Agent-in-the-Loop: Conversational Agent Support in Service of Reflection for Learning During Collaborative Programming

Sreecharan Sankaranarayanan(✉), Siddharth Reddy Kandimalla, Sahil Hasan, Haokang An, Christopher Bogart, R. Charles Murray, Michael Hilton, Majd Sakr, and Carolyn Rosé

Carnegie Mellon University, Pittsburgh, PA, USA
{sreechas,skandima,sahilh,haokanga,cbogart,rcmurray,mhilton, msakr,cprose}@andrew.cmu.edu

Abstract. Dynamic conversational agent-based support for collaborative learning has shown significant positive effects on learning over no-support or static-support control conditions in prior studies. In order to understand the boundary between human-led and AI-led support for collaboration, we compare in this study an approach where the agent's primary role is to help students regulate their own collaboration with two more typical prompting strategies that are used only during a reflection phase: one designed to provide a specific informational focus for the reflection, and the other designed to draw out evaluation, elaboration, and exploration of alternative perspectives. Significant positive effects on learning over and above just the human-led form of support are observed when either of the prompting strategies are used.

Keywords: Conversational agents · Human-AI collaboration · Reflection prompts · Group conversational agents · Adaptive Collaborative Learning Support (ACLS) · Collaborative programming

1 Introduction

In an article in the 25th anniversary issue of IJAIED, Rummel and colleagues contrast two possible futures for adaptive collaborative learning support (ACLS) [11]: In one more dystopian future, an intelligent agent has tremendous AI-enabled capabilities and the resulting blind trust in these abilities leads to practices experienced by students as inscrutable and lacking in nuance. In the second, more utopian vision, the system not only takes into account multiple dimensions of support [4,16] but also balances this adaptivity with user freedom and shared user/system control [11]. While a great many studies have demonstrated a significant positive impact on learning for fully AI-enabled support for collaborative

I. I. Bittencourt et al. (Eds.): AIED 2020, LNAI 12164, pp. 273–278, 2020.
https://doi.org/10.1007/978-3-030-52240-7_50

learning compared to no-support control conditions [1,7–10], contrasting AI-enabled support to human-led support will allow us to understand the boundary between the two and work towards the more utopian vision.

We situate our study in a synchronous programming activity in an online graduate-level course on Cloud Computing offered at Carnegie Mellon University and its international branch campuses. The activity is divided into several tasks. Within each task, students work in groups of 4 in complementary roles designed with the purpose of assisting each other and furthering the progress of the group as a whole. Thus, the locus of support resides with the students themselves, in an effort to embody the more utopian vision of AI. In this human-led design, the conversational agent only serves as the agent-in-the-loop to provide automated feedback regarding how well students perform their roles – in effect, helping students help each other.

Added to this human-led support, we investigate two more traditional fully AI-enabled conversational agent supports in the form of agent-led reflective discussions at the end of each programming task: one designed to provide a specific informational focus for the reflection, and the other designed to draw out evaluation, elaboration, and exploration of alternative perspectives. The experimental manipulation enables us to test whether the addition of fully automated support produces learning gains over-and-above the human-led support (The Automated Support Benefit Hypothesis).

Results of the 2×2 experimental study show that specific portions of the programming activity lend themselves to pre- to post-test learning and within those portions, a significant improvement in learning is observed over-and-above that of the human-led support when either of the two agent-led supports are offered.

2 Method

A summary of the course structure and the location of the study within it is shown in Fig. 1. Within the first sub-unit of the fourth project unit of the course, students work with our synchronous collaborative software development activity, called the Online Programming Exercise (OPE) in an 80-min long session. A total of 101 students from across three campuses completed the activity to build an inverted index using the Scala programming language, and 100 of these students completed the subsequent project.

Based on instructional design best practices [2], we divide the overall programming activity into five different tasks which target five learning objectives (LOs). Each task is divided into a problem-solving phase where students work on the programming task, and a discussion phase where they participate in a reflective discussion based on the task. This task structuring can be considered a macroscript [3] that sequences the activity into learning phases as described in the Script Theory of Guidance [5]. Each LO is assigned two multiple-choice questions on the pre- and post-tests to measure student learning from the activity. Student performance on the subsequent individual project associated with the task then serves as a delayed post-test as show in Fig. 1.

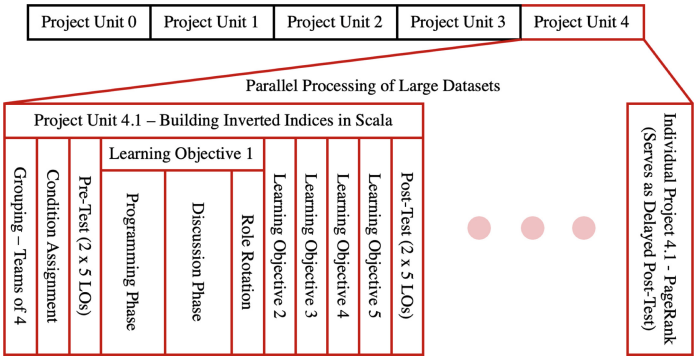

Fig. 1. Course Structure, Pre-Test, Post-Test and Delayed Post-Test Alignment

Within each task, based on the industry paradigm referred to as Mob Programming [6,12–14,17,18], we specified four interdependent roles with well-defined responsibilities that students are assigned to. The *Driver* is the only participant who writes the code, based on high level instructions received from the *Navigator*, who makes decisions on the next course of action based on discussion with the rest of the team members which include the *Researcher* who assists the group with ideation and implementation by consulting external support material, and the *Project Manager* who is responsible for making sure the rest of the team members are complying with and adequately performing their roles. The roles rotate after each task. This role-scaffolding paradigm can be considered a microscript that provides support for the collaboration within a learning phase. The control condition consists of the task structuring macroscript used in combination with the role-scaffolding microscript.

In the experimental conditions, we additionally investigate two more traditional conversational agent supports in the form of discourse level prompts during the discussion phase which can also be considered microscripts. *Information Prompts* support learners in warranting their claims (Ex: *"@Researcher, what is the advantage of writing OS-aware code like you did here?"*) and *Elaboration Prompts* explicitly prompt another learner to build on an existing argument towards knowledge construction (Ex: *@Driver, How would you improve the implemented approach?*) [15].

We tested the Automated Support Benefit Hypothesis with a 2 × 2 factorial design in which the first factor was the presence or absence of information prompts, and the second factor was the presence or absence of elaboration prompts. The teams were randomly placed into the four conditions: 7 groups in the control condition where no prompts were presented, 6 groups presented with elaboration prompts, 5 with information prompts only and 9 groups where both prompts were presented.

3 Results

We first test pre- to post-test learning gains from the exercise. For LOs 3 and 4 there was a significant pre- to post-test gain as measured with a 2-tailed paired t-test, $t = 2.43$, $p < .05$ indicating that these two tasks lent themselves to learning during programming much more than the other tasks. For LO 2, average pre-test score was 1.7, post-test score 1.8, and standard deviation .6. For LO 3, average pre-test score was 1.5, post-test score 1.6, and standard deviation .6. Because of the learning gains achieved in the two LOs, we are able to test our hypothesis regarding the intensification of learning in the experimental conditions.

We used a repeated measures ANCOVA model, with LO and role as random variables, pre-test score (per LO) as a covariate, elaboration prompts and information prompts and the interaction between the two as independent variables, and post-test score (per LO) as the dependent variable. As an aside, there was no statistically significant difference in learning between roles.

In terms of pre- to post-test gains, there was no significant main effect of the elaboration prompt factor; $F(1, 440) = .21$, $p = n.s.$ However, there was a significant interaction effect between the two experimental factors $F(1, 440) = 11.6$, $p < .0001$. In a post-hoc analysis, we determined that both of the conditions with only one type of prompt were associated with significantly more learning than the control condition, and the condition with both types of prompts was not significantly different from control. The effect size of the addition of elaboration prompts over no prompts was .32 s.d., which is a medium effect size. The effect size of the addition of information prompts over no prompts was .42 s.d., which is a medium effect size.

To test the impact on a subsequent individual programming task, we built an ANOVA model, with elaboration prompts and information prompts and the interaction between the two as independent variables, to measure the impact of the experimental manipulation separately on three outcome measures related to task performance: time on subsequent programming task, number of submitted attempts on that task, and score. Here, there was a trend for the elaboration condition to improve performance in terms of time-on-task, number of submission attempts, and score, though none of these were statistically significant. For the information prompts also, the trend was consistently that they were associated with lower time on task, lower number of submissions, and higher scores.

Thus, the use of elaboration prompts or information prompts alone significantly improve on pre- to post-test learning from the task and exhibit positive trends for the subsequent delayed test.

4 Conclusion

Based on the results, we can conclude that agent-led support shows promise for augmenting and significantly improving over primarily human-led support.

Acknowledgements. This work was funded in part by NSF grants IIS 1822831, IIS 1917955 and funding from Microsoft.

References

1. Adamson, D., Dyke, G., Jang, H., Rosé, C.P.: Towards an agile approach to adapting dynamic collaboration support to student needs. Int. J. Artif. Intell. Educ. **24**(1), 92–124 (2014)
2. Carver, S.M.: Cognition and instruction: enriching the laboratory school experience of children, teachers, parents, and undergraduates. In: Cognition and instruction: Twenty-Five Years of Progress. pp. 385–426. Lawrence Erlbaum Associates (2001)
3. Dillenbourg, P., Hong, F.: The mechanics of cscl macro scripts. Int. J. Comput.-Support. Collaborative Learn. **3**(1), 5–23 (2008)
4. Diziol, D., Rummel, N.: How to design support for collaborative e-learning: a framework of relevant dimensions. In: E-collaborative Knowledge Construction: Learning from Computer-Supported and Virtual Environments, pp. 162–179. IGI Global (2010)
5. Fischer, F., Kollar, I., Stegmann, K., Wecker, C.: Toward a script theory of guidance in computer-supported collaborative learning. Educ. Psychol. **48**(1), 56–66 (2013)
6. Hilton, M., Sankaranarayanan, S.: Online mob programming: effective collaborative project-based learning. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, p. 1283. SIGCSE 2019, Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3287324.3293774, https://doi.org/10.1145/3287324.3293774
7. Kumar, R., Rose, C.: Architecture for building conversational agents that support collaborative learning. IEEE Trans. Learn Technol. **4**(1), 21–34 (2011)
8. Kumar, R., Rosé, C.P.: Triggering effective social support for online groups. ACM Trans. Interact. Intell. Syst. (TiiS) **3**(4), 24 (2014)
9. Kumar, R., Rosé, C.P., Wang, Y.C., Joshi, M., Robinson, A.: Tutorial dialogue as adaptive collaborative learning support. Front. Artif. Intell. Appl. **158**, 383 (2007)
10. Rosé, C.P., Ferschke, O.: Technology support for discussion based learning: from computer supported collaborative learning to the future of massive open online courses. Int. J. Artif. Intell. Educ. **26**(2), 660–678 (2016)
11. Rummel, N., Walker, E., Aleven, V.: Different futures of adaptive collaborative learning support. Int. J. Artif. Intell. Educ. **26**(2), 784–795 (2016)
12. Sankaranarayanan, S.: Online mob programming: effective collaborative project-based learning. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, p. 1296. SIGCSE 2019, Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3287324.3293709, https://doi.org/10.1145/3287324.3293709
13. Sankaranarayanan, S., et al.: Online mob programming: bridging the 21st century workplace and the classroom (2019)
14. Sankaranarayanan, S., et al.: An intelligent-agent facilitated scaffold for fostering reflection in a team-based project course. In: Isotani, S., Millán, E., Ogan, A., Hastings, P., McLaren, B., Luckin, R. (eds.) Artificial Intelligence in Education, pp. 252–256. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23207-8_47
15. Stegmann, K., Weinberger, A., Fischer, F.: Facilitating argumentative knowledge construction with computer-supported collaboration scripts. Int. J. Comput. Support. Collaborative Learn. **2**(4), 421–447 (2007)

16. Walker, E., Rummel, N., Koedinger, K.: Beyond explicit feedback: new directions in adaptive collaborative learning support (2009)
17. Wilson, A.: Mob programming-what works, what doesn't. In: International Conference on Agile Software Development. pp. 319–325. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18612-2_33
18. Zuill, W., Meadows, K.: Mob programming: a whole team approach. In: Agile 2014 Conference, Orlando, Florida, vol. 3 (2016)