11-2012

Tournament-based Teaching

Shannon Christopher BOESCH
Singapore Management University, cboesch@smu.edu.sg

Sandra BOESCH
Pivotal Expert Pte Ltd, sandracboesch@gmail.com

Citation

Boesch, Chris and Sandra Boesch. 2012. "Tournament-based Teaching." Paper presented at the 3rd Annual Conference on Computer Science Education: Innovation and Technology, Bangkok, October 28-29. doi:10.5176/2251-2195_CSEIT12.35.

Tournament-based Teaching

Chris Boesch & Sandra Boesch
School of Information Systems (SIS) Singapore
Management University (SMU) Singapore
and Pivotal Expert, Pte. Ltd. Singapore

Abstract— **Over the past two years we have collaborated to develop a process and set of online games to enable additional feedback to both students and instructors in a classroom setting. We have named the resulting process Tournament-based Teaching due to the extensive use of tournament-based feedback for groups and individuals throughout course delivery. Tournament-based Teaching enables individualized and peer-based learning in a classroom setting and provides additional motivation for students to prepare for classroom sessions. It also provides feedback to instructors, which can be leveraged to provide better schedule classroom sessions.**

Keywords-education, programming, team-based learning

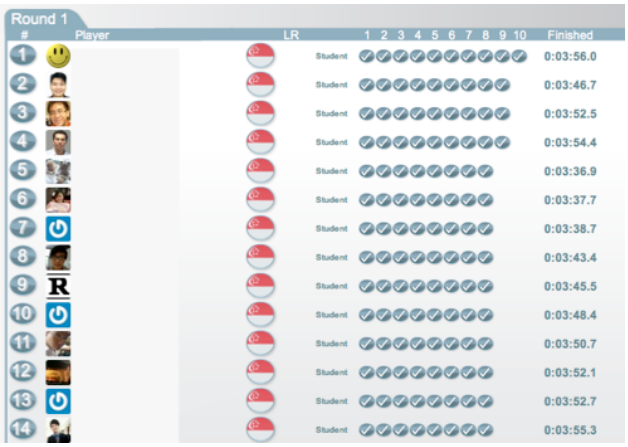
I. INTRODUCTION

When setting out to develop a more effective method to teach basic computer science, the authors were looking for innovative ways to provide additional, individualized feedback to students learning software languages such as Python, JavaScript, and Java for undergrad university courses. The authors took the approach to enable students to practice software languages on their own by having them solve short programming problems in an online system (see Figure 1) in a variety of software languages.

Figure1. SingPath Play Screen

Students were able to practice solving these problems on their own time, from their own systems, wherever they had Internet access. This method enabled the authors to provide additional feedback to students in a more real-time manner than had been previously possible with live, in-class quizzes and weekly problem sets turned in as homework. Students were still assigned problems to solve as in previous terms, but by requiring students to solve all problems in an online system, the authors were able to provide students with real-time feedback on their progress and at the same automatically track which students were on pace to solve all required problem prior to weekly deadlines. Once the online system had been developed to verify problem solutions and provide real-time feedback, it quickly became apparent that the same system could be used for in-class practice sessions as well as out-of-class practice sessions. To support a more interactive group experience, the online system was extended to support software tournaments. The in-class tournament experience for students was designed to be exactly the same as solving a set of problems at home with the addition that the class instructor was able to project every student's progress on a large screen at the front of the class.

Figure 2. Tournament ranking screen



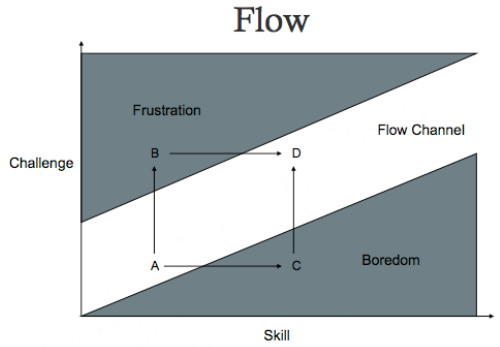
This process provided the class and the instructor with more real-time feedback on how the class was progressing overall as well as providing visibility to better understand which students were demonstrating a higher degree of proficiency. This additional feedback provided the instructor with more information to better allocate time and to identify when the first students where finished with all of the problems. This same information was also instantly available to the students participating in the tournament. After holding

approximately twenty tournaments to start weekly technical classes, a repeatable process began to emerge which enabled the classroom instructor to manage the time allocated for weekly tournaments, provide personalized feedback for each student, and enable peer-based learning. This overall process has come to be referred to as Tournament-based Teaching.

Tournament-based teaching begins with students being asked to solve some number of problems prior to coming to class. The authors most often assign problems using SingPath. SingPath.com is a free, online game that can be played by anyone anywhere in the world. SingPath has been developed and refined over the years by the authors to help individuals practice multiple software languages and to help educators provide additional practice opportunities for their students. SingPath enables educators to create challenges that students must unlock prior to coming to a particular classroom session. Students can be required to solve all of the assigned problems on their own or be permitted to work in teams to unlock challenges. Either way, each student is still required to type a solution for each problem into SingPath while logged in to his or her own account. After unlocking their assigned challenges, students have the opportunity to continue practicing as much or as little as they desire before the next class session.

Each class session begins with a short, twenty to thirty-minute tournament consisting of problems selected by the class instructor. The instructor can adjust the number of problems in each weekly tournament based on the complexity of the material or the current proficiency of the students. These tournaments serve many of the same goals as the formative assessments used to start classes in Team Lead [1] and Team-based Learning [2] classroom sessions. In order to manage class time and enable team-based learning, the following tournament process has been refined over the past few years. When the students first begin a tournament, all problems must be solved without access to any external information from other websites. This is intended to encourage students to practice solving problems without always looking at a language reference or example problems. The idea is that memorizing the basic syntax of a programming language will increase the likelihood of students getting into a state of flow (See Figure 3) when programming and potentially make the process of programming a little more fun [4].

Figure 3. State of flow



Starting tournaments in a closed-book fashion also provides an advantage to the most prepared students and a slightly greater challenge for the most advanced students. In the authors' experience, there is usually at least one student that comes to class prepared to solve all the problems in the tournament very quickly. The most prepared student often completes the entire tournament in less than five minutes. This five-minute mark, or the point at which the first student finishes the tournament, has turned out to be a good point at which to allow the remaining students to access the Internet to view examples or references for the programming language syntax. The initial closed-book phase also provides students with an opportunity to assess how well they are prepared to solve problems in the given programming language. There are often a few tournament participants that realize how much they have been relying on their notes. Usually after a frustrating closed-book session one week, the same student will come to class better prepared for the tournament the following week. The duration of the closed-book phase of the tournament should be timed to provide students with the opportunity to reflect on their preparation and understanding of the programming language yet this phase needs to be short enough to avoid overly frustrating the students that may simply not recall a particular detail of the programming language syntax. To alleviate frustration, it is possible to skip problems during the tournament and then return to the problems after the closed-book phase of the tournament has finished. Seeing a fellow classmate complete the entire tournament, or most of it, during the closed-book phase of the tournament demonstrates to the class what is possible by the most prepared of their peers.

Once the closed-book phase of the tournament has finished, the instructor leading the class announces to the class that they may access the Internet for assistance. This phase of the tournament is referred to as the open-book phase. Students are permitted to access any online materials just as they would if they were working on a real-world software project. The only restriction is that students are not permitted to access any personal notes or prepared code snippets that they may have previously created. Reviewing prior work is considered an unfair advantage and as cheating. As might be expected, the open-book phase of the tournament tends to go more quickly than the closed-book phase of the tournament for most students since students are able to reference details about the software language syntax and more quickly debug their code. Later during course terms, we have observed that students access external materials less frequently as logic errors rather than syntax errors begin to consume the majority of their debugging time. During the open-book phase of the tournament, additional students finish solving all of the problems and are expected to sit quietly while other students finish, or they are allowed to begin practicing for the next upcoming tournament. Students are not allowed to talk to their peers during either the closed-book or open-book phases of the tournament. This guideline has proven to be frustrating at times for students that have finished and must watch helplessly as peers to their right or left struggle with a logic error for which the finished student sees the solution and is unable to help. The closed-book phase of the tournament usually lasts until the first ten students have finished the material or until about ten minutes of the allotted class time for the tournament is remaining. These guidelines have been developed over time to balance the opportunity for students to work on their own while also providing an

opportunity for peer-based learning during the last tournament phase. The authors prefer to announce the first ten students' names out loud as they finish in order to acknowledge their accomplishment of finishing in the top ten as well as to update the class on how many top ten slots are remaining and how much longer the open-book phase of the tournament is likely to last. Continuing the open-book phase of the tournament until ten students have finished has proven to be motivational for many students and entertaining for the students that have already finished.

Once approximately ten students have finished, the open-book individual phase of the tournament is completed and the students who have finished are asked to find a fellow student to assist. This is where the peer-based learning phase of the tournament begins. Stronger students are provided with an opportunity to practice coaching less proficient students and the less proficient students are provided with a personal coach to work with them as they solve their remaining problems that they were unable to solve on their own during the open-book time provided. Once the peer-based phase of the tournament begins, students begin to complete the tournament problems more quickly. As soon as half of the class has finished all of the problems, every student that has not completed the tournament should have a peer coach that has completed the tournament. During the peer coaching time, the student coaches can help their peers to recall features of the software language that can be used to solve problems or help them to identify logic or syntax issues with the still working student's current solution. The authors have also observed that even the weakest students are motivated to have a few problems solved on their own before their peers arrive to help them during the peer-based learning phase of the tournament. Few students are content to have less than half of their tournament problems solved at the point in time that ten of their classmates have completed the tournament. This knowledge of the coming peer-based learning phase of each tournament helps to motivate some students to come more prepared for class and at the same time ensures others during the tournament that personalized help will soon be on the way [5].

II. Findings & Results

Although a stressful way to start class for some, students have provided mostly positive feedback on the Tournament-based Teaching approach. Tournaments are not graded but are rather part of the participation grades for courses. Students are given complete freedom to prepare for the tournaments as much or as little as needed. At a minimum, the tournaments provide students with additional feedback on how well they understand a software language, how well they are able to solve problems in the software language, and how their proficiency compares with their peers. This supplementary feedback enables the students to do additional self-guided learning outside of class and make better-informed decisions on how much more or less they should be practicing their programming skills. The increased feedback for the class instructors has proven highly valuable as well. After just a few weekly tournaments, instructors can begin to see trends of which students are most proficient at new material and which students are likely to struggle during the course. The tournament data also enables the instructor to better pace class sessions overall by better understanding which students are most likely to get bored by a slower class pace and which are most likely to become frustrated by a faster introduction of new and more complex material. The instructor can also observe which problems in a tournament students were most likely to struggle with and which were the easiest for students to solve. This information can be used to better design review sessions in future classes. Overall, the authors have found the use of Tournament-based Teaching a valuable tool to continually assess different class sections while scalably providing more personalized feedback for individual students. The process has also been leveraged as part of the Singapore Management University School of Information Systems Second Chance Admissions Tournament process [3].

III. Discussion & Conclusions

We are continuing to experiment with ways to balance the potential frustration and boredom of students learning and practicing software languages. We hope to find innovative ways to move a larger percentage of class participants into a state of "flow" – a psychological state correlated with motivation and future ability [4]. We are also planning to introduce more team-based games where pre-defined teams of students compete as teams rather than as individuals. Our hope is that this will lead to even more effective peer-based coaching and learning.

Acknowledgments

We are grateful to the Singapore Management University School of Information Systems for enabling us to conduct tournaments in live classroom settings. We are also grateful to the staff of Pivotal Expert for maintaining and enhancing the SingPath platform and for making it free to students and faculty around the world.

References

[1] Kamei, Robert; Cook, Sandy; Puthrucheary, Janil; Starmer, Frank. *21st Century Learning in Medicine: Traditional Teaching versus Team-based Learning* in *Medical Science Educator.* Voluem 22. Issue 2. 2012.

[2] Michaelsen LK, Knight AB, Fink LD. Team-Based Learning, A Transformative Use of Small Groups in College Teaching. 1st edition. Sterling, Virginia: Stylus Publishing, LLC; 2002.

[3] Shernoff, D.J.C., Mihaly; Shneider, Barbara; Shernoff, Elisa Steele *Student engagement in high school classrooms from the perspective of flow theory.* School Psychology Quarterly, 2003. 18(2): p. 158-176.

[4] Csikszentmihalyi, M. *Finding flow: The psychology of engagement with everyday life.* 1st edition. Basic Books: New York, NY;1997.

[5] Stump, G.S., Hilperta, J. C., Husman, J., Chung, W.T., Kim, W. *Collaborative Learning in Engineering Students: Gender and Achievement.* Journal of Engineering Education. July 2011, 100(3), p. 475–497.