Developing a Work-based Software Engineering Degree in Collaboration with Industry

Matthew Barr
Matthew.Barr@glasgow.ac.uk
University of Glasgow

Jack Parkinson
Jack.Parkinson@glasgow.ac.uk
University of Glasgow

ABSTRACT

Work-based learning has been in practice in Software Engineering for some time, but only in recent years has it been introduced as a pathway to an honours-level undergraduate degree across the UK. Through the lens of one such scheme, the Graduate Apprenticeship programme in Scotland, we have investigated what challenges work-based learning degree programmes are likely to face and took this question to 26 industry partners. Also, since we are aware of a persistent skills gap between Software Engineering graduates and entry-level industry roles, we investigated the skills that Software Development teams are looking for in Scotland. This paper details our findings concerning perceived challenges to industry, the skills and knowledge to be imparted at university and the workplace learning opportunities which can be exploited by companies.

CCS CONCEPTS

• **Social and professional topics → Industry statistics**.

KEYWORDS

software engineering, development, work-based learning, apprenticeships

ACM Reference Format:
Matthew Barr and Jack Parkinson. 2019. Developing a Work-based Software Engineering Degree in Collaboration with Industry. In *UK & Ireland Computing Education Research Conference (UKICER), September 5–6, 2019, Canterbury, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3351287.3351292

1 INTRODUCTION

Work-based Learning (WBL) is a largely untested mode of delivering Computing Science or Software Engineering education, and a genuine apprenticeship in Software Engineering goes far beyond traditional approaches to incorporating industry experience into university courses. In a truly work-based degree, the concept of work placements forming part of a taught programme is reversed: the apprentice spends a significant majority of their time in the workplace, rather than on campus. Thus, university staff must possess a deep understanding of employer partners' Software Engineering practices and be clear about employer expectations relating to the development of apprentices' skills and competencies. This paper outlines the results of extensive industry consultation, intended to develop an understanding of employer practices and expectations which, in turn, underpins the design of a new Graduate Apprenticeship in Software Engineering programme.

UKICER, September 5–6, 2019, Canterbury, United Kingdom

ACM ISBN 978-1-4503-7257-2/19/09…$15.00
https://doi.org/10.1145/3351287.3351292

2 BACKGROUND

WBL has technically been practiced for centuries in the form of apprenticeships. However, WBL towards a university degree - Bachelors or Masters - is a relatively new concept in the UK: in 2014 Degree Apprenticeships were announced in England, a WBL programme which results in a university degree. Degree Apprenticeships have since been made available in Wales, and Northern Ireland have expanded their Higher Level Apprenticeships to include university degrees. Scotland has followed suit with Graduate Apprenticeships, building on the Modern Apprenticeships scheme monitored and funded by Skills Development Scotland (SDS), who have also developed the frameworks for specific subject areas on which the programmes are focused.

Based on the wide adoption of these programmes across the UK and figures provided by the UK Government indicating a steady growth in their uptake [14], it is clear that WBL with the goal of attaining a degree is increasing in popularity. A Scottish Graduate Apprenticeship for an honours degree is a four-year programme, a significant time investment for both students and companies. Companies are also expected to pay students throughout their degree, which could be considered risky given that some of these apprentices begin their degree with few practical skills, or substantial evidence of them, in the areas they are employed. The degree is pitched at the same level as existing undergraduate degrees, putting pressure on universities to ensure that quality is of an equal standard whilst typically being delivered with fewer contact hours.

As a result of these risks, it's important that institutions offer the right WBL degree. This is not possible without fully understanding the challenges faced by companies, education providers and students engaging in these degrees. Several studies in the past few decades have identified challenges in WBL and adjacent means of study: organisational culture and social attitudes must support WBL, which is something that cannot be effectively controlled [4, 7, 17]; linking personal development with expected organisational development is rarely straightforward [7, 17]; attitudes towards learning materials are not always shared by the learning provider and the company [17]; WBL is resource intensive [11]; education providers and companies are struggling to recruit students [4, 11].

In the UK, one study reviews the development of a Chartered Manager Degree Apprenticeship from the perspective of the programme designers, citing misinformation about the value of the degree, grappling with standards bodies and internal institutional

resistance as challenges faced [16]. Rowe *et al.* later break down several challenges in the areas of skills development, mentoring and employer-driven pedagogic approach, and collected qualitative data about these challenges from managerial positions in companies [15]. The organisations' feelings were overwhelmingly positive, but the authors are very clear that the sample size was too small and the programmes too young to infer that these challenges have been overcome, and call for further research with larger subject pools.

The stated challenges offer a well-established basis for formulating and delivering a WBL programme, but do not paint a full picture of the challenges likely to be faced with launching these degrees in Scotland, specifically in Software Engineering (SE). In order to truly understand what the key sticking points are going to be, it is important to communicate with companies and establish what their perception of the landscape is and how they feel challenges are likely to arise.

It must be acknowledged that in the past, SE graduates have not fully met the standards required by industry in terms of knowledge and skills. The Software Engineering Body of Knowledge (SWEBOK) represents a framework designed to solve this issue by discerning the key fields of SE required in the workplace and providing practitioners and course designers with a collection of skills and knowledge to be ticked off [10]. This work is, however, out of date, with the most recent update to the framework in 2004 [1], likely too long ago to be of great relevance as reference material in such a fast moving domain [6]. In later research, Azuma *et al.* reflect on the issues with SWEBOK's basis in Bloom's taxonomy [5] and propose alternatives and adjustments, aiming to reduce the gap between what could be called academic and industrial SE [3].

Other researchers have observed the skill gap between SE graduates and the industries they go on to work in and have attempted to provide various solutions [9, 12, 13], with no particular method surfacing as a standout best approach. This presents us with a major concern when designing WBL programmes: we must ensure that we are delivering the right content. Furthermore, given that these students will be in post from their first day of study, there is an added requirement, not required of typical undergraduate degrees, that the order of teaching is structured in such a way that these students are of value in the workplace early on.

3 RESEARCH QUESTIONS

As a result of the background knowledge collected about these programmes and the recognition of an obvious need for a better understanding of the area, we have formulated the following research questions:

(1) What are the perceived challenges in delivering work-based education in Software Engineering?
(2) What skills and competencies are companies looking for? This can be split into three further questions:
   (a) What skills and competencies should be taught in a university environment?
   (b) What are the learning opportunities that companies can provide?
   (c) What skills and competencies need to come first so that both the university and the companies can provide useful learning experiences?

4 METHODS

In total, 26 companies completed an initial online survey and 22 took part in subsequent interviews. A form of purposive sampling was employed, acknowledging that random sampling is not feasible in SE research [2]. Our sample was drawn from employers known to have expressed interest in software-related Graduate Apprenticeships, with additional companies approached in order to ensure that a range of sectors and company sizes was included. Sectors comprised technology, finance, health, education, defence, energy, construction and games. The number of software engineers employed by companies ranged from two to an estimated 1500.

The online survey was intended to establish employer expectations, skills requirements and existing recruitment and training practices. The survey also served to prime employers for the follow-up interview, the schedule for which followed a similar structure but allowed interviewees to expand upon points made in their survey responses. Surveys were administered using the Jisc Online Surveys system[1] in accordance with UK General Data Protection Regulations (GDPR). Each employer was provided with a unique survey URL and advised that the questions may be completed in multiple sittings. Respondents were encouraged to complete the survey in collaboration with colleagues, as appropriate. For example, some questions were better answered by a colleague in the HR department, while others were more relevant to colleagues in IT.

Interviews were conducted with a mixture of personnel, usually from an IT background but with HR or senior management personnel contributing as required. All but one of the employers consented to audio recording of the interviews, which were conducted on company premises. The employer that did not consent did so on security grounds. The mean interview duration was 49 minutes and audio recordings were subsequently transcribed for analysis, which was conducted using NVivo qualitative data analysis software. The analysis was conducted using a broadly deductive approach, following the Framework Method [8]. As such, the analytical framework was largely defined *a priori* by the interview schedule, with additional codes developed through a process of open coding and grouped according to the predefined analytical framework. Analysis was conducted by two researchers, following transcription by a professional transcription service. Both researchers familiarised themselves with the transcripts in their entirety before independently coding five of the 22 interviews. Following this initial coding, the researchers met to agree a common set of codes which were then applied to the analysis of the remaining transcripts. These codes are reflected in the structure of the Analysis & Discussion section below. Finally, results of the analysis were presented to interviewees for verification. The study - comprising surveys and interviews - received ethical approval from the relevant College Ethics Committee.

5 ANALYSIS & DISCUSSION

Many questions were asked of companies in the various stages of the consultation, resulting in rich data about the operation and needs of Software Development houses in Scotland. Detailed here are just the most significant responses providing answers to the research questions posed.

[1] https://www.onlinesurveys.ac.uk/

5.1 Perceived Challenges

This section details the challenges that employers having highlighted prior to taking on an apprentice.

5.1.1 Balancing Time. Perhaps the most commonly-cited challenge associated with Graduate Apprenticeship delivery concerns getting the right balance between time spent working and time spent studying. This challenge is exacerbated by the sheer range of employer expectations: there is little agreement on what that balance should be. Employers' stated preferences included fully online delivery of taught material with no time spent on campus, one or two days per week on campus, or one week per month on campus.

For employers who preferred longer contiguous blocks of university time, the advantages lay in requiring "less switching of focus between learning and work", with another employer noting that "five days would allow someone to understand concepts, then have three weeks to apply what they have learned". Still another employer suggested that longer university blocks would provide Graduate Apprentices with "a concentrated period of study, rather than dipping in weekly, and a proper break from the workplace". At the other end of the spectrum, employers in favour of wholly online learning provision cited "cost reasons and to reduce the amount of time the student is away from home".

In short, there is little consensus among employers about the optimal balance, and this is a challenge. However, employers generally did not see issues with timetabling, bar "unexpected workplace demands" or "any big events that occur at the workplace during the week out". These challenges aside, most employers suggested that advance planning would address any concerns.

5.1.2 Content Delivery. Related to the issue of mode of study is the challenge of learning at a distance: there are employers in Scotland for whom regular travel to even the nearest university is expensive and time-consuming. One of the most remote employers consulted made the following suggestion:

> Online learning would be good but a blended learning approach for us would be best so there is some face-to-face interaction. This could be a two days block bimonthly or even lectures through VC [video conferencing].

Several employers, however, did note the importance of face-to-face access with lecturers and that it was "important that [an] apprentice gets to be regularly in contact with other students".

5.1.3 Privacy and Information Security. Another 'non-issue' worth highlighting is the need for university staff to view students' work, raising potential data sharing, privacy and IP concerns. However, most employers simply assumed that a Non-Disclosure Agreement (NDA) would cover all parties concerned. Indeed, NDAs were expected to cover more general concerns about IP that might arise from hosting a student on work premises.

5.1.4 Entry Requirements. Aside from getting the right balance between university and workplace activity, perhaps the next most contentious issue with employers was the academic entry tariff imposed by our particular institution. In short, the required grades were believed to be too high, and likely to discourage applicants from certain backgrounds. As entry requirements differ between institutions, this issue is not considered here in further detail.

5.1.5 Providing for Extended Periods of Time. Finally, the challenge of committing to a four-year apprenticeship was identified by several employers. Concerns related to the fact that such an approach "differentiates from [the existing] business model so will require a new way of working", with another employer stating that: "our business changes to meet the evolution of consumer and partner needs. The course needs to be flexible enough to accommodate significant changes in technology and business practise."

Another respondent noted that such change could come about "if the company was sold to an investor and they did not want apprenticeships". The greatest concern related to such a commitment was that the Graduate Apprentice may not perform well enough, with several employers noting that they would require clear means of terminating their relationship with a student. For example: "given the size of the company, there may be issues if the apprentice isn't performing at the expected level after a significant period of time."

5.2 University-based Learning

This section details what employers expect to be taught on WBL programmes, focusing on skills and knowledge that should be addressed by the education provider.

5.2.1 Theoretical Foundations. Understandably, each organisation defined theoretical foundations a little differently. We deliberately did not assign a clear-cut definition to the term, and instead expected employers to provide their own. The most common subject areas considered to fall under this term were: how compilers work; database fundamentals; theory of Agile; operation of processors; efficiency; object oriented concepts; program architecture; disk scheduling; networking fundamentals; and operating systems.

Obviously some of these subject areas are more granular or more theoretical than others, but ultimately the majority are similar in that they do not directly contribute to practical, day-to-day development. Despite the essentially non-practical nature of these fundamentals, they were overwhelmingly regarded as being useful (or even essential) skills by many of the companies consulted. Where possible we directed the discussion to ascertain why this might be the case and identified several key reasons.

Firstly, this knowledge is required for conceptual understanding of technology, as one company said: "...fundamentally the theoretical side transcends most of it, if not all, so it's the point it goes to the heart, I guess, of our view whereby in some form the tech doesn't matter."

These skills were also identified as being required for learning and picking up new concepts, as an existing apprentice recalled: "...I felt it was valuable to have the conceptual understanding because it meant I didn't need to know specific languages. I was able to pick them up because I understood the concepts."

Although not considered to be directly practically applicable skills, computational theoretical fundamentals also seem to improve the quality of written code, as one company indicated: "...it's not something that you usually have time to look into when you're working to deadlines. But, it's good to have that understanding, it helps you to write better code in the first place."

The knowledge of theoretical concepts is regarded as essential for career progression, and it was also noted that it creates opportunities and encourages further personal development on an internal, individual level.

> ...it's the ones that really go out their way and learn a bit more of theory, do a bit more reading, do a bit more... watching the videos that excel and become our senior developers... And the ones that do really lean in to it will progress a lot faster...

These fundamentals provide people with better problem-solving skills, logic and confidence, with one employer stating that: "I know the ones that have a better education, and to me, I can tell the difference in terms of their work." Expanding on confidence, it was noted that awareness of these foundations gave the *employer* more confidence in the abilities of the student, not just affecting the students' own confidence:

> And it gives us a bit of confidence that they've got that, that skill set when they're coming in that at least when we start speaking techie talk to them, then at least they know what we're on about.

It is important that these concepts are taught at university because it is not considered as something that can be effectively taught by companies. One employer at a small software development company reflected on their own university experience:

> ...we did a lot of operating systems. How many people are going to do an operating system? Nobody. But knowing how it works is very important, and companies don't have the resources to teach that.

These concepts are applicable in many contexts, making the skills learned versatile, flexible and persistent, as reflected by another employer:

> I might not have to go and implement, like, a scheduling algorithm for a disc drive but it's going to be quite useful for me to know that because the same thought processes and ideas, that can apply to anything else.

Three companies indicated that these skills would not strictly be useful to them: one large defence contractor indicated that they can be filled in later. Another employer did concede that they would be useful for progression but less so for the day-to-day work of the apprentice:

> For them, that will hold them in very, very good stead for the rest of their career because the world will change, and actually understanding at a deep level how stuff works. People can be useful to us very, very quickly without that. So how important is it? For the individual, yeah, it's quite important. For us, we can live without it.

Finally, another company noted these concepts as "historic", meaning that they are not applicable in a modern context.

> Yeah, that's a really interesting one because I think it is steeped a little bit in history as well. And I'm all for the history of it, but do we still need to focus so much on it? I'm not entirely convinced.

Despite these comments, the industries consulted generally consider these skills to be useful in many different ways and should be considered valuable in shaping well developed graduates.

5.2.2 Languages and Technologies. Questions were included to probe roughly what technology stacks, languages, frameworks and tools were being used in industry in order to establish which areas focus should be placed.

As might be expected, any discussion of the tools used in SE is likely to relate to the adopted methodology. For example, when asked if Agile was used within one public sector organisation, the response was "We're using Jira" . Similarly, several participants referred to tools such as Cucumber, Gherkin and GitLab when asked about their approach to testing.

Given the breadth of technologies in use across the sector, the tools used to develop software are similarly broad. Jenkins is typically, but not universally, used for continuous integration. Git is widely used for version control, but Subversion and other tools are also in use. Jira is frequently used for issue tracking, but not across the board. In short, the types of tools used across the industry are broadly similar, but the specific products in use varies considerably.

Companies with a web development function reported using up to seven frameworks on current projects, although some companies use "bits and pieces" of many more frameworks and technologies in their web and platform development.

> We have C#, ASP.NET, web API projects. We also have Java projects such as Java Spring. But we also have JavaScript, Node, UJS, React and Angular. And I think there's Python as well.

Several companies reported that their web development stack has changed recently, with one company highlighting the challenge of teaching web development in a rapidly changing landscape.

Object oriented (OO) development proved ubiquitous. Four companies stated that they do a lot of Java development, with one suggesting that an understanding of Java was an "obvious" requirement. Three companies indicated that they struggle to recruit sufficient Java developers, with one referring to Java developers as "gold dust". Other companies were more interested in OO concepts and paradigms being taught, rather than specific languages, and it was suggested that OO concepts were the most important topic to front-load in teaching.

Full stack development was raised explicitly by multiple companies, who all indicated that the ability to move across the stack as required would be useful for an apprentice to have.

> The guy who is doing the front end just now, he would do back end things as well; so most of the team is full stack... they have preferences, but they are able to do everything.

Cloud development, mobile app development and functional programming across several platforms were also mentioned as highly desirable skills for students to be taught.

Given the extensive range of languages and technologies used in industry preparing students to program in the specific language and environment used by the workplace is a daunting task, which would require a largely bespoke teaching experience for every student and would be a huge drain on resources.

Our proposal to address this issue is to teach the fundamental underpinnings of these many different languages and technologies, providing the skills and knowledge necessary to understand them in abstract terms, and allow their wealth of industry time to fill out the practical application of these concepts. For example, a university may provide extensive teaching on OO development, and the student will be able to apply this knowledge regardless of whether their workplace uses Java, C# or OO Python.

5.3 Workplace Learning Opportunities

We used the term "Ironing" in our industry consultation to refer to initial WBL opportunities. The term refers to simple tasks that an apprentice can perform from day one which can only have a limited, isolated impact on day-to-day operations to manage risk - just as an apprentice tailor can be asked to iron fabric. In the initial survey employers were asked what tasks they were likely to assign as ironing to incoming apprentices. The most mentioned ironing practices are detailed here.

5.3.1 Testing. Testing was one of the activities most frequently cited by employers as being suitable for apprentices and other junior employees to take on. As one large financial services company suggests:

> I feel that we're starting to move more towards a world which we don't have segregated QA in development, it's all kind of intermingled. And I believe that Graduate Apprentices and early graduates fulfill this role quite nicely and it gives them the ability to grow up through the organisation and get exposure to existing systems and build test harnesses around them, and the like.

Another large technology firm goes so far as to suggest that testing roles are desirable for people new to the industry:

> In fact, that's a role that we find a lot of young people are attracted to these days. It's no longer seen as testing as being sort of like second-rate career. It's now seen as being up there with core software development. So being a test engineer is actually quite desirable for a lot of people.

It is clear from the varied responses to questions about testing in the consultation that approaches to testing vary from team-to-team in many organisations. One organisation described having "thousands and thousands of tests running with every single check-in" on some teams, whereas other teams have "not really got the automated testing there, so it is very heavily manual testing that's run there".

The most common reason for putting students on testing early was to give them a non-destructive task which will accelerate their understanding of a system.

> We found that even beginning to look at how we would automate our testing with things like Cucumber, having to go through that process is going, 'if this does this and then that does that'. Then it gives people a much better understanding of how the application works in the first instance.

Some employers specifically asked for testing to be taught upfront though one company then withdrew this after thinking that it might be too early, and another was specifically interested in the mindset of testing. Mindset was raised a few times, with the suggestion that developers are inherently bad at writing tests because they do not consider it important enough or do not enjoy doing it, so three companies mentioned that the importance of testing is conveyed to the students. Another organisation also mentioned that understanding testing results in cleaner code: "I get the feeling that [support, testing and documentation] are the things that the developers don't really like doing very much, but they're actually really quite key things."

Companies also suggested they would use testing as ironing exercises to introduce something new to the organisation, in one case implementing a new automated testing system and in another diverting control from external QA processes to internal staff.

Our primary take-away point from the discussion of testing as a workplace learning opportunity was two-pronged. Companies wished the university to provide an understanding of the fundamentals of testing along with a focus on instilling the correct attitude and mindset towards testing. Meanwhile, companies will find practical places for students to apply this knowledge and give them opportunities to demonstrate the worth of their theoretical basis.

5.3.2 Code Review and Bug Fixing. Bug fixing was a very common ironing exercise; however, a few organisations went on to qualify that they would be allowing this in a very specific environment early in the apprentices' careers. For example, the bug fixing would be very small changes in a non-critical system, essentially "locked off" from main development. Two other companies specifically said they would be interested in bug fixing because it will aid navigation and understanding of new systems.

> I think the onus is really around a little bit of bug-fixing, but I would hope, and I would rather the emphasis I'll be putting to their managers is that I want to see small isolated projects, things like a test harness, where you can, to a certain extent, say to them, there you go, there's a piece of work.

Refactoring was mentioned as a step parallel to or immediately following a period of bug fixing, with the expectation that after fixing simple bugs apprentices begin to get a feel for the process of improving code. Code review was also mentioned as a way to improve Graduate Apprentices' understanding of the code base, as were support tasks. The former was also useful to raise questions in areas where existing software developers may be stuck in their ways or focused on divergent issues.

> ...we found after about three months [of code review], even the ardent, you know, the dyed in the wool software developers ... were very surprised at what they could learn. So, what they could learn from these young whippersnappers. Because they were asking the stupid questions.

5.4 Ordering of Teaching

The final part of our second research question addresses what needs to be front-loaded to ensure apprentices are of use to the company as early as possible.

5.4.1 Programming Ability. The ability to program was considered a prerequisite by many employers. However, two employers stated that an understanding of programming theory was more important than knowing a specific language. Others sought apprentices with a "natural aptitude" for programming or a "personal interest" in the subject, rather than coding experience. Indeed, given the extensive range of languages used in industry, preparing students to program in the language and environment used by a workplace is impractical.

Instead, the way forward appears to be to drill down into programming concepts. Only a limited amount of time should be dedicated to learning specific languages, and more time should be spent on learning skills and knowledge required to pick up new languages quickly. This should prepare apprentices for the working environments they will encounter early on and also make them better suited to a rapidly changing workplace.

5.4.2 Professionalism, Responsibility and Ethics. As stated as a challenge above, a concern for companies is privacy, particularly given that apprentices will be actively encouraged to talk about their work in class. It is important to many companies that they know what they can and cannot talk about. Most indicate that they will be setting very clear boundaries for apprentices to follow and guidelines for what they are allowed to share and will expect them to stay within these parameters.

> "...when they join us [a large financial services company] they will go through a day and a half of initial training about: this is proprietary information, these are the things you can and can't share, and they are employees, so they'll understand that there are some things they can and can't share."

One company stated that they expect students to have "common sense" about these issues, tying human-centred security to their maturity. An adequate solution to dealing with sensitive data, then, would be to provide general rules for sharing company information at university, with specific instructions to discuss anything they are unsure about with their managers. Managers must be very clear about what students can and cannot share, and should express an interest in what students are likely to share in order to be aware of the potential risks.

Only one company explicitly mentioned client communication as an important professional skill, but it was noted that other companies intend on placing apprentices in client facing roles from early in their degree, so these skills would also be useful to them near the start of teaching.

Two organisations specifically mentioned a dedicated ethics course and noted it will be something they would hope to demonstrate in their working environment but will not be able to directly teach. Since Graduate Apprentices will need to show an awareness of ethics and related issues, such as workplace etiquette, from the outset, it is essential to cover these early.

5.4.3 Team Working. Team work was referred to by many companies, in a variety of contexts: development skills alone are simply not enough, and may be less important than team working skills: "...if somebody is quite introvert and quite closed and perhaps likes to work on their own then they're not going to succeed in [this organisation]."

The ability to interact and communicate with people is very important and not only in person but also over the phone and via email: "...if someone is really good at development, they still need to interact with folk, so it's just making sure that they're prepared to move into an environment where they're not necessarily just working on their own and that there are expectations from others."

University group projects only teach team work on a superficial level. Students could be taught to critique and assess how well their team is working: "...the amount of time that's spent operating as a team can be limited at university, because fundamentally it's an individual degree, so most of the time it'll be spent as an individual, whereas actually the workplace, the vast, vast majority of time is spent in a team."

Apprentices will be assessed in work on their ability to work as a team: "How engaged they are, because it's very collaborative here, so that would also be something that we would be kind of assessing them, is how well they work with other people." Since most Graduate Apprentices will be placed in a team from the first day of their apprenticeship, they need to be made aware of the basics of team communication and collaboration from the start.

6 CONCLUSION

Based on our industry consultation, it's clear that we still face many challenges in establishing WBL programmes in SE. However, this consultation highlighted the importance of communicating with employers about the issues involved, as we found that in some cases they subverted our expectations. They reacted overwhelmingly positively to some areas of teaching which we considered to be of less value to them as employers, and raised concerns about some of the content and proposed programme structure which had not previously been considered.

We do not consider this work to be complete. Our consultation, though far-reaching within Scotland, capturing the working practices and requirements of a dynamic range of companies, only captures the needs of Scottish employers. Requirements may differ in other parts of the UK and internationally. Acknowledging that this work is not applicable to everyone, we encourage many more institutions to conduct similar consultations. It is to be expected that many universities are aware of what industry consider important through collaborations and advisory committees, but we have found the experience of hearing so many voices to be instructive.

While we are, of course, examining our responses through the lens of an institution intending to launch a Graduate Apprenticeship programme, it is our expectation that this work will be of use to many other educators of Computing Science and Software Engineering at various levels. With the knowledge that a skills gap still exists between what universities are providing and what industry needs, closing this gap should be at the forefront of any educators' minds who wish to produce well-regarded graduates.

REFERENCES

[1] Alain Abran, James W. Moore, Pierre Bourque, Robert Dupuis, and Leonard L. Tripp. 2004. *Software engineering body of knowledge.* IEEE Computer Society, Los Alamitos, CA, USA.

[2] Bilal Amir and Paul Ralph. 2018. There is No Random Sampling in Software Engineering Research. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings (ICSE '18).* ACM, New York, NY, USA, 344–345. https://doi.org/10.1145/3183440.3195001

[3] Motoei Azuma, François Coallier, and Juan Garbajosa. 2003. How to apply the Bloom taxonomy to software engineering. In *Eleventh annual international workshop on Software Technology and Engineering Practice (ICSTE '03).* IEEE, 117–122. https://doi.org/10.1109/STEP.2003.13

[4] Uschi Backes-Gellner. 2014. *Benefits of apprenticeship training and future challenges: Empirical results and lessons from Switzerland and Germany.* Technical Report. Working Paper 97.

[5] Benjamin S Bloom. 1956. *Taxonomy of educational objectives, Handbook 1: Cognitive domain.* David McKay Co Inc, New York.

[6] Barry Boehm. 2006. A view of 20th and 21st century software engineering. In *Proceedings of the 28th international conference on Software engineering (ICSE '06).* ACM, New York, NY, USA, 12–29. https://doi.org/10.1145/1134285.1134288

[7] Pierre Dillenbourg. 2002. Over-scripting CSCL: The risks of blending collaborative learning with instructional design.

[8] Nicola K. Gale, Gemma Heath, Elaine Cameron, Sabina Rashid, and Sabi Redwood. 2013. Using the framework method for the analysis of qualitative data in multi-disciplinary health research. *BMC medical research methodology* 13 (Sep 2013), 117–117. https://doi.org/10.1186/1471-2288-13-117

[9] Orit Haller-Hayon. 2011. Learning by Sharing: Does it Improve Graduates' Preparation for the Working World? *International Journal of Interdisciplinary Social Sciences* 5, 9 (2011), 143–161.

[10] Stephanie Ludi and James Collofello. 2001. An analysis of the gap between the knowledge and skills learned in academic software engineering course projects and those required in real: projects. In *31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No. 01CH37193)*, Vol. 1. IEEE, T2D–8. https://doi.org/10.1109/FIE.2001.963881

[11] Belinda McLennan and Shay Keating. 2008. Work-integrated learning (WIL) in Australian universities: The challenges of mainstreaming WIL. In *ALTC NAGCAS National Symposium.* ALTC, Melbourne, Australia, 2–14.

[12] Kyung J Min, John K Jackman, and Douglas D Gemmill. 2013. Assessment and Evaluation of Objectives and Outcomes for Continuous Improvement of an Industrial Engineering Program. *International Journal of Engineering Education* 29, 2 (2013), 520.

[13] Emily Oh Navarro. 2005. A survey of software engineering educational delivery methods and associated learning theories. (2005).

[14] Andrew Powell. 2018. Apprenticeship statistics: England. (2018).

[15] Lisa Rowe, Daniel Moss, Neil Moore, and David Perrin. 2017. The challenges of managing degree apprentices in the workplace: a manager's perspective. *Journal of Work-Applied Management* 9, 2 (2017), 185–199.

[16] Lisa Rowe, David Perrin, and Tony Wall. 2016. The chartered manager degree apprenticeship: trials and tribulations. *Higher Education, Skills and Work-Based Learning* 6, 4 (2016), 357–369.

[17] Päivi Tynjälä and Päivi Häkkinen. 2005. E-learning at work: theoretical underpinnings and pedagogical challenges. *Journal of workplace learning* 17, 5/6 (2005), 318–336.