12-2011

Case Study on Using a Programming Practice Tool for Evaluating University Applicants

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

Kevin STEPPE
Singapore Management University, kevinsteppe@smu.edu.sg

Citation

Boesch, Chris and Kevin Steppe. 2011. "Case Study on Using a Programming Practice Tool for Evaluating University Applicants." Conference Proceedings of 2nd International Conference on Computer Science Education: Innovation and Technology, Singapore, December 5-6. doi:10.5176/2251-2195_CSEIT37

Case Study on Using a Programming Practice Tool for Evaluating University Applicants

Chris Boesch & Kevin Steppe
School of Information Systems (SIS)
Singapore Management University (SMU)
Singapore

Abstract—**We used a programming practice tool to test basic programming skills of prospective students. A live competition was used to test those skills. Students who did well were asked for further interviews. Most students had no prior background and reported learning the basics of two programming languages within two weeks of self-study.**

Keywords-education, programming, programming aptitude,

I. INTRODUCTION

Teachers of programming have long noted that many students have difficulty in learning programming. It has even been reported that many computer science students never really learn to program[1]. For schools and departments in which programming has a significant place in the curriculum and which have the luxury of selecting students from many applicants, a method to identify early those who are likely to succeed is desirable.

Many tests for programming aptitude without requiring actual programming have been proposed. These started with the IBM Programmer Aptitude Test (PAT) and have been followed up with Wolfe's tests. More recent attempts come from Winrow's PAAT/I-PAT [2] and Tukiainen's test [3]. Despite the early popularity of the PAT, its actual prediction of performance was often poor [4]. While Tukiainen's test first appears to have statistical correlation with exam scores, after selecting only those with no prior programming experience, the test shows no predictive power. More recently, Bornat [5] attempted to create an aptitude test based on the 'mental models' work of Johnson-Laird [6]. This work is particularly interesting first because it attempts to determine if the students' mental models match the models of simple program operations, and secondly because it focuses on areas Bornat had observed to be significant hurdles for first-time learners. Despite initially promising results, Bornat too did not find consistent predictive power.

Further, in much of the world, pre-tertiary instruction in programming continues to be the exception rather than the rule, even in science-oriented curriculums. In Singapore for example, despite thousands of students taking a heavily science-dominated secondary curriculum, only about two hundred take any programming instruction. At Singapore Management University we do not require prior programming experience for admissions to the School of Information Systems program. The admissions process goes to great lengths not to bias the selection process in favor of those with prior programming experience. Our first round of admissions is based on prior academic results and problem-solving skills demonstrated during a live interview. While the concept of programming aptitude has been discussed as a desirable criterion, the lack of effective methods for determining aptitude has prevented inclusion of this criterion.

Our experience with both first-year programming courses and advanced courses which require programming are similar to those reported elsewhere. Like Bornat we observe a bimodal distribution of performance early programming courses. Roughly speaking, students with stronger academic backgrounds show up at the top of the results more often. However there are enough exceptions, both in unexpectedly good and unexpectedly bad performances, to resist making any strong claims. In fact, in a case studies of seven universities from seven countries, Alexander et. al. report that none of the studies produced convincing evidence that high school achievements identify programming success[7].

In 2011, we decided to pilot an attempt to identify applicants who were not accepted after the usual interview, but may have had significant technical aptitude which was not evaluated during the interview process and was not obvious from their previous academic results. Approximately one hundred and eighty students that had interviewed with the university but had not been accepted were invited to work with an online programming practice tool called SingPath in order to qualify for a live competition and a chance at a second interview.

This method differs from the previously mentioned aptitude tests by requiring actual programming as a criterion. Because we were looking at students after the main admissions process, we were willing to evaluate early actual programming performance rather than aptitude. Of course success in university requires much more than programming skill, so we only invited those who had met the standards for prior academic performance. Therefore we had reason to believe they would do well in areas such as math and economics where they have prior background and results. With the additional process we hoped to find a few applicants who also had demonstrable programming skills and be more likely to pass first year programming courses.

We originally expected a large percentage of students who signed up for the competition would have had prior programming knowledge. However, we found that the majority of those who signed up and passed the practice portion to qualify for the live tournament had no programming experience prior to the competition. Eight of the top ten in the live tournament had taught themselves programming in the two weeks preceding the tournament.

The rest of the paper is organized as follows: part II) the competition process; part III) the SingPath tool; part IV) results and findings; part V) discussion and conclusions

II. The Competition Process

All students not accepted for admissions to the School of Information Systems at Singapore Management University are informed by letter and email. In 2011 at the end of the letter an explanation of the second chance admissions tournament and interview were included. Each letter contained a link to the online practice tool SingPath where applicants were invited to register for the challenge. This challenge required applicants to solve sixty small programming problems in order to be invited to a live programming tournament. All applicants were free to use the tool before registering for the tournament.

On the day that the notification letters were delivered to applicants, a few applicants registered for the admissions challenge and began to solve simple problems. By the end of the first week following the notification letter, thirty-four students had registered for the challenge and were solving problems online. Students were given two weeks to complete the problems to get an invitation to the live competition.

The admission challenge that was constructed required students to solve thirty problems using the Python programming language and thirty problems using the Java programming language. These thirty problems covered the basics of each language syntax and required the applicants to learn how to write basic programming functions in each language. The idea behind this approach was that students who were able to learn to program in more than one language in a short period of time would be capable of learning and applying other new technical subjects in the future. Most of the students chose to start solving Python problems first and then moved on to solving Java problems.

After the two week 'practice' period was over, twelve students had solved at least the minimum sixty problems and had submitted all of the requested information. These twelve qualifying students were then notified of the time and place for the live admissions tournament which would be held on the school campus.

On the day of the tournament, all twelve students that had been invited showed up for the tournament. Applicants brought their own notebook computer to use in the tournament. The tournament consisted of one round of ten Python problems and one round of ten Java problems. Because the number of applicants at the tournament was small, our focus was to determine whether the applicant could repeat their performance from the practice phrase, and thus filter out anyone who had asked a friend to complete the initial phase for them. Therefore nine of the ten problems in each round to come from the problems that students should have already solved. We included one new problem, that was more challenging, to provide differentiation among the applicants.

At the live tournament all the students logged into SingPath.com and the rules of the tournament were explained. The tournament itself was conducted as an open book test. Applicants were allowed to access the Internet to research solutions to their problems. The only restriction was that applicants were not allowed to use any personal notes that they may have created when solving the problems at home. Two faculty members observed to ensure that applicants were not accessing restricted material or communicating with anyone online. No inappropriate behavior was observed.

Each round lasted for approximately forty-five minutes. In the Python round, the first applicant to solve all ten problems completed in approximately thirteen and a half minutes. It took thirty-nine minutes for the eighth player to complete all ten problems before the round was halted. Four players were unable to complete the Python round in the allotted forty-five minutes. In the second round, the first applicant solved all ten Java problems in approximately nine and a half minutes. The sixth player in the second round then solved all ten problems in approximately thirty-five minutes. In the Java round six applicants were unable to solve all ten problem within the forty-five minute time limit.

Two of the twelve applicants ranked at the bottom of the group in both rounds and were dismissed. The remaining ten students were interviewed immediately after the tournament in order of their ranking. The interview was used to find out more about the applicants' technical background, interests, and ability to communicate. Singapore Management University places an emphasis on communication and presentation skills so it was necessary to ensure we did not recommend admission for anyone who did not meet our standards in those areas. Of the ten students interviewed after the tournament, eight were recommended for admission and informed of their admissions the following week.

III. SingPath

SingPath is a free online programming practice tool structured as a game, which enables players (users) to practice software development by solving short problems. Most players solve the easier problems in one or two attempts within a minute. More difficult problems will take several attempts and several minutes. SingPath offers problems in programming languages such as Java, Python, Ruby, Javascript, and Objective-C. The problems are arranged in levels which focus on specific topics such as how to create functions or how to work with strings in a particular language. Players progress by solving enough problems in a level to 'unlock' the following level. Levels are organized into paths. Players are expected to unlock levels in a particular order along a path to ensure that they have mastered basic language concepts that will be needed to solve problems in later levels. In addition to the provided problems, players are able to create their own problems for other users to solve.

Problems consist of a problem name, a problem description, examples and public tests. Problems may include private tests in addition to the public tests. Problems can also include starter code which will be provided to players. Public and private tests are simple unit tests which will be familiar to any developers who have worked with unit test frameworks like JUnit. Tests are simple assertions used to determine if the code provided by players meets the criteria specified for solving the problem. Public tests are intended to test most cases for the problem and are used to provide detailed feedback to players when their solutions are not correct. Often players will solve the majority of tests but fail a few corner cases which require extra code or alternate logic. Private tests are used to ensure that players do not overfit their solutions to the provided public tests. For example, players might be tempted to add a simple if-then statement to their solution to make a final failing test case pass rather than modify their solution to properly handle all test cases in a general manner. When players fail private tests, they are only asked to generalize their solution further rather than being provided with detailed test failure information as they are for public tests. The core features of SingPath provide players with clear goals and objectives, timely feedback, and ample time to practice.

SingPath also provides support for live tournaments where players can demonstrate their new skills in a timed, competitive environment. The tournament feature is used in both classroom settings and technology conferences to provide users with an opportunity to demonstrate their skills in a proctored environment. When used for tournaments, SingPath provides a live ranking based on how many problems each participant has solved. SingPath also displays the time at which each participant last solved a problem, a feature which can be useful in deciding whether to extend or shorten the tournament.

IV. Findings & Results

One of the most unexpected findings from the competition was how few applicants with prior programming skills qualified for the tournament. In Singapore, there are several polytechnics that provide diplomas in technical subjects were programming courses are taught. We originally expected that the tournament would provide students from these institutions with an enhanced opportunity for admission. However, eight of the twelve qualifying for the tournament were not from one of these programs and had in fact taught themselves programming in the two weeks they had been given to complete the sixty problems on SingPath. Further, of the eight who completed at least one of the two rounds in the live tournament, only one had taken programming courses previously. Three of the four students in the tournament who had taken programming courses previously failed to finish either round in the time allocated.

Those who had taught themselves to program during the practice phase all reported having spent from twenty to eighty hours on the challenge. Most reported getting some help from friends or family, though all of those stated that they eventually had to learn on their own. Equally interesting, all reported that they found programming to be "frustrating and fun". Thus, while they found the experience challenging they also, overall, found it enjoyable and worth further pursuit. These reports suggest that for these students the process of learning to program put them in "flow" – a psychological state correlated with motivation and future ability[8].

Of the top ten from the tournament, eight were eventually offered admissions to the School of Information Systems. All eight accepted the admissions offer and enrolled in August 2011.

V. Discussion & Conclusions

We originally conceived the challenge and tournament as a way to identify applicants with programming talent not recognized in the usual interview. We feel that the challenge succeeded in this – eight applicants capable of solving basic programming problems in two different languages were identified. This is considerably more programming skill than required for most of our applicants.

In designing the usual interview, we have long sought for a way to identify students who are interested in computing and eager to meet our work load expectations. Both of these are hard to determine from an interview or even from past results. In the usual interview we commonly hear that students are interested in computing because they enjoy using a computer at home – a rational that often ceases during the first programming course. Willingness to work hard is similarly impossible to measure when the applicant is trying to give a favorable impression. Usually the best we can do is to warn them that computing projects can take a lot of work.

The process discussed in this paper seems to highly select for students who in their initial exposure to learning programming are sufficiently interested to put in a lot of work. While it is possible that the students were more interested in admission to university than in programming, the challenge did find the twelve who were interested enough to solve sixty problems, as compared to the other hundred and sixty-eight who were only interested enough to come to the first interview. It is also possible that some of those who did not participate accepted offers from other universities. If it is the case that those who registered for the challenge were primarily students with no other offers, they showed a high level of desire to gain admissions to our university.

The challenge also appears to select students willing to work harder than their peers. The applicants put in a remarkable effort to learn to program. While we cannot guarantee that they will maintain that level of effort, we have more direct evidence of their willingness to work hard than we have for other applicants.

Third, the challenge – especially the practice phase – is highly scalable. With no effort on the part of the faculty or staff, the number of applicants was reduced from one hundred and eighty to twelve. An increase in the number of applicants registering for the challenge would require no additional effort.

We still need to pursue a longitudinal study on the success of the students selected through this process. While our experience with the tournament and interviews suggests that the process selects for characteristics we want, we still need to verify that the students do well in actual courses over the long term. These data will only be available as the students progress through the degree program.

Given the success of the process in our pilot, we are looking to use it to select a larger portion of the next cohort. Despite the desire to expand this process, we still feel that we cannot use it as the principal criterion for the majority of our admissions. We will need to analyze the long term success of these students compared to the usual interview process to find the right balance.

ACKNOWLEDGMENTS

We are grateful to the admissions department at Singapore Management University and the staff in the School of Information Systems for their efforts in ensuring the smooth operation of the tournament and admissions process.

REFERENCES

[1] McCracken, e.a. *A multi-national, multi-institutional study of assessment of programming skills of first-year CS students*. in *Working group reports from ITiCSE on Innovation and tchnology in computer science education*. 2001. Canterbury, UK.

[2] Winrow, B., *The Walden programmer analyst aptitude test.* Dr. Dobb's Journal. Fall (1999).

[3] Tukiainen M, M.E., *Programming aptitude testing as a prediction of learning to program*, in *14th Workshop of the Psychology of Programming Interest Group*. 2002: Brunel University.

[4] Mayer D. B., S.A.W. *Selection and evaluation of computer personnel - the history of SIG/CPR*. in *1968 ACM National Converence*. 1968.

[5] Bornat R, S.D., Simon, *Mental models, Consistency and Programming Aptitude*, in *Tenth Australasian Computing Education Conference*. 2008: Wollongong, Australia.

[6] Johnson-Laird P, B.V. *A model theory of modal reasoning*. in *Nineteenth Annual Conference of the Cognitive Science Society*. 1997.

[7] Alexander S, A.J., Boyle R, Clark M, Daniels M, Laxer C, Loose K, Shinners-Kennedy D. *Case Studies in Admissions to and Early Performance in Computer Science Degrees*. in *Working Group at ITiCSE*. 2003. Thessaloniki.

[8] 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.