Experiences in Developing and Delivering a Programme of Part-Time Education in Software and Systems Security

Andrew Simpson*, Andrew Martin*, Cas Cremers*, Ivan Flechais*, Ivan Martinovic*, and Kasper Rasmussen*
*Department of Computer Science, University of Oxford
Wolfson Building, Parks Road, Oxford OX1 3QD
United Kingdom
Email: {Andrew.Simpson, Andrew.Martin, Cas.Cremers, Ivan.Flechais, Ivan.Martinovic, Kasper.Rasmussen}@cs.ox.ac.uk

***Abstract*—We report upon our experiences in developing and delivering a programme of part-time education in Software and Systems Security at the University of Oxford. The MSc in Software and Systems Security is delivered as part of the Software Engineering Programme at Oxford — a collection of one-week intensive courses aimed at individuals who are responsible for the procurement, development, deployment and maintenance of large-scale software-based systems. We expect that our experiences will be useful to those considering a similar journey.**

I. Introduction

Much has been written about the need for quality software engineering education over the past 25 years, with the contributions of Lethbridge [1] and Shaw [2], [3] being notable in this respect. Some authors have reflected upon experiences in particular countries (see, for example, [4], [5] and [6]), while others have considered the various difficulties associated with delivering software engineering education (see, for example, [7], [8], and [9]). Our particular concern in this paper is a programme of education that exists at the academia / industry interface — an area explored by a number of other authors, including Mead *et al.* [10], Fraser *et al.* [11], Vaughn and Carver [12], Subrahmanyam [13], and Almi *et al.* [14].

The Software Engineering Programme at the University of Oxford[1] was launched in 1993.[2] The Programme, which was originally a partnership between the University's Department of Computer Science (previously the Computing Laboratory) and its Department for Continuing Education, had the aim of providing graduate-level education to professional software engineers who wished to study on a part-time basis. The approach of the Software Engineering Programme has been to deliberately target its programme of education at those associated with the procurement, development, deployment and maintenance of large-scale software-based systems — which inevitably requires its courses to be both relevant to commercial needs and accessible to those who have been away from the higher education context for some time (or, in some cases, those who have no prior higher education experience).

Initially concentrating on Oxford's traditional strengths in topics such as Functional Programming and Formal Methods, the Software Engineering Programme has evolved into a significant enterprise offering one-week intensive courses in approximately 40 different subjects. In recent years, the Programme has become wholly owned by the Department of Computer Science and now offers professionals the opportunity to study for an MSc in Software Engineering or an MSc in Software and Systems Security. The development and delivery of the latter is the focus of this paper. We give consideration to the relationship between security and software engineering, and pay particular attention to how we have tried to raise awareness of security issues, challenges and techniques to those whose primary concern is software development. In addition, we reflect upon some of the trends we have witnessed and some of the lessons we have learnt along the way.

The structure of the remainder of this paper is as follows. In Section II, we discuss the Software Engineering Programme at the University of Oxford. We present our MSc in Software and Systems Security in Section III, and focus on a particular course (Data Security and Privacy) in Section IV. In Section V, we discuss our approach to supervising and assessing projects. In Section VI, we reflect upon our experiences, indicate some trends, and present some lessons learnt. Finally, we summarise the contribution of this paper in Section VII.

II. The Software Engineering Programme at the University of Oxford

What is now known as the Software Engineering Programme at the University of Oxford was established in the early 1980s as a collection of 'industrial courses': professionals would attend one-week, intensive courses that introduced them to formal methods such as Z [17], [18], Communicating Sequential Processes (CSP) [19], [20], and B [21], [22]. An 'integrated programme' of six one-week courses was established in 1993, with the first intake being drawn entirely from staff of IBM's UK Development Laboratories at Hursley; this

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

[2] Various aspects of the Software Engineering Programme have been reported upon previously: issues pertaining to the supervision and assessment of part-time postgraduate projects are discussed in [15]; the challenges of teaching formal methods to professional software engineers studying on a part-time basis are discussed in [16].

has evolved into a comprehensive programme of education, which now delivers one-week courses in approximately 40 different subjects. At present, approximately 300 students are registered with the Software Engineering Programme, pursuing either an MSc in Software Engineering or an MSc in Software and Systems Security. Many of these students 'commute' from overseas to attend one-week courses.

The aim of the Programme is to provide students with an understanding of software engineering and security principles and techniques; this is stated formally thus:[3]

> "The aim is to provide students with an understanding of software engineering principles and techniques, enabling them: to choose the most appropriate technique to apply in a software design, development, or management situation; to apply that technique, or to identify the resources (intellectual or material) necessary for its application; to explain their choice in terms that can be understood by anyone with a basic knowledge of the field."

One's interpretation of the term *software engineering* will inevitably influence the philosophy of a programme of education in the subject. The Programme team has found Blanchard's definition of *systems engineering* [23] — which encompasses technical, social and other aspects associated with the design and development of large-scale systems — to be an appropriate reflection of the activity that motivates it:

> "The application of scientific and engineering efforts to (1) transform an operational need into a description of system performance parameters and a system of configuration through the use of an iterative process of definition, synthesis, analysis, design, test and evaluation, and validation; (2) integrate related technical parameters and ensure the compatibility of all physical, functional, and program interfaces in a manner that optimises the total definition and design; and (3) integrate reliability, maintainability, usability (human), safety, producibility, supportability (serviceability), disposability, and other such factors into the total engineering effort to meet cost, schedule, and technical performance objectives." [23]

The Programme's requirements for entry are flexible, taking into account prior industrial experience and academic background. The formal requirements are stated thus.[4]

> "To be accepted for postgraduate study in software engineering (or software and systems security), you should have:
>
> - at least two years' close engagement with software (or security) issues in a professional environment (but see below);
> - a university-level qualification in a related subject (but see below);
> - a good command of both written and spoken English;
> - an appreciation of the challenges and practices in software engineering (or in software and systems security);
> - an appropriate level of logical, mathematical, or analytical skills;
> - a good understanding of the nature of the programme, and the level of commitment required; and
> - the support of their employer, if necessary, in embarking on this course of study.
>
> More extensive experience may compensate for the lack of a related qualification, and a strong, immediately-relevant qualification — e.g. in engineering or computing — may compensate for a lack of experience."

The wide diversity of the student body gives rise to a number of challenges: few assumptions can be made about the nature of previous experience, meaning that the complexities of teaching principles and techniques are inevitably different from those associated with teaching cohorts of full-time student that are (typically) more homogeneous. (This heterogeneity is not exclusive to our programme; Hjelmås and Wolthusen [24], for example, make similar observations in the context of their programme.)

Each course consists of three components: a period of preparatory study, typically involving the reading of textbook chapters or research papers and, perhaps, a small exercise; an intensive teaching week, consisting of lectures, exercises and practicals; and a written assignment. The relatively small class sizes (the maximum class size is 18) lead to a high degree of interaction between students and instructors. The structure of the course will depend upon the subject, but will typically involve anywhere between 10 and 20 hours of lectures, supported by practical sessions, group exercises or tutorial sessions, as appropriate. Assignments — which are undertaken over a six-week period — allow students to reflect upon and apply the techniques taught during the week. Some assignments are more research-oriented, requiring students to engage with the wider academic literature; others are more applied, with a particular scenario having to be tackled.

The assignments are considered to be formal examinations by the University of Oxford. For each assignment, extensive personalised feedback (on average, of the order of 2–3 pages) is provided to each candidate. There are good reasons for this choice of mode of assessment. First, our students often travel from all over the world to attend our courses; to expect them to travel back to sit examinations would be impractical. Second, by being provided a six-week period in which to undertake an assignment, students are given a great deal of opportunity to reflect upon, and immerse themselves in, the material that was delivered during the one-week course. We would argue that this approach to assessment allows students to gain a deeper understanding of a topic than would be possible were they to be assessed via a sit-down examination.

Many of the Programme's students bring significant experi-

[3] http://www.cs.ox.ac.uk/softeng/handbook/specification.html
[4] http://www.cs.ox.ac.uk/softeng/study/apply.html

ence: it is not uncommon for a course attendee to have greater in-depth knowledge of a particular aspect of the course than the teaching team. This presents great learning opportunities; it also presents challenges in terms of ensuring that enthusiasm is channeled effectively.

A candidate for the MSc has up to four years (with the possibility of a one-year extension and suspensions of status, if necessary) in which to attend ten courses and complete a dissertation. The time available reflects the professional and personal pressures that many of our students will face during their time with the Programme.

While students are free to choose any 10 courses with a view to designing their own unique programme of study (with the only constraint being that students registered for the MSc in Software and Systems Security must choose at least six courses from the Software and Systems Security provision), it has been helpful to collect courses into three separate themes:

- *Software Engineering Methods*: Agile Methods; Concurrency and Distributed Systems; Enterprise Architecture; Interaction Design; Management of Risk and Quality; Model Checking; Performance Modelling; Process Quality and Improvement; Requirements Engineering; Safety Critical Systems; Software Development Management; Software Engineering Mathematics; and Specification and Design.
- *Software Engineering Tools*: Agile Engineering Practices; Concurrent Programming; Database Design; Design Patterns; Extensible Markup Language; Functional Programming; Mobile and Sensor Networks; Object-Oriented Design; Object-Oriented Programming; Semantic Technologies; Service Oriented Architectures; and Software Testing.
- *Software and Systems Security*: Building Information Governance; Cloud Security; Data Security and Privacy; Design for Security; Forensics; Mobile Systems Security; Network Security; People and Security; Risk Analysis and Management; Secure and Robust Programming; Security and Incident Management; Security Principles; Security in Wireless Networks; and Trusted Computing Infrastructure.

These themes have developed over time, reflecting demands from the student body and advice provided by the Programme's Industrial Advisory Board. A crucial part of the review process comes from students themselves: should any course be deemed irrelevant, registration numbers would decline accordingly.

The courses delivered by the Programme are all concerned with the teaching of principles in a way that is sympathetic to the context in which they might be deployed. The course that is called eXtensible Markup Language isn't simply a training course on XML; rather, it is a course that is concerned with transformation and manipulation of information, with XML as a vehicle. As another example, the Object-Oriented Programming course is not simply a course on Java: rather, it concentrates on principles — with Java as a vehicle.

This approach is important: the prior experience of students is varied (an attendee on the Database Design course, for example, might have been working with relational databases for many years; an attendee on the aforementioned Object-Oriented Programming course might have significant experience of working on an implementation of the Java Virtual Machine) and there is a need for all students to ensure that they receive value from the course — no matter how experienced they are. Thus, the courses support the students' practical knowledge by discussing principles and foundations within the broader software engineering context.

The Programme limits the number of dependencies between courses as much as possible. Nevertheless, there are certain natural starting points: the Security Principles course is the typical starting point for those pursuing the Software and Systems Security MSc; the Software Engineering Mathematics course underpins many of the more formal courses. As students can start at any point during the year, it may very well be that a student doesn't actually start their career with the Programme with one of these two courses: for this reason (as well as others), very few assumptions can be made about prior academic experience.

The Programme currently employs 14 full-time academics and three administrators. This number has grown steadily from a single academic in 1993, to four by the year 2000, to seven by 2003, through to the present total of 14. Of these, six (the present authors) are primarily concerned with the delivery of courses in the area of Software and Systems Security.

III. The MSc in Software and Systems Security

Technological and societal developments over the past two decades have led to increased security concerns and challenges across a wide range of sectors. The Software Engineering Programme has attempted to meet these demands. The Programme delivered its first course on a security-related topic — Security Principles — in November 2000, with others (including Design for Security, People and Security, and Risk Analysis and Management) coming on-line in the subsequent couple of years. These developments led to the Programme deciding to initially offer a Postgraduate Certificate in Computer Security — requiring candidates to attend, and submit assignments for, four courses. In recent years, the Programme's provision in this area has expanded significantly, meaning that students now have the opportunity to pursue an MSc in Software and Systems Security. Students pursuing this path are required to ensure that six of their 10 courses are drawn from the Software and Systems Security theme.

If the MSc were described as consisting of a total of 180 credit points, then each course and assignment would be associated with a notional 15 credit points, and the dissertation would be associated with a further notional 30 credit points.

The learning outcomes are stated thus:[5]

> "Graduates in Software and Systems Security will be able to: evaluate the security risks and threats

[5]http://www.cs.ox.ac.uk/softeng/handbook/specification.html

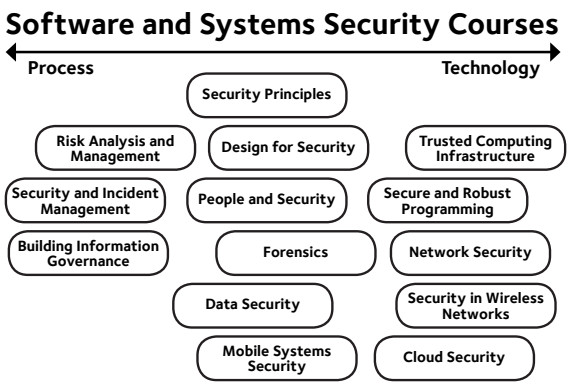

Fig. 1. The range of courses

> associated with a proposed software development; explain the technologies used to achieve a suitable level of security in a given situation; and design security solutions that are both technically sound and usable in practice."

We give brief descriptions of each course in Appendix A. The breadth of the provision — with individual courses placed on what might be characterised as the 'process–technology spectrum' — is illustrated in Figure 1.

Half of these courses are delivered by full-time academic staff; the remainder are delivered by 'external lecturers', all of whom are subject experts. At the heart of many of the security courses are case studies and 'war stories': examples of errors and mishaps that have led to security and privacy breaches at a variety of levels — ranging from the individual through the commercial to the national. This makes the employment of subject experts from industry a credible addition to the Programme's teaching team: the input and expertise of these individuals adds significant value and credibility to the students' learning experience. Importantly, the nature of the student body means that the students themselves often bring interesting and unique perspectives on particular problems.

The UK Government has made 'cyber security' a strategic priority, largely in recognition of the perceived threat to the nation's 'cyber infrastructure'. In 2014, the MSc in Software and Systems Security was one of a handful of Master's degrees to be accredited by the Government Communications Headquarters (GCHQ), the UK's security and intelligence organisation. As well as meeting the requirements for the MSc in Software and Systems Security, to receive a 'GCHQ certificate' each student should have received a passing grade for each of the following courses:

- Security Principles.
- Security and Incident Management.
- Forensics.
- At least two from Design for Security, Risk Analysis and Management, and People and Security.
- At least one from Data Security and Privacy and Building Information Governance.
- At least one from Network Security and Secure and Robust Programming.
- At least one from Cloud Security, Mobile Systems System, Trusted Computing Infrastructure, and Security for Wireless Networks.

This reflects GCHQ's perception of what a qualification in 'general Cyber Security' should cover and our desire that our graduates should cover an appropriate mix of theory and practice.

IV. An example: Data Security and Privacy

The Programme's *Data Security and Privacy* course was delivered for the first time in June 2010, and has subsequently been delivered on three further occasions. The course is an interesting case study as, while very clearly a 'security course', it does require an understanding of some software engineering subjects (relational databases, in particular) and utilizes some techniques from computer science (set theory; predicate logic; graph theory). To this end, the course 'requirements' for participants are stated thus:

> "Participants should have a basic understanding of computer security to the level provided by the Security Principles course; participants should also have some familiarity with predicate logic and set theory to the level provided by the Software Engineering Mathematics course."

The motivation for the course is stated as follows:

> "As increasing amounts of data are captured about patients, consumers and citizens, and as more ways of linking and utilising such data emerge, so do concerns about the treatment of personal data — with these concerns emerging from a variety of stakeholders. As such, issues pertaining to database and applications security have increased in importance in recent years. Understanding how existing and emerging legislation might be considered in designing secure databases, as well as how such designs might be mapped to practical security measures, will be essential in an increasingly data-driven world."

The structure of the course — which is broken into three parts — follows naturally from the above.

1) Context: the changing landscape; privacy, data security and the law.
2) Access control: theory and practice; mandatory policies; role-based access control; policy languages.
3) Privacy: statistical database security; balancing privacy and utility; k-anonymity and related techniques.

The current course text — *Security, Privacy, and Trust in Modern Data Management* by Petkovic and Jonker [25] — reflects the breadth of the course. The fact that previous instances have leveraged very different books — *The Spy in the Coffee Machine* by O'Hara and Shadbolt [26] and Bishop's *Introduction to Computer Security* [27] — is indicative both

of the fact that new courses evolve over time and also that these courses are often not a natural fit for standard academic textbooks — given the nature of the student body and the fact that teaching techniques 'within context' is important to the Programme team.

The course involves 18 sessions — as is typical for our courses (with two sessions per half-day). There are four sessions per day for Monday to Thursday and two sessions on Friday (courses finish on Friday lunchtime).

The current course structure is as follows:

- Monday AM: Introduction: the changing landscape.
- Monday PM: Privacy, data security, and the law.
- Tuesday AM: Access control: theory and practice.
- Tuesday PM: Mandatory policies.
- Wednesday AM: Role-based access control.
- Wednesday PM: Policy languages.
- Thursday AM: Balancing privacy and utility.
- Thursday PM: Statistical database security.
- Friday AM: k-anonymity and related techniques.

The nature of the sessions varies according to the nature of the topic. For example, the Monday AM session is lecture-based, whereas the Monday PM material is delivered in the style of an interactive workshop. The access control part of the course is relatively theoretical, utilising formal techniques to explore different models of access control. Finally, the third component of the course consists of lectures, a workshop-based discussion, theoretical exercises, and practical exercises.

The nature of the assignment reflects the breadth of the course and the fact that no prior experience (in terms of security, database or systems administration skills, formal description techniques, etc.) can be assumed. To this end, assignments with a research focus are set for this course. As an example, the assignment for the first instance of this course consisted of two questions. The first, which makes reference to [28], was stated as follows:

> "One important model of access control that was not covered in the taught material is the Clark-Wilson model, which was first described in the paper *A comparison of commercial and military computer security policies*, published in the Proceedings of the 1987 IEEE Symposium on Research in Security and Privacy. After consulting both this paper and the wider literature (constituting any relevant textbooks and conference or journal papers),
>
> (a) give an overview of the model;
> (b) show, via an extension of the running example from the lectures (or any other example of your choice), the principles of the model;
> (c) compare and contrast the Clark-Wilson model with some of the other models of access control that we have studied; and
> (d) give some indication of the implementation issues associated with this model either in general terms, or with respect to a specific DBMS.
>
> As an illustration of scope, it would be surprising if an answer were to be longer than 10 pages."

The second question, which makes reference to [29], was stated thus:

> "First, consider the discussion between Korff and Shadbolt on the pros and cons of www.data.gov.uk. Second, download some data sets from www.data.gov.uk to get a feel for the kind of data that is available and the potential privacy issues involved. Describe where you stand on the discussion. You should feel free to make use of any resources to support your arguments; a strong answer should both leverage relevant literature and give consideration to some of the data sets available from the site.
>
> Remember that a scientific argument needs to be backed up with evidence (don't simply give opinions!), and that you should describe the strengths and weaknesses of each position before presenting a reasoned conclusion.
>
> It would be surprising if an answer were to be longer than 5 pages."

The assessment criteria for the course are stated as follows:

- Context: have you demonstrated an awareness of the issues pertaining to privacy and confidentiality derived from relevant legislation, guidelines and ethical concerns?
- Access control: have you demonstrated an understanding of the underlying theory and principles?
- Statistical database and microdata security: have you demonstrated an understanding of both the broad issues and the underlying theory and principles?"

Students' experiences in tackling this particular question are summarised (in part) in [30].

V. Projects

The project is a crucial part of the MSc — it allows students to address some aspects of security than is the case in undertaking post-course assignments. There are several aspects that complicate the project process for part-time students. For example, (in the UK, at least) full-time MSc students will typically spend an extended period of several months on their project work: this is not possible for part-time MSc students. Such students will typically spend spare-time — at irregular intervals — working on their project. There are consequences of this. First, there is the overhead associated with 'picking up' from where things were left at the end of the last session. Second, by not being in the higher education context on a full-time basis, many of the lessons learnt by osmosis by full-time students are missed out on. One consequence is that part-time students can overestimate what is required of them and underestimate potential risks.

To counter these problems, we deliberately encourage our students to pursue project topics that are closely related to their professional activity; this has the additional benefit of increasing the potential for employer 'buy-in'. If this is not possible, basing a project around the topics covered in one

of their favourite courses is the next best option. Only as a last resort will a student pursue a project suggested by a staff member. Crucially, the subject must be of interest to the student, rather than of interest to an academic, if it is to sustain them over what is a significant period of time

The Programme team has established a 'Project Week' to mitigate some of these problems: students are required to attend a one-week preparatory course prior to embarking upon their project work. Topics such as academic writing, the assessment and examination process, research skills, and project planning are covered during the week. The intention is that successful completion of the course will reduce many of the risks. The most important aspect, though, is that, together with the academics responsible for running that week's activities and the student's supervisor, a sensible, coherent and well-scoped project proposal is identified. To help in this, the students take part in an assessment exercise — whereby they assess previously submitted dissertations to help understand the various parts of the process; this enables them to judge what constitutes a well-scoped and well-executed project.

The allocation of project supervisors happens during these Project Weeks. All students are assigned a supervisor (drawn from the academic staff who teach Security courses on the MSc in Software and Systems Security) at the start of their period of study. Should the student's proposed project be something that their supervisor can supervise, then they will typically remain with that supervisor for the project; if, on the other hand, the intention is to pursue a topic that is more closely aligned with another academic's expertise, then they will change supervisor during the Project Week and it will be the new supervisor that 'signs off' the proposal.

All dissertations are assessed by two academics, one of whom is an examiner (the examiners are a subset of the group of academics responsible for delivering the course) and neither of whom can have been involved in the supervision of the student. The assessors provide independent grades according to clearly stated criteria. If the grades differ by less than 10% and don't cross either of the fail/pass or pass/distinction boundaries, then the final mark is determined by averaging the two provisional marks; should the difference be significant or cross a boundary, the two assessors are invited to arrive at an agreed, consolidated grade. Should such agreement be impossible to arrive at, the external examiner will be invited to review the dissertation and suggest a grade. (The external examiner is at liberty to review any other dissertation.)

VI. Evaluation and lessons learnt

A. Evaluation

Each course is evaluated by attendees via a paper-based questionnaire. Students respond (anonymously) to 12 statements on a scale of 1 to 5 (1 and 2 disagree; 3 indifference; 4 and 5 agree):

1) The lectures added significant value to the course material.
2) The lecturer took the time needed to explain the key concepts.
3) The lectures included valuable contributions from the other students in the class.
4) The lecturer was helpful and ready to answer questions.
5) The exercises helped me to understand the topics covered in the lectures.
6) The lecturer or tutor was knowledgeable and encouraging.
7) Help was available — from the lecturer or tutor — when I needed it.
8) Issues raised were adequately addressed — through model solutions or discussion.
9) I think that the techniques taught during the course will be valuable to me in the future.
10) The course was well constructed: the various components worked well together.
11) The course material was appropriate, and of good quality.
12) The course administration was efficient and effective.

At the time of writing, the average score across all courses delivered by the Software Engineering Programme since 2010 is 4.53 (4 represents "agree" and 5 represents "strongly agree"); the highest 'overall' score is 4.80 for Specification and Design. The scores for the Software and Systems Security courses are:

- Building Information Governance: 4.69
- Cloud Security: 4.27
- Data Security and Privacy: 4.58
- Design for Security: 4.52
- Forensics: 4.64
- Mobile Systems Security: 4.78
- Network Security: 4.54
- People and Security: 4.65
- Risk Analysis and Management: 4.58
- Security and Incident Management: 4.73
- Security Principles: 4.53
- Secure and Robust Programming: 4.61
- Security in Wireless Networks: 4.61
- Trusted Computed Infrastructure: 4.59

B. Lessons learnt

1) The relationship between Software Engineering and Software and Systems Security is a complicated one: Our first course in a security-related subject was Security Principles, which was first delivered in November 2000. All of the students were studying for an MSc in Software Engineering and were drawn from organisations such as IBM, Panasonic, Nokia and the Royal Air Force — which would have been typical for the Programme at the time. The most recent instance at the time of writing (delivered in July 2014) was evenly split between those studying for an MSc in Software Engineering and those studying for an MSc in Software and Systems Security. This shift reflects the fact that the Security Principles course tries to achieve two different things: to provide Software Engineering students with an exposure to security issues and to provide Software and Systems Security students with an appropriate entry point to their MSc.

The provision of the two MScs gives rise to a broad programme of courses. This expansion of the Programme — and the fact that students can 'mix and match' courses — brings some interesting challenges. For example, the prior experience that software engineers bring to courses will (typically) be different to that brought by security engineers: the fact that security engineers might attend, for example, the Database Design course can make for an interesting teaching challenge — as might the fact that a software engineer might attend, for example, the Network Security course. The small class sizes mean that it is possible to 'tailor' that week's delivery for those in attendance to suit their background and experience.

2) Funding and support from employers: The Software Engineering Programme had its foundations in one-week courses delivered to employees of several companies. While the Programme team still delivers courses on a one-off basis to some companies, this is not the core business of the Programme. 20 years ago, it was not uncommon for companies to invest in the education of their charges: companies would often pay the course fees of those attending. Ten years ago, companies providing funding for their employees was less common, but 'support in kind' might be provided: time off from work was given to attend courses; workloads might be reduced; 'study days' might be given. Now, it is not uncommon for students to not only pay their own fees but also to attend courses in their annual vacation time: while the need for security (and software engineering) education is often recognised by many companies, their actions do not always reflect this.

3) The value of classroom contact: The rise of Massive Open Online Courses (MOOCs) in the past few years has, in conjunction with increases in tuition fees in the UK, resulted in a long series of debates questioning whether more traditional approaches to higher education can survive in the 21st century.

The delivery of courses within the Software Engineering Programme typically involves one lecturer and one teaching assistant (sometimes supported by guest lecturers). Class sizes of up to 18 (with an average of approximately 12) encourage interaction. We believe that the value delivered in this mode is extremely high — with, as we have discussed, the prior experience of the course attendees providing significant 'added value'. A drawback is that this mode of study is relatively expensive. Nevertheless, recruitment remains good — with year-on-year increases in terms of both numbers of applications and numbers of admissions.

4) Prior knowledge changes: The typical student on the Software Engineering Programme 20 years ago was a relatively experienced software engineer, who had been based in the industry for at least five years. This meant that the prior knowledge that one might use in delivering courses was relatively uniform. As an example, when teaching discrete mathematics, one might use a binary tree as a motivating example when discussing recursive functions. Unfortunately, this is no longer true: it is not unusual to be met by blank faces (by even those with a first degree in an IT-related subject) when mentioning binary trees. This is for (at least) two reasons. First, the level of abstraction has been raised: developers don't have to define their own tree-like structures as libraries exist that can be leveraged. Second, the student body of the Software Engineering Programme now reflects the healthy heterogeneity that is the workforce in software engineering, security, and related industries.

The problem is even more complex when it comes to security courses: some will have backgrounds in software; others will have backgrounds in hardware; some will have neither, and are taking the courses from the point of view of needing to be an 'informed customer'. As we have discussed, this brings (and will continue to bring) challenges in terms of course design and delivery. The breadth of courses (as per Figure 1) is sympathetic to this: a purely managerial route through the MSc (drawing on courses from the Software Engineering MSc) is possible — but wouldn't, for example, meet the aforementioned GCHQ accreditation requirements.

5) Principles endure: As alluded to in Section II, much has changed in the software industry over the past two decades. Yet principles still endure. For example, issues such as the healthiness of transactions were important to know in the 1970s. Then, for a period, they were less important to know as database management systems could be relied upon to provide good transactional support. Later, as developers were required to write code to engage with several data sources, developers themselves had again to think about such issues. (Similar points might be made for, for example, query optimisation.) With respect to other aspects, Formal Methods and Functional Programming — while never attaining the heights predicted several decades ago — now have a role to play, in, for example, underpinning software testing frameworks and code assertion frameworks.

Similarly, one could argue that while technological developments often bring new security challenges, it is often the case that they breathe life into old challenges — or present existing ones in new guises. We would argue that providing professionals the opportunity — and space — to step back and consider principles and fundamentals is a benefit of programmes such as ours.

6) The team: An important aspect of the delivery and running of the Software Engineering Programme is that there is a dedicated team whose teaching responsibilities are exclusively associated with the Programme: although all of the academics supervise full-time doctoral students and undertake research and administrative tasks, all of their teaching is exclusively for the Programme. In addition, all of the academics' teaching tends to be in subjects closely related to their research interests. Subject experts are contracted to deliver courses where in-house expertise does not exist. Going further, the Programme's students are all supervised by members of this team. This is important: the demands, needs and expectations of part-time students are very different from those of full-time students; as such, academics and administrators alike need to be sensitive to this. For example, personal or professional commitments can sometimes mean that long-planned course attendances are cancelled at short notice — or, even worse, those delivering the course are unavailable at short notice. Well thought through

contingency measures are more vital in a context such as this than they would be in a 'typical' educational context, as is a solid 'team ethos'.

We acknowledge that this is an unusual situation. However, it would be difficult to reproduce an enterprise such as that described here by employing a team who attempted to deliver such a programme in addition to other duties.

VII. Conclusions

We have described the MSc in Software and Systems Security, which is part of the Software Engineering Programme at the University of Oxford. The difference between our MSc and other, 'more traditional' Master's in Security (such as that of Royal Holloway, University of London (RHUL) [31], for example) pertains to heritage and motivation: our MSc grew out of the University of Oxford's Software Engineering expertise; that of RHUL grew out of their research expertise in cryptography (and related areas). In some ways, our experiences have more in common with those who have given consideration to the development of courses on software assurance (e.g. [32]–[35]). Further, the focus on contextual and practical issues means that the course is closer in nature to, for example, Carnegie Mellon's MS in Information Technology—Privacy Engineering (MSIT-PE) (see [36] for a description of the motivation for the development of that course) than more traditional Master's in Security.

The MSc (and, indeed, the Software Engineering Programme as a whole) is operating in a context in which the demand for a highly skilled work force is increasing at precisely the time that government support for higher education is declining [15]. In addition, there are ongoing debates about how Computer Science and Software Engineering curricula prepare graduates for the 'real world'. While there have been many shifts over the past two decades — in terms of employer-investment in individuals, in terms of technology, and so on — the MSc in Software and Systems Security and the Software Engineering Programme (which has just celebrated its 20th anniversary) continue to benefit from the gap that exists between formal software engineering and security education at the undergraduate level and the needs of industry; they also benefit from the fact that these needs will continue to evolve — meaning that the demand for post-experience education such as that described in this paper will continue to exist for the foreseeable future.

References

[1] T. C. Lethbridge, "What knowledge is important to a software professional?" *IEEE Computer*, vol. 33, no. 5, pp. 44–50, 2000.

[2] M. Shaw, "Software engineering education: A roadmap," in *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)*. ACM Press, 2000, pp. 371–380.

[3] ——, "Continuing prospects for an engineering discipline of software," *IEEE Software*, vol. 26, no. 6, pp. 64–67, 2009.

[4] C. O'Leary, D. Lawless, D. Gordon, L. Haifeng, and K. Bechkoum, "Developing a software engineering curriculum for the emerging software industry in China," in *Proceedings of the 19th IEEE International Conference on Software Engineering Education and Training (CSEET 2006)*, 2006, pp. 115–122.

[5] G. Taran and M. Rosso-Llopart, "Software engineering education in Russia: A comparative study of people, process and technology: A four year pespective," in *Proceedings of the 20th IEEE International Conference on Software Engineering Education and Training (CSEET 2007)*, 2007, pp. 19–28.

[6] K. Garg and V. Varma, "Software engineering education in India: Issues and challenges," in *Proceedings of the 21st IEEE Conference on Software Education Education and Training (CSEET 2008)*, 2008, pp. 110–117.

[7] M. R. Ali, "Imparting effective software engineering education," *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 4, pp. 1–3, 2006.

[8] N. R. Mead, "Software engineering education: How far we've come and how far we have to go," *Journal of Systems and Software*, vol. 82, no. 4, pp. 571–575, 2009.

[9] M. A. Ardis and P. B. Henderson, "Software engineering education (SEEd): Is software engineering ready for MOOCs?" *ACM SIGSOFT Software Engineering Notes*, vol. 37, no. 5, pp. 14–14, 2012.

[10] N. R. Mead, H. J. C. Ellis, A. Moreno, and P. MacNeil, "Can industry and academia collaborate to meet the need for software engineers?" *Cutter IT Journal*, vol. 14, no. 6, pp. 32–39, 2001.

[11] S. Fraser, R. Bareiss, B. Boehm, M. Hayes, L. Hill, G. Silberman, and D. Thomas, "Meeting the challenge of software engineering education for working professionals in the 21st century," in *Proceedings of the 18th Annual SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2003)*, 2003, pp. 262–264.

[12] R. B. Vaughn and J. Carver, "Position paper: The importance of experience with industry in software engineering education," in *Proceedings of the 19th IEEE International Conference on Software Engineering Education and Training (CSEET 2006)*, 2006, pp. 19–19.

[13] G. V. B. Subrahmanyam, "A dynamic framework for software engineering education curriculum to reduce the gap between the software organizations and software educational institutions," in *Proceedings of the 22nd IEEE International Conference on Software Engineering Education and Training (CSEET 2009)*, 2009, pp. 248–254.

[14] N. E. A. M. Almi, N. A. Rahman, D. Purusothaman, and S. Sulaiman, "Software engineering education: The gap between industry's requirements and graduates' readiness," in *Proceedings of the IEEE Symposium on Computers and Informatics (ISCI 2011)*, 2011, pp. 542–547.

[15] A. C. Simpson, A. P. Martin, J. Gibbons, J. W. M. Davies, and S. W. McKeever, "On the supervision and assessment of part-time postgraduate software engineering projects," in *Proceedings of the 25th International Conference on Software Engineering (ICSE 2003)*. IEEE Computer Society Press, 2003, pp. 628–633.

[16] J. W. M. Davies, A. C. Simpson, and A. P. Martin, "Teaching formal methods in context," in *CoLogNET/FME Sympsoium, TFM 2004*, ser. Lecture Notes in Computer Science, C. N. Dean and R. T. Boute, Eds. Springer, 2004, vol. 3294, pp. 185–202.

[17] J. M. Spivey, *The Z Notation: A Reference Manual*, 2nd ed. Prentice-Hall International, 1992.

[18] J. C. P. Woodcock and J. W. Davies, *Using Z: Specification, Refinement, and Proof*. Prentice-Hall International, 1996.

[19] C. A. R. Hoare, *Communicating Sequential Processes*. Prentice-Hall International, 1985.

[20] A. W. Roscoe, *Understanding Concurrent Systems*. Springer-Verlag, 2010.

[21] J.-R. Abrial, *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[22] S. A. Schneider, *The B-Method: An Introduction*. Palgrave Cornerstones in Computer Science, 2001.

[23] B. S. Blanchard, *System Engineering Management*. Wiley, 1998.

[24] E. Hjelmås and S. D. Wolthusen, "Full-spectrum information security education: Integrating B.Sc., M.Sc., and Ph.D. programs," in *Proceedings of the 3rd Annual Conference on Information Security Curriculum Development (InfoSecCD 2006)*, 2006, pp. 5–12.

[25] M. Petković and W. Jonker, Eds., *Security, Privacy, and Trust in Modern Data Management*. Springer, 2007.

[26] K. O'Hara and N. Shadbolt, *The Spy In The Coffee Machine: The End of Privacy As We Know It*. Oneworld Publications, 2008.

[27] M. Bishop, *Introduction to Computer Security*. Addison Wesley, 2004.

[28] D. D. Clark and D. R. Wilson, "A comparison of commercial and military computer security policies," in *Proceedings of the 1987 IEEE Symposiym on Research in Security and Privacy*, 1987, pp. 183–193.

[29] D. Korff and N. Shadbolt, "Public information: Cause for celebration or concern?" *Public and Science*, pp. 10–11, March 2010.
[30] A. C. Simpson, "On privacy and public data: A study of data.gov.uk," *Journal of Privacy & Confidentiality*, vol. 3, no. 1, pp. 51–65, 2011.
[31] C. Ciechanowicz, K. M. Martin, F. C. Piper, and M. J. B. Robshaw, "Ten years of information security masters programmes: Reflections and new challenges," in *Security Education and Critical Infrastructures*, C. Irvine and H. Armstrong, Eds. Kluwer Academic Publishers, 2003, pp. 215–230.
[32] N. R. Mead, J. McDonald, J. H. Allen, M. Ardis, T. B. Hilburn, A. J. Kornecki, and R. C. Linger, "Development of a master of software assurance reference curriculum," *International Journal of Secure Software Engineering*, vol. 1, no. 4, pp. 18–34, 2010.
[33] N. R. Mead, J. H. Allen, M. Ardis, T. B. Hilburn, A. J. Kornecki, R. C. Linger, and J. McDonald, "Software assurance curriculum project volume I: Master of software assurance reference curriculum," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-2010-TR-005/ESD-TR-2010-005, August 2010.
[34] N. R. Mead, T. B. Hilburn, and R. C. Linger, "Software assurance curriculum project volume II: Undergraduate course outlines," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-2010-TR-019, ESC-TR-2010-019, August 2010.
[35] N. R. Mead, J. H. Allen, M. Ardis, T. B. Hilburn, A. J. Kornecki, and R. C. Linger, "Software assurance curriculum project volume III: Master of software assurance course syllabi," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-2011-TR-013/ESD-TR-2011-013, March 2011.
[36] L. F. Cranor and N. Sadeh, "A shortage of privacy engineers," *IEEE Security & Privacy*, vol. 11, no. 2, pp. 77–79, 2013.

APPENDIX

We describe the learning outcomes of the individual courses in the following.

A. Building Information Governance

The successful participant will:

- achieve an understanding and ability to explain the diverse sources of requirements that give substance to the complexity and value of information governance;
- acquire a capability to analyse these requirements and map their detailed criteria into process models and controls that govern the creation and use of digital information assets throughout their lifecycle;
- exercise the requisite skills at analysing and navigating conflicting and non-aligned rule systems in order to structure unified approaches to governing information that are defensible and capable of rapid adaptation to changing requirements;
- evaluate alternative strategies for implementing information governance objectives across various tools, including policies, procedures, contracts, application designs, and cloud-based services; and
- use the knowledge gained to develop an integrated, substantive proposal for establishing or improving information governance within a defined scope of application.

B. Cloud Security

The successful participants will:

- be able to explain cloud architecture, properties, management services, and security challenges;
- understand security risks associated with different deployment models, and what could be done to address such risks; and
- experience a real demonstration, provided by on building a simple cloud using an appropriate management tool, managing it, and hosting a web-application on it.

C. Data Security and Privacy

The successful participant will:

- have an awareness of both the risks and threats associated with data security, as well as the relevant legislative and regulatory frameworks;
- be able to utilise established and emerging theory in the design of secure access mechanisms; and
- be in a position to reason about issues of privacy with respect to data release.

D. Design for Security

The successful participant will:

- know the strengths and weaknesses of different security design techniques; and
- be able to specify a security solution to fulfill specific design requirements.

E. Forensics

The successful participant will:

- understand how forensic principles and techniques relate to the investigation of software and systems;
- have a detailed knowledge of a systematic methodology for undertaking forensic investigations; and
- understand how the results of an investigation relate to those security measures that were taken during the design, development and implementation of a software-centric system.

F. Mobile Systems Security

The successful participant will:

- be able to describe the threat landscape for mobile devices and applications, and be able to map its co-evolution with security controls and anticipated trajectories for the future;
- have a working knowledge of the main sources of vulnerabilities in mobile applications — deriving from the whole hardware and software stack — and their impacts;
- understand the subject of mobile handset forensics, the difficulties to be encountered, and how the objectives for extracting evidence often conflict with keeping a device secure;
- understand the differing security and privacy requirements of sets of users and be able to implement privacy and security elements by design into mobile applications;
- be able to form a coherent design strategy for usable, friendly security in mobile applications whilst minimising the risk to users;
- be able to describe the future threat landscape for mobile and connected devices, understanding the physical security impacts of emerging technologies used in smart cities such as machine-to-machine; and

- understand the strengths and weaknesses of the mobile application lifecycle from digital signing of applications, application distribution through to methods for detecting maliciousness in applications, software upgrades and kill switches.

G. Network Security

The successful participant will:

- be able to explain how the architecture of the internet gives rise to security challenges;
- know and understand the major classes of security technologies used in best practice to improve internet security;
- understand how technology, practice, and procedure work together to deliver security in networked systems; and
- be able to extend their understanding to encompass the security of new and emerging kinds of network.

H. People and Security

The successful participant will:

- be able to specify usability criteria that a security mechanism has to meet to be workable for end-user groups and work contexts;
- be able to chose and configure mechanisms for best performance in a given organisational context; and
- be able to specify accompanying measures (policies, training, monitoring and ensuring compliance) that a user organisation needs to implement to ensure long-term security in practice.

I. Risk Analysis and Management

The successful participant will:

- be able to understand the main issues of risk in computer and information security;
- be able to conduct a security risk analysis and make cost-benefit decisions based on this; and
- have an overview of how risk analysis can be used to make a business case for security.

J. Secure and Robust Programming

The successful participant will:

- be able to explain the sources of failures in software written using modern high-level languages;
- have an understanding of the conceptual tools needed to mitigate and eliminate those failures;
- have gained practical experience in using tool-sets which permit the development of robust and correct software; and
- be able to place such practices appropriately within a systems development methodology.

K. Security and Incident Management

The successful participant will:

- have an understanding of the key themes and principles of security incident management, and be able to apply these principles in designing systems and models for managing security incidents;
- understand how to apply the principles of incident management in a variety of contexts, and be able to make a case to argue the extent to which technology can assist in the resolution of security incidents and how this is changing over time; and
- have an appreciation of the wider context of security incident management, and in particular of the relationship with business continuity and crisis management disciplines.

L. Security Principles

The successful participant will:

- understand the main issues in computer and information security;
- have practical experience in the analysis of secure communication protocols;
- have an overview of the scope of the current leading technologies and standards; and
- be able to evaluate security solutions.

M. Security in Wireless Networks

The successful participant will:

- understand the main security goals and adversarial models of wireless and mobile networks;
- gain a broad knowledge regarding real-world security architectures of WLANs, GSM/UMTS, WSNs, RFIDs, etc.;
- be able to reason about wireless security protocols and protection techniques, and discuss proposed solutions and their limitations; and
- have an overview of the recent advances regarding lightweight authentication, key management for wireless networks, secure localization, and wireless device pairing.

N. Trusted Computing Infrastructure

The successful participant will:

- be able to explain critically the notion of trust as embodied in trusted computing devices, and the requirements upon those devices;
- know the role and purpose of each element of the trusted platform module;
- be able to use the Trusted Software Stack API to interact with the TPM;
- understand how technologies of virtualization can combine with trusted platform modules to yield trusted infrastructure; and
- describe some systems architectures which use these capabilities to provide innovative and strong security solutions.