RQ1, we looked at the chat contributions and file revisions made by students separated by the roles they assumed in the mob. If the structure of the mob was followed, we would expect to see that most of the file revisions (code contributions) come from students in the driver role. Moreover, we would expect students in all roles to consistently make chat contributions as an indicator of active participation in the activity. Table 3.1(a) shows the number of chat contributions and file revisions by mob role for each mob session. Examining chat contributions, we see that across all the mob sessions, about a third came from students assuming each of the roles. This suggests that regardless of the roles assumed, students were not disengaged from the activity. Further, over 80% of the file revisions across all the mobs were made by the driver indicating that the intended role of the driver was largely followed.

Inspecting the chat contributions being made by the driver, most of them are requests for direction such as "We passed that test case, what next?" or requests for clarification such as "Are we reading from file or standard input?". This is consistent with the role of the driver but will require further analysis to quantitatively verify. Similarly, messages sent by the navigator mostly involve marshalling the mob or providing directions to the driver based on the mob decision. Another factor that needs to be investigated in more detail in the future is deviations from mob orthodoxy. For instance, one mob used comments in the file to organize their thoughts which are currently counted as code revisions. In some other cases, the role switches ended up not happening instantly even with advance warning. The driver would often finish up what they were currently working on before switching roles. Contributions made after the switch are currently attributed to their new roles thus possibly explaining some of the deviation described above.

#### 3.4.2 Group Size Effects

To infer that the mob structure continues to be followed with increasing group size, we would first expect the driver to make a majority of the file revisions (code contributions). We would also expect students in all roles to actively participate in the chat but expect students in the mob role specifically to account for a larger percentage of contributions because the number of members in the mob role increases with increasing group size. To investigate this, we examine
Table 3.1: Number of chat contributions and file revisions in total and separated by role per mob session.

(a) Number of chat contributions and file revisions separated by role per mob session

<table>
<thead>
<tr>
<th>Mob Session</th>
<th>Total Users</th>
<th># of Mobs</th>
<th>Number of chat messages</th>
<th>Number of File Revisions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total</td>
<td>Driver</td>
</tr>
<tr>
<td>mob_0</td>
<td>51</td>
<td>17</td>
<td>2697</td>
<td>866 (32.11%)</td>
</tr>
<tr>
<td>mob_1</td>
<td>28</td>
<td>10</td>
<td>1449</td>
<td>387 (26.71%)</td>
</tr>
<tr>
<td>mob_2</td>
<td>24</td>
<td>7</td>
<td>963</td>
<td>292 (30.32%)</td>
</tr>
<tr>
<td>mob_3</td>
<td>19</td>
<td>5</td>
<td>927</td>
<td>263 (28.37%)</td>
</tr>
</tbody>
</table>

(b) Number of chat contributions and file revisions in total and separated by role for mobs of different sizes.

<table>
<thead>
<tr>
<th>Group Size</th>
<th>Total Users</th>
<th># of Mobs</th>
<th>Number of chat messages</th>
<th>Number of File Revisions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total</td>
<td>Driver</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>12</td>
<td>2117</td>
<td>643 (30.37%)</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>10</td>
<td>2161</td>
<td>558 (25.82%)</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>3</td>
<td>624</td>
<td>170 (27.24%)</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>2</td>
<td>486</td>
<td>131 (26.95%)</td>
</tr>
</tbody>
</table>

(c) Number of chat contributions and file revisions normalized to one mob by role and separated by role for successful and unsuccessful mobs.

<table>
<thead>
<tr>
<th>Task Result</th>
<th>Total Users</th>
<th># of Mobs</th>
<th>Number of chat messages (Normalized by Mob)</th>
<th>Number of File Revisions (Normalized by Mob)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total</td>
<td>Driver</td>
</tr>
<tr>
<td>Pass</td>
<td>103</td>
<td>32</td>
<td>5229</td>
<td>48.1 (29.43%)</td>
</tr>
<tr>
<td>Fail</td>
<td>19</td>
<td>7</td>
<td>807</td>
<td>38.43 (33.33%)</td>
</tr>
</tbody>
</table>

We see that students were not disengaged in any role (At least 20% of the chat contributions came from students in each role regardless of its size). We can further note that the percentage of chat contributions made by students in the mob role increases with increasing group size from 35.71% for mobs of size 3 (1 student in mob role) to 51.85% for mobs of size 6 (4 students in mob role) which is consistent with what we would expect to find. Looking at the file revisions, we can say that the driver role was largely followed. However, there seem to have been deviations in groups of size 3 and 6. These differences are not statistically significant - The number of file revisions made by drivers in a group of size 3 for instance was 313.25 on average per mob (S.D. = 269.34) which was not significantly different from the same for groups of size 4 which was 356 (S.D. = 316.74) (p = 0.7356 < 0.05). Similarly, the difference in average file revisions per mob made by groups of size 5 which was 488 (S.D. = 308.95) and groups of size 6 which was 274.5 (S.D. = 283.55) was also not statistically significant (p = 0.4934 < 0.05). Even so, it might have been the case that in smaller groups, there is less social pressure to uphold rules, more instances of possible domineering by one member or deference to one person to code. In groups of larger sizes, decisions may take longer or the driver role takes longer to come around leading to impatience on the part of mob members and these warrant further investigation in order to identify the ideal group sizes for mob programming.
3.4.3 Problem Difficulty Effects

As problems that the groups solve become harder, we might expect that they spend more time thinking individually and less time actually discussing or making code revisions. We would therefore expect to see more chat contributions and less file revisions with increasing problem difficulty. The course team rated Mob 3 as the most difficult followed by Mob 1 and Mob 2. Mob 0 cannot be used because even though the problem was trivial, it served as the introduction to mob programming for most students. Keeping this in mind, we can once again look at Table 3.1(a). While there is no clear trend, students seem to have made fewer chat contributions and less code revisions on average in the case of the hardest problem suggesting that they may have spent more time thinking. Further investigation that can help characterize the content of the conversations rather than just the number might help us better understand the interaction between problem difficulty and student behavior.

3.4.4 Successful Problem Completion

Understanding ways in which the behavior of unsuccessful students differs those of successful ones can help us better support them. In cases where the mob was unsuccessful, we might expect students to make fewer file revisions and more chat contributions if they were discussing the problem but could not come to a consensus about the path forward. We might also expect them to make more file revisions and fewer chat contributions if they were wheel spinning by trying many different things instead of discussing and coming to a consensus before implementing [10]. Table 3.1(c) shows file revisions and chat contributions normalized per mob by role based on whether students passed or not. Most students were successful. In the minority of the mobs that were unsuccessful, students made almost twice as many file edits and fewer chat contributions. The number of file revisions made by students in the driver role for unsuccessful mobs was on average 460.71 (S.D. = 350.47) which was marginally significantly more than the same for successful mobs which was 254.19 (S.D. = 220.65) (p = 0.1455). The differences in chat contributions made by members in any role was not statistically significant. This may suggest that the students were possibly wheel-spinning. It was further observed that these students made more submissions to the auto-grader hoping for feedback that they could use to get to the right answer. This unproductive form of engaging with the support, termed "gaming the system" has been identified as one major predictor of failure [9]. Both spaces provide opportunities for the intelligent conversational agent to provide support or correct unproductive behavior.

3.4.5 Student Perception

The goal of OMP is to provide an opportunity for jointly optimizing student learning and productivity on the task. To that end, we are faced with the challenging task of shifting student perceptions away from the current culture of fixating on grades to caring about the process and mechanisms they used for learning from their peers. To gather evidence for whether we at least contributed to this culture shift, and to get a deeper understanding of what students’ perceptions about the activity were, and what they thought they gained from it, we administered
surveys after each mob session and at the end of the course. The post-mob surveys solicited immediate feedback on student perceptions while the post-course survey provided an opportunity for students to reflect on the four mob programming exercises as a whole and suggest possible improvements.

Three Likert scale questions asked how students liked the activity as a whole, how much they felt it improved their learning and how was their experience with Cloud9. The feedback was generally positive with average scores of over 3 on a 5 point scale. Answers to the open response follow-up questions helped contextualize these scores. Collaboration and teamwork were most often cited as a part of the OMP exercise that worked best. Several students commented about the advantage of being exposed to different approaches to the problem, having multiple pairs of eyes to catch errors and edge cases, and even commented on the benefits of collaborative programming in real-time. The mob structure and rotation of roles was cited as the part of the OMP exercise that helped with the collaboration the most. The overall sentiment is best captured by this student’s experience - "I think working as a team and rotating roles worked best. It had people in the driver’s seat to perform the actual programming and then had you sit back and see the whole picture. It also allowed us to see different approaches to problems by other people.”. However, some students complained about their group members’ lack of proficiency in programming, the mob slowing them down on things they already knew or the speed with which the problem was being solved as a whole. This was especially pronounced in the harder mobs indicating that although there were positive examples of learning, the culture shift was not entirely embraced and it is important to provide a problem at the right difficulty in order to strike a balance between productivity and learning.

3.5 Experimental Setup - Study 2

Reward structures used to evaluate students on their performance in team projects often incentivize behaviors like divide and conquer where students self-select into performing tasks they are already good at rather than challenging themselves to learn new skills. The OMP framework discourages the allocation of tasks purely on the basis of prior expertise, thus allowing students to not only contribute in roles they are already good at, but also learn from their teammates to contribute in roles when they are not. We can hypothesize therefore, that (1) the OMP scaffold, if effective, will produce distinct collaborative behaviors associated with each role in mob programming. If we also believe that students would default to optimizing for productivity in the absence of such as a scaffold, we can hypothesize that (2) these distinct collaborative behaviors will not be adopted in self-organized groups, which might result in student behavior looking far more consistent throughout the activity. By enforcing the OMP scaffold for collaboration however, we run the risk that productivity may be harmed because students with less expertise at each subtask may get in the way. Furthermore, the cognitive load from role-switching might reduce productivity on the task and putting students in roles they are not familiar with could increase discomfort and therefore negatively affect their perception of the task. We hypothesize therefore, that (3) students from the OMP scaffold groups might feel more negatively about their experience compared to students from the self-organized groups and might perform worse on their project.
this second study therefore, we experimentally contrast the mob programming scaffold against student self-organization in a between-subjects design embedded within a completely online Cloud Computing course offered to the students of Carnegie Mellon University and its campuses worldwide [89]. The course offers hands-on experience on the applications of Cloud Computing using three leading cloud computing platform providers: Microsoft Azure, Google Cloud Platform and Amazon Web Services. Students are expected to have strong programming competency in at least one programming language prior to taking the course. A total of 120 students took the course organizing themselves into teams of 3 for a half-semester long course project.

In the first week of the course, as a part of primers that students had to complete, they were grouped randomly into teams based only on their time availability to participate in an OMP training session. Prior to this training session, students were provided materials explaining the mob programming paradigm and how it will be used for collaborative programming exercises in the class. The training session itself required students to solve a relatively simple programming task collaboratively. The task was kept simple so as to allow students to familiarize themselves with the mob programming paradigm as well as the Cloud9 interface, which was the industry-grade software development platform that housed their collaborative activity. Each activity lasted 80 minutes with roles switching every 8 minutes. The role-switching was kept relatively frequent in order to promote observation of the problem from multiple perspectives. All students regardless of condition first took part in an OMP training session two weeks prior to the experimental manipulation, thus allowing them to familiarize themselves with the roles and the workings of the OMP framework and the Cloud9 interface.

The experimental manipulation took place two weeks after the OMP training session when students had acquired the prerequisites necessary to complete the programming task. As in the training session, the activity lasted a total of 80 minutes with switches happening every 8 minutes. Students, in their 3-person teams decided on a time when they will be available to work on the activity together and informed the course instructors. The programming task required students to perform data analysis on Twitter data using tools on the cloud. Many possible solution paths and ways of implementing the solution provided room for the responsibilities associated with each role to manifest in concrete ways towards producing the end product. Students in both conditions performed their collaborative work during the manipulation in the same Cloud9 environment. However, only in the OMP condition did the students receive instruction to engage in OMP practices and experience automated support for the role taking and timed role switching. The following data was collected in both conditions to facilitate our analyses - Code contributions and chat logs from the team programming exercise? These allow us to get an insight into the way students organized themselves to solve the exercise in the self-organized condition and see if they followed the structure imposed by the scaffold in the OMP condition; Grades - Grades on individual assignments and the team project prior to and post the team programming exercise help control for differences in prior knowledge among students assigned to either condition. The grades on the team programming activity itself, help us see if there was a difference in group product quality across conditions; Post team programming exercise survey - This survey asked about prior familiarity with teammates, how this activity helped discover teammates, how they chose to structure the activity, how their experience with the OMP training session helped structure this activity, how effective they...
felt their organization was, both in the training session and in this activity, and their experience with the Cloud9 interface. The survey allowed for us to better understand the mechanisms behind the observed results.

3.6 Analysis and Results

**Hypothesis 1: The OMP scaffold, if effective, will produce distinct collaborative behaviors associated with each role in mob programming.** A goal of OMP is to orchestrate rotation of team members through a set of three distinct but interdependent roles. If the manipulation was successful, we would expect to see distinctive behavior patterns associated with the roles. As both a manipulation check and a lens to elucidate the effect of the manipulation on collaborative processes, we conducted a quantitative discourse analysis.

In order to test the distinctiveness of the roles that team members took up in the discussions, we identified themes within them using an automated technique. First, we aggregated the transcripts for all discussions such that each turn was labeled with a condition and a role. We removed the contributions from the automated support agent. In total there were 3,396 remaining turns produced by 36 groups and a total of 108 students. There are several topic analysis techniques for identifying themes automatically within textual data, however, many of them require very large text corpora in order to operate properly. Thus, we adopted a very simple technique based on the same principles. In particular, we first computed a term frequency by document matrix where each document was the text uttered within a conversational turn, and its representation within its row within the matrix contained a count for each word signifying how many times it appeared in the corresponding document. Then using a Principal Components Analysis with Varimax Rotation, we identified the top 15 latent factors that distinguished conversational turns that explained the most variation in the matrix. We then saved the loading onto the 15 latent factors in the data table so that each turn was represented in terms of the 15 factors.

We selected 6 latent factors to interpret shown in Table 3.2, which were the most distinguishing between roles within the OMP condition and behavior within the unsupported condition based on a Chi-squared analysis between the factor loadings per turn and the turn status, which was the role the speaker was in when uttering the turn (Supported Driver, Supported Navigator, Supported Mob, or Unsupported Mob).

Next, we conducted a student-t post-hoc analysis to determine which comparisons between turn status were significant. If Hypothesis 1 is supported, we would see turn status between roles to be more distinctive in the OMP condition, meaning different roles are showing different patterns (factors) in the way they talk.

Here we see that turns from Drivers were distinctive in their relatively high loading on the Reporting Structure and Interpretation factor. Turns from Navigators were distinctive in terms of their relatively high loading both on Direction and Code. Turns from Mob members in the supported condition were high on Evaluation and Brainstorming and Code. However, turns from members in the unsupported condition did not load high on any of the distinguishing factors. The distinctions we see in interpretation of the factors that distinguish between supported roles are consistent with what we would expect based on their definition.
We do see different roles in the OMP condition adopting different discourse patterns, with Driver displaying more reporting, structure and interpretation, Navigator displaying more direction, and the mob displaying more on evaluation and code. This is consistent with our definition of the three roles and Hypothesis 1 is thus supported. **Hypothesis 2: The distinct collaborative behaviors associated with roles in the OMP condition will not be observed in self-organized groups, which might result in student behavior looking far more consistent throughout the activity.** We considered that it is possible that within self-organized groups that members took up roles despite not having been assigned. If this was the case, we might not see those distinctively in the analysis above since turns from all team members are taken together in the self-organized condition as Unsupported Mob and thus, we would only be able to see average behavior across roles (if any). In order to test this hypothesis therefore, we had to adopt a different methodology - cluster analysis. We clustered turns using K-means clustering in order to identify cross-cutting factor profiles across turn statuses. Since there are 4 Turn Statuses (Supported Driver, Supported Navigator, Supported Mob, and Unsupported Mob), we set the number of clusters to 4. In order to test whether the cluster analysis identified patterns that distinguished some subset of unsupported mob turns from those of the supported roles, we conducted a Chi-Squared test between the cluster assignment per turn and the Turn Status, however this was not significant. One cluster contained only turns from the 3 supported roles, and in particular contained all turns from the Driver role. However, none of the clusters were distinguishing for the unsupported condition in contrast to the other supported roles. In particular, the other clusters contained a mixture of unsupported turns, the Navigator and the supported Mob.

This cluster analysis shows that we are unable to identify any distinctive behavior patterns.
for turns within the self-organized groups, suggesting that the collaborative behaviors associated with each role in mob programming were not naturally adopted by students if not explicitly scaffolded. Hypothesis 2 is thus supported.

**Hypothesis 3:** Students from the OMP scaffolded groups might experience more discomfort as compared to students from the self-organized groups if the scaffolds are effective in countering the natural tendencies of students to gravitate towards what they are experienced doing. They may also produce lower quality work. In the post-programming exercise survey, we asked students about their experience with the Cloud 9 environment and how prepared they were for future group projects. We compared student responses in the two conditions in order to test Hypothesis 3. The outcome measures used were student responses to the likert scale questions 1) “How prepared do you think your team is to start working on the project? (5 meaning very prepared) and 2) “What was your experience with Cloud9?” (5 meaning very positive experience). We also operationalized covariates to control for students prior familiarity with group members and prior knowledge. Prior knowledge was operationalized as the average score of the 2 individual exercises that the students had completed prior to this activity. Familiarity was the average of the response to 2 likert scale questions asking students to rate their familiarity with their teammates (5 meaning very familiar). We then built linear regression models to investigate students self-reported level of preparedness and experience with Cloud 9 across the OMP scaffold and unsupported conditions. We used the likert-scale response as the outcome variable, condition as the main factor, and students prior grades and familiarity with the team as covariates. Students in the OMP scaffold condition perceived themselves as less prepared to start working on the project compared with students in the unsupported condition (p = 0.0413). Students in the OMP scaffold condition were also less satisfied with Cloud 9 compared with students in the unsupported condition (p = 0.0627). The effect is marginally significant. Importantly however, a linear re-

<table>
<thead>
<tr>
<th></th>
<th>Driver</th>
<th>Navigator</th>
<th>Supported Mob</th>
<th>Unsupported Mob</th>
<th>Comparison (student-t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reporting Structure and Interpretation Evaluation and Brainstorming Direction Code</td>
<td>.14</td>
<td>-.07</td>
<td>.06</td>
<td>-.03</td>
<td>D&gt;N</td>
</tr>
<tr>
<td></td>
<td>-.09</td>
<td>.02</td>
<td>.14</td>
<td>-.02</td>
<td>SM&gt;D,UM</td>
</tr>
<tr>
<td>Code Status Code Abstraction</td>
<td>-.06</td>
<td>.14</td>
<td>-.03</td>
<td>-.02</td>
<td>N&gt;D,SM</td>
</tr>
<tr>
<td></td>
<td>.06</td>
<td>-.07</td>
<td>.11</td>
<td>-.01</td>
<td>N,SM&gt;D</td>
</tr>
<tr>
<td></td>
<td>.08</td>
<td>-.08</td>
<td>-.05</td>
<td>.01</td>
<td>n.s.</td>
</tr>
</tbody>
</table>

Table 3.3: Comparison of factor loadings from PCA associated with each role, focusing on the 6 identified themes
gression model built using the mob session grade as the outcome variable, the condition as the main factor while controlling for students' prior grades and familiarity with their team members found no significant differences between the two conditions suggesting that while students provided with the scaffold experienced marginally significantly more discomfort, this did not manifest itself in actual performance differences. We discuss below some reasons for why we think this might be the case. Hypothesis 3 is thus partly supported.

### 3.7 Discussion

In our analysis we find support or partial support for all three hypotheses. Although it might seem trivial that students in the OMP condition would take on behavior patterns consistent with the roles we instructed and scaffolded during the activity, and students in the self-organized condition did not, it is notable that just two weeks earlier, all of the students went through an OMP training session in which they were instructed about the roles and experienced the scaffolding. As a validation that students felt benefit from that experience and sought to apply it with or without instruction to do so, we included questions in the post team exercise survey asking students to report whether they felt the training session before this programming exercise was. As reported in the survey, students in general felt the training session influenced their behaviors in the programming exercise in both conditions (with an average of 3.55, and a standard deviation of 1 on the likert scale from the survey question: “Earlier in the semester, you participated in a training session for Online Mob Programming (OMP). Do you feel that experience influenced your behavior during this team exercise?”). The self-organized groups reported higher influence than the OMP scaffold groups, with an average of 3.72 (s.d. = 0.96) for the self-organized groups and an average of 3.43 (s.d. = 1.01) for the OMP scaffold groups. The difference between the two conditions is not statistically significant.

We also saw evidence that the students felt more organized in the team programming exercise compared to first training session regardless of the condition they were in. Two likert-scale questions asking 1) “How effectively did the organization work between participants during the training session?” and 2) “How effectively did the organization work between participants during this exercise?” (5 meaning very effectively organized) showed increases. Specifically, for the self-organized groups, the self-reported organization measure went from an average of 3.55 (s.d. = 1.12) in the training session to an average of 4.24 (s.d. = 0.83) in the team programming exercise. For the OMP scaffolded groups, the self-reported organization measure went from an average of 3.49 (s.d. = 1.07) in the training session to an average of 4.08 (s.d. = 0.80) in the programming exercise session. There was no statistical difference between the two conditions on the organization measure. This suggests that students start to get more proficient in their roles with practice and shows promise for reduction of discomfort associated with the scaffold as well as an increased possibility of producing beneficial learning opportunities over time. There could have been several reasons for why students experienced discomfort with the OMP scaffold. An open ended question in the survey asking “What could be done to improve the exercise?” points to a few possible explanations. Several students were disgruntled that the roles were not assigned separately by task or their ability to do those tasks - "the roles can be changed according to tasks and individual's ability". This supports our hypothesis that students
are focused on productivity rather than using the task as a learning opportunity and require a scaffold to make best use of the activity for learning. In a similar vein, several students thought that a voice communication channel would produce more efficient and quicker communication reinforcing once again the fixation on productivity. A third possibility is discomfort with the Cloud9 IDE which several students expressed needing more practice with. Like we observed above, this form of discomfort is likely to decrease with practice.

One hope for OMP is that it offers the opportunity to embed learning opportunities during work. This was not an explicit focus of this investigation of OMP. Nevertheless, apart from the formal analysis above, we also conducted analyses to investigate whether the OMP scaffold session had any impact on students’ grades later in the course. We computed the post-OMP session grade by averaging the four assignment grades that happened after the OMP session and refer to it as post-OMP grade. We used post-OMP grade as the outcome variable, condition as the main factor, and controlled for students’ prior grades and familiarity with the team. We do not see a difference in post-OMP grade between the two conditions. This suggests that in the OMP scaffold session, students picked up the distinctive collaborative behaviors associated with each role, while at least not harming their individual learning to prepare them for the later stages of the course. Exposure to more OMP sessions could help us see decreases in the initial discomfort associated with the adoption of the scaffold and start to see more of the learning gains.

3.8 Experimental Setup - Study 3

Finally, in order to study the effects of transactivity in this context, in a 3x3 Latin Square design, we compare students working alone and in two OMP configurations (with and without transactivity-maximization team formation designed to enhance reflection) [90].

3.8.1 Hypotheses

Because the OMP paradigm separates responsibilities into three distinct roles associated with brainstorming ideas, selecting among the ideas being considered by analyzing their pros and cons, and implementing the selected idea, we can expect that the more that teams adhere to the OMP paradigm, the more substantive their discussions will be. The distribution of responsibilities to roles in an inter-dependent fashion can also ameliorate group coordination difficulties and process losses, thus ensuring that the group product quality does not drop. Moreover, the rotation of roles affords a more even distribution of responsibilities among all of the participants. We can hypothesize therefore that -

• **Hypothesis 1** - Teams that demonstrate an elevated level of compliance to the OMP paradigm will discuss project-relevant conceptual content more substantively, contribute work towards the group solution more evenly, and produce a group product with as high of quality as individuals or teams with lower compliance to the OMP paradigm.

In designing this current study, we build on prior work developing a team formation strategy that is associated with idea sharing [114] and reduced problems with distribution of labor and conflict [86]. This team assignment paradigm uses a measure of observed exchange of trans-
active discussion as an estimate of pairwise collaboration potential between students and then
groups teams within a class in such a way as to maximise the estimated pairwise collaboration
potential within teams across the class as a whole. Transactivity as a conversational construct
is known to be more prevalent within groups with more is mutual respect [6] and a desire to
build common ground [35]. The benefits for transactive idea exchange can potentially interface
well with the hypothesized benefits of OMP including more even distribution of work and more
substantive discussions. Thus, we hypothesize that assignment of teams using this transactivity
maximisation approach might increase adherence with OMP practices and further amplify its
benefits. We can hypothesize therefore that -

• **Hypothesis 2** - Groups formed transactively will demonstrate higher compliance with
OMP practices that will be associated with an intensification of the observed benefits of
OMP compliance.

### 3.8.2 Participants and Experimental Design

In order to test the two above specified hypotheses, we experimentally contrast the mob pro-
gramming scaffold in randomly formed and transactively formed groups against individual pro-
gramming in a 3x3 Latin square between-subjects design embedded within a completely online,
graduate Cloud Computing course offered to the students of a large North American univer-
sity and its campuses worldwide. The comparison between individual programmers and groups
will allow us to investigate Hypothesis 1 and the comparison between transactively formed and
random groups will allow us to investigate Hypothesis 2.

The Cloud Computing course offers hands-on experience on the applications of Cloud Com-
puting using three leading cloud computing platform providers: Microsoft Azure, Google Cloud
Platform and Amazon Web Services. Students are expected to have strong programming com-
petency in at least one programming language prior to taking the course. A total of 120 students
took the course allowing 40 students to be assigned to each condition for each exercise. In the
first week of the course, as a part of primers that students had to complete, they were grouped
randomly based only on their time availability to participate in an OMP training session. Prior
to this training session, students were provided materials explaining the mob programming
paradigm. The training session itself required students to solve a relatively simple program-
ming task collaboratively. The task was kept simple so as to allow students to familiarize them-

The experimental study was run as a 3 (Condition) x 3 (Programming Assignment) Latin
square design, where the order of conditions was counter-balanced between three tracks in or-
der to control for potential order effects. The three conditions were Individual, Random Team
Assignment, and Transactivity-Based Team Assignment. Students were assigned randomly to
the three tracks. Each time team formation was done, it was done within a track for an assignment such that all students within that track on that assignment were assigned to teams using the same paradigm. Teams of 4-5 students were formed. Since within each track, team assignment was performed for two different programming assignments, we ensured that teams were formed with students who had not worked together at the other time points, including the training activity that occurred prior to the experimental manipulation.

The transactivity-based groups were formed by maximizing the pairwise transactive exchange observed from a Reflection-Feedback discussion forum exercise prior to the OMP exercise, using the same software published previously and shared by the authors [114]. To provide an estimate of collaboration potential for the transactivity-based team assignment, students were required to post a reflection to the discussion forum after some individual project activities prior to the experimental study and to offer feedback to three other students of their choosing. This feedback exchange provided both the opportunity for students to experience more social interaction in the course, as well as data for estimating the collaboration potential of pairs of students based on their exchange of transactive feedback contributions. The transactivity-based groups were then formed by using a constraint satisfaction algorithm that globally maximizes the average pairwise transactivity across all groups of students [113].

The first assignment of the experimental manipulation started four weeks after the OMP training session and the other two assignments were administered in two week intervals after that. As in the training session, each assignment lasted a total of 80 minutes with 10 minutes being reserved for introductions and wrap-up and role switches prompted by the Intelligent Conversational Agent facilitator happening every 7 minutes. In the first assignment, students worked on the concept of thread synchronization to simulate a bank system that handles deposits and withdrawals; in the second assignment, students used functional programming in Scala to implement a simple application using Spark Distributed Databases; in the third assignment, students solved a binary classification problem by extracting and engineering numeric and categorical features in Python. Students in all 3 conditions performed their work during the manipulation in the same Cloud9 environment. They were supported by the Intelligent Conversational Agent, which provided instructions and managed the time for the activity, informed students of their roles and when they needed to switch roles.

### 3.9 Analysis and Results

#### 3.9.1 Operationalization of Variables from Logged Data

The following data was collected in all conditions to facilitate our analyses -

- **Code contributions and chat logs** - The code contributions allow us to analyze how work was distributed among the group members. Because the student in the Driver role is required to do all of the code changes, we can also measure if students complied with the structure of OMP or not, using the code contributions. Chat logs allow us to analyze the content of the discussions between students.

- **Grades** - Grades on individual assignments and the team project prior to and post the OMP exercises help control for differences in prior knowledge and ability among students.
assigned to any condition. The grades on the exercises themselves, help us see if there was a difference in group product quality across conditions.

- Post-assignment and post-course surveys - Apart from asking for feedback on the activities, the surveys were used to gather insight into student perception of the activity, the learning from the activity, and the group product. The survey provides additional context to analyze the quantitative data obtained from the logs.

In order to quantify whether students complied with the structure suggested by the OMP activity, we calculated a Compliance Score for each of the groups. The compliance score was measured as

\[ \text{compliance\_score} = \frac{\text{code\_driver}}{\text{code\_not\_driver}} \times \frac{n}{n-1} \]

where code\_driver is the number of edits to the code made by participants in the driver role, code\_not\_driver is the number of edits to the code made by everyone else and n is the number of group members. A higher compliance score therefore means that the ratio of the code contributions made by the driver to the number of contributions made by the rest of the members is more, and that constitutes more compliance with the OMP structure.

In order to further pin down how work was being distributed in the group, we calculated the percentage of code contributions made by each group member. From this we computed an Evenness Deviation score, which measured the difference between this percentage and what percentage would be observed for the group if work was distributed evenly. This number was scaled in order to allow for values to vary between 0 and 1 regardless of group size.

Finally, in order to quantify the extent to which activity-relevant Conceptual Content was being discussed in the chat, we measured the vector similarity of the topic representation of the chat contributions of a student with that of the primer corresponding to that activity. The topic representation was constructed using a latent semantic indexing model over a bag of words representation of the set of primer documents using the number of topics set to 5. A higher document similarity score meant that more of the conceptual content from the primer was discussed in the chat.

### 3.9.2 Results and Discussion

The formulation of our hypotheses and accompanying study design were based on the concern that when students are assigned team work in a project course, in dysfunctional teams function like the most capable student within the team because that person takes on the lion’s share of the work, thus undercutting the practice opportunities of team-mates. So, as a foundation for the evaluation of the hypotheses, we first quantified the extent to which this problem was in evidence within the course we chose to study.

We begin by checking to ensure that students in all three conditions achieved equivalent grades on the assignments, and indeed, there was no significant difference. For this test we computed an ANCOVA model with Condition at a time point as the Independent variable, average grade prior to the experiment as Covariate, Assignment time point as a random variable, and Grade at time point as the Dependent variable. There was no significant effect of Condition either for the auto graded portion of the assignments (F(2,348) = .17, p = .88) or for the manually
graded portion of the assignments (F(2,348) = .83, p = .43). Thus, teams (regardless of condition) and individuals achieved the same grade on average. For the remainder of the analysis, we will focus only on teams.

We have observed students defaulting to an uneven distribution of labor in order to achieve an advantage on their grade. In order to test the extent to which the reward structure substantially does encourage an uneven distribution of labor, we computed an ANCOVA model with a three-way split on the Evenness Deviation variable (Top Quartile, Middle, Lower Quartile) as the independent variable, average grade prior to the experiment as Covariate, Assignment time point as a random variable, and Grade at time point as the Dependent variable. The upper quartile had deviation scores of higher than .33, and the lower quartile had deviation scores less than .08. The median deviation score was .2. In 3 percent of teams, a single member did all of the work. There was no significant effect of the deviation variable on grade, though the trend was in the expected direction both for the auto graded and manually graded portions of the assignment. Thus, students may falsely believe it is necessary to deviate from an even distribution of labor when in fact it does not help their grade.

Hypothesis 1: Teams that demonstrate an elevated level of compliance to the OMP paradigm will discuss project-relevant conceptual content more substantively, contribute work towards the group solution more evenly, and produce a group product with as high of quality as individuals or teams with lower compliance to the OMP paradigm.

We first tested for an association between Compliance scores with Conceptual Content scores and found the association to be highly significant (R=.2, p < .005) such that more highly compliant groups focused more on conceptual content in their chats. Then we compared the Conceptual Content scores for students in compliant teams with those in non-compliant teams. We computed a median split on the Compliance score in order to compare students in groups that were highly compliant to OMP practices versus those in groups that were not. Then we compared Evenness Deviation scores for students in compliant teams with those in non-compliant teams using a t-test and found that Evenness Deviation was significantly higher in non-compliant teams F(1, 161) = 4.2, p < .05, effect size .3 s.d. Just as we tested the effect of condition on grade, we also tested compliance to Mob practices. For this test we again computed an ANCOVA model with the median split on Compliance at a time point as the Independent variable, average grade prior to the experiment as Covariate, Assignment time point as a random variable, and Grade at time point as the Dependent variable. There was no significant effect of Compliance. Thus, we have correlational, though not causal, evidence to support the hypothesis that Mob practices are associated with more conceptual focus and more even distribution of labor.

Hypothesis 2: Groups formed transactively will demonstrate higher compliance with OMP practices that will be associated with an intensification of the observed benefits of OMP compliance.

To test this hypothesis we computed an ANOVA model with Condition as the independent variable and Assignment time point as a random variable. Compliance score was the dependent variable. Here we found a trend consistent with the hypothesis, but it was not significant. Thus, Hypothesis 2 is not supported. One explanation is that the OMP structure acts as its own scaffold for idea exchange, which might also make the transactivity manipulation less necessary. This might explain the lack of significant support for Hypothesis 2.
3.10 Proposed Work

We propose to extend this work to the community college context, allowing us to take our experience from injecting workplace-relevant collaborative experiences into university learning environments and apply them to a context where employment is the salient goal. Planning is currently underway and preparation for deployment will happen over the summer of 2019 with the first studies starting in the Fall.
Chapter 4

Data-Driven Improvement of Support for Knowledge Co-Construction (Proposed Work)

This chapter proposes two strands of work that takes advantage of the data collected from several iterations of conversational agent facilitation in the online collaborative programming context for improving this support in future iterations. The first strand involves the use of reinforcement learning to adaptively improve support. We first describe how the conversational agent framework Bazaar that we use to offer collaborative conversational agent support can be adapted to enable reinforcement learning. This includes the derivation of states and actions from the Bazaar logs, how the learning happens, how the results from the learning are converted back into knowledge sources that Bazaar uses, and what happens in each episode and across episodes of the learning. In order to make more data available for learning, we propose to use past iterations of OMP to build a user simulator which will enable the reinforcement learning algorithm to be trained in an interactive fashion enabling us to learn strategies offline before implementing them in a real student context.

The second strand uses the simultaneous conversational and code data generated from the Online Mob Programming sessions to learn code explanations for the current state of the code. These code explanations can be used to provide additional shared context for knowledge co-construction especially for the roles that aren’t tasked with actually writing the code (Navigator and Mob).

4.1 Reinforcement Learning over Knowledge Abstractions

The Knowledge Construction Dialogue [50] (KCD), is a mainstay in tutorial dialogue systems and can be thought of as a stack-based dialogue system architecture that encodes how a conversational system interacts with a student. The KCD can not only be used to control the flow of the conversation with students, it can also be used to "push" sub-dialogues for remediation or additional clarification in cases where that is inferred to be useful. The KCD is generally designed by a human tutor and can be considered as an abstraction of their understanding of the
optimal strategy for maximizing the effectiveness of the tutoring. This can however be turned into a data-driven learning opportunity by allowing the knowledge abstractions to be subtly modified based on feedback from interacting with students. A sample schematic with various actions that the agent can take based on student responses in the Physics domain is shown in 4.1. This knowledge abstraction is encoded into Bazaar specific XML representation which is then interpreted by the framework while interacting with the student.

4.1.1 Reinforcement Learning System

The KCD results in a sequence of actions by the dialogue system that eventually lead to success or failure. We cannot infer much from each individual action that the system takes. However, we can assign a positive or negative reward to the sequence of actions that the dialogue system takes. These rewards are associated with subtly different knowledge abstractions allowing us to eventually converge on the best abstraction using a Reinforcement Learning paradigm.

As dialogue progresses as a sequence of interspersed actions (meaningless utterances, answers, silence, questions), the dialogue moves from one state to the other, where each state can be uniquely identified by the conversation history. While this is a comprehensive definition, it leads to an unwieldy growth of the state space. We propose to reduce the state space by simplifying each action as a binary indicator which suggests if the answer was correct or not. This not only enables rapid learning through the shrunken space but also narrows the scope of the agent. As a result, every state is uniquely identified by the dialogue history which can be visualised in 4.2. Issuing positive and negative rewards are specific to the context but as an example, a significant positive reward and negative reward can be associated with task success and task failure respectively. Smaller penalties can be assigned to cases where more prompting from the agent is necessary to achieve task success. This incentivizes the agent to choose the right set of prompts that can lead to task success quickly.

4.2 Reinforcement Learning of Dialogue Behavior

Another important component of the dialogue system is the output coordinator. In order to prevent the conversation feed from being clogged by too many agent messages (in some cases irrelevant by the time they appear), all of the output messages are associated with a priority and timeout and sent to the output coordinator. The output coordinator acts as the gatekeeper using these priority and timeout values to decide which messages actually get sent to the output. Higher priority items are promoted over lower priority ones, and items that timeout are dropped from the queue. An example series of output coordinator actions is shown in the figure 4.3. The priority and timeout values produce interesting variations in behaviors at the output level and are once again assigned by the designed of the system without much of an understanding for how they interact with each other and influence the output. We can once again frame this problem as a reinforcement learning problem using an offline update methodology.

We learn the optimal behavior of the agent iteratively, learning from previous episodes. In this regard we collect the log files after every episode and use it to update the agent's behavior for the next episode. Essentially, we calculate the reward values after each episode. The logs
Figure 4.1: A sample physics knowledge abstraction with various actions the agent can take based on the student's responses
Figure 4.2: Example State Space Representation

capture various aspects of the dialogue. These include - The timestamp: This captures the timing of an utterance from the student or the tutor agent. The actor: The entity delivering the dialogue. The dialogue: The tutor agent's dialogue or the student's response.

4.2.1 Reinforcement Learning System

We propose a model based approach for framing the above problem as Markov Decision Process. A Markov Decision Process is a tuple $(S, A, P, R, \gamma)$ where

- $S$ is a finite set of states
- $A$ is a finite set of actions
- $P$ is a state transition probability matrix
- $R$ is a reward function
- $\gamma$ is a discount factor $\gamma \in [0, 1]$

**State Space** Every state is represented by a vector representation of priorities (for instance) which can take a set of particular values. In a simplistic scenario, priority values can be $[0, 0.25, 0.5, 0.75, 1]$.

**Actions** Every action can be seen as either incrementing or decrementing single priority value based on predefined step size [e.g.: 0.25]. Assuming three possible events we are choosing from starting with priority values of 0 for all three, the sample actions are as follows -

- 0, 0, 0.25 (increasing priority of the third event)
- 0, 0, -0.25 (decreasing priority of the third event)
- 0.25, 0, 0 (increasing priority of the first event)
- 0, -0.25, 0 (decreasing priority of the second event)

and so on.

**Reward Function** In supporting transactive exchange between students, the reward can...
Figure 4.3: Output Coordinator Actions using Priority and Timeout
be the number of unique pairs of transactive interactions observed in a conversation. In each episode, agent is deployed with a set of priority values for given events. With the same set of values, multiple iterations are executed. The chat logs are collected for each such episode and reward function is calculated at the end of it. The reward function value is averaged across all episodes to account for the non deterministic nature of multi party dialogue. This averaged reward value is used to update the agent for the next episode by determining the value function of each state with the new reward value. The optimal value and policy is determined and a new agent is deployed with new set of priority values. This process is repeated until the agent reaches a state that consistently gives optimal value across different episodes.

4.3 Code Explanation Generation

The OMP context is unique because it captures both code edits, as well as the conversations that led to those edits being made. Building off of prior attempts at generating pseudo-code and structured representations from code [4, 70] as well as code from natural language [42, 118] we propose producing explanations of code that can then be used as shared context for knowledge co-construction especially for the roles that aren’t tasked with actually writing the code (Navigator and Mob).
Chapter 5

Post-Operative Conversational Support for Patients and Caregivers (Proposed Work)

A Left-Ventricular Assistive Device (LVAD) is a surgically implanted mechanical pump attached to the heart to assist its function in patients with advanced heart failure. They have become important and widely accepted as a treatment option for severe, acute, and chronic heart failure for use in four main scenarios - as a bridge to a heart transplant, as a bridge to a decision (heart transplant or other treatment options), as a bridge to recovery or as a destination therapy (in patients not suitable to receive heart transplants) [48, 62, 78]. Even as their prevalence has increased, with over 40% of the patients waiting for a heart transplant receiving an LVAD [92], patient support tools, including two validated paper-based decision support tools, have focused primarily on the decision of whether to get an LVAD or not [23, 117].

A recent interview study recorded the issues experienced by both patients and caregivers before and after the LVAD decision in order to assess their values and concerns in the context of the decision-making experience. One striking finding of this interview study was the sentiment that most patients felt that they "had no choice" in getting the LVAD implant. Similar themes such as "LVAD or death" and "I didn’t have any alternatives" were echoed in a majority of the interviews. In contrast, the primary concerns of the patients and their caregivers revolved around adapting to living with an LVAD with significant mentions of stress of waiting or fear of another surgery impacting the future decision to get a heart transplant and concerns about managing everyday activities like showering. Caregivers also talked about not being able to take care of their own health because of a singular focus on the patient’s needs [60]. We believe therefore, that the decision support tool should not focus on the LVAD itself, which most patients and caregivers felt was a non-decision and focus instead on supporting both patients and caregivers in living healthfully together post undergoing this life-changing medical procedure. The conversational agent support is situated in the home sphere which can serve as an environment separated from the stresses of the hospital and the involvement of the doctor allowing patients and caregivers to discuss at their leisure. Furthermore, prior research indicates that participants are more likely to discuss about sensitive topics with an agent than with a human [75]. The conversational agent can therefore initiate conversation about sensitive
topics without the need for either participant to incur the social cost of starting it themselves.

5.1 Proposed User Experience

The patient and caregiver are given separate logins to the system that they can access on a phone, a desktop, a laptop or a tablet. The proposed user experience flow is shown in Figure 5.1. As a part of Sorting Activity 1, patients and caregivers are first shown a subset of cards, each highlighting a different concern that they may have including bathing, swimming, diet, weight management, risk of infection and maintaining the LVAD. The full list of cards is shown in Figure 5.2. After receiving additional information (if needed) about the subset of cards, the patient and caregiver are asked to sort them using criteria that makes sense to them. For example, by increasing order of importance or increasing order of burden. The activity then proceeds to the discussion phase moderated by the conversational agent. Here, they are allowed to view their partner’s ordering and prompted to explain how they think their partner’s ordering was done. This allows the patients and their caregivers to understand the concerns that are most important to each of them and provide them an opportunity to discuss more substantively about those concerns. Conversational agent support for transactive exchange in this context is hypothesized to help patients and their caregivers achieve common ground on their concerns. The final sorting activity involves the patient role-playing as the caregiver and vice versa. They choose cards that they think would be the primary concern for the role they are playing and are then prompted to discuss about the difference in their choices. This activity allows them to once again work towards common ground as they build off of the understanding they have gained in the previous activity. The activities are summarized in the Figure 5.3.
• Bathing
• Swimming
• LVAD implant surgery
• Driving a car
• Risk of stroke
• Smoking
• Driveline maintenance/dressing changes
• Chances of Heart Transplant
• Getting out of the house
• Diet
• Weight management
• Traveling
• Taking a shower
• Battery
• Sex and intimacy
• Disease prognosis
• Risk of infection
• Mental Wellness
• Medications
• Trips to the hospital
• Finances

Figure 5.2: List of Information Cards

Sorting Activity 1
Subset of cards are presented to each participant (patient, caregiver)
Prompt: Please organize these cards, using any criteria that makes sense to you. E.g., by order of importance or order of burden
Time limited activity – Confirm when task is complete for both participants

Discussion Activity
Organized cards are presented to the other participant (patient sees caregiver’s cards and visa versa)
Prompt to caregiver: View your partner’s cards, explain how you think they did the ordering.
Discuss aloud.
Prompt to patient: Explain how you organized your cards
Once complete, prompts switch and the patient explains how they think the caregiver organized their cards.
Indicate when second discussion and sharing are complete

Sorting Activity 2
Whole set of cards are presented to each participant (patient, caregiver)
Prompt: Pretend you are your partner, please select and order 5 cards that you think would be top concerns
Time limited activity – Confirm when task is complete
System assesses card choices and differences
Prompt: You selected X card, while your partner did not – please discuss
Patients and caregivers talk about why they made their decisions
Indicate when discussion and sharing are complete

Figure 5.3: Sequence of Activities
Chapter 6

Conversational Agent Support for Civil Political Discourse (Proposed Work)

As political tribalism continues to increase in recent years, it is worth recognizing that the skills required to engage in productive civil discourse are generally only taught implicitly in traditional education contexts (if at all). There is no doubt that social media and cultivated news feeds have probably exacerbated this problem [47], but the roots of political tribalism likely run much deeper, in what we value and how we judge morality. Some have argued that it is differences in values that make it difficult to communicate across tribes. Moreover, because these values manifest primarily as intuitions [97], merely injecting more logical reasoning into the dialogue is unlikely to improve the communication [37]. The features of this problem space suggest that the productiveness of dialogue between two people of differing beliefs may be improved by the use of some mediating agent, and we therefore hypothesize that collaborative conversational agent support can play a role in facilitating productive civil discourse.

6.1 Research Questions

- Can conversational agent prompts encouraging transactive interaction make the civil discourse between two people with opposing views more productive?
- How do the dynamics of the interaction change as the metaphorical distance between values increases or decreases?

**Hypothesis:** The experimental condition, where a conversational agent is prompting participants to interact more transactively with each other, leads to more productive discourse when compared to the control condition where this prompting is absent.

To test our hypothesis, an experiment to collect discourse data will be conducted according to the experimental design described in the following section. In brief, a verbal protocol analysis will be conducted to measure transactivity and civil productivity. Concurrently, a simulated conversation think-aloud will be conducted to produce new knowledge that we hope will illuminate the shortcomings and potential future improvements to the conversational agent. Each of these two stages are described in detail below.
6.2 Proposed Experimental Design

Participants will be recruited from online participant recruiting platforms (Amazon Mechanical Turk/Prolific) with the restriction that they are United States citizens and over 18 years of age. The primary benefit of online recruitment is that it results in a political diverse sample (as demonstrated in [22]). Participants will first be asked to complete the Moral Foundations Theory Questionnaire [31], as well as some questions about their specific political beliefs. For example:

The government has no right to regulate firearms. Agree or Disagree

These specific questions will be used to pair each participant with another participant who holds the opposing viewpoint. Participants will be held in an online waiting room while waiting for a match. Once a match is found, they will be redirected to an online chat room, where they will be asked to help their partner better understand why they made the choice they did, and discuss for 10 minutes.

In the control group, participants will be left to discuss the issue on their own (i.e., without intervention). In the treatment group, participants will be periodically prompted by a conversational agent to engage more transactively with each other. In each condition, the discourse will be logged for analysis.

After the discourse data is collected, the conditions will be blinded by removing agent dialog and coded for two constructs -

Transactivity: the extent to which learners act on each other’s reasoning [101]

Civil Productivity: interaction that fosters democratic goals [73]

After the data is coded, the condition will be unblinded, and the frequencies of each measure from each condition will be compared with a Chi-Squared Test, to test our hypothesis.

While the verbal protocol analysis provides information about the effectiveness of the current iteration of the conversational agent, what it fails to provide is meaningful information about the underlying thought processes that motivate any differences across conditions. Making these thought processes explicit will likely highlight the problems with the current iteration of the agent, as well as provide inspiration for future improvements.

Unfortunately, conducting a traditional think aloud during a dialogue is impossible, both because it would be too intrusive (i.e., would interrupt the natural flow of the conversation), and also because it violates the privacy of unshared thoughts, which may undermine the social good-will built from the shared thoughts.

To overcome these limitations, a simulated conversational think aloud will be conducted, in which one of the participants in the discussion will be a collection of statements made in prior, real-world conversations. Specifically, the statements will come from data collected during our experiment. Participants in this think aloud will then respond to each statement as if they are having a conversation with a person, but will verbalize their thought processes while they craft their response.

We hope to understand the applicability of the theoretical concepts behind collaborative conversational support such as knowledge building, common ground and transactivity to better support civil political discourse.
Bibliography


Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001. 2


[38] Regine Hampel and Mirjam Hauck. Towards an effective use of audio conferencing in
distance language courses. 2004. 3.2.3


[40] Catherine Harder, Geoff Jackson, and Janet Lane. Talent is not enough. Canada West Foundation, 2014. 2


[51] Mahesh Joshi and Carolyn Penstein Rosé. Using transactivity in conversation for summarization of educational dialogue. In Workshop on Speech and Language Technology in


symposium on Computer science education, pages 177–182. ACM, 2013. 2.1


[86] Sreecharan Sankaranarayanan, Cameron Dashti, Chris Bogart, Xu Wang, Majd Sakr, and Carolyn Penstein Rosé. When optimal team formation is a choice-self-selection versus intelligent team formation strategies in a large online project-based course. In International Conference on Artificial Intelligence in Education, pages 518–531. Springer, 2018. 2.1 3.8.1

[87] Sreecharan Sankaranarayanan, Cameron Dashti, Chris Bogart, Majd Sakr, and Carolyn Rose. Transactivity-based team-formation as a choice: Evaluation in a large online project course. 7th International Collective Intelligence Conference, Jun 2019 (In Press). 1.1

[88] Sreecharan Sankaranarayanan, Cameron Dashti, Chris Bogart, Xu Wang, Majd Sakr, and


[99] GR Stockwell. Cmc for language learning: Examining the possibilities. In *JALTCALL 2004 Conference, Tokiwa University, Mito, Japan*, 2004. [3.2.3]


