How the First Year of a Work-based Degree in Software Engineering Prepares Students for Industry

Matthew Barr
School of Computing Science
University of Glasgow
Glasgow, Scotland
Matthew.Barr@glasgow.ac.uk

Derek Somerville
School of Computing Science
University of Glasgow
Glasgow, Scotland
Derek.Somerville@glasgow.ac.uk

Sebastian Dziallas
Computer Science
Fulbright University Vietnam
Ho Chi Minh City, Vietnam
sebastian.dziallas@fulbright.edu.vn

***Abstract*—This Practice Work-in-Progress paper presents a work-based undergraduate degree program in Software Engineering, designed in consultation with industry. Work-based learning is often seen as a means of ensuring university graduates are equipped with the skills and knowledge required to succeed in the workplace. The purpose of this study was to determine the extent to which the program has prepared students for working in industry. To this end, the students were surveyed after they had completed their first semester, and returned to their workplace for two months. Qualitative analysis of survey data revealed the aspects of their first year courses that apprentices valued most in the workplace. These aspects included exposure to multiple programming languages and existing codebases, as well as knowledge of Software Engineering tools and practices. Underpinning many of the students' responses, however, was the increased confidence that their university instruction afforded them in the workplace.**

***Index Terms*—work-based learning, software engineering, apprenticeships, higher education, confidence**

I. Introduction

Introductory computing course sequences have received substantial attention in the literature over the years [1], [2]. Examples include the Media Computation approach aimed at non-majors [3], which was also later adopted for computing majors [4], as well as courses with an emphasis on data analysis [5] and games [6]. Prior work has also examined different pedagogical approaches, including peer instruction and pair programming [7]. However, in contrast to the traditional CS1-CS2 course sequence, the first year presented here takes a different approach.

Apprenticeships are a long-established form of learning that remains prevalent in vocational and professional subject areas. However, despite the practical and professional nature of Computing Science - and Software Engineering, in particular - this form of work-based learning has not typically been associated with university-level CS education. This situation is changing, not least in response to suggestions that university courses do not adequately prepare graduates for the workplace [8], [9]. Work-based degree programs, then, are intended to better prepare students for the working world; however, such programs are relatively untested in CS, and come with a particular set of challenges. For example, the taught material may not align with workplace practices [10], workplace cultures [11], or organisational development approaches [12]. Despite these challenges, degree-level apprenticeships are increasingly common across the UK [13]. Here, such programs may be seen as a response to the issues identified in the government-commissioned Shadbolt review of graduate employability [14], which recommended "extending and promoting work experience" in university programs. Indeed, since the inception of Degree Apprenticeships in England ('Graduate Apprenticeships' in Scotland), this is how work-based degrees have been framed: as a means of addressing the skills gaps reported by employers and developing a skilled workforce [15]. A Graduate Apprenticeship in Scotland, specifically, comprises four years of study, with approximately 20% of the students' time spent at university, and the remaining 80% spent in the workplace.

The Graduate Apprenticeship described here is a work-based undergraduate degree in Software Engineering, designed from the ground up in consultation with industry (see [16]). The first year of this program comprises two eight-week blocks of university-based instruction, designed to equip students with the skills and knowledge required to succeed in the workplace. Outside of these blocks, students are based with their employer, learning on the job and applying what they have been taught at university. Partner employers recruit a wide range of students with mixed levels of prior knowledge of computer programming. Thus, we offer an optional summer school that teaches the students basic Python programming, with the aim of reducing the imbalance in prior knowledge before the apprentices start the program [17]. Newly-recruited apprentices are given a programming test [18] and the weaker students encouraged to take the summer school.

On the Graduate Apprenticeship program, the taught material is intended to ensure that students are productive in the workplace as soon as possible, covering a range of professional SE practices in the first teaching block. The material is delivered via two concurrent modules: one that focuses on programming languages and concepts, and another that focuses on the fundamentals of working as a SE professional. The programming module is novel in that it does not concentrate on any single programming language, drawing instead on examples of many different technology stacks, and working with existing codebases. The professional practice module covers ethical issues, design life cycles, development methodologies, and security factors - material not typically taught until later in a SE degree.

The purpose of this work is to determine the extent to which the first teaching block prepared students for working in industry, and to better understand which aspects of the taught material have proved most beneficial in the workplace. It is important that we evaluate our approach, to inform our own practice: flipping a CS degree around to focus on work-based learning is new territory for us. Furthermore, our block-based approach is far from traditional: most apprenticeship programs are delivered using a 'day release' model, whereby students are typically at university or college one day each week [16]. It is also notable that we 'front load' our teaching, with a greater proportion of the students' time spent at university in the first 18 months of the program. The more usual approach - certainly in the UK - is to evenly distribute the 20% of apprentices' time spent at university over the entire duration of the program. In terms of both content and delivery, this program, therefore, represents an innovative – and largely untested – approach to teaching Software Engineering. An evaluation of our approach, then, is of interest to any CS educator tasked with designing (or re-designing) a work-based degree program. Indeed, there are implications for the design of more traditional CS programs, too.

II. Methods

An online survey was distributed to first year students on the program after they had completed their first semester and returned to their workplace for two months. Ethical approval for evaluating the program in this way was obtained from the College Ethics Committee. The survey was structured around the two modules' Intended Learning Outcomes (ILOs), with students asked how their attainment of each ILO had helped them in the workplace. The first year cohort comprised 48 students; 42 (88%) students provided responses. Participants were assigned pseudonyms to ensure anonymity. In line with the process of thematic analysis established by Braun & Clarke [19], students' free-text responses were first coded by one of the authors. These initial codes were then reviewed and refined in discussion with all three authors, before being grouped into themes. Following another round of discussion - which resulted in the consolidation of codes related to SE tools into a single group - the themes were more precisely defined and named.

III. Results and Discussion

We identified themes related to programming, practices in the workplace, and students' confidence. These themes are reflected in the headings below.

A. Programming

During the module on programming languages and concepts, students learn to write object-oriented programs and are exposed to multiple programming languages. In doing so, the apprentices are encouraged to understand the underlying similarities between programming languages, as well as the differences ("Common programming misconceptions or aspects of more obscure languages which may not map directly to commonly used languages" – Bob). The idea that this experience prepared apprentices for programming in the workplace ("[It] allowed me to understand languages much faster." – Lisa) is one of the strongest themes identified in the data. As one apprentice put it:

> Most programming languages share the same basic concepts and my workplace uses a range of languages, therefore it's incredibly useful to focus on these concepts so that you can understand new languages quickly. – Muhammad

However, the same student does suggest that still more languages could be introduced to the course, specifically those used in the apprentices' workplaces:

> I think the course addressed most of the concepts that I could think of but I would maybe say it could dip a little bit into the specifics of some languages, perhaps the languages which are most common among the class. – Muhammad

This is a fair comment, of course. However, it should be noted that the languages used for examples provided in class are informed by a pre-sessional survey, which asks apprentices to identify the languages they believe they'll be using in the workplace. It is not possible to include everything - some proprietary technologies are difficult to incorporate, for example - but most respondents understood the value of a course that included examples in multiple languages, even if their specific language was not the focus. For example:

> [The course] gave me a basic knowledge of coding that I could transfer to the language I use in work easier and with less adjustment period. [...] It has helped me use a language and software that is not covered in university. – Frank

> It helped me with being able to understand the concept of what a program is doing, as I've maybe not done the concept in one language but have seen it in another so I can follow along with what is going on. – Ahmed

> Similarities and differences in programming languages allowed me to look at different codes that I don't recognize from university and break it down based on my understanding of programming languages in general to get a grasp on what I am looking at. – Arlene

Apprentices also reported that the course helped equip them to deal with the multiple languages encountered in a single workplace:

> Not all code I have seen in the workplace is from the same language. So having the ability to figure out what the code is doing based on similarities to other language is very useful. [...] Although understanding the differences is also crucial as knowing that techniques that works for one language may not work

> for others is useful in developing a larger knowledge of programming in general. – Daniel

Students are also introduced to how to navigate large codebases, as the codebases they encounter in their companies are substantially larger than those first-year undergraduate students typically come across. For instance, they explore the structure of projects on GitHub and make small changes to existing open source projects. Students commented that having this prior experience prevented them "from being overwhelmed" (Nadia) and allowed them to "make a difference very quickly in my company" (Sven).

Many of the apprentices appreciated that working with existing codebases reflected the reality of what they would be doing in the workplace, for example:

> You aren't going to be writing new systems everyday, it's more likely that you will be fixing problems or making adjustments to an existent codebase so learning how to make additions to large codebases has been extremely useful in understanding the impact of my work and role. – Jose

> This was incredibly useful as workplace codebases come in different shapes and sizes, having the experience of navigating large codebases and the basics of how to add to a codebase was crucial. – Brett

B. Practices

Students also reported understanding practices that are commonly used in industry. One example here was recognizing the importance of writing clean code:

> Knowing the 'professional' way of doing things helped me get in the mindset of coding with cleanliness and standards in mind, rather than just writing a lot and cleaning it up after. – Michal

This was even the case in contexts where there were no explicit standards:

> I asked a mentor about the standards in my company, and was surprised to realise we didn't have any, however in practice we limit our line length and have name conventions for our classes. – James

It is also interesting to note that these first-year apprentices recognised the value of readable code ("It helped me to understand why people in my team use comments and have clean code" – Karen), which novice programmers typically do not [20]:

> In my workplace readability of code is very important as there are a lot of different developers that have to work with the code, so if it is difficult to read then developers from different teams or even within the team may struggle to understand what the function of the code is, so learning about readability is a lot of help. – Jacob

> Learning about easy-to-read code has massively helped my progress in the workplace as I didn't have much prior experience with coding. Easy-to-read code helps developers to pick up work on codebases they may not have written originally and understand what they are then working on. – Susan

The university courses also covered agile development principles and software development tools, which apprentices found to be useful:

> Learning about development methodologies has been extremely relevant to my role in [company]. Most of the development methodologies we covered in university exist in some capacity here, which has been very helpful in understanding how my team work to produce an end-product for stakeholders. – Susan

> [It was] useful to learn about git, IDEs etc. which I have begun to use. – Omar

In particular, understanding industry development practices has enabled apprentices to make a contribution in the workplace from the outset, engaging in what might be termed 'legitimate peripheral participation' [21]:

> My workplace uses the principles of Agile quite extensively so understanding the detail and theory has helped a lot in the reasons why we work the way we do and how I can better contribute. – Jose

Another common topic was related to testing. Prior work found that many apprentices were writing unit tests in their workplaces [22] and we found that in these data as well. For example:

> Very useful – a lot of my initial changes were in adding small unit tests and doing this with the background given in the testing sections was far easier. – Michal

C. Confidence

A significant theme was related to students' confidence. Begel & Simon have previously observed that: "Asking questions, however, reveals to your co-workers and managers that you are not knowledgeable, an exposure that most new developers felt might cause their manager to reevaluate why they were hired in the first place." [23] However, the apprentices here seemed confident to ask questions:

> Again having an understanding of these at a basic level enabled me to identify specific areas I didn't fully understand and therefore have the ability to ask the right questions when it came to the workplace. – Nadia

Students also described feeling more confident about approaching problems in different programming languages, as they had previously experienced different languages in their studies ("it allowed me to approach other languages I had not seen before with confidence" – Lucas). Other examples included:

> Had to make changes to the front end of a web app written in JavaScript using the React framework - was much less intimidated having been exposed to similar concepts in multiple languages. – Grant

> Seeing the broad similarities in basic concepts being used across languages really helped me when starting on a C# project while only knowing Java - knowing that I didn't have to learn an entirely new way of doing it and just had to use the same concepts but with different syntax made it less scary! – Michal

This improved confidence extended also to the tools encountered in the workplace, for example:

> Also helped me feel more confident when using software such as Jira as I wasn't seeing it for the first time in a work environment. – Frank

The professional practice module was also reported to have given apprentices the confidence to challenge ethical and professional behaviour in the workplace:

> It helps me understand my companies policies and why they are there, along with helping me to identify any breaches and feel confident enough to address them. – Arlene

IV. Conclusion

Based on these data, it appears that our initial eight-week block of teaching successfully prepares students for the workplace. This is achieved by exposing students to multiple programming languages to develop a deeper conceptual understanding, and by requiring students to work with existing codebases from the outset. Covering development practices including agile, software development tools, and testing methodologies has also been important in preparing apprentices for work. By demystifying the concepts and practices encountered in the workplace, students are equipped not only with the requisite skills and knowledge but also the confidence required to apply these in the workplace. We will continue to refine the modules described here, adding additional programming languages to the repertoire of examples and introducing some larger codebases. In doing so, our underlying purpose will be to nurture the confidence that our students indicate is so important in the workplace. We believe that this evidence for the efficacy of our approach to delivering the apprenticeship degree may be of interest to others developing programs in Software Engineering.

We are also excited about the emerging theme of student confidence in the workplace, as recent research has linked students' self-efficacy to persistence in the field [24]. The participants in this work described feeling more confident in approaching problems in the workplace. In contrast, Begel and Simon, in their work on novice software developers in industry, noted that recent graduates experienced a power inequality and social anxiety that prevented them from seeking help from colleagues [23]. They observe that many of their participants only asked questions "after flailing for a long time and spending many hours ineffectually trying to solve a problem." [23] While the program described in this paper is not the first to emphasize student confidence [25], we believe that future work then has the potential to identify specific aspects that support student confidence in the classroom and in the workplace.

References

[1] B. A. Becker and K. Quille, "50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. Minneapolis MN USA: ACM, Feb. 2019, pp. 338–344.

[2] A. Luxton-Reilly, Simon, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, M. J. Scott, J. Sheard, and C. Szabo, "Introductory programming: A systematic literature review," in *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. Larnaca Cyprus: ACM, Jul. 2018, pp. 55–106.

[3] M. Guzdial, "A Media Computation Course for Non-majors," in *Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '03. New York, NY, USA: ACM, 2003, pp. 104–108.

[4] B. Simon, P. Kinnunen, L. Porter, and D. Zazkis, "Experience report: CS1 for majors with media computation," in *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE '10*. Bilkent, Ankara, Turkey: ACM Press, 2010, p. 214.

[5] R. E. Anderson, M. D. Ernst, R. Ordóñez, P. Pham, and B. Tribelhorn, "A Data Programming CS1 Course," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. Kansas City Missouri USA: ACM, Feb. 2015, pp. 150–155.

[6] J. D. Bayliss and S. Strout, "Games as a "flavor" of CS1," in *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education - SIGCSE '06*. Houston, Texas, USA: ACM Press, 2006, p. 500.

[7] A. Salguero, J. McAuley, B. Simon, and L. Porter, "A Longitudinal Evaluation of a Best Practices CS1," in *Proceedings of the 2020 ACM Conference on International Computing Education Research*. Virtual Event New Zealand: ACM, Aug. 2020, pp. 182–193.

[8] K. Min, J. Jackman, and D. Gemmill, "Assessment and evaluation of objectives and outcomes for continuous improvement of an industrial engineering program," *International Journal of Engineering Education*, vol. 29, no. 2, p. 520–532, Jan 2013.

[9] E. O. Navarro, *A Survey of Software Engineering Educational Delivery Methods and Associated Learning Theories*. University of California, Apr 2005, no. UCI-ISR-05-5. [Online]. Available: https://isr.uci.edu/sites/isr.uci.edu/files/techreports/UCI-ISR-05-5.pdf

[10] P. Tynjala and P. Hakkinen, "E-learning at work: Theoretical underpinnings and pedagogical challenges," *Journal of Workplace Learning*, vol. 17, p. 318–336, 2005.

[11] U. Backes-Gellner, *Benefits of Apprenticeship Training and Recent Challenges - Empirical Results and Lessons from Switzerland and Germany*. Swiss Leading House, Feb 2014, no. 97. [Online]. Available: https://econpapers.repec.org/paper/isoeducat/0097.htm

[12] P. Dillenbourg, "Over-scripting cscl: The risks of blending collaborative learning with instructional design." in *Three worlds of CSCL. Can we support CSCL?* Open Universiteit Nederland, 2002, p. 61–91, publisher: Heerlen, Open Universiteit Nederland. [Online]. Available: https://telearn.archives-ouvertes.fr/hal-00190230

[13] A. Powell, "Apprenticeship statistics: England," House of Commons Library, Tech. Rep., 2018.

[14] N. Shadbolt, "Shadbolt review of computer sciences degree accreditation and graduate employability," Department for Business, Innovation & Skills, London, UK, Tech. Rep., 2016.

[15] S. Smith, M. Caddell, E. Taylor-Smith, C. Smith, and A. Varey, "Degree apprenticeships - a win-win model? a comparison of policy aims with the expectations and experiences of apprentices," *Journal of Vocational Education & Training*, vol. 73, no. 4, p. 505–525, 2021.

[16] M. Barr and J. Parkinson, "Developing a work-based software engineering degree in collaboration with industry," in *Proceedings of the 1st UK & Ireland Computing Education Research Conference*, ser. UKICER. Association for Computing Machinery, Sep 2019, p. 1–7. [Online]. Available: https://doi.org/10.1145/3351287.3351292

[17] D. Somerville, Q. Cutts, M. Barr, and J. Parkinson, *Addressing mixed levels of prior knowledge by individualising learning pathways in a Degree Apprenticeship Summer School*. New York, NY, USA: Association for Computing Machinery, Jan 2020, p. 1–5. [Online]. Available: https://doi.org/10.1145/3372356.3372370

[18] R. Bockmon, S. Cooper, J. Gratch, and M. Dorodchi, “(re)validating cognitive introductory computing instruments,” in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE ’19. New York, NY, USA: Association for Computing Machinery, Feb 2019, p. 552–557. [Online]. Available: https://doi.org/10.1145/3287324.3287372

[19] V. Braun and V. Clarke, “Using thematic analysis in psychology,” *Qualitative Research in Psychology*, vol. 3, no. 2, p. 77–101, Jan 2006.

[20] S.-N. A. Joni and E. Soloway, “But my program runs! discourse rules for novice programmers,” *Journal of Educational Computing Research*, vol. 2, no. 1, pp. 95–125, 1986.

[21] S. Dziallas, S. Fincher, M. Barr, and Q. Cutts, “Learning in context: A first look at a graduate apprenticeship,” in *21st Koli Calling International Conference on Computing Education Research*, ser. Koli Calling ’21. Association for Computing Machinery, Nov 2021, p. 1–11. [Online]. Available: https://doi.org/10.1145/3488042.3490020

[22] M. Barr and D. Somerville, “Preparing software engineering apprentices for industry,” in *Proceedings of the 2020 ACM Conference on International Computing Education Research*, ser. ICER ’20. New York, NY, USA: Association for Computing Machinery, Aug 2020, p. 310. [Online]. Available: https://doi.org/10.1145/3372782.3408116

[23] A. Begel and B. Simon, “Novice Software Developers, All over Again,” in *Proceedings of the Fourth International Workshop on Computing Education Research*, ser. ICER ’08. New York, NY, USA: ACM, 2008, pp. 3–14.

[24] J. Gorson and E. O’Rourke, “Why do cs1 students think they’re bad at programming? investigating self-efficacy and self-assessments at three universities,” in *Proceedings of the 2020 ACM Conference on International Computing Education Research*, ser. ICER ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 170–181. [Online]. Available: https://doi.org/10.1145/3372782.3406273

[25] S. Rosenthal and T. R. Chung, *A Data Science Major: Building Skills and Confidence*. New York, NY, USA: Association for Computing Machinery, 2020, p. 178–184. [Online]. Available: https://doi.org/10.1145/3328778.3366791