Adopting Commercially Inspired Practices within an Academic Teaching Course: A Case Study of a Computer Games Engineering Degree

G. Ushaw[1], W. Blewitt [1] and G. Morgan [1]
[1]*School of Computing Science, Newcastle University, UK*
gary.ushaw, w.f.blewitt, graham.morgan@newcastle.ac.uk

Keywords: Software Engineering, Project Based Learning, Engineering Education

Abstract: A case study of a computer games engineering course is presented. The course has been designed with close input from industry and is achieving a high rate of success in the number of graduates being recruited by the target industry. The organisers of the course have extensive experience in both the software engineering industry, and in delivering academic teaching. These experiences are combined so that commercial software development practices, technologies and philosophies are adopted throughout the delivery of the academic course. The paper discusses the specifics of how and why this was achieved, and uses the Game Engineering course as an exemplar for encouraging the adoption of commercially inspired techniques within the teaching of software engineering and computer science more generally.

1 INTRODUCTION

Learning to program effectively is recognised as one of the most difficult tasks facing a prospective software engineer or computer scientist (Jiménez and Villalobos, 2010; Corney et al., 2010) (it has been claimed that a novice programmer may take ten years to become an expert (Winslow, 1996)). Teachers must regularly update and redesign courses on this topic to make the learning process more accessible, and to maintain relevance to the latest developments in the field. Further to this, the software industry has high expectations of specific skills in graduate software engineers. Some of these expected skills are specifically technical, while others are related to good working and engineering practices.

Teaching of the specific technical aspects of programming is a challenging prospect, that can be researched, defined and updated to meet both academic and commercial needs. The aspects related to the practicalities of working within a larger software engineering environment are more difficult to define and present interesting challenges to teaching within an academic environment. It has been widely noted that there is unease from industry on the practicalities of what is taught on university courses (Perkmann et al., 2011; Baekkelund, 2006), and a general acceptance that new graduates need varying amounts of specific training after joining a software development company. Anecdotally, academics can be seen as existing in their ivory towers with little interest in the practicalities of industry, while the commercial sector is at the coal-face with little interest in seeing what is beyond the most immediate challenges. Of course, these are wild exaggerations.

The authors of this paper have a combined background in both the software industry and academia, specifically in the field of computer game engineering. We propose that a more fruitful relationship between academia and industry can be fostered by adopting and promoting aspects of working methodologies and philosophies from industry within the academic course framework. This, coupled with the utilisation of technologies, equipment and software tools which are industry standard, should result in graduates who are better prepared for a career in software engineering, and are perceived as being more employable by their target industry.

In this paper we present a case study of a Computer Games Engineering Masters course which has been running for over five years, and enjoys high levels of support from the games development industry. This has been achieved by working closely with prominent members of the industry, and by combining commercial working practices and philosophy with the pedagogical demands of an advanced course on software engineering. The use of commercially inspired practices within the academic teaching has led to a success rate of over 70% of graduates gaining employment in the target industry within six months

of graduation (this compares to an average success rate of around 12% from game development courses across the UK (Livingstone and Hope, 2011)). The Games Engineering course is used as a case study, but the general points made are applicable to the wider fields of computer science and beyond.

2 METHOD

In designing the course in a way that incorporates commercially inspired practices, while maintaining pedagogical standards, a number of different elements were addressed. In this section, each of those elements is described in turn, focusing on the details of how and why each aspect was implemented.

2.1 Industrial Involvement

The most important way in which we have ensured our course incorporates commercially inspired ideas is to directly involve industry in the course. Industry involvement with the course is at the core, curriculum level, and not limited to mere endorsements. These industrialists are an integral part of the degree. From helping to identify course content to providing jobs for graduates, industry is involved at every stage of the course.

2.1.1 Industrial Advisory Board

The course is guided by an industrial advisory board. The board consists of experienced software engineers from industry who meet once a year to review course content. When inviting a developer to join the board, we focus on technical people (i.e. lead programmers and technical directors rather than managers or academic liaison), as the course content is heavily technical.

Commercial software developers are very busy people. When dealing with our board members, and especially when approaching a potential new member, a great deal of care is taken in respecting their crammed timetables and priorities. In the short term, the industrial advisers are providing a favour for the course; any benefit for them is much longer term (i.e. more employable graduates). Keeping commercial representatives from industry involved in the course is time consuming and requires significant effort, but the long term benefits for the course justify the effort required.

The presence of the industrial advisory board clearly states two things about the course. Firstly, it states to potential students that this is a course which is taken seriously by the target industry, increasing the chance of employment on graduation, and therefore encourages them to engage with the academic content. Secondly it states to members of that industry that the directors of the course take their advice and opinions seriously, crafting graduates in a manner which is beneficial to them.

2.1.2 Industrial Seminar Series

A series of seminars are presented by the members of the industrial advisory board to the student class (the seminar is optional for the industrialists, but most contribute). These seminars are typically on either a technical subject or an organisational one, and are designed to allow representatives of industry direct interaction with the student cohort. The speakers can use the seminars as a window to directly inform the students about an aspect of the subject which they feel is especially relevant.

Further to this, a session immediately after each seminar provides a more informal feedback opportunity between the speaker (often a potential employer) and the students. This session is used for further Q&A, and for the industry speaker to see the students' work and provide comment and feedback on it. One cannot overstate the impact that leading video game developers have on students. Video games is a creative industry and many developers are "famous". A faculty member indicating that a piece of work is good is nothing compared to an industry expert praising a student's work.

2.1.3 Industrial Placement

Credit for the second half of the Masters course is achieved from an individual project and dissertation. It is strongly encouraged that this project work takes place within a commercial software development studio if possible and practical. We have found that our industrial partners are very likely to take on one or more student to work on a project within their organisation. The combination of showing that we can evolve the course content in light of their input through the industrial advisory board, and the face-to-face interaction that we provide through the industrial seminar series, leads to an increased level of trust and interest in taking on our project students.

We decided early in constructing the course that any paperwork associated with an industrial placement should be kept to an absolute minimum for the industrial partner. This applies equally to any bureaucracy pertaining to the logistics of the student working away from the faculty, and to any assessment of the student's work from an academic perspective. In

our experience, reassuring a potential provider of a student placement that there is minimal red tape involved yields more interest in providing a placement. The student project must be subject to scrutiny and assessment for academic purposes, but this is achieved through direct dialogue with the industrial supervisor rather than supplying forms to be completed.

Building fruitful two-way relationships with industrial parties is not an easy task. The authors have many years of experience working within the target industry and, consequently, a wide network of contacts which can provide a rich source of collaboration. However these contacts are not taken for granted, and we are constantly aware that our relationships now, as academics, are different to what they were as industrialists. It should be apparent from this section that we take great care to treat industrialists with respect and to demonstrate that we understand their priorities, while ensuring the provision of an academically rigorous programme of study.

2.2 Hardware and Tools

A key feature of the case study course is that the hardware and software tools employed by the students during practical sessions are equivalent to those used by the industry. A further key feature is that the use of these tools is focussed on education rather than training - ii.e. the students are taught why the tool is useful and how it provides that utility, rather than a more straightforward guide to how to use a particular tool. The intention is that this approach results in graduates who can take on a new tool with confidence as they understand what it is for, and how it relates to the tools with which they are already familiar.

A simple example is the use of source control for all practical work. Any commercial software developer employs a source control system, and there are many professional packages available. From the first day of our course we issue each student with a source control account, and instruction on how and, more importantly, why to use it. This focus on the benefits of using source control, and the mechanism by which it is achieved, ensures that the choice of which professional source control package to use during the modules is almost irrelevant. When entering industry, the graduate will expect to use source control as part of the daily programming routine, and will understand and appreciate the reasons for doing so; the specifics of the particular solution will then be picked up within a few hours of use.

Combining the use of industry-standard tools with discussion of why they are useful, and how they work for the developer (rather than focussing on how the developer interacts with the specific tool) extends to all aspects of programming during our course. This includes the hardware itself (in our case games consoles, high-end PCs, tablets and smart-phones), and the development environment (compilers, debuggers, SDKs, IDEs, etc). The intention is that, by educating the student about why a tool is useful, and how it works, the resulting graduate will be much more comfortable moving onto new development environments or hardware, due to a better understanding of what the new technology is actually doing.

2.3 Collaboration not Competition

Working within a commercial enterprise is a collaborative process, often involving work with developers elsewhere on the planet (Matthes et al., 2011). In the authors' experience, the majority of students regard coursework and practical sessions as a competition with their classmates. In designing our case study M.Sc. course we have encouraged a collaborative atmosphere throughout. This applies to the nature and openness of the discussion and tutorial sessions, and the layout of the desks in the practical teaching environment (desks should face each other, rather than face a wall, to encourage student interaction). Each student has a workstation within the lab, which is where the entire day is spent, as would be the case in a commercial software development organisation. Lectures take place within this space (i.e. the teachers come to the students, rather than the students go to each lecturer's theatre). Students are encouraged to carry out all practical work in the lab, rather than taking it home. This approach is intended to imbue the class with a sense of belonging (i.e. it is *their* lab), and give the environment the feeling of a shared workplace.

It is important to note that the collaborative experience is not limited to the aspects of the course which include a team project. We encourage discussion and problem solving between students during individual coursework and practical sessions. While working within a commercial software development enterprise, one of the main tools available to a software engineer is the developer at the next desk: many problems can be resolved by a brief discussion with a colleague.

Team projects are a feature of most computer science degree courses, and are regarded as a crucial step toward becoming a fully-developed software engineer (Hilburn and Humphrey, 2002). While working in the software industry, the authors interviewed over one hundred graduate programmers from many educational institutions over a number of years. Al-

most without exception, when questioned about a team project, the response involved a claim that the interviewee did almost all of the work, and the rest of the team did almost nothing (indeed, the author stopped asking about team projects in interviews as this repeated response rendered it pointless). By encouraging a collaborative atmosphere throughout the course, we hope to achieve a better group dynamic during the parts of the course involving teamwork, and a more positive experience of how teams should work together toward a common goal, which should then be apparent in subsequent job interviews.

Encouraging collaboration may also encourage plagiarism. Careful design of coursework, and clarity on which aspects gain marks, mitigates this issue. As discussed later in this paper, coursework is defined in a purposefully open-ended way so that each student can make decisions which test their own individual ability. The coursework does not have a right or wrong set of results, but consists of a framework in which to develop many related ideas. Consequently no two pieces of student coursework should be similar; if similarities are apparent then a code inspection and comparison should help resolve any potential plagiarism issues.

When the team project commences, the students are already familiar with source control. Using a shared code-base for the team project provides additional experience with standard working practices, while also inculcating effective teamwork and intra-team learning. A less gifted student may learn from changes made to the project by more gifted students. Conversely the more advanced student may realise that code generated for a team must be straightforward to interface with, and clearly structured and commented for other team members to understand and debug.

2.4 Increasing Autonomy for Students

Many studies show that encouraging student autonomy leads to greater engagement with the subject (Boud, 1988; Reeve et al., 1999; Reeve et al., 2004). Our case study course has been designed in such a way that the student cohort is given increased autonomy in the nature and direction of their study and coursework as the modules progress. Preparing a student for the workplace entails instilling a degree of self-reliance and the ability to move forward from one task to another without direct instruction. Note that encouraging both autonomy and collaboration is not contradictory; a well-rounded software engineer knows when other members of a team can be useful in progressing a problem or decision (i.e. the engineer can autonomously decide to collaborate).

Our case study course in Computer Game Engineering consists of three phases, each allowing the students more autonomy than the last. The first phase consists of sequences of lectures stripped daily across each week of the modules, combined with practical sessions and tutorials directly related to the lecture topics. These modules provide the basis of knowledge and ability hat is required across the remainder of the course. The tutorial and practical work is heavily prescribed, with direct instruction on how to achieve success (further exercises and suggestions are supplied, to push the more gifted or committed students). The modules are assessed by a combination of written closed-book examinations, and practical coursework. The coursework definitions become more open-ended as the modules in the first phase progress, allowing for more varied features to be added, and individual research to be rewarded, but the work required to achieve a pass mark is clearly prescribed.

The major element of the second phase of the course is the team project. The teams are allocated by the course supervisor so that each includes students with a full range of ability. The definition of the team project provides a specific set of goals, but is not specific on how they are attained; it also provides a high-level design for the product which is to be developed, but instructs the team to design the detail themselves. This approach encourages a combination of teamwork and autonomy. The course supervisor is available throughout the week on an informal basis, and reviews progress more formally (offering advice where required) on a weekly basis. The assessment of this phase involves no written examinations, but includes a demonstration of the finished product, technical reviews of the implementation, and evaluation of how the team interacted. There is also an individual research paper required during this phase - the subject is prescribed in the definition, and training is provided in researching a subject and presenting the work in the form of a conference paper.

The final term of the course consists of an individual project and dissertation for each student who has successfully progressed from the first semester. This work is self-led research over a period of five months. By this stage, the students are working completely autonomously, with a fairly informal review once a week with the course supervisor. The project topic is chosen from a list of possibilities provided by the course supervisor (many are generated by industrial or academic partners). Assessment is split evenly between implementation and dissertation.

In the authors' experience, this course structure successfully introduces, and increases, the autonomy

of each student to the point where they can work independently and reliably, while having the confidence and self-knowledge to ask for advice or discussion (whether from academic staff or fellow students) when it is useful. An interesting observation is that, while the individual project work could at be carried out elsewhere, almost all students not engaged in industrial placement work in the teaching laboratory, treating it as their workplace for eight hours a day, five days a week. This professional attitude impresses our industrial guests to the laboratory.

2.5 Focussed Module Content

A core philosophy in constructing and updating our case study course is to ensure that the subject matter of each module is focussed on the course's overall objectives and learning outcomes. The aim of our case study course is to educate potential software engineers on the core technologies utilised in developing high quality video games. It would be relatively easy to consider the various aspects of that topic, and to amalgamate some existing modules from other masters level courses into a new course with a little tailoring. However the focus of these modules would not be sufficiently relevant to be accepted by our industrial advisory board.

Consider an example: artificial intelligence is a topic to which many university modules are devoted, and is also a technology utilised in modern video games. However, our industrial contacts (and the authors' own industrial experience) would agree that most of the content of a more generalised artificial intelligence module is not relevant to developing video games, due to practicalities of either performance, memory requirement or reliability (Baekkelund, 2006; Blewitt et al., 2013). An artificial intelligence module for a video games engineering course must be created bespoke with that application in mind. This approach to module construction applies to other aspects of the course and, the authors would argue, to any course which is to maintain good relationships with a target industry.

Although broad subject study is encouraged in laying foundations (e.g., learning to program is a key skill to become a video game programmer), when focusing on video games the subject material is constructed solely with video games in mind. For example, building a physics engine for a fast-paced first person shooter is not the same as building a physics simulation. Physics engines are primarily concerned with an illusion of reality for real-time game-play whereas simulation is concerned with an illusion of reality for off-line analysis. The products arising from two such programmes of study will be distinct; although related, their focus is different and the output (learning outcome) is not compatible.

When developing the course structure, we could have included some "easier" subjects and avoided the more challenging aspects of the core technologies used in video game engineering. This approach may well have attracted more students, and therefore increased the revenue for the faculty. However such an approach would not provide potential employers with sufficiently skilled graduates, which would lead to a poor relationship with industry, and ill-equipped graduates attempting to enter the careers market.

2.6 Coursework as Portfolio

We have mentioned coursework throughout this paper as it is a vital aspect of assessing students ability on an industry-led engineering course. The coursework is designed so as to have a dual purpose: firstly to provide practice and to measure student ability, and secondly to furnish the students with a set of demonstrations and experiences for discussion during a job selection process.

Potential employers in the engineering and programming sectors want to know whether a graduate applicant is capable of understanding and exploring a technical topic in depth. In the authors' experience a major part of a graduate's interview is focussed on the final year project. The projects we offer on our course are designed so as to involve in-depth exploration of a subject; further to this, as many as possible are designed in partnership with either an industrialist or an academic from another department. Consequently, projects usually have an interested party who is considered as the customer. This approach imparts the project with a sense of professionalism, with a target standard to be met to an agreed deadline.

The pieces of coursework attached to each module are also designed to form part of a portfolio, for use by the student at interview. Each coursework definition includes aspects which will appeal to the target industry (admittedly this is easier to define for games engineering than many topics), and students are encouraged to provide videos of their work online, so that potential employers can see their work and discuss it at interview. Feedback from industry has been very positive toward the quality of the coursework and its accessibility.

Year	Total Graduates	Known Employed	Total Percentage	Lost Contact	Known Percentage
2008-09	11	8	72.7	3	100
2009-10	4	4	100	0	100
2010-11	20	14	70	6	100
2011-12	29	22	75.9	5	91.6
2012-13*	12	8	66.7	0	66.7
Total	76	56	73.7	14	90.3

Table 1: The number of students known to have gained employment in target industries or academia within 6 months of graduation for each year of the course. (*)In the case of 2012-13 less than 6 months have elapsed since graduation.

3 Results and Discussion

In this section we present the success rate of graduates from our case study course who enter full-time employment in the target industry, followed by some anecdotal feedback from representatives of that industry. We then discuss the successful elements of the course and consider possible improvements alongside some words of advice for course designers looking for inspiration from industrial practices.

3.1 Graduate Recruitment

Table 1 shows, for each year the course has run, the number of students that graduated from the course, the number of those students that are known to have achieved employment within the target industry or academia within six months of graduation, and that figure expressed as a percentage of the total number of graduating students. The total over the five years that the course has ran represents a percentage success rate of more than 70%. This compares very favourably to the UK national average from computer games development courses of around 12% (Livingstone and Hope, 2011).

The figures in the ”known employed” column of Table 1 only account for the students with whom we have maintained contact, whereas the ”total graduates” column includes all graduating students, so the actual success rate is likely to be even higher than stated. Indeed if we only consider students with whom we have maintained contact, then the percentage success rate is close to 100% (as seen in columns 4 and 5). Furthermore the 2012-13 cohort has only recently graduated (i.e. less than six months prior to the time of writing), so this figure is expected to rise somewhat. The course is at Masters level, and entrance requirements are high; consequently the intake numbers vary somewhat over the years. We maintain a high entrance requirement because the work on the course is difficult, and it would be unfair to accept the registration of a less able student who is very likely to fail.

We endeavour to remain in contact with students after graduation, usually through a professional networking website. There are three reasons for doing so. Firstly this allows us to keep track of the success rate from our course, as illustrated above. Secondly, software development companies regularly approach us looking for potential employees; by maintaining contact with graduates we can match suitable candidates even after they have left the university. Thirdly, and with much longer term effect, when our graduates succeed in the industry the hope is that they will become involved in the course, presenting seminars and eventually recruiting graduates.

3.2 Industrial Feedback

Feedback received from industry is overwhelmingly positive on the quality of the graduates from our case study course. The high percentage of graduates gaining employment is evidence that the industry is impressed with the results of our approach. Many of the members of the industrial advisory board take on one or more student project during the course, which then invariably leads to an offer of full time employment after graduation. Furthermore, those employers return the following year, often looking to increase the graduate intake from our course.

The authors make a point of engaging with commerce at all levels, from multinational software publishers to local SMEs and start-ups. As we have a long career in the target industry, we have many contacts who may be interested in getting involved in our course, or employing graduates. Maintaining a presence at industry events is important, but we elect to keep contact informal and friendly. We have also found that word-of-mouth between developers, especially SMEs and start-ups, is very important and positive. In the words of one CEO of a local SME, the authors are ”helping keep local game development alive, due to the number of high calibre graduates produced”.

3.3 Discussion

It should be apparent from this paper that the authors have designed a coherent course consisting of a series of inter-related modules which combine pedagogic demands with commercially inspired practices. Care has been taken throughout the course to ensure that the students' core skills are gradually built up, and that an understanding of why those skills are important pervades both the taught subject matter and the coursework and practical sessions.

Our use of projects involving incremental autonomy is intended to motivate students, and to build up high-level programming skills throughout the course. During each module on the course students are engaged in activities that complete a scaffold of the overall pedagogic program. It has been noted (Vega et al., 2012) that care should be taken that the scaffolds and activities needed to complete the exercises are arranged carefully in order to stress the relevant content, and also to help those students acquire programming skills effectively.

The authors have found that industrial involvement and feedback should not always be taken at face value. We are wary of crafting module elements toward specific needs of a particular industrialist. In particular, we are regularly asked to provide some training and knowledge of specific tools, engines, or SDKs (not coincidentally the technology which that industrialist is currently using for a project). We are open to integrating such tools into individual projects where they are relevant, but our module content must focus on the science and engineering which is the basis for those tools, rather than providing hands-on training with them.

Perhaps the key factor which we have employed in building this course is maintaining a good understanding of what the targeted industry expects of graduates, in terms of their longer term prospects. We focus on core technologies and underpinning theory, rather than specific implementations. This approach puts an onus on the teaching staff to not only remain *au fait* with developments in the field, but to recognise which developments stem from underlying conceptual ideas. The course is entirely delivered by instructors who are actively involved in research in the core technologies which drive the industry; without that research backbone, we believe the course would be significantly less successful.

4 Conclusions

We have discussed how commercially inspired practices and philosophies can be integrated into an academic teaching course, in a manner which supports the pedagogic needs of the teaching institution and student cohort. We have used our own M.Sc. course in Computer Game Engineering as a case study, and have attempted to present our experiences in a way which is applicable more generally to computer science degrees and other subjects.

We have achieved a very high success rate of students gaining employment within an industry which is notoriously difficult to enter. Coupled to this, we have built up a strong relationship with industry incorporating technically minded representatives into the planning and execution of the course. The authors' own experience has influenced the style of teaching, and has resulted in successfully inculcating a professional work ethic in the students, achieved through treating some aspects of the teaching lab in a comparable way to a commercial development studio. This is coupled with our philosophy of teaching the more difficult and challenging aspects of game engineering, and having a high entrance requirement for students.

Student satisfaction is measured by the University via a series of module feedback questionnaires. This feedback is anonymous and is optional. The scores attained by the course and the teaching staff are consistently high over the five years that the course has ran. Invariably the highest student feedback scores in the school of computing science are attributed to the teachers on this course. Perhaps more importantly, the informal feedback received from students after a few months in their first job is consistently positive in terms of how well they were prepared by our commercially inspired teaching methods.

REFERENCES

Baekkelund, C. (2006). Academic ai research and relations with the games industry. *AI Game Programming Wisdom*, 3:77–88.

Blewitt, W., Ushaw, G., and Morgan, G. (2013). Applicability of gpgpu computing to real-time ai solutions in games. *IEEE Transactions on Computational Intelligence and AI in Games*, 5(3):265–275.

Boud, D. (1988). *Developing student autonomy in learning*. Psychology Press.

Corney, M., Teague, D., and Thomas, R. N. (2010). Engaging students in programming. In *Proceedings of the Twelfth Australasian Conference on Computing Education-Volume 103*, pages 63–72. Australian Computer Society, Inc.

Hilburn, T. B. and Humphrey, W. S. (2002). Teaching teamwork. *Software, IEEE*, 19(5):72–77.

Jiménez, C. and Villalobos, J. (2010). Learning/teaching a computer programming course. *Analysis of State-of-the-Art Solutions for Personalised Learning Support*, page 3.

Livingstone, I. and Hope, A. (2011). Next gen.: Transforming the uk into the worlds leading talent hub for the video games and visual effects industries. nesta.

Matthes, F., Neubert, C., Schulz, C., Lescher, C., Contreras, J., Laurini, R., Rumpler, B., Sol, D., and Warendorf, K. (2011). Teaching global software engineering and international project management. In *Third International Conference on Computer Supported Education*.

Perkmann, M., King, Z., and Pavelin, S. (2011). Engaging excellence? effects of faculty quality on university engagement with industry. *Research Policy*, 40(4):539–552.

Reeve, J., Bolt, E., and Cai, Y. (1999). Autonomy-supportive teachers: How they teach and motivate students. *Journal of Educational Psychology*, 91(3):537.

Reeve, J., Jang, H., Carrell, D., Jeon, S., and Barch, J. (2004). Enhancing students' engagement by increasing teachers' autonomy support. *Motivation and emotion*, 28(2):147–169.

Vega, C., Jiménez, C., and Villalobos, J. (2012). Implementing an incremental project-based learning solution for cs1/cs2 courses. In *Second International Conference on Computer Supported Education*, pages 15–27.

Winslow, L. E. (1996). Programming pedagogya psychological overview. *ACM SIGCSE Bulletin*, 28(3):17–22.