software engineering programme 2009

UNIVERSITY OF
OXFORD

Software engineering at Oxford

Welcome to the *Software Engineering Programme*, a centre for advanced education and applied research at the University of Oxford. Established in 1993, the Programme exists to make the connection between theory and practice in software engineering, and to make the expertise of the University available to those who wish to study while continuing in full-time employment.

Software engineering

'Software engineering' is the application of scientific and engineering principles to the development of software systems: principles of design, analysis, and management. The application of these principles makes it easier to develop software that meets its requirements, even when these requirements change; to complete the development on time, and within budget; and to produce something of lasting value, by being easy to maintain, re-use, and re-deploy.

An opportunity to learn

The Programme teaches the principles of modern software engineering, together with the tools, methods, and techniques that support their application. It offers working professionals an opportunity to learn more about the technological advances that are changing their lives, through a programme of part-time study at Master's level at one of the world's leading universities.

Courses and qualifications

The Programme offers seven postgraduate qualifications — *Postgraduate Certificate*, *Postgraduate Diploma* and *Master of Science*, both in *Software Engineering* and in *Software and Systems Security*, and a *Postgraduate Certificate in Object Technology*. These qualifications can be obtained through attendance on courses in a variety of software engineering subjects, such as requirements engineering, security risk and threat analysis, service-oriented architectures, design patterns, software testing, formal techniques, and development management. The courses can also be taken separately, without registering for a qualification, as short programmes of advanced professional training.

"Conceptual integrity
is the most important consideration
in system design."
(Fred Brooks)

Mode of study

Each course is delivered by an expert in the subject, and includes an intensive teaching week of classes, lectures, and practical work, followed by a written assignment. This mode of study is particularly effective for those with significant professional or personal commitments. Courses take place at the Programme's purpose-built facilities in the centre of Oxford, and class sizes are kept small to allow for a high degree of interaction.

Who should apply?

The Programme is intended for those with an awareness of software development issues; existing students include architects, programmers, managers, and informed customers. The entry requirements for the postgraduate qualifications reflect this: relevant industrial experience is valued as highly as previous education. There are no formal entry requirements for the individual courses, although prospective attendees are asked to confirm that they satisfy the informal requirements for any course that they wish to attend.

Research activity

The Programme is also a centre for research activity. The academic staff are involved in a number of national and international projects in the areas of: large-scale data integration and sharing; medical informatics; virtual research environments; ubiquitous computing; applications of trusted infrastructure technologies in distributed systems; languages and tools for object model transformation; programming languages; and model-driven software engineering.

Postgraduate qualifications

All of the courses offered by the Programme are taught at Master's level: the various postgraduate qualifications differ only in focus and extent. All successful applicants for postgraduate study are registered initially for a Postgraduate Certificate. Registered students may change from one specialism or qualification to another at any time, provided that they are making satisfactory progress.

Postgraduate Certificate

This qualification requires attendance on courses in four subjects, and the successful completion of the corresponding written assignments. For the specialised Postgraduate Certificates in *Software and Systems Security* and *Object Technology*, at least three of the subjects must be chosen from the specialism; the fourth subject may be freely chosen. The courses and assignments must be completed within two years of the date of admission.

Postgraduate Diploma

This qualification requires courses in eight subjects. For the specialised Postgraduate Diploma in *Software and Systems Security*, at least five of these subjects must be chosen from the specialism; the remainder may be chosen freely. The courses and assignments must be completed within three years of admission.

Master of Science

This qualification requires courses in ten subjects; it also entails the completion of a project and dissertation, involving participation in a project module and a dissertation module. For the specialised MSc in *Software and Systems Security*, at least six of these subjects should be chosen from the specialism, with the remainder being freely chosen. The courses and assignments should be completed within four years of admission; an additional year is available, if needed, in which to complete the dissertation.

Admission

Applications for part-time, postgraduate study are invited from anyone with sufficient experience or proven ability in the field of Software Engineering.

A typical applicant might have an undergraduate degree in a related subject, and at least two years' experience of software development in an industrial context. However, relevant experience may compensate for a lack of formal qualifications, or vice versa — candidates may contact the Programme Office for further information.

The open nature of the entry requirements means that a formal interview is an essential part of the admissions process. In advance of the interview, applicants are asked to present two references from people who are familiar with their work or study achievements and — if appropriate — to secure the support of their employer.

Successful applicants will become registered students of the University with effect from the beginning of the next University term. New students who have already attended courses on the Programme can use those courses as credit towards their qualification, provided that the course dates are within one year of the date of admission.

Accreditation

The MSc in Software Engineering is accredited by the British Computer Society: graduates from the Programme can use the MSc to gain exemption from the Professional Graduate Diploma and PGD Project. The individual courses can be used as credit in the IEE, IAP and BCS continuing professional development schemes, and have a CATS value of 15 points for credit transfer between postgraduate programmes.

Software engineering methods

The pressing problems in the development of large and complex systems are typically matters of communication as much as anything else — getting the right people to talk to each other, addressing the concerns of multiple stakeholders, producing models to promote shared understanding and validate assumptions, and so on. One theme of courses offered by the Programme is in methods for modelling, reasoning about, and managing the software development process.

Software Engineering Mathematics (SEM)

An important characteristic of a specification is the ability to reason about the objects it describes, and thus about the system it models. If that specification is written mathematically, the reasoning can take the form of calculation: we can use simple, logical methods to check consistency, and to predict the consequences of design decisions. This course is an introduction to mathematical description, using basic set theory and logic in the Z notation. It serves as an introduction to the Software Engineering Methods theme.

Specification and Design (SDE)

We deal with complexity by structuring our specifications: identifying patterns, and finding appropriate abstractions. This course shows how to organise specifications using the 'schema' language of Z, to produce descriptions of component functionality that are concise, precise, and comprehensible. It shows also how these descriptions can be analysed to check for consistency of requirements, to validate invariants or assertions, and to determine the preconditions of operations.

Concurrency and Distributed Systems (CDS)

The consequences of design decisions are particularly hard to predict in the presence of concurrency or complex patterns of interaction. This course presents a powerful technique for describing the intended behaviour of concurrent systems, and for reasoning about the interactions that emerge. The technique is based upon the language of Communicating Sequential Processes (CSP).

Model Checking (MCH)

It is impossible to prove the correctness of a complex design without automated reasoning. One of the most powerful forms of automated reasoning is model-checking, in which every configuration of a given design is automatically explored and validated. This course presents a set of practical techniques for the analysis of patterns of interaction, and shows how the application of these techniques can be supported by model-checking tools.

Performance Modelling (PMO)

This course shows how simple techniques from continuous mathematics can be used to predict the behaviour of systems. Formulae and theorems from the theories of probability and random processes may be applied in software and communications performance modelling. It explains how to predict congestion, calculate expected loads, and develop strategies for integrated services.

"Correctness is clearly the *prime quality*. If a system does not do what it is supposed to do, then everything else about it matters little." *(Bertrand Meyer)*

Software Development Management (SDM)

Management is an essential element of successful software development. This course gives an introduction to the fundamental aspects of project and people management that are required to successfully run a software development project. It is aimed at those with some experience of working in a project-based software environment but with little or no previous experience of project management or team leadership.

Agile Methods (AGM)

Agile methods are challenging conventional wisdom regarding systems development processes and practices, effectively 'putting process on a diet' and investing in people and teams. This course enables today's software development professional to understand the heart of agility, and covers both the theory and practice of agile methods such as XP, Scrum, Crystal, FDD, Lean, and DSDM.

Requirements Engineering (REN)

Establishing firm and precise requirements is an essential component of successful software development. Requirements may be technical, although these are often the least problematic; successful analysis requires broader investigation, addressing the human context of current and future work practices. This course covers a range of methods from 'hard' semi-formal approaches, to 'softer' people-oriented ones.

Process Quality and Improvement (PRO)

Every software development organisation needs to be focused on the delivery of quality. The software engineering discipline responds by calling for a managed process for the construction and testing of software, and for the improvement of that process. This course explains the necessary concepts within the frameworks provided by three important international standards.

Management of Risk and Quality (MRQ)

Too many project planning approaches concentrate on just the estimating and network aspects of planning. This is of little value if the project is given the wrong shape or the wrong activities are chosen in the first place. The *Strada method* taught in this course builds the project from an analysis of the specific risks to be faced. It then uses an analysis of the specific quality requirements to complete the planning. The two perspectives of risk and quality prove sufficient to give the basis for a reliable and robust plan.

Software engineering tools

For the fundamental principles of computing to scale up to the engineering of industrial-size systems, it is essential to make good use of tools to automate the mechanical aspects of development, freeing up the human involvement to address the more challenging aspects that require creativity and insight. A second theme of courses offered by the Programme covers tools for software development, particularly languages for design and programming.

Object Orientation (OOR)

Object orientation involves the encapsulation of functionality and associated data as objects, and the classification of those objects into a hierarchy of classes. This makes it easier to control complexity, and encourages the development of flexible, robust, and re-usable components. This course covers the principles of object orientation, offering both an introduction for beginners and a wider perspective for experienced users of object technology; it also serves as an introduction to the Software Engineering Tools theme.

Object-Oriented Design (OOD)

This course teaches standard techniques for the specification of software from a variety of perspectives: conceptual and physical, static and dynamic, requirements and designs. The course is based around a carefully-chosen subset of the Unified Modeling Language (UML), and places the techniques in a formal software engineering context.

Object-Oriented Programming (OOP)

This course teaches the concepts and principles of object-oriented programming. The language used is Java, although most of the material covered will apply equally well to any other object-oriented language: objects, methods, interfaces, messages, and events.

Design Patterns (DPA)

This is an advanced course in the structure and behaviour of object-oriented systems, covering both design and programming. It is based around the notion of a *design pattern*: an abstraction of a proven solution to a recurring problem in a specific context in system design.

Software Testing (STE)

This course presents realistic, pragmatic steps for rigorous and organised software testing. It clarifies testing terminology and covers the different types of testing performed at each phase of the software lifecycle, together with the issues involved in these types of testing. The course discusses how tests can be derived from requirements and specifications, design artifacts, or the source code, and introduces appropriate testing tools with hands-on exercises.

Database Design (DAT)

Relational database technology is the dominant approach to information storage, with products that offer an unmatched combination of abstraction and performance. To use these products effectively, however, requires an understanding of the underlying principles and concepts: relational modelling, normalisation, query optimisation, transactions, and distribution.

"If you don't think carefully, you might think that *programming is just typing* statements in a programming language." *(Ward Cunningham)*

Service-Oriented Architecture (SOA)

The current consensus on best practice for building component-based distributed applications is to use a *service-oriented architecture*. Services are encapsulated behind carefully-designed simple interfaces, and the realities of heterogeneity, decentralisation and fault tolerance are embraced rather than ignored. This course provides an understanding of the strengths and weaknesses of service orientation, informed by an ability to implement and deploy simple web services using a suitable development platform.

Mobile and Sensor Networks (MOB)

Recent advances in wireless mobile and sensor technologies have changed computing, enabling application scenarios in which large numbers of pervasive computing devices are connected to a wireless networking infrastructure in an ad hoc manner. This course covers communication protocols for mobile ad hoc networks, and provides an overview of distributed data management techniques for resource-constrained sensor networks.

Functional Programming (FPR)

In functional programming, computations are modelled as expressions rather than actions. This offers significant opportunities for parametrisation and modularisation, beyond those available in conventional, imperative programming. It also results in the production of programs that are clearer, simpler, and often surprisingly concise. The techniques taught in this course will be useful in any language — particularly for the definition of transformations on structured data.

Extensible Markup Language (XML)

The Extensible Markup Language (XML) is a notation for the definition of document structures, and the production of structured documents. It can be used to define application-specific representations that are easy to process and transform, facilitating the interchange of information between different systems and components. This course teaches the essentials of the language, document validation, the creation of stylesheets, and the use of core XML technologies to solve software engineering problems.

Software and systems security

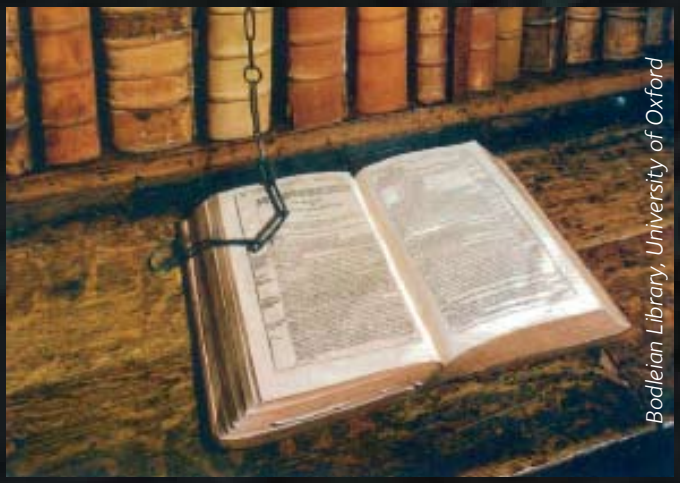
As computing systems become more essential to our daily lives, it becomes ever more important that the services they provide are available whenever we need them. We must also be able to rely on the integrity of the systems, and thus the information that they hold and provide. What is more, our society and our economy depend upon certain pieces of information being held in confidence. The third theme of courses on the Programme concerns software and systems security, from both social and technical points of view.

Security Principles (SPR)

This course combines a treatment of the fundamental principles of cryptography and security protocols with a practical treatment of current best practice. It explains the need for computer security, and the scope of the available technical solutions; presents techniques for evaluating security solutions; and provides an overview of the current leading technologies and standards in the security arena. It also provides an introduction to the whole Software and Systems Security theme.

Design for Security (DES)

This course explores how cost-effective solutions to security needs can be achieved by following well-established architectural practices and detailed security principles. Central to these considerations is meeting requirements with established solutions, and striking a balance between security and other system requirements.



Bodleian Library, University of Oxford

Secure and Robust Programming (SRO)

Many system failures and security vulnerabilities arise at the programming level. These can often be attributed to inadequate handling of exceptional situations, poor understanding of the details of the programming language in use, and incomplete descriptions of the interfaces between components. This course addresses those problems from a programming perspective, with the aim of improving the practitioner's capability in writing and reviewing code. Topics include animation of specifications, static analysis, design by contract, run-time assertion checking, and compile-time verification.

Trusted Computing Infrastructure (TCI)

A secure system is the product of numerous layers that operate together to provide in-depth protection. This course looks at the various platforms upon which a secure system operates, with an emphasis on practical and repeatable means of implementing these platforms securely. Topics covered include buffer overflows, cryptographic libraries, sandboxing, virtualisation, trusted computing, and database security, building towards a toolkit of sound principles for secure systems implementation.

Security Risk Analysis and Management (RIS)

Security is a property of an entire system in context, rather than of a software product, so a thorough understanding of system security risk analysis is necessary for a successful project. This course introduces the basic concepts and techniques of security risk analysis, and explains how to manage security risks through the project lifecycle.

"If you think technology can solve your security problems, then you don't *understand the problems* and you don't *understand the technology.*"
(Bruce Schneier)

Safety Critical Systems (SCS)

Computers are often placed in control situations within safety-critical systems. Safety is an emergent property of whole systems; software may play only a small part. This course will enable the systems engineer to determine whether a safe system can be built, and what requirements must be placed on software in order to keep risk at an acceptable level.

Forensics (FOR)

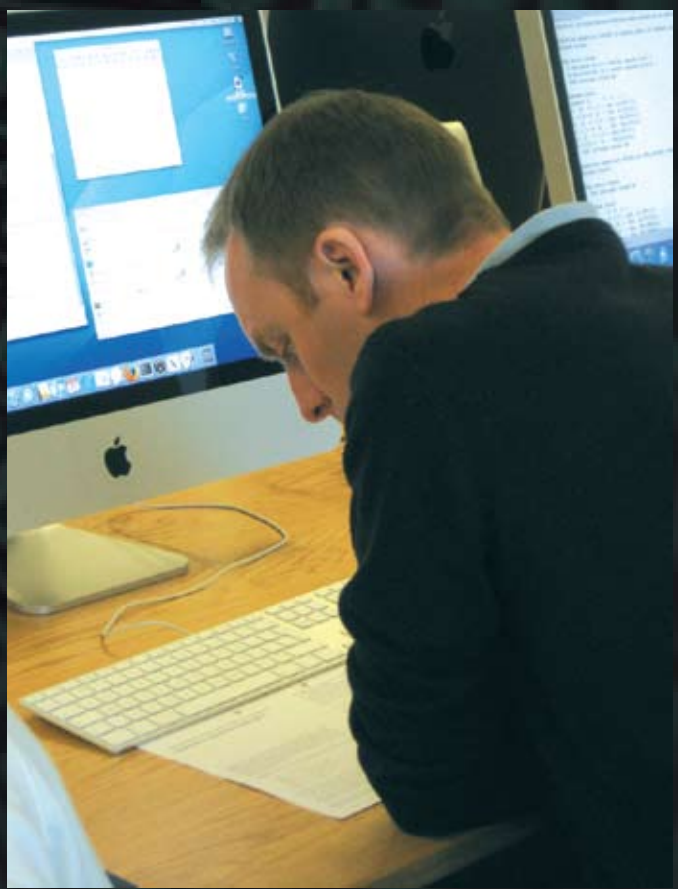
Investigating computer crime is a delicate and involved process that requires an understanding of the evidential standards necessary in various contexts where electronic forensic data may be needed. This course describes the current best practice in both understanding and deconstructing an attack whilst preserving evidence, and also explores how to design and evaluate systems in order to facilitate forensic examination.

People and Security (PAS)

A very high proportion of failures in security can be attributed to human weakness, misunderstanding, misinformation, or failure to grasp the importance of the processes individuals are expected to follow. This course draws on work from human-computer interaction, and more widely from psychology, relating these issues back to hard technical implementation decisions.

Network Security (NES)

Networking technologies play a critical role in almost all modern software-based systems, whether the fixed networks of computers we now regard as commonplace, or the growing cloud of pervasive devices which have increasingly diverse profiles of network connectivity. As a result, networks provide a potential vector for many forms of attack, and are an ideal location for many threat mitigations and isolation technologies. Much benefit has been derived from a layered approach to network architecture, and most approaches to security are aligned to those boundaries. This course will consider the prevention, detection, mitigation, and remediation of security problems in the network at each level of abstraction, as well as looking at cross-cutting concerns across the whole stack.



Background

The Software Engineering Programme is the result of a collaboration between Oxford University Computing Laboratory (OUCL) and Oxford University Department for Continuing Education (OUDCE). OUCL has an international reputation for linking mathematical theory to industrial practice. OUDCE has a distinguished history in promoting life-long learning, including delivering high-quality education to working professionals.

The discipline of software engineering has changed considerably during the lifetime of the Programme: although the principles remain the same, the context in which they are applied is evolving. As a result, the Programme is in a state of constant development, placing greater emphasis on different principles, and finding new ways to relate them to industrial practice.

Programme staff

There are seventeen core members of staff, all of whom work exclusively for the Programme: *Alessandra Cavarra*, Lecturer; *Andrew Cooper,* Teaching Assistant; *Edward Crichton*, Researcher; *Jim Davies*, Professor and Director; *Melissa Endacott*, Administrator; *Ivan Flechais*, Lecturer; *Jeremy Gibbons*, Reader and Deputy Director; *Ralf Hinze*, Reader; *Jackie Jordan*, Manager; *Eric Kerfoot*, Teaching Assistant; *Andrew Martin*, Lecturer and Deputy Director; *Steve McKeever*, Lecturer; *Shirley Sardar*, Administrator; *Clint Sieunarine*, Teaching Assistant; *Andrew Simpson*, Lecturer; *Niki Trigoni*, Lecturer; *Chen-Wei Wang*, Teaching Assistant.

Subject specialists

There are also a number of subject specialists, who teach courses in their particular areas of expertise: *Paul Beaven,* Independent Consultant; *Rob Collins*, Independent Consultant; *Gareth Digby*, Director, Packer Engineering; *Paul Gibson*, Development Operations Manager, IBM; *Conrad Hughes*, Research Fellow, Edinburgh; *Marina Jirotka*, Reader, Oxford; *Nigel Kermode*, Independent Consultant; *Robert Leese*, Director, Smith Institute; *Angela Martin*, Independent Consultant; *Bill Roscoe*, Professor, Oxford; *Angela Sasse*, Professor, London; *Mark Slaymaker*, Researcher, Oxford; *Perdita Stevens*, Reader, Edinburgh.

Organisations

The Programme benefits greatly, in terms of advice and sponsorship, from a number of leading companies and organisations, especially EPSRC, IBM, Motorola, Marconi, QinetiQ, SEEDA, the Smith Institute, and Symbian. Representatives from most of these organisations sit on the advisory panel for the Programme, helping to ensure that its teaching reflects and anticipates industrial needs.

The students come from a wide range of organisations: software companies; healthcare providers; finance houses; government agencies; bioinformatics start-ups; management consultants; power companies; car manufacturers. A significant number are self-employed, or work for small enterprises. They share an awareness of software engineering practice, and a desire to learn more about the underlying principles.

Kellogg College

Every student registered for an MSc degree must become a member of an Oxford college. There are nearly forty colleges, each with its own style and tradition. Not all of them take part-time students; one of them is particularly associated with part-time study. It was named in honour of William Keith Kellogg, industrialist and philanthropist, in recognition of the support given to adult and continuing education by the W. K. Kellogg Foundation.

Kellogg College has approximately 100 fellows (including all of the Programme's University Lecturers) and approximately 430 registered students. Most of the Programme's students who progress to the MSc apply to Kellogg. The College organises regular events, and acts as an additional point of contact between MSc students and the University.

Testimonials

Hundreds of students have taken part in the Software Engineering Programme since it has started. They have benefited in a variety of different ways. Here are some testimonials from past and present students.

Formal qualifications

A significant proportion of students on the Programme do not have formal qualifications in software engineering or computer science, but have moved into the software industry from other fields. Many of them value the opportunity to validate their experience with a recognised academic qualification.

- *"I joined the Programme because I work in IT, but my background is in physics. It was an ideal opportunity to get a formal qualification in the area that I work in."*
- *"When I started the Programme, I didn't know anything about software engineering apart from what I learned on the job. I thought that it was about time that I found out really how to engineer software, so that I could lay claim to the title of software engineer."*
- *"I started the Programme at a time in my life when I had just entered the software industry. I had a background in industrial engineering, but no formal knowledge of software engineering. It provided an excellent background for me, allowing me to choose the subjects that were most important to my career."*

Promotion and mobility

Students often find the additional qualification a help when it comes to promotion or to changing jobs.

- *"Studying on the Programme has made people treat me differently: they see that I am investing in my long-term career."*
- *"The Programme has had a significant effect on my career. Just after I got my degree, I went for a new job; the one thing I was told in my interview was that the Oxford University MSc looks very impressive on a resumé. It has certainly opened a number of doors for me."*

Practical applications

Many students have found immediate practical applications of the material they have studied on the Programme.

- *"Just after the XML course, I had a contract that involved building a report generator. The skills I gained on the course helped me solve that problem better and faster. My client was so happy that they extended my contract. The additional income far outweighs the cost of studying."*
- *"The Programme has provided a lot of technical advantages to my company — the knowledge I have gained has been put to good use. We've had a huge amount of fun in applying all the techniques we have learned. We now have the basis of what we believe to be a unique and world-beating product and we are in the process of marketing it."*

Taking part

To reserve a place on a course, you need first to be registered with the Programme (just click on *Sign in or Register* on the front page of the website). If a place is available, you will receive a confirmation from the Programme Office. For courses in certain subjects, you may be asked to confirm that you have sufficient experience before your reservation can be approved.

Postgraduate study

To apply to study for a postgraduate qualification, you should follow the procedure on the Programme website, or request an information pack from the Programme Office. Either route provides instructions on what supporting documentation you should supply.

Only those applicants with an appropriate combination of education and experience will be called to interview. Applicants should not make travel plans, or accept contingent financial (or other) commitments, unless and until a formal, written offer of a place has been made. The allocation of a place on an individual course does not imply admission to a postgraduate qualification.

Fees

There is a standard fee of £1400 for each course attended, payable in advance. This includes course materials, and lunches during the teaching week, but not accommodation. The fee applies whether or not the attendee is working towards a postgraduate qualification, and regardless of nationality and residency. The Postgraduate Certificate will typically entail four course attendance fees, the Postgraduate Diploma eight, and the MSc ten. The MSc also requires attendance at a week-long project module in Oxford, and passing a dissertation module. There is no course fee for either of these modules — the cost is included in the student registration fee for the MSc.

Students may cancel attendance, provided that the cancellation is received well enough in advance. Cancellations at short notice may not receive a full refund of the course attendance fee.

The examination and assessment of the course assignment is included in the course attendance fee. Should a student be granted permission to take the assignment for a later course in the same subject, an additional examination fee of £100 will apply.

There is an annual student registration fee of

£2520 for any of the postgraduate qualifications; this is payable for one year for a Postgraduate Certificate, two years for a Postgraduate Diploma, and four years for an MSc. Students who are citizens of a member state of the European Community, and have been ordinarily resident in the European Economic Area for the past three years, may qualify for Home/EU status, and a reduction of this annual award fee to £1490.

The following table gives the current cost of each qualification, assuming the expected, minimum number of course attendances.

	Home	*Overseas*
PGCert	7090	8120
PGDip	14180	16240
MSc	19960	24080

All payments must be completed before any postgraduate qualification will be awarded by the University. Award fees are based upon the date of admission. Attendance fees are based upon the date of the course, and may increase during the period of study, typically in line with the rate of inflation in the UK.

Contact

For enquiries, contact the Programme Office:

Software Engineering Programme
Wolfson Building, Parks Road
Oxford OX1 3QD, UK
+44 1865 283525 (phone), 283531 (fax)

The most effective means of contact is email:

office@softeng.ox.ac.uk

Further information about the Programme, together with application forms, can be found on the Programme website:

www.softeng.ox.ac.uk