

Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Computing and
Information Systems

School of Computing and Information Systems

4-2014

Automated Mentor Assignment in Blended Learning Environments

Chris BOESCH

Singapore Management University, cboesch@smu.edu.sg

Kevin STEPPE

Singapore Management University, kevinsteppe@smu.edu.sg

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research



Part of the [Higher Education Commons](#), and the [Software Engineering Commons](#)

Citation

BOESCH, Chris and STEPPE, Kevin. Automated Mentor Assignment in Blended Learning Environments. (2014). *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T): Proceedings: April 23-25, 2014, Klagenfurt, Austria*. 94-98.

Available at: https://ink.library.smu.edu.sg/sis_research/2046

This Conference Proceeding Article is brought to you for free and open access by the School of Computing and Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Computing and Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email cherylids@smu.edu.sg.

Automated Mentor Assignment in Blended Learning Environments

Chris Boesch and Kevin Steppe
School of Information Systems, Singapore Management University
{cboesch, kevinsteppe}@smu.edu.sg

Abstract

In this paper we discuss the addition of automatic assignment of mentors during in-class lab work to an existing online platform for programming practice. SingPath is a web based tool for users to practice programming in several software languages. The platform started as a tool to provide students with online feedback on solutions to programming problems and expanded over time to support different of blended learning needs for a variety of classes and classroom settings. The SingPath platform supports traditional self-directed learning mechanisms such as badges and completion metrics as well as features for use in classrooms, such as tournaments. We evaluate the addition of the mentor assignment feature during two short workshops designed to introduce students to the Python and JavaScript programming languages. The introduction of the mentor assignment features provided a more collaborative and engaging experience compared with previous courses.

KEYWORDS

Game-based learning, personalized learning, blended learning, software education

1. Introduction

Personalized, one-to-one teaching and mentoring has been recognized as one of the most effective methods of maintaining student engagement and enhancing learning[1]. We have been looking for ways to scale-up more personalized assessment and learning using tool support. In 2011, we invited nearly 200 students who were on the borderline of being accepted or rejected for university admission to learn basic Java and Python on their own. They were instructed to use SingPath, our tool for practicing programming in a self-directed manner, to demonstrate minimal competence, and then take part in a short software tournament[2]. Since then, we have extended the platform to support software lab delivery in classroom settings [3] and extended to support more personalized, self-directed learning in preparation for classroom sessions [4]. After observing numerous students preparing for in-class tournaments and observing how students interacted during tournaments, we looked for ways to improve engagement and mentoring opportunities. One opportunity was to use the tool to match better-prepared and more capable students as mentors for the less prepared and capable students.

In our software tournaments, participants are usually asked to solve around ten small problems. A picture of the interface is shown below in Figure 1. These are sometimes run at the beginning of a class period to check students preparation on homework. Other times they are run at the middle or end of the period to evaluate the participants' understanding of the material. In a classroom where the objective is to move all the students through the same material, varying levels of student capability can cause a variety of issues. First, the most capable students are likely to finish solving the problems in much less time than the least capable students. In our experience, it is not uncommon to see some students finish problems

in as little as one fifth of the median class completion time while the least prepared students might not be able to finish solving all of the problems regardless of the time provided. This can make allotting time for tournaments challenging since many students in a class will be finishing early while others will continue to need time. Secondly, it can be frustrating and discouraging for the least capable students to continue working on the first few problems as they realize that many of their peers have completed all of the problems. Thirdly, the most capable students in class can become bored and distracted as they wait for their peers to finish. And finally, it is difficult for a single instructor to assist the least capable students quickly and efficiently enough to close the completion time gaps between the fastest and slowest students. We felt that student mentoring could help alleviate all these issues. The less capable students would receive more individual assistance and be less frustrated; the more capable students would remain engaged; and the total time to completion would be reduced.

The screenshot displays the SingPath.com interface for a JavaScript practice problem. At the top, it shows 'Level Progress: 13/14' with a green progress bar and 'Game progress 0 of 5'. The 'Current Practice Question' section includes the name 'Double String' and a description: 'Create a function called double_word() that takes a string as a parameter and returns the string duplicated.' To the right, it specifies 'Path Name: Javascript', 'Level: 3', and 'Difficulty: Easy', with a 'Skip' button. Below the question is a code editor with a 'Run' button. The editor contains the following code:

```
1 - function double_word(w){
2   return w;
3 }
```

To the right of the code editor is a 'Sample Answer' table with the following data:

	Expected	Recieved	Correct
	funfun	fun	false
ssippi'	MississippiMississippi	Mississippi	false

Figure 1: Practice and tournament problem solving on SingPath.com

2. Automated Mentor Assignment

During software tournaments, a live ranking is usually displayed on a screen to show the class progress. This ranking provides feedback to the students about how the class is progressing and provides the class instructor with information as to which students may need the most assistance and how long the class is likely to take to finish the tournament. In some classes we have experimented with asking early finishers to help their peers. This informal process suffers from various social frictions. First the early finishers usually will turn to help their friends, who generally are not the ones who need help. Second, the mentored student is not directly informed that help is coming, thus the pair does not have process cues about what to do, frequently resulting in no mentoring actually occurring. An automated process can solve these issues.

In preparation, we arranged empty seats next to all students so that finishing students could move to the location of the students that they will be mentoring. Students were also given

instructions on how the mentors would be assigned. When a student solves the final problem in the tournament, they are assigned to serve as a mentor to the student currently at the bottom of the ranking. Students are notified on the ranking board of whom they will be mentoring. Additionally next to the mentored student is placed the name of the mentor. See Figure 2 for an example from our study. The mentors then move to the location of their mentee. Once the mentor arrives, they are not allowed to touch the mentee's keyboard or mouse, but are free to offer suggestions as to what the mentee should attempt to do next.































#	Player	Country	Type	1	2	3	4	5	6	7	8	9	10	Last solve time	Now Mentoring	Mentored By
1	 Erwin		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 04 min 35 sec	Mrchamp	
2	 Melvin		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 04 min 50 sec	Jifei	
3	 Ben Chan		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 05 min 40 sec	Cheuk	
4	 Binh		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 07 min 28 sec	Xin Yi	
5	 Fu Mei		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 07 min 39 sec	jinzaw	
6	 Bear		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 07 min 41 sec	sara	
7	 iPython		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 08 min 18 sec		
8	 Zoey		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 06 min 15 sec		
9	 Wei Song		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 05 min 10 sec		
10	 sara		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 04 min 48 sec		Bear
11	 jinzaw		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 06 min 07 sec		Fu Mei
12	 Jifei		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 08 min 20 sec		Melvin
13	 Cheuk		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 08 min 52 sec		Ben Chan
14	 Mrchamp		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 09 min 00 sec		Erwin
15	 Xin Yi		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 05 min 31 sec		Binh

Figure 2: JavaScript tournament ranking with mentor assignment shown

This approach leverages many aspects of the Keller Plan[5]. In the Keller plan, students that had mastered material were drafted as mentors to assist with evaluating and assisting other students. Often, these students were teaching assistants that had mastered the material in previous terms. The automated mentor assignment feature in software tournaments identifies and drafts mentors on the fly based on their mastery of specific material for a specific class. As long as a student is the first person to be able to solve all of the problems out of a class of students, they are considered as qualified to mentor their least capable peer. And since the first to finish are always assigned to assist the slowest progressing students, the most capable are assigned to assist the least capable.

3. Study Methodology

This process was evaluated in early December 2013 at two short programming workshop courses – one to teach JavaScript the other for Python. Each workshop took three hours and was attended by 15 second-year university students with prior experience programming in Java. The majority of the students attending the workshops had no prior experience with the software language being. Approximately four days prior to the workshop, students were sent

an email notifying them that these workshops would be blended learning courses[8] and that they were encouraged, but not required, to try coding in either JavaScript or Python on their own, online prior to attending the. The students were directed towards Codecademy[6] and SingPath[7] as potential online resources to explore the basics of each language.

Each classroom session started with a ten-problem software tournament consisting of simple problems involving functions. After this first tournament was completed, the students were informed that there would be a second tournament at the end of the session where the winner would receive a small prize. The automated mentoring assignment process was used in both the starting and ending tournaments.

4. Results

All four of the tournaments – two for each workshop – proceeded as expected. The tournaments took between twenty-five and forty minutes to complete. There were no significant differences between the tournaments held in the JavaScript workshop and the tournaments held in the Python workshop. Each course took place in a forty-seat seminar room and was attended by approximately fifteen students. The two-to-one seat-to-student ratio considerably improved the logistics of mentoring. There was no difficulty in students and mentors sitting together or finding appropriate space. Compared to previous ad-hoc arrangements this saved consider time.































#	Player	Country	Type	1	2	3	4	5	6	7	8	9	10	Last solve time	Now Mentoring	Mentored By
1	 Erwin		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 04 min 35 sec	Mrchamp	
2	 Melvin		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 04 min 50 sec	Jifei	
3	 Ben Chan		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 05 min 40 sec	Cheuk	
4	 Binh		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 07 min 28 sec	Xin Yi	
5	 Fu Mei		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 07 min 39 sec	jinzaw	
6	 Bear		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 07 min 41 sec	sara	
7	 iPython		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 09 min 26 sec	Wei Song	
8	 Zoey		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 10 min 13 sec	Ronald	
9	 Wei Song		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 10 min 58 sec		iPython
10	 sara		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 11 min 22 sec		Bear
11	 Jifei		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 13 min 54 sec		Melvin
12	 Cheuk		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 14 min 12 sec		Ben Chan
13	 jinzaw		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 15 min 58 sec		Fu Mei
14	 Xin Yi		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 16 min 03 sec		Binh
15	 Mrchamp		Student	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	0 hour 17 min 15 sec		Erwin

Figure 3: JavaScript tournament ranking at end of tournament

Explaining the process upfront as well as indicating to mentees who would come help them considerably smoothed over social frictions. Both mentor and mentee knew what they should be doing and were prepared for the process. We observed that mentors stayed engaged with the class and mentee's frustration was reduced. By automating assignment, mentors did not

stick to their prior friend groups. Our impression was that this helped form a more cohesive class cohort and gave the tournaments more of a 'team-sport' feel rather than individual work. Figure 3 shows the final assignment of mentors and completion times.

The workshops benefited from the small size as many of the students already knew each other. For larger and more diverse groups of students, it would be necessary to provide name cards for students to make it easier for mentors to find mentees. For much larger groups the logistics of moving around may require that the assignment be restricted to those nearby – we have plans to work on such a feature.

4. Conclusions

The automated mentor assignment feature provides a way to shorten the time needed for tournaments since the earliest finishing students serve as mentors to help the slowest progressing students. This also allows the instructor to assist a larger number of students rather than helping just one or two students who are struggling the most. The mentor assignment also keeps the first finishers engaged as they practice reading other students' code and offering advice. At the same time, less capable students are provided with a personal mentor to assist them with any parts of the material that they may be struggling with. This appears to reduce frustration and speed completion. The ability to provide opportunities for self-directed learning, opportunities to mentor, and opportunities to be individually mentored to students enables courses to provide a more personalized experience. Our observation indicates that this increases student engagement and allows more flexibility in allocating instructor time. This process also leads to a more social class experience since students do not assist a wider portion of the cohort than their immediate friends. Since mentors are assigned across the class, it gives students an opportunity to interact with other classmates with different levels of capability. The process also helps to reinforce the idea of working through labs and completing tournaments as a full class exercise rather than an individual exercise since the goal is for the entire class to finish and the class cannot move on until everyone has solved the selected problems.

Acknowledgment

The authors thank Singapore Management University for supporting this research.

References

- [1] Bloom, B. (1984). "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring", *Educational Researcher*, 13:6(4-16).
- [2] Boesch, C., & Steppe, K. Case Study on Using a Programming Practice Tool for Evaluating University Applicants. 3rd Annual International Conference on Computer Science Education: Innovation and Technology. 2011.
- [3] Boesch, C., & Boesch, S. (2012). Tournament-based Teaching. 4th Annual International Conference on Computer Science Education: Innovation and Technology. 2012.
- [4] Boesch, C., & Boesch, S. (2013). Adaptive Gameplay for Programming Practice. 5th Annual International Conference on Computer Science Education: Innovation and Technology (CSEIT 2013).
- [5] Keller, F. S. (1968). Goodbye teacher... *Journal of Applied Behavior Analysis* 1, 79-89.
- [6] <http://codecademy.com>
- [7] <http://singpath.com>