Online Mob Programming: Bridging the 21st Century Workplace and the Classroom

Sreecharan Sankaranarayanan, Xu Wang, Cameron Dashti, Marshall An, Clarence Ngoh, Michael Hilton, Majd Sakr, Carolyn P. Rosé
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213
{sreechas, xuwang, cdashti1, haokanga, pcn, mhilton, msakr, cprose}@cs.cmu.edu

Abstract:
We investigate how a collaborative software development paradigm in the workplace can be adapted for collaborative project-based learning in the classroom. The paradigm, called Mob Programming, where a group of co-located developers work on one problem concurrently, inspires Online Mob Programming which structures groups of 3-6 students collaborating online, in a platform integrated with automated support for role rotation. Results from this study comparing OMP scaffolding with self-organization in a university computer science course shows OMP scaffolds help students adopt OMP roles without a significant drop in group product quality.

Introduction

Increasing automation is largely blamed for the projected loss of over 5 million jobs by the year 2020, as argued by the World Economic Forum. Even as the use of automation for collaborative learning support is increasingly cast in a dystopian light (Rummel et al., 2016), we can choose instead, to adopt an optimistic view working toward the use of CSCL for on-the-job training and employee reskilling. As a step closer to that end goal, we start in the software engineering context by adapting an industry inspired paradigm for collaborative software development called Mob Programming (Zuill and Meadows, 2016; Wilson, 2015) into a collaborative learning paradigm for the classroom called Online Mob Programming (OMP). By focusing on a paradigm already popular in the industry, we aim for two distinct benefits: first, allowing students to engage in an industry relevant group practice in order to prepare them for work in industry; and second, developing an empirical foundation for structuring work practices in ways that might facilitate injecting collaborative learning opportunities into the workplace of the future. Evidenced by the rise of standards such as 21st Century Skills (Burrus et al., 2013), providing workplace-relevant learning opportunities for students is an increasing concern even as the rate at which the job market is changing is rapidly increasing. Against this backdrop, we offer a foundational study.

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 participant or group of participants who consider and deliberate between multiple alternative implementations ultimately informing the decision of the Navigator**;** ***Facilitator***: A single participant 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 roles 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.

The concern with group collaborative activities is that group processes could come in the way of learning and productivity, and without structure, it can descend into chaos. 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 (Johnson and Johnson, 1984). In our work, the role of the facilitator would be performed by an intelligent conversational agent based on the open source Bazaar framework (Adamson et al., 2014). 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. Assigning students to roles and periodic rotation of the role assignments can prevent the adoption of suboptimal strategies. Moreover, the role assignments can mitigate the problem of social loafing by introducing an element of social pressure but at the same time balancing that pressure with support from team members in other roles. Students are exposed to different perspectives in solving problems, building solutions, experimenting, debugging and writing readable code. They are also forced to externalize their thinking, which provides the opportunity for knowledge gaps to be revealed and addressed. Finally, they also have the opportunity to observe knowledge and expertise in action as they are learning. This study will serve as a first step towards realizing these benefits of Mob Programming.

Experiment and Results

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 less expert 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.*** In order to test our hypotheses, we experimentally contrast the mob programming scaffold against student self-organization in a between subjects design embedded within a *completely online* software development course. A total of 120 students took the course organizing themselves into teams of 3 for a semester long course project. In the first week of the course, they were grouped randomly into teams based only on their time availability to participate in an OMP training session. The experimental manipulation took place two weeks after the OMP training session when students had acquired the prerequisites necessary to complete the programming task. We collected logs of code contributions and chat logs from the team programming activity, 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, and a Post team programming exercise survey asking 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.

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 within the collected data streams 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. We expected see characteristic patterns between roles to be more distinctive in the OMP condition. The distinctions we found in interpretation of a factor analysis that distinguish between supported roles are consistent with what we would expect based on their definition. Hypothesis 1 was 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 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. One cluster contained only turns from the 3 supported roles. 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. 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. The evidence that students in the Mob programming condition were less comfortable was marginal. Importantly however, a linear regression 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. Hypothesis 3 is thus partly supported.

This work was funded in part by NSF grants IIS 1822831 and IIS 1546393, and funding from Google.

References

Rummel, N., Walker, E., & Aleven, V. (2016). Different futures of adaptive collaborative learning support. *International Journal of Artificial Intelligence in Education*, *26*(2), 784-795.

Burrus, J., Jackson, T., Xi, N., & Steinberg, J. (2013). Identifying the most important 21st century workforce competencies: An analysis of the Occupational Information Network (O* NET). *ETS Research Report Series*, *2013*(2), i-55.

Zuill, W., & Meadows, K. (2016). Mob Programming: A whole Team Approach. In *Agile 2014 Conference, Orlando, Florida*.

Wilson, A. (2015, May). Mob programming-what works, what doesn't. In *International Conference on Agile Software Development* (pp. 319-325). Springer, Cham.

Adamson, D., Dyke, G., Jang, H., & Rosé, C. P. (2014). Towards an agile approach to adapting dynamic collaboration support to student needs. *International Journal of Artificial Intelligence in Education*, *24*(1), 92-124.

Buchan, J., & Pearl, M. (2018, June). Leveraging the Mob Mentality: An Experience Report on Mob

Johnson, D. W., & Johnson, R. T. (1984). Structuring groups for cooperative learning. *Organizational Behavior Teaching Review*, *9*(4), 8-17.