On The Supervision and Assessment Of Part-Time Postgraduate Software Engineering Projects

Andrew Simpson, Andrew Martin, Jeremy Gibbons, Jim Davies, and Steve McKeever
Software Engineering Programme
Wolfson Building, Parks Road, Oxford OX1 3QD, United Kingdom

Abstract

This paper describes existing practices in the supervision and assessment of projects undertaken by part-time, postgraduate students in Software Engineering. It considers this aspect of the learning experience, and the educational issues raised, in the context of existing literature—much of which is focussed upon the experience of full-time, undergraduate students. The importance of these issues will increase with the popularity of part-time study at a postgraduate level; the paper presents a set of guidelines for project supervision and assessment.

1. Introduction

In [25] a lack of Computer Science-specific literature concerning projects is noted. This has started to be addressed within the community. For example, the United Kingdom's Effective Projectwork in Computer Science (EPCoS) consortium [30, 17] and the collection of papers of [22] represent positive moves in this respect. Internationally, journals such as *Computer Science Education* have also published many articles on the subject.

A significant proportion of this literature, as one would expect, is primarily concerned with projects undertaken by full-time undergraduates. However, the literature does step outside the boundaries of the traditional 'design-build-report'-style project module: some projects are run with an industrial partner [1, 13, 23], while others involve international collaboration [7, 27].

No matter what form projects take, efforts should be directed towards evaluating their role in the Computer Science curriculum [11]. This is done in [17], where a collection of examples of good practice for various forms of computing project are presented: allocation methods, supervision techniques and assessment methods are detailed for industrial-based projects, taught MSc courses, and so on. Our specific concern—projects undertaken by part-time postgraduate Software Engineering students—is not represented there. This paper has been written in the spirit of that collection. We pay particular attention to the topics of intellectual property and confidentiality, as this is perhaps more of a concern for our students (and their employers) than it is for full-time undergraduate students.

The structure of the remainder of this paper is as follows. In Section 2 we describe the University of Oxford's Software Engineering Programme, and outline the context in which the projects undertaken by the Programme's students are supervised and assessed. In Section 3 we relate our experiences of supervising and assessing projects to the published literature. In particular, we consider the six themes identified in [17]: *allocation*, *supervision*, *assessment*, *reflection*, *team or group projects*, and *motivation*. In Section 4 we offer some guidelines for the supervision of part-time postgraduate Software Engineering projects that may be of benefit to similar programmes. Finally, we summarise the contribution of this paper, and indicate areas for future work.

2. Software Engineering at the University of Oxford

Both in the United Kingdom and in the United States there is, on the one hand, the need for a highly-skilled work force, and, on the other hand, there is a decline in public support for higher education [4]. In addition, as more young people enter higher education in the United Kingdom, the possession of a postgraduate degree is becoming a more significant factor in the job market—at the same time as the level of funding for postgraduate taught courses is being reduced. This pattern is being replicated worldwide.

In [15], several themes that characterise higher education's response to these problems are identified. Two of these themes—"a shift from faculty-centered to learner-centered institutions" and "lifelong learning"—are at the heart of the Software Engineering Programme at the University of Oxford, the philosophy of which echoes the sentiments of [33], in which a vision for software engineering education and training for the twenty-first century is outlined.

There has also been a shift in higher education's mission to include three additional goals: providing knowledge to the workforce, retooling people for new careers, and catering to the need for mental simulation [2]. Universities are now being asked to consider what it means to be a graduate [20] and to debate the transferable skills that such graduates should possess [18].

The Software Engineering Programme at the University of Oxford[1] is a joint venture between Oxford University Computing Laboratory and the University's Department for Continuing Education, which has the aim of making the educational resources of the University of Oxford more accessible to a wider public. The Programme offers a number of short courses—which may be taken individually, combined to form short programmes of study, or used as credit towards a postgraduate qualification—that covers a wide range of software engineering topics.

At any one time, approximately 200 students are registered for a part-time qualification: either a Postgraduate Certificate, a Postgraduate Diploma, an MSc, or a doctorate in Software Engineering (although we do not concern ourselves with doctoral students in this paper). These students—many of whom come from overseas—attend intensive, one-week courses at the Programme's own teaching facility. The Programme currently has a core staff of eight: five university lecturers, supported by an administrative staff of three.

The Programme allows students to register initially for one of three qualifications: a Postgraduate Certificate, a Postgraduate Diploma, or an MSc. It is only for the latter of these that the completion of a project is required (although, in previous years, this was also a requirement for the Diploma). Students on the MSc undertake a project carrying about a quarter of the weight of the degree; the dissertation has a 20,000 word limit. The project typically involves application of techniques taught on the Programme to a real-world problem or product of the student's employer. The project is supervised by one of the Programme lecturers, and graded by two people: one of the Programme lecturers other than the supervisor, and one of a committee of three examiners.

As well as allowing students to move from registration for a lower qualification to a higher one, the Programme also allows students to move back down the ladder: many part-time students find themselves in the position of being unable to make sufficient time to undertake the research associated with a project or to write up their findings (or, for that matter, both)—even though they have completed (and passed) the required number of taught courses. It would be a pity for such students to leave the Programme without an award of any sort; hence, the possibility of transferring to a lower award.

In this mode of delivery, the Programme is relatively novel, though by no means unique. For example, the Safety Critical Systems Engineering and System Safety Engineering[2] courses run by the University of York are both organised in a similar way. The University of Kansas also offers a part-time MSc in Software Engineering aimed at students from industry.[3] In addition, as is noted in [16], some institutes have gone further and are offering the electronic delivery of such courses.

3. Software Engineering projects: theory and practice

In [14], it is argued that the Graduate Engineer must be capable of demonstrating practical competence in action. This is the fundamental basis for including projects in the Computer Science curriculum. Furthermore, it is a requirement for the Programme's accreditation by the British Computer Society (BCS) [4] that a project component be included in the MSc. However, we feel that the project aspect of our degree is significantly different to the traditional project aspect—in terms of, for example, its role in the curriculum, the demands on supervision, the nature of topics undertaken, and the nature of assessment. We discuss such issues in this section.

The role of project work In [3], the view that the role of project work can be unclear is presented: students enter it with expectations that are often based upon ill-defined preparation, resulting in a preoccupation with technical issues [32]; prospective research students may be exercised in areas that they might later pursue [31]; and so on. Such issues are of little relevance to our students, as the role of project work for them is very clear. They are now in a position to use, or evaluate, these techniques with respect to problems that they face within their working environment; this is the role played by project work within our Programme. However, there are—of course—exceptions. For example, a student who works on a test team for an industrial strength Java Virtual Machine implementation must be stretched in different ways.

Industry relevance In the United Kingdom, criticisms have been made of the preparedness of graduates for entering the 'real world': the Dearing Report [12] and the UK Engineering Council's SARTOR (Standards and Routes to Registration) Report [34] have both made this point forcefully in recent years. As a result, some admirable efforts have been made to make projects of full-time Computer Science students more relevant to industry. For example, [35] describes the use of 'University based industrial projects' to provide full-time students with the advantages of industrial contacts,

[1] www.softeng.ox.ac.uk

[2] www.cs.york.ac.uk/MSc/SCSE

[3] www.eecs.ku.edu

[4] www.bcs.org.uk

while [21] describes an MSc course involving a real client and a professional software project manager. Another example is described in [36], which reports on "classroom experiences in software engineering coursework where students are placed in an industrial environment and given a real customer, a real project, and held to commercial practices and accountability." Such efforts are not, of course, restricted to the United Kingdom. Other examples include [19], which describes experiences at Monash University, and [10], which details experiences at Uppsala University of project-oriented courses in which the courses are chosen to be similar to genuine industrial projects. The projects undertaken by our students are, by their very nature, relevant to industry. Indeed, the projects have the added bonus of determining the industrial relevance of the techniques and methods that they have been taught as part of the Programme: this is an extremely valuable means of validating the material taught. Our students, however, face slightly different problems here. For example, a company whose student has submitted a project that includes a detailed critique of their internal practices risks losing a lot if such sensitive information were to leak out.

Supervisor influence In [6], the view that "individual students' experience of project work will depend heavily on the attitudes and beliefs of their supervisor" is given. The projects undertaken by our students are very much work-oriented, and, therefore, the potential impact of supervisors' attitudes and beliefs is restricted—and, at times, restricted to workload planning.

Technical challenges Many full-time students, at the end of their course of study, may have to demonstrate a significant amount of technical competence simply to use the development environment that they have chosen. As far as most of our students are concerned, they demonstrate this every day of their working lives. Therefore, the challenges presented by completing a project as part of their course are different to those of full-time students: work and family pressures have a far greater impact than the technical challenges presented to them.

Learning objectives The learning objectives for project work can "encompass the exercise and development of a wide range of skills varying from generic personal transferable skills, such as communication and teamwork, to technical skills such as the ability to design, construct and deliver a system meeting stated requirements . . . The required deliverables should reflect and reinforce the learning objectives" [5]. The impact of the learning outcomes can be increased if the objectives are stated clearly and explicitly at the start of the project process—something that we endeavour to do.

Allocation As with undergraduate courses, the Software Engineering Programme allows a degree of choice with regards to the selection of MSc project topics. There are, however, two main aspects that determine this choice. First, the project is seen as a culmination of the students' course of study. As such, the project is expected to involve the application of some of the techniques learnt during their studies. The specific problem to be solved is the second aspect that determines choice—it is typically expected that the problem area will be one associated with the students' workplace. It follows that the 'assignment problem' identified in [9], which requires the "variation of a classical approach to the Assignment Problem" [29] is not a concern here.

Supervision Due to the very nature of the Programme's students, the form of supervision taken is unusual. Supervision meetings are more infrequent than is the case for full-time undergraduate students. This is mainly due to the fact that the Programme's students are not (typically) located in Oxford. There are, of course, extremes: one student may visit every couple of weeks; another may visit right at the start of their project work and never be seen again. When students do make the trip to see their supervisor, this means that such meetings are longer and—arguably—more productive than is the case for full-time students. Another result of this distance is that regular email contact is far more important than is the case for full-time students.

Assessment A number of authors have discussed the concept of assessment authenticity (see, for example, [28], [8], and [37]). As noted in [17], there are many forms of such assessment. The fundamental basis for assessment of projects submitted by Programme students is the level to which they have demonstrated skill at understanding and applying the techniques taught.

Reflection To quote [17], "reflection on experience underpins the process of successful learning and is essential to the success of education." Nowhere is this more true than in project work, where, typically, this is the main opportunity for such activity. Our students are at an advantage in this regard: because the project is the culmination of their course of study (and also because of the nature of the project being undertaken), the reflection process is more widespread: there is the opportunity for reflection on the benefits of the whole course, as well as on the project work undertaken.

Team or group projects The projects undertaken by our students are all individual projects: given that there is no set start or end date (or duration), that the students are geographically distributed, and some projects may involve commercially sensitive material, it cannot be otherwise. That is

not to say, however, that we do not value the roles of team or group projects: the final module undertaken by all students prior to them starting on their project work is a compulsory module in which students work in groups to specify, design, implement, and test a system. In addition, some students report on work they have individually carried out within a team context in their place of work, with the other members of the team being unconnected with the Programme.

Motivation In [24], four particular points in the learning cycle at which motivation may be addressed are listed: it can be affected by their *interest* in their work; their perception of its *relevance*; their *expectation* that they will succeed; and their *satisfaction* in their achievement. This has a neat fit with the philosophy of the Software Engineering Programme and—in particular—the project work undertaken.

Planning As detailed in [17], work schedules stretching over weeks or months are new to most full-time students: they should be encouraged to produce a time plan to enable both the student and supervisor to monitor progress. Most part-time students *are* used to generating time plans: typically the drive to produce a time plan comes from the student rather than the supervisor.

4. Guidelines

In this section we present some guidelines (in the spirit of [17]) that can be employed with regards to part-time postgraduate Software Engineering projects.

Understand the student's subject area The process of choosing of subject area for a student's project is significantly different than it is when dealing with a full-time undergraduate Computer Science student. Full-time undergraduates choose their project topics for a variety of reasons: they are keen on the subject area and want to explore it further; they want to work with a particular member of staff; or maybe they are assigned a project by a 'project supremo'. Part-time postgraduate students go through a very different process in their choice of project area. Typically, they will want to extend or reanalyse something associated with their everyday work (indeed, the Software Engineering Programme at Oxford encourages this)—the choice then becomes very simple. The difference here is that there is unlikely to be a supervisor who is familiar with the project, working practice, or work place in which the project is to be embedded—thus, there is a learning curve for the supervisor as well as the student. This brings us to the second guideline.

Understand that the supervisor-student relationship is genuinely a two-way relationship The supervision of students who actually work in the industry results in a flow of information is genuinely two-way. For example, part-time students can be the source of potential (applied) research ideas for their supervisors. The relationship between part-time students and their supervisors is unlike that between full-time students and their supervisors. In the latter, there is nearly always a huge gulf in age, experience, status, and authority; with regards to the former, it is more of a relationship between peers, with each party bringing different contributions to the table.

Understand the pressures faced by the student Full-time students typically have time pressures with regards to project submission: increasingly such students have to finance their studies through part-time (and, in some cases, full-time) jobs. Part-time students also have time pressures, but of a slightly different variety. Full-time students (usually) have a set date by which they should submit their project; part-time students (again, usually) do not. Without a firm target, real life gets in the way and progress is slow: therefore a realistic work plan with plenty of slack and a fixed submission date are essential.

Be realistic with regards to assessment Part-time students are not immersed in an academic culture, day-in, day-out. They do not have easy access to the university library—although they can access Web resources easily; they do not have easy access to their supervisor—although email contact is easy to establish. Most part-time software engineering students cannot be expected to produce research-based projects; they cannot be expected to work with recent research results; they cannot be expected to work with a supervisor in his or her specialist area. They can, however, be expected to demonstrate, apply, and evaluate what they have learnt during the taught part of the course. This must be the basis for the assessment of such projects.

Provide both formative and summative feedback We have found that our students have a strong desire for summative feedback: they feel that they have invested a tremendous amount of time and energy into their project work, and wish to know on what basis the grade awarded to them was arrived at. However, formative feedback is also a vital part of the project process. The students have gone through a process of employing techniques taught during their study to genuine problems in their workplace: many of these students wish to carry on doing so, and to be able to exploit this new knowledge successfully. In this regard, a final bit of constructive advice can be valuable.

Confidentiality The very nature of the projects undertaken by our students means that the issue of confidentiality is of

concern, and so there is a responsibility on the Programme to ensure that—for some projects—only approved individuals (such as, for example, the supervisor and examiners) may have access to the final project.

As students are encouraged to undertake projects which are directly connected with their work, the majority are under a duty of confidentiality to their employer, and most are well aware of this. As a result, projects are undertaken within the confines of a non-disclosure agreement. The Programme has a pro-forma agreement to which most companies are happy to give their consent. This covers discussion with lecturers and examiners, and allows the project to be kept on very restricted access in the library. This, of course, has a negative impact on the pool of previous projects that can be made available for current students to read.

Intellectual property The University's normal practice is to place the projects in a library after they have been examined, and the University claims intellectual property rights in (among other things) software developed *in the course of, or incidentally to* the student's studies, if it possesses commercial potential.

Intellectual property raises a wider range of issues. The University's statutes are so phrased in order to protect it against exploitation by others of ideas substantially developed by its staff. In the case of full-time students this may be a significant concern. Part-time students, by the nature of their studies, have less interaction with University staff, and are much less likely therefore to discover a valuable idea with their aid. The delineation of work related or incidental to their studies, and work undertaken for their employer is often impossible; the two are intertwined.

Most students appear unaware of such concerns—consent to the University statutes is a small matter when they register for a degree. For very few will it be remotely of interest, the majority of projects generating little by way of valuable (or directly-exploitable) intellectual property. For many, however, if the issue is raised it will generate concern, even if they fall into this group.

Of course, where the student is in employment, they will have already (probably) assigned intellectual property rights to their employer, and so at the point of registering for a degree are not able to give any such rights to the University.

It is frequently necessary to advise a student to construct their project in such a way as to avoid placing interesting and/or ground-breaking developments within the scope of the project. For example, one student is developing a framework to enable companies in a particular sector to construct e-commerce solutions with particular ease. Rather than describe this work as an MSc project, and gain credit for doing so, they find themselves writing instead a project describing an application of the framework, developed alongside it in order to validate and refine its definition. Another student has developed a tool which will substantially aid their company in maintenance and development of a class of product. This student might happily submit a description of that tool for their MSc without fear of the company's rights being damaged. These two examples represent extremes. The inequity between students due to their employment situation with regard to intellectual property is seldom great, and very seldom likely to affect the outcome of the student's degree. Very few MSc projects develop intellectual property of significant value; the situation is nevertheless far from satisfactory.

5. Summary

In [17] a collection of examples of good practice are produced to aid potential supervisors of various forms of Computing projects. Our specific concern—projects undertaken by part-time postgraduate Software Engineering students—is not represented there. As a consequence, this paper has been written in the spirit of that collection. We paid particular emphasis to intellectual property and confidentiality because it is of considerable relevance to the projects undertaken by our students.

The role of projects within Computing curricula is very much an accepted part of such degree programmes—mainly because it helps prepare students for life in the 'real world'. Although the benefits that come from including a project component as part of our degree are very different to those associated with full-time undergraduate projects, the benefits are very real—for both students and supervisors.

In [26] four stakeholders in project work are identified: academic staff, students (before the work), students (after the work), and employers. In this paper we have focussed on the views of one of these stakeholders—the academic staff. We shall focus on the views of the others in future papers. Furthermore, it may be that many of the themes identified in this paper are relevant to similar courses in other fields—this shall also be a subject for future work.

References

[1] C. Bergman. Senior design projects with industry. In *Proceedings of FIE'98*. IEEE, November 1998.

[2] H. Blustain, P. Goldstein, and G. Lozier. Assessing the new competitive landscape. In R. N. Katz and associates, editors, *Dancing With The Devil: Information Technology And The New Competition In Higher Education*, pages 51–72. Jossey-Bass inc., 1999.

[3] R. D. Boyle and M. A. C. Clark. Non-technical issues in undergraduate CS project work or what are we (all) here for? In M. Holcombe, A. Stratton, S. Fincher, and G. Griffiths, editors, *Projects in the Computing Curriculum: Proceedings of the Project '98 Workshop*, pages 117–128. Springer, 1998.

[4] D. W. Breneman, J. E. Finney, and B. M. Roherty. Shaping The Future: Higher Education Finances In The 1990s. San Jose: California Higher Education Policy Centre, 1997.

[5] P. Capon. Maximizing learning outcomes of computer science projects. *Computer Science Education*, 9(3):184–199, 1999.

[6] M. A. C. Clark and R. D. Boyle. A personal theory of teaching computing through final year projects. *Computer Science Education*, 9(3):200–214, 1999.

[7] T. Clear. A collaborative learning trial between New Zealand and Sweden using Lotus Notes Domino in teaching the concepts of human computer interaction. In *Proceedings of ACM Innovation and Technology in Computer Science Education*, 1999.

[8] A. Collins. Portfolios for science education: issues in purpose, structure and authenticity. *Science Education*, 76:451–463, 1992.

[9] R. Cooper and R. Welland. Computing science projects at the University of Glasgow. In M. Holcombe, A. Stratton, S. Fincher, and G. Griffiths, editors, *Projects in the Computing Curriculum: Proceedings of the Project '98 Workshop*, pages 169–183. Springer, 1998.

[10] M. Daniels and L. Asplund. Full scale industrial project work: a one semester course. In *IEEE Frontiers in Education, San Juan, Costa Rica*, November 1999.

[11] M. Daniels, A. Berglund, and M. Petre. Some thoughts on international projects in the undergraduate education. In *Project 99, EPCoS, Exeter, UK*, September 1999.

[12] Report Of The National Committee On Higher Education. HM Government, HM Stationary Office, 1997.

[13] D. Dekker. Issues when using company sponsored projects to provide a design experience for students. In *Proceedings of FIE'97*. IEEE, November 1997.

[14] P. J. Denning. Educating a new engineer. *Communications of the ACM*, 35(12):82–97, December 1992.

[15] J. J. Duderstadt. Can colleges and universities survive in the information age? In R. N. Katz and associates, editors, *Dancing With The Devil: Information Technology And The New Competition In Higher Education*, pages 1–26. Jossey-Bass inc., 1999.

[16] G. C. Farrington. The new technologies and the future of residential undergraduate education. In R. N. Katz and associates, editors, *Dancing With The Devil: Information Technology And The New Competition In Higher Education*, pages 73–94. Jossey-Bass inc., 1999.

[17] S. Fincher, M. Petre, and M. Clark, editors. *Computer Science Project Work: Principles and Pragmatics*. Springer, 2001.

[18] A. Griffin. Transferring learning in higher education: problems and possibilities. In R. Barnett, editor, *Academic Community: Discourse or Discord?* Jessica Kingsley, 1992.

[19] D. Hagan, S. Tucker, and J. Ceddia. Industrial experience projects: A balance of process and product. *Computer Science Education*, 9(3):215–229, 1999.

[20] HEQC. Assessment In Higher Education And The Role Of 'Graduateness'. Higher Education Quality Council, London, 1997.

[21] M. Holcombe and H. H. Lafferty. Using computer professionals for managing student software projects. In D. Bateman and T. Hopkins, editors, *Proceedings of the Conference on Developments in the Teaching of Computer Science*. University of Kent at Canterbury, 1992.

[22] M. Holcombe, A. Stratton, S. Fincher, and G. Griffiths, editors. *Projects in the Computing Curriculum: Proceedings of the Project '98 Workshop*. Springer, 1998.

[23] A. Jackson. An industry-centered capstone experience for aeronautical management technology students at Arizona State University. In *Proceedings of FIE'98*. IEEE, November 1998.

[24] J. M. Keller. Motivational design of instruction. In C. M. Reigeluth, editor, *Instructional-Design Theories and Models*. Lawrence Erlbaum Associates, 1983.

[25] M. Knudsen and T. Vinther, editors. *Project Work in University Studies*. Roskilde University, Denmark, September 1997.

[26] M. R. Luck. Undergraduate research projects — why bother? In *EPCoS workshop, Leeds, UK*, January 1998.

[27] T. Macek, B. Mannova, J. Kolar, and B. Williams. Global cooperation project in computer programming course. In *Proceedings of SIGCSE'99*. ACM Press, March 1999.

[28] R. Mitchell. What is 'authentic assessment'? Portfolio: The Newsletter of Arts PROPEL, Harvard University, December 1989.

[29] C. H. Papdimitrou and K. Sterglitz. *Combinatorial Optimization*. Prentice Hall, 1982.

[30] M. Petre and S. Fincher. Using other people's experience of project work: realising fitness for purpose. In M. Holcombe, A. Stratton, S. Fincher, and G. Griffiths, editors, *Projects in the Computing Curriculum: Proceedings of the Project '98 Workshop*, pages 19–30. Springer, 1998.

[31] S. Rowett. Experience of a student-led final year degree project. In M. Boyle, editor, *Student led Projects at the University of Leeds*. University of Leeds, 1995.

[32] J. Ryder and J. Leach. Research projects in the UG science course: students learning about science through enculturation. In G. Gibbs, editor, *Improving Student Learning Through Course Design*. Oxford Centre for Staff Learning and Development, 1997.

[33] H. Saiedian. Software engineering education and training for the next millennium. *Journal of Systems and Software*, 47(12), December 1999.

[34] Standards and routes to registration (SARTOR). Engineering Council, 1997.

[35] A. Stratton, M. Holcombe, and P. Croll. Improving the quality of software engineering courses through university based industrial projects. In M. Holcombe, A. Stratton, S. Fincher, and G. Griffiths, editors, *Projects in the Computing Curriculum: Proceedings of the Project '98 Workshop*, pages 47–69. Springer, 1998.

[36] J. B. Vaughn jr. Teaching industrial practices in an undergraduate software engineering course. *Computer Science Education*, 11(1):21–32, 2001.

[37] G. Wiggins. A true test: toward more authentic and equitable assessment. *Phi Delta Kappan*, pages 703–713, 1989.