Addressing mixed levels of prior knowledge by individualising learning pathways in a Degree Apprenticeship Summer School

Derek Somerville, Quintin Cutts, Matthew Barr, Jack Parkinson
University of Glasgow
Glasgow, UK
(Derek.Somerville,Quintin.Cutts,Matthew.Barr,Jack.Parkinson)@glasgow.ac.uk

ABSTRACT

Teaching an introductory programming course is beset by two core challenges: students enter the course with different levels of prior experience; and, for whatever reason, they progress at different rates. This is at odds with a typical face-to-face course, where material is delivered to the whole class in a lock-step fashion. Addressing these two challenges lies at the heart of an eight-week Summer School, described here, designed to bring a broad intake of students up to a common level ready to start on an degree apprenticeship programme that assumes a certain level of programming ability at the outset.

A programming test of students' understanding of programming constructs (e.g. conditional logic, loops, methods etc) allowed us to select for whom the course was mandatory, but it was open to all. Successful completion of the Summer School required students to demonstrate mastery of nine competencies. Each competency had an initial test, if the student did not meet the required level they were asked to complete coursework using Jupyter Notebooks. If the student passed the initial test they moved straight to the final test and if they passed, they moved to the next competency. There were also practice quizzes for students to do before the final test. This provided an individualised learning pathway for the students.

Students improved in the programming test and have given favourable reviews of the Summer School in a focus group.

CCS CONCEPTS

•Theory of computation → Apprenticeship learning; •Applied computing → Education; E-learning; •Software and its engineering → General programming languages;

KEYWORDS

Python, Introduction, Programming, Degree Apprenticeship, Graduate Apprenticeship

ACM Reference format:
Derek Somerville, Quintin Cutts, Matthew Barr, Jack Parkinson. 2020. Addressing mixed levels of prior knowledge by individualising learning pathways in a Degree Apprenticeship Summer School. In *Proceedings of Durham '20: ACM Computing Education Practice, Durham, UK, January, 2020,* 5 pages.

Durham '20: ACM Computing Education Practice, Durham, UK

DOI: 10.1145/3372356.3372370

1 INTRODUCTION

The University of Glasgow launched its first Graduate Apprenticeship degree programme in 2019-20 [1], choosing to do so with the subject of Software Engineering. The School of Computing Science at the University partnered with local businesses across Glasgow to recruit candidates via the company's own recruitment process. The 2019-20 cohort includes students with a broad range of computing science experience from a Higher National Diploma (HND) and Highers (one of the Scottish national school-leaving certificate exams and university entrance qualifications) to little or no computing science experience.

This paper describes a Summer School developed in the context of a degree apprenticeship programme in Software Engineering. The programme expects apprentices to have reached a certain level of mastery in programming before starting. Given the wide range of apprentices recruited by partner companies onto the programme, the Summer School must be effective across these ability levels to ensure that the whole cohort are suitably prepared by the first day of the programme.

The team at the university used a programming and problem solving test with all apprentices to determine who should attend the Summer School prior to beginning the undergraduate degree programme. To ensure all students could participate in the Summer School if required, it was presented primarily online with optional daily drop-in sessions. This made it possible for students to continue in existing employment and enabled employers to begin the work element of the apprenticeship programme earlier if they wished. The test and nature of the course enabled a number of individualised learning pathways to be supported across the cohort.

Students took the same test at the end of the Summer School, and all students' performance improved. A focus group session was held at the end of the course with students who had followed different learning pathways and they were very positive about the learning experience.

While the course presented here addresses a specific need identified in the design of a degree apprenticeship programme, the more general issue of students entering a programming course with different levels of prior experience, and then proceeding through the course at different speeds, will be familiar to any programming instructor.

2 SELECTION - APTITUDE TESTS

A battery of aptitude tests was compiled from recognised sources to assess both problem solving and programming skills. All apprentices selected by companies for the programme completed the tests, which is now described.

Course	Problem Solving (30)	Programming Test (9)
1P	21.25 (70.83%)	6.58 (73.11%)
1CT	18.30 (46.22%)	4.16 (61.00%)
Total	20.81 (69.38%) $\sigma = 4.67$	5.91(65.63%) $\sigma = 2.12$

Figure 1: Aptitude Test mean results for first year students

2.1 Problem Solving

The duration of the problem-solving test was 40 minutes with 30 questions - 20 questions on logical reasoning and ten on spatial skills. This test was not used to determine the need for the Summer School, but to help identify students that may require additional support during their studies. It is included here, because, as will be shown later in the paper, the Summer School appears to have had an effect on problem solving skills as well as programming.

2.1.1 Logical Reasoning. The Logical Reasoning measures the ability to identify patterns and sequences in obscured contexts, demonstrating logical thinking and cognitive flexibility to avoid distractors and deliberate contextual misleading elements. Logical reasoning has been shown to be a predictor of academic success by multiple researchers [9], where a correlation has been observed between logical reasoning and grades in an introductory computing course.

2.1.2 Spatial Skills. The Spatial skills aspect of the problem-solving test represents the ability to identify, generate and manipulate structures internally, such as the ability to visualise all the sides of a cube when they cannot all be visible, or to internally rotate a 3D object [10]. Spatial skills have been connected with success in Science Technology Engineering and Maths (STEM) for decades [5], and in recent years evidence has grown for a specific relationship with computing ability. Spatial skills are associated with many factors of computing success: grades, academic attainment and even code tracing and comprehension.

2.2 Programming Test

Students with prior programming ability tend to fare far better than their peers without, in an academic context [2]. Results from this section helped us to identify gaps in the student's understanding.

The test used to measure programming ability is the SCS1 [4]: a language independent, well-validated measure of fundamental concepts expected to be covered in an introductory computing course (i.e. the level of experience one would typically expect from someone who is competent in a programming language). A reduced version of this test with nine questions [6] was taken in 20 minutes.

2.3 Benchmarking the Test

In order to benchmark the test results, a sample of 30 first year undergraduate students on our traditional campus-based programme were selected who had completed a two semester Python course. In semester one these students are split into two groups, those who already have some programming experience (1P course) and those who do not (1CT course), please see figure 1.

3 INDIVIDUALISING LEARNING PATHWAYS

Twenty nine apprentices started the Summer School, three passed the programming test, but decided to attend anyway. The course was designed to be run over eight weeks, seven apprentices started two weeks before the end. To allow each apprentice to learn at their own pace we created several learning pathways, the benefits of individual learning pathways have been demonstrated [7].

3.1 Programming Constructs

Similar to other programming courses a programming competency model was used [8]. The basic programming competency constructs the apprentices learned are as follows:

- `Declaring, Manipulating and Displaying Variables`: Basic assignment and printing of variables.
- `Conditional Logic`: If, elif and else statements.
- `Data Storage`: Dictionaries, lists, arrays and tuples.
- `Iterations`: While and for loops.
- `Python File Management`: Reading and writing of files.
- `Methods`: Function and method calls.
- `Exceptions`: Exception handling.

For each of the above competencies, the programming constructs, had an initial test. If the apprentice got 100% they would move on to the final test. If the apprentice was above 80% they had the option to move to the final test or look at the lessons. The apprentices that did the lessons used Jupyter Notebooks to learn about the competency and this enabled them to experience writing and executing Python. The apprentices then had the option to do practice quizzes before they took the final test. Once the apprentice got above 80% in the final test they moved to the next competency. Some apprentices chose to jump between the lessons to help their understanding.

One apprentice highlighted they found the initial tests helpful to gauge how much effort was required for the competency, so spent longer on the lessons and doing the practice quizzes. The quizzes focused on learning to read code [3]. Only a few questions in each competency had missing code which had to be completed to give experience of reading and writing Python.

Below are the different learning pathways for each competency:

- `High prior knowledge and high confidence`: Initial quiz and final quiz.
- `High prior knowledge and low confidence`: Initial quiz, practice quiz and final quiz .
- `Medium prior knowledge and high confidence`: Initial quiz, lesson sections for mistakes and final quiz.
- `Medium prior knowledge and low confidence`: Initial quiz, lesson sections for mistakes, practice quiz and final quiz.
- `Low to no prior knowledge and time`: Initial quiz, lessons, practice and final quiz.
- `Low to no prior knowledge and low time`: Initial quiz, lessons and final quiz.
- `Low to no prior knowledge and inquisitive`: Initial quiz, lessons, future lessons, previous lessons, practice and final quiz.
- `No prior knowledge and very low time`: Lessons and final quiz.

As the Summer School progressed the apprentices used the face to face drop in sessions to collaborate with each other and with their team members in their workplaces; this collaboration element significantly enhanced the learning pathways.

3.2 Reading Solutions

It was felt that all apprentices would gain from reading programs [3]. A Playing Cards, Snap and Black Jack Python files were created for the apprentices to read. There were optional lessons explaining the Python files in more depth. The apprentice had a final test to check confirm their understanding of the Python files.

The apprentices used the following learning pathways:

- `High prior knowledge and high confidence`: Read Python files and completed the final test.
- `Low prior knowledge or low confidence`: Read Python files, completed lessons and completed the final test.

3.3 Writing Programs

The Writing Programs competency had a number of mandatory problems to solve, which increased in difficulty. There were a number of optional simpler problems using the same programming constructs which they could solve fist.

The apprentices used the following learning pathways:

- `High prior knowledge and high confidence`: Mandatory problems.
- `Low prior knowledge or low confidence`: Optional problems and mandatory problems.

4 SUMMER SCHOOL EVALUATIONS

Twenty-seven out of the twenty-nine apprentices have fully completed the course. One has not completed due to personal reasons and the other invalidated their results.

4.1 Aptitude Tests

As shown in Figure 2, for all apprentices who attended the Summer School, the problem solving average score before was similar to the undergraduate first year ICT (limited or no prior knowledge) average score (18 out of 30). The problem solving scores after the Summer School were marginally below the 1P (prior knowledge) first year students. The increase could be due to the apprentices sitting the test a second time, however it was noted for an apprentice that almost doubled their problem solving score that they had not completed the test the first time.

As shown in Figure 3, the increased improvement in the programming tests for apprentices who required the Summer School was sizeable at 31%. Although no re-testing was done to see if there was an effect of repeating the test, it was felt this would be small, since it was only ten weeks since they last sat the programming test. The apprentices did not see the correct answers or their score and the questions were also reasonably complex, so difficult to remember. The apprentices who required the Summer School had an average above the 1CT score of 4.16, but less then the 1P score of 6.58.

The average for the apprentices who did not require the Summer School was 7.5, but this is a very tiny sample. These apprentices all improved in the programming test, although only minimally since they already had a relatively high score.

No individual who has re-sat the programming test got a lower score, all improved. Only three students got one or two marks less for the problem solving, the rest improved.

Test	Before	After	Increase
Programming Test (9)	2.61 (28.97%) $\sigma = 1.71$	5.26 (58.48%) $\sigma = 1.79$	2.59 (28.81%) $\sigma = 1.62$
Problem Solving (30)	18.21 (60.71%) $\sigma = 4.30$	20.72 (69.07%) $\sigma = 3.73$	3.39 (11.30%) $\sigma = 3.43$
Logical Reasoning (20)	13.21 (66.07%) $\sigma = 2.71$	14.52 (72.60%) $\sigma = 2.49$	1.48 (7.40%) $\sigma = 2.77$
Spatial Skills (10)	5.00 (50.00%) $\sigma = 2.64$	6.20 (62.00%) $\sigma = 2.31$	1.22 (12.22%) $\sigma = 2.48$

Figure 2: Aptitude Tests mean results for all apprentices

Test	Before	After	Increase
Programming Test (9)	2.20 (24.44%) $\sigma = 1.12$	5.13 (56.94%) $\sigma = 1.70$	2.75 (30.56%) $\sigma = 1.65$
Problem Solving (30)	18.20 (60.67%) $\sigma = 4.33$	20.68 (68.94%) $\sigma = 3.97$	2.64 (8.79%) $\sigma = 3.53$
Logical Reasoning (20)	13.28 (66.40%) $\sigma = 2.73$	14.63 (73.13%) $\sigma = 2.60$	1.45 (7.23%) $\sigma = 2.78$
Spatial Skills (10)	4.92 (49.20%) $\sigma = 2.72$	6.38 (63.75%) $\sigma = 2.66$	1.53 (15.25%) $\sigma = 2.36$

Figure 3: Aptitude Test mean results for apprentices requiring the Summer School

No	Learning Style	Prior Coding	Programming Now
1	Mainly Online	Good	Slightly Polished
2	Mainly Online	None	Significantly Better
3	Drop in plus 70% online	None	Significantly Better
4	Drop in plus 70% online	Limited	Significantly Better
5	Drop in plus 70% online	None	Significantly Better
6	Drop in only	Good	Improved quite a bit
7	Some drop and half online	Limited	Significantly Better
8	Online for two weeks	Very Good	Slightly Polished
9	Some drop for two weeks	Limited	Significantly Better
10	Mainly online	Limited	Definitely Improved

Figure 4: Attendance and knowledge levels of focus group

4.2 Focus Group

Ten participants from a total of twenty-nine Summer School attendees were selected for the focus group. These were specifically chosen to represent a broad range of knowledge and experience of those who attended the course, as shown in Figure 4.

Some students had a negative preconception about the Summer School using Python when they knew they would be coding in Java in their new job, although they felt they benefited from learning an additional programming language and actually liked Python by the end of the Summer School. Other students asked to submit the written code in a language other than Python (which was allowed). Those students with no prior programming felt that Python restricted the syntax and so made it easier to learn to program.

One student with no prior programming jumped between competencies in both directions to help give context. With other students re-visiting prior lessons and questions. Some felt they just learnt the answers to the practice questions without fully understanding the solution.The use of Jupyter Notebooks was felt to aid learning, allowing apprentices to write code, although some felt there was a steep gap from the Jupyter Notebook lessons to the final quiz for some competencies.

The apprentices felt that the eight week time period for the Summer School was a good length. They also liked the lunch time drop in sessions from 12pm till 2pm. This allowed a few to work on problems in the morning, get help at lunch time and then work on the course again in the afternoon. It was noted by some apprentices who had started work that they were restricted and asked to "do Uni work in your own time". They also mentioned that going back to work after the drop in session gave their manager confidence that they were taking part in the school. A number of apprentices liked the online nature of the course as they felt that they could lead the learning, but they appreciated the help and explanations that they received at the face-to-face drop in sessions.

One apprentice said that they got a "big high" when they completed and fully understood their first program, whilst another liked improving on their solutions during the writing programs section. A number of apprentices felt they now have confidence in programming, even when looking at Java code. They now feel they understand code, even Java and were happy to "but in" in the workplace.

Some apprentices felt the Summer School helped them integrate with their new team, when they asked teammates for help with problems they faced during the course. One student noticed that they were receiving more help from a fellow student with collaborative work at the start of the Summer School, but by the end they felt they were collaborating equally. The apprentices felt they benefited from the chance to collaborate with their classmates and viewed the drop in sessions as a welcome opportunity to meet and discuss their work.

5 IMPROVEMENTS

Following the Summer School course survey and focus group a number of improvements were identified.

- `Parsons Puzzle`: Apprentices did not like these questions and the formatting indentation was lost within Moodle.
- `Writing Programs after a Competency`: Although it is recognised that learning to read programs first is important, the apprentices requested a mix. Adding a written program at the end of some competencies will help give some early successes. It was also suggested that the written program could be expanded as more competencies are completed.
- `Test Cases`: To give apprentices early feedback on written programs it was suggested test cases should be provided. This would complement the Test Driven Development (TDD) elements covered in the degree programme. It would also help to reduce the staff burden of grading written programs. One down side of this approach is it might dictate what methods to create and the overall approach of the solution. Apprentices did highlight they liked solving a problem in their own way.
- `A variety of prior coding templates`: To help apprentices with the Writing Program problems, a variety of levels of coding templates could be provide to give apprentices a helping start.
- `Add more difficulty to lessons in Jupyter Notebooks`: It was highlighted that for some competencies there was a large jump from the level of the lessons in Jupyter Notebooks to the final quiz. The proposal is to add more advanced lessons to the Jupyter Notebooks, bringing the apprentices up to the level of the final quiz.
- `Further reading`: One apprentice asked for recommended reading on computing history and other topics.

6 CONCLUSION

The apprentices of all levels of coding experience liked the approach of the individual learning pathways. The Summer School improved the apprentices understanding of programming constructs and helped them gain confidence. It also provided the apprentices with an excellent opportunity for collaborating with both their fellow classmates and their workplace teams.

REFERENCES

[1] Matthew Barr and Jack Parkinson. 2019. Developing a Work-based Software Engineering Degree in Collaboration with Industry. *UKICER Proceedings of the 1st UK Ireland Computing Education Research Conference* 9 (2019).

[2] Finland Helsinki. 2000. Does it help to have some programming experience before beginning a computing degree program? *ACM SIGCSE Bulletin* 32, 3 (2000), 25–28.

[3] Phil Robbins Raymond Lister Mike Lopez, Jacqueline Whalley. 2008. Relationships between reading, tracing and writing skills in introductory programming. *ICER '08 Proceedings of the Fourth international Workshop on Computing Education Research* (2008), 101–112.

[4] Mark Guzdial Shelly Engleman Miranda C. Parker. 2016. Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment. (English). *ICER '16 Proceedings of the 2016 ACM Conference on International Computing Education Research* (2016), 93–101. DOI : http://dx.doi.org/10.1145/2960310.2960316

[5] Jack Parkinson and Quintin Cutts. 2018. Investigating the Relationship Between Spatial Skills and Computer Science. *ICER '18 Proceedings of the 2018 ACM Conference on International Computing Education Research* (2018).

[6] Jonathan Gratch Mohsen Dorodchi Ryan Bockmon, Stephen Cooper. 2019. (Re)Validating Cognitive Introductory Computing Instruments. (English). *SIGCSE '19 Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (2019), 552–557. DOI : http://dx.doi.org/10.1145/3287324.3287372

[7] Dechawut Wanichsan Parames Laosinchai Sasithorn Chookaew, Patcharin Panjaburee. 2014. A Personalized E-Learning Environment to Promote Studentsfi Conceptual Learning on Basic Computer Programming. *Social and Behavioral Sciences* 116 (2014).

[8] Carsten Schulte. 2008. Block Model fi?! an Educational Model of Program Comprehension as a Tool for a Scholarly Approach to Teaching. *Proceeding of the Fourth international Workshop on Computing Education Research* (2008), 149–160.

[9] Ricky Baker E A. Unger. 1983. A predictor for success in an introductory programming class based upon abstract reasoning development. (English). *ACM SIGCSE Bulletin - Proceedings of the 14th SIGCSE technical symposium on Computer science education* 15, 1 (1983), 154–158. DOI : http://dx.doi.org/10.1145/952978.801037

[10] So Yoon Yoon. 2011. *Psychometric properties of the Revised Purdue Spatial Visualization Tests: Visualization of Rotations (the Revised PSVT:R).* Ph.D. Dissertation.

Purdue University.