Combining Mastery Learning with Project-Based Learning in a First Programming Course: An Experience Report

Mehdi Jazayeri
University of Lugano

***Abstract*—One of the challenges in teaching a first programming course is that in the same course, the students must learn basic programming techniques and high level abstraction abilities, and the application of those techniques and concepts in problem solving and (engineering) design. To confront this challenge, in previous years, we have included a project-based learning phase at the end of our course to encourage the acquisition of high level design and creativity. To address some of the shortcomings of our previous editions, we have recently included a mastery phase to the course. While project-based learning is suitable for teaching high-level skills that require design and creativity and prepare the students for the study of software engineering, mastery-based learning is suitable for concrete skills such as basic programming tasks. Our particular innovation is to allow students into the project phase only if they have demonstrated a minimum predefined competency level in programming. The combination of the two approaches seems to address most of the requirements of a first programming course. We present our motivation for combining the two pedagogical techniques and our experience with the course.**

***Index Terms*—first programming course; mastery learning; project-based learning; CS1/CS2; introductory programming; software engineering education**

I. Introduction and motivation

The curriculum of informatics at the University of Lugano is influenced heavily by software engineering ideas [1]. Indeed, the admittedly ambitious goal of our curriculum is to produce software designers in only three years. The curriculum has four major ingredients: Theory, Technology, System Thinking, and Communication and Teamwork. The system thinking part of the curriculum emphasizes that the students must not only learn how to build systems but also analyze why they are building a system and how the system will fit in, and influence, the environment in which it is deployed. One of the ways that we emphasize both teamwork and system thinking is to engage students in projects from the very first semester. The students work on their first group project in the latter half of their first programming course. So, in addition to the usual challenges in teaching the introductory course, we also face the challenges of including a group project in the course. The first programming course is thus a foundation for the rest of our software engineering curriculum.

Introductory programming courses are challenging to design and teach. Students come in with different computing backgrounds and abilities. From the course, the students must learn a varied collection of skills and concepts. They must not only learn the syntax and semantics of a particular programming language, but also the mechanics of program creation (editing) and debugging, and abstract concepts of problem solving, modeling and design. In the same course, the students must learn basic programming techniques and high level abstraction abilities and the application of those techniques and concepts in problem solving and (engineering) design. The traditional lecture style of instruction is not particularly suited to the requirements of such a course. In previous years, we have included a project phase at the end of our course to encourage the acquisition of skills that promote the learning of of high level design and creativity. To address some of the shortcomings of our previous editions, we have recently included a mastery phase to the course. Mastery learning is a pedagogical technique that is suitable for concrete skills such as basic programming tasks, while project-based learning is suitable for high-level skills that require design and creativity and prepare the students for the study of software engineering.

Since 2004, we have been teaching our first programming course at the University of Lugano with the goal of educating students not only in programming but also design and team work [2]. The original version of the course used Scheme as the programming language. We also have experience with a Java course [3] in Lugano. The current version of the course uses Python, which enables the combination of different styles of programming. Until this year, the course started with a phase that emphasized individual programming skills. In the second phase, the students took on a larger assignment which we called a personal project. In this phase they individually chose a topic, and designed and implemented an application around that topic. In the final phase of the course, the students formed teams of 3 or 4 and worked on a team project. While we had many positive experiences and the best students achieved a great deal in the course, we had persistent problems in the third (team project) phase. We experimented with different ways of team formation: a team with mixed abilities, teams selected by instructors, teams self-formed, and so on. But none of these techniques proved completely satisfactory. The learning by students was highly non-uniform. The strong students provided motivation to the weaker students and intimidated them at the same time. The weaker students tended to lose self-confidence and some floundered in the project phase.

Last year, we hypothesized that while projects are in general

a good motivating device that challenge the strong students, they do not serve the needs of the weaker students as they are not prepared to make use of the learning opportunity. We realized that we needed to separate the students into a group that can benefit from doing a project and those who are not prepared to do so. In last year's midterm quiz of the course, the lowest score, on a scale of 0 - 100, was 2 and the highest score was 98. It is clear that these two students have different needs to help them advance in their learning. This is an instance of the 2-Sigma problem in teaching [4]. Our solution was to use mastery learning [5] in the first phase of the course. Mastery learning is ideally suited to teaching basic programming skills. Only if the students demonstrate competency on those skills, are they admitted to the second, project phase, which emphasizes high level design skills.

We implemented an approximate version of mastery learning in 2013-14 and a more systematic version in 2014-15. We report mostly on the experience with the version of 2014-15 here. We believe that the experience is timely and relevant to other software engineering educators.

II. Course structure

We structured the course in two phases: a programming mastery phase, and a project design phase. In the first phase, the students are expected to learn basic programming skills and demonstrate their ability through specific activities. Only if they are able to pass this phase will they be admitted to the second phase.

A. Programming skills mastery phase

Mastery learning [5] is a pedagogical technique adapted to concrete skills and topics that can be stratified in levels of difficulty. In such skills, it is important to learn one skill before moving on to the next, more difficult skill. For example, it is important to learn addition completely before attempting to learn multiplication. And such skills are relatively easy to demonstrate. Thus, it is possible to set up the learning environment in such a way that the student works on successive skills and moves to a new one after they have demonstrated competence in the previous skill. This technique seems well-suited to beginning programming. For example, the sequence of variables, types, assignment statement, if-statement, loops, functions, can define a logical sequence of learning skills for introductory programming. Indeed, most programming books follow such a sequence, with minor variations. It is difficult to try to learn if-statements before mastering boolean variables and assignment statements, although many students may try to learn if-statements after they have had only a partial understanding of the boolean type. Once they have received a 50% score on types and assignment statements, they move on to do if-statements. According to the mastery learning method, they must achieve 100% competency on types and assignment statements before moving on to if-statements.

One major difficulty with implementing mastery learning in traditional academic calendars is that lectures are fixed, with topics being introduced at the pace the professor has decided, rather than the (individual) pace of students. In our case, we compromised on a middle ground. We defined a set of skills that the students had to master. We had lectures on these skills. And we outlined a set of skill check lists that the students could use to demonstrate their mastery of the skills. Ideally, the students would go at their own pace. In practice we held a mastery check meeting each week in which the students demonstrated the skills they had mastered. At the end of the mastery phase (week 10 of the semester), the students who had completed all the skills, moved on to the project phase. Those who had not completed the skill set continued working on the skills. The check list for each skill consisted of a set of preliminary questions, a program to modify, and a program to write. The mastery check meeting required each student to demonstrate their work with a teaching assistant. That is, rather than just submitting an assignment, they had to present their solution and engage in a discussion about its structure and behavior. Once they demonstrated that they understood the solution, they were given a pass and asked to move on to the next skill.

B. Creativity and design project phase

Those students who completed the mastery phase on time were judged to have sufficient programming skills to be able to benefit from engaging in a group project activity. Project-based learning [6], [7] is an increasingly popular pedagogical technique with a long history in software engineering. In this approach, the students are usually grouped in teams and given an open-ended problem of their choosing to explore and solve. Software projects offer many opportunities for such learning. Having guaranteed a minimum skill level of all the students engaged in the project phase, the students could count on reliable team members. The advantage of working in teams is motivation and peer learning. There are two main reasons why project-based learning is ideally suited to programming courses. The first is that it promotes the idea of learning by doing. Programming cannot be learned simply by studying from a book. The learners must get their hands dirty. By exploring various programming ideas, the students enhance their programming skills. The second factor that project based learning is suitable for programming is that it addresses the wide range of abilities and interests of the students. In choosing the projects, the students are guided by their interests. As they develop the project, they can decide which aspects to explore further. Some students may be interested in graphical aspects while others are interested in communication and security. A team of three students is usually capable of producing a significantly more sophisticated application than a single student, thus increasing the students' sense of accomplishment.

III. Combining mastery and projects

To address the difficulties we had faced in previous editions of the course, last year (2013-14) we combined mastery learning and project based learning. The way we ran that course could be considered a prototype version in that the

TABLE I
MASTERY STEPS OF PYTHON, 2014-15.

Skill	Learning objective	Test
Install and run Python	Python as calculator	Mathematical computations
Names and values	Variables and statements	Defining and solving mathematical equations
Function abstraction (no parameters)	Turtle graphics	Drawing shapes: square, circle
Repetition 1	For loop	Producing patterns
Fruitful functions	return statement; function input parameters	Approximation algorithms; Archimedes algorithm for estimating Pi
Selection	if-statements	Monte Carlo simulation
Text processing	Strings	Cyphers and Codes
Collections	Lists and List processing; text files	recursive function; statistics
Repetition 2	While loops	Processing text files; Stock market charting
Associative data	Dictionaries	Histogram; memoization
Preventing and handling error	Exceptions; assert statement	Processing files and user input
Functional programming	Lambda, map, filter, reduce; list comprehension	map-reduce
Objects	Classes and methods	Graphical User Interface Objects

mastery phase was based on traditional assignments. Using that experience, in 2014-15 we defined and ran the course with the mastery program applied explicitly. The skill list is shown in Table I. The table shows (first column) the high level concept, (second column) the Python instantiation of it, and (third column) the main program used to test the student's mastery. The test shown in the third column is both a specific example of a program used to test the student and also indicative of the kind of things that the student is supposed to know how to do. Indeed, we give the students lots of questions that they can use to test their own knowledge. We give them a set of elementary questions about the language feature, a program that they are asked to modify for a certain goal, and a program to write to achieve a certain function. During the mastery test, they have to explain orally how they have achieved the three levels of mastery.

It is clear that this kind of mastery test requires a reasonable level of student/teacher ratio. In our case we have four teaching staff for about 40 students. The technique is not immediately scalable for much larger classes. But as Bloom [4] has shown, the mastery approach is a compromise between larger lecture classes and one-on-one tutoring. Both in terms of teaching effort and learning outcomes, mastery learning falls in between one-on-one tutoring and lecture-based teaching.

The students who did not complete the mastery phase are not admitted to the project phase. They still can pass the course by working independently to master the basic skills. We encouraged them to learn on their own by using resources available on the Web. Indeed, a few students found YouTube videos more useful than our own lectures! The skill list provided concrete and clear self-learning goals for these students. In this way, the mastery group gets a second chance to learn the basic material. The project groups, on the other hand, are more equal in terms of abilities and do motivate each other. The strong groups are freed to achieve at their maximum level of ability, unconstrained by fixed assignments. A possible disadvantage of excluding some students from the project phase is that students finishing the course will have different experiences. One set has learned only the basic programming skills while the other has also honed their project skills. But this is inevitable even in traditional courses.

IV. RESULTS

We started the class with 45 students. Of these, 29 completed the mastery phase successfully and were eligible to do the project phase. Of these 29, 27 decided to do the project phase and the other two decided to accept the minimum passing grade for the course. The 18 that were not allowed to do the project continued working on their skill mastery. Of these, 6 were able to successfully master the skills and obtain a passing grade for the course. The other 12 failed the course. In the past, we would have allowed these 18 students to go on to the project phase putting unreasonable burden on them and their team-mates. We believe that the 6 that were able to work on their skills benefited from not doing a project, and the 27 that did the projects benefited from having competent colleagues.

While we want to divide the students into two groups of competent and not, in practice there are three groups of students. The really strong group and the really weak group at the two extremes and a group in the middle. Looking more closely at this middle group, we can further divide it into two groups. The upper group has the possibility to advance to the top group while the lower group has to work hard to avoid falling into the bottom group. In our case, we gave this middle group a second chance to gain mastery and some were able to take advantage of that chance.

The project groups were divided into seven groups. Six of the groups completed their projects. All six projects were selected to make presentations at a faculty-wide event. These six projects serve to showcase the remarkable ability of students when left to explore their own interests, motivated by friends and peers.

V. RELATED WORK

Both mastery learning (sometimes called competence-based learning [8]) and and project-based learning are well-known pedagogical methods applied in many disciplines and studied

by education researchers. Mastery learning was defined and studied by Benjamin Bloom and his students [5] in particular to address the 2-sigma problem [4]. In comparing traditional classroom instruction against one-on-one tutoring, Bloom observed that under tutoring, the final achievement of the average student was two standard deviations above the performance of the average student in the traditional classroom learning. With mastery learning, the average student was able to achieve one standard deviation above the traditional classroom student. Mastery learning is particularly suited to well-defined learning tasks and has been applied with success to many pre-college classes. There has been surprisingly little application to programming, although the approach seems well-suited to the task. We have only found one brief report on the topic [9].

"Project-based learning is a comprehensive approach to classroom teaching and learning that is designed to engage students in investigation of authentic problems." [10] The motivations for project-based learning are that it engages the students in the learning by working on an open-ended project, discovering problems and finding solutions as they go along. Projects play a major role in software engineering courses but they are not used exactly in the way project-based learning [10] is supposed to be applied from a pedagogical point of view. Project courses in software engineering are used primarily to give students a chance to experience practical problems that they don't encounter in their theoretical learning. The project is not typically open-ended with the student in charge of the learning. Projects typically have milestones and the students are expected to follow a very specific path to the solution. Nevertheless, project based learning has been used also in computer science, and especially in software engineering [7]. In any case, projects are rarely used in first-semester programming courses because the common wisdom says that the students are not mature enough at this point to work on projects, especially team projects. We have found, on the contrary, that students are motivated by projects and look forward to the project phase. Once the project starts, they in fact put too much time in the project, at the expense of their other courses.

VI. Conclusions and future plans

Combining mastery and project-based learning addresses many challenges faced in introductory programming courses. Our combined approach helped the strong students more than the weak students. The strong students get to do more than they would have done in the usual course because they reach the project phase and work with similarly prepared students. Even the weaker students who reach the project phase appreciate the opportunity of working on projects. The weaker students who do not reach the project phase get a second chance to work on their basic skills.

Among the lessons we have learned in this experience are:

1) Mastery based techniques are applicable to basic programming skills and may be used to qualify students for entry into team projects. It is surprising that they are not used more widely in programming courses. We have used mastery learning in combination with project based learning. Our experience shows clearly that mastery learning may be used by itself in first programming courses to benefit a majority of students.
2) Mastery based techniques may be embedded in a semester structure by careful design of programming questions that enable the students to test their own understanding of the material. Especially with the availability of so much educational material on line, the mastery technique allows the course to be structured so that the students can pace their own progress.
3) Project based learning helps students in introductory programming courses but a threshold of competence level is required before a student can reap the benefits of working in a project. Enforcing the threshold ensures that the students in the project are all able to contribute to the project thus enhancing the group experience and achievement.
4) Student motivation and engagement in projects is enhanced when members of the project team possess uniform levels of subject mastery.

References

[1] M. Jazayeri, "The education of a software engineer," in *Proceedings of the 19th IEEE international conference on Automated software engineering*. IEEE Computer Society, 2004, pp. 18–xxvii.

[2] M. Lanza, A. L. Murphy, R. Robbes, M. Lungu, and P. Bonzini, "A teamwork-based approach to programming fundamentals with scheme, smalltalk & java," in *Proceedings of the 30th International Conference on Software Engineering*, ser. ICSE '08. New York, NY, USA: ACM, 2008, pp. 787–790. [Online]. Available: http://doi.acm.org/10.1145/1368088.1368199

[3] M. Hauswirth, D. Zaparanuks, A. Malekpour, and M. Keikha, "The javafest: A collaborative learning technique for java programming courses," in *Proceedings of the 6th International Symposium on Principles and Practice of Programming in Java*, ser. PPPJ '08. New York, NY, USA: ACM, 2008, pp. 3–12. [Online]. Available: http://doi.acm.org/10.1145/1411732.1411734

[4] B. S. Bloom, "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring," *Educational researcher*, pp. 4–16, 1984.

[5] J. H. Block, P. W. Airasian, B. S. Bloom, and J. B. Carroll, *Mastery learning: Theory and practice*. Holt, Rinehart and Winston New York, 1971.

[6] S. Matsuura, "An evaluation method of project based learning on software development experiment," *SIGCSE Bull.*, vol. 38, no. 1, pp. 163–167, Mar. 2006. [Online]. Available: http://doi.acm.org/10.1145/1124706.1121394

[7] M. Gorlatova, J. Sarik, P. Kinget, I. Kymissis, and G. Zussman, "Project-based learning within a large-scale interdisciplinary research effort," in *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '13. New York, NY, USA: ACM, 2013, pp. 207–212. [Online]. Available: http://doi.acm.org/10.1145/2462476.2465588

[8] P. Thomas, J.-M. Labat, M. Muratet, and A. Yessad, "How to evaluate competencies in game-based learning systems automatically?" in *Proceedings of the 11th International Conference on Intelligent Tutoring Systems*, ser. ITS'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 168–173. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30950-2_22

[9] J. Jacková, "Learning for mastery in an introductory programming course," *SIGCSE Bull.*, vol. 40, no. 3, pp. 352–352, Jun. 2008. [Online]. Available: http://doi.acm.org/10.1145/1597849.1384394

[10] P. C. Blumenfeld, E. Soloway, R. W. Marx, J. S. Krajcik, M. Guzdial, and A. Palincsar, "Motivating project-based learning: Sustaining the doing, supporting the learning," *Educational psychologist*, vol. 26, no. 3-4, pp. 369–398, 1991.