MSc in

Software Engineering

(part-time)

UNIVERSITY OF OXFORD

flexible, professional education

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.

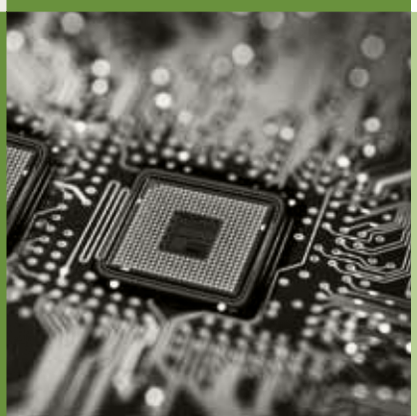
The Software Engineering Programme at the University of Oxford teaches the principles of modern software engineering, together with the tools, methods, and techniques that support their application. It offers a flexible programme of short courses to those working full time in industry or in the public sector. It is accessible to anyone with the right combination of previous education and practical experience.

The courses on the Programme can be used as individual programmes of professional training in specific subjects, or as credit towards a Master of Science (MSc) degree in Software Engineering from the University of Oxford. Students on the MSc take between two and four years to complete a minimum of ten courses, typically at a rate of three courses per year, earning a degree while in full time professional employment. The courses may be taken in any order and combination, depending upon previous experience and education.

Each short course is based around a week of intensive teaching in Oxford, with some initial reading to consider beforehand, and a six-week assignment to complete afterwards. The teaching week allows you the chance to explore a subject in depth, with expert teaching and supervision, away from the demands of work and family. The reading gives you the opportunity to prepare yourselves; the assignment, an opportunity to deepen and to demonstrate your understanding.

Course Structure

Pre-Study	Teaching Week	Assignment
4 weeks	1 week	6 weeks



Courses in Software Engineering

Software Engineering Methods

These courses assume an understanding of the issues and challenges of software development.

Agile Methods AGM

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

Software Development Management SDM

SDM presents the skills required for the successful delivery of complex and innovative software projects, giving students a thorough grounding in the methodologies and practice of people, project, and development management, enhanced by industry guest speakers and by syndicate exercises. On completion of the course students will be able to assess a software development situation and select an appropriate management strategy.

Process Quality & 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 & Quality MRQ

Too many project planning approaches concentrate on just estimating and network aspects. This is of little value if the project is given the wrong shape or the wrong activities are chosen in the first place. The approach taught in this course builds the project from an analysis of the specific risks to be faced, in order to 'manage quality up and risk down'.

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. .

Software Engineering Mathematics SEM

It is well known that software-based systems are extremely complex entities; it is also well known that abstraction offers the opportunity for software engineers to focus on key aspects of behaviour. This course shows how to use basic logic and set theory to describe, reason about, and understand properties of software and systems. The techniques presented are given in terms of the Z formal description technique.

Specification & Design SDE

State-based modelling and analysis provides the opportunity to develop and reason about models of systems in terms of the potential effects of operations on the system under consideration. This course builds upon the material presented in SEM, and considers how an approach based on well-founded mathematical concepts might be incorporated into the software development process. In doing this, the course gives consideration to both the Z and B languages.

Concurrency & 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

Model-checking, one of the most powerful forms of automated reasoning, automatically explores and validates every configuration of a given design. Without such techniques, it is impossible to prove correctness of complex designs. This course presents practical techniques for the analysis of patterns of interaction, and automated tools for the application of these techniques.

Software Engineering Methods *...continued*

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 considers the specific issues, problems and techniques associated with analysis, design, development and verification of systems that will be used in safety-critical applications.

Performance Modelling PMO
This course presents techniques for modelling the performance of computing and communications systems. It covers tools, techniques, and analytical methods to improve the efficiency or productivity of existing or planned computer systems. In particular, it addresses the problem of how to design for the best balance of system behaviour, performance, and workload.

Software Engineering Tools

These courses assume a familiarity with modern programming languages, tools, and techniques.

Functional Programming FPR
In functional programming, computations are modelled as expressions rather than statements. This offers significant opportunities for parametrisation, modularisation, and optimisation, beyond those available in imperative or object-oriented programming. It also results in programs that are clearer, simpler, and often surprisingly concise. This course uses Haskell, but the techniques and concepts are useful in any language – particularly for transformations on structured data.

Concurrent Programming CPR
The next generation of soft real-time server-side applications will only scale through massively concurrent programs executing on multi-core processors in a distributed environment. Erlang is an open source language with lightweight processes, no shared data, and built-in distribution, catering for these kinds of problem. This course uses Erlang to implement highly concurrent, massively scalable, soft real-time systems, with an emphasis on fault tolerance and high availability.

eXtensible Markup Language XML
XML is a universal notation for creating languages, be they data or instructions. This course teaches one how to create such languages, how to validate them and how to transform them. All using the same generic framework of XML. We use a variety of practical examples that highlight fundamental issues and demonstrate how XML can be applied to a variety of Software Engineering disciplines.

Service Oriented Architecture SOA
SOA represents a convergence of ideas from object orientation, distributed systems, and component-based development, underpinned by cross-platform protocols based largely on XML. This course provides an understanding of the strengths and weaknesses of SOA, informed by an ability to implement simple web services using a suitable development platform. It covers the definition of applications as combinations of services, and emergent properties of those compositions.

Agile Practices in Engineering APE
In this course we cover engineering practices that support frequent, reliable delivery of software in an agile environment. We look at techniques such as continuous integration and pair programming, as well as automated quality assurance, release and deployment. We show how these methods may be applied to greenfield or legacy projects.

Software Testing STE
Software Testing is a key aspect of the system development. This course provides all of the key knowledge and skills required to both lead a software test organisation and to be actively engaged in software testing. The course details processes, plans, methods and tools for test and presents a full life-cycle approach to testing. It covers functional, non-functional, performance and security testing.

Database Design DAT
Database Design introduces the fundamentals of the relational model, including the relational algebra and calculus. It explores how to design relational databases that fit business requirements and covers the topics of normalization and orthogonal design. It teaches how to query a relational database using SQL, and highlights where SQL deviates from the relational model. Finally, it touches upon query optimization, transaction management and distributed databases.

Mobile and Sensor Networks MOB
This course presents communication protocols and management techniques for wireless, mobile, and ad hoc networks. It introduces application scenarios, models and challenges of these networks; it then focuses on wireless sensor networks, and presents in-network processing and storage management techniques for resource constrained sensor systems. Finally, it introduces the concept of delay-tolerant networks, and touches upon epidemic and gossip-based protocols.

Software Engineering Tools

...continued

Object Orientation OOR

Objects are fundamental to object orientation: an entity that binds together code and data, and is accessed through an interface. When programming in the large, an object-based design controls complexity and encourages code reuse. This course takes a closer look at the cohesion, coupling, and subtyping of objects, offering an introduction for beginners and a wider perspective for more experienced programmers.

Object Oriented Design OOD

This course teaches standard techniques for the specification and design of software systems. The notation of the Unified Modeling Language (UML) is presented, via a number of case studies. The course describes fundamental principles of object-oriented modelling, requirements development, and design, showing how to effectively use system requirements to drive design and development. It also introduces design-by-contract and the Object Constraint Language.

Object Oriented Programming OOP

This course teaches the concepts and principles of object-orientation. OOP builds on OOR, however, it shifts the focus from logical design to the impact that the concept of an object has on practical programming. While the language used is Java, most of the material covered will apply equally well to any other object-oriented language: objects, messages, interfaces, exceptions and generics.

Design Patterns DPA

This is an advanced course in the structure and behaviour of object-oriented systems. 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. The course covers both the philosophy and the practice of patterns, in both design and programming.

Software Product Lines SPL

The SPL approach to software development promises significant improvements in time-to-market, cost, and reliability, through the systematic identification and exploitation of commonalities and variations in software systems. The approach promotes asset reuse throughout the software life cycle, and facilitates product customisation; it has been applied successfully in a number of different domains, by large and small organisations alike.

Courses in Software and Systems Security

A range of other courses are available, addressing subjects in software and systems security. These may address complementary topics, or provide useful background, for the study of software engineering.

Security Principles SPR

This course teaches the fundamental principles of information and systems security. It explores a wide range of technologies, examines security standards and expectations, and explains techniques for the evaluation of security requirements and solutions. It places theoretical work on protocol design, cryptography, and information flow firmly in the context of existing and emerging practice.

Design for Security DES

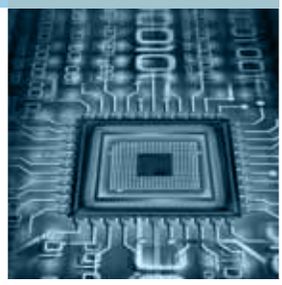
Security is a system-level property, and emerges from the coordinated design of components and processes. This course shows how a range of factors, from architectural patterns to detailed technical controls, can be considered together in the production of cost-effective solutions. It addresses the challenge of providing security, through a combination of infrastructure, mechanisms, and procedures, while satisfying requirements for functionality and usability.

Security Risk Analysis & Management RIS

The concept of risk is central to software and systems security. Understanding the ways in which systems are vulnerable to threats needs to inform the selection and prioritisation of security measures. This course teaches a principled approach to risk analysis, explores the techniques and practices of risk management, and demonstrates their application through a realistic set of examples and case studies.

Forensics FOR

Investigating computer crime is a delicate process that requires a deep understanding of the evidential standards expected of electronic forensic data. This course describes the best practice in deconstructing an attack whilst preserving evidence, and explores how to design and evaluate systems in order to facilitate forensic examination. It combines a principled approach with practical work, recovering data using low-level tools.



"The MSc looks impressive on a resume. It has certainly opened a number of doors for me"

Trusted Computing Infrastructure TCI

A secure system relies on numerous layers operating together. This course looks at the platforms underpinning secure systems, with an emphasis on practical means of implementing these securely. It examines roots and chains of trust, operating systems, trusted platforms, and virtualisation. It shows how these are applied to secure networking, remote working, trusted storage, and remote computation in grids and clouds.

People & Security PAS

Many failures in security can be attributed to human weakness, misunderstanding, or failure to grasp the importance of prescribed processes and procedures. The interaction between people and technology often presents a significant challenge to secure operation. This course teaches techniques drawn from human-computer interaction and psychology, addressing this challenge within the context of hard, technical decisions.

Network Security NES

Networks are a potential vector for many attacks, and are an ideal location for threat mitigation technologies. This course teaches approaches to prevent, detect, and remediate security problems in the network at each layer, as well as looking at cross-cutting concerns across the networking stack. It examines the strengths and weaknesses of boundary protections, intrusion detection and prevention, and privacy-preserving routing.

Data Security DAS

Issues of data security are becoming increasingly important in many contexts; further, it is becoming more difficult to separate the design and implementation of technical solutions in this area from overarching ethical and legal concerns. This course gives consideration to recent developments in the fields of information security and privacy, and reflects upon how they impact upon models of data access, release and aggregation.

Security Incident Management SIM

Managing security incidents is critical to business continuity. Incidents range from the small and predictable, which can be eliminated through operation controls, to the large and unpredictable, where standard management controls may not work. This course teaches the principles of incident management in practice, and identifies key themes for effective response to events that impact upon businesses, governments, and individuals.

Secure & Robust Programming SRO

Secure and Robust programming focuses on the low level aspect of high integrity code generation. We revise fundamental aspects of logic to provide a robust foundation to specification. We show how logic programming can be used to animate specifications and create effective static analysis tools. We then proceed to Design by Contract and through the use of a case study, show how to annotate programs with behavioral constraints to ensure a weak form of correctness.

Cloud Security CLS

Automated self-managed services – for software, platforms, and infrastructure – can provide significant convenience for local administration, yet also remove many tools and controls commonly used, while introducing new risks and threats. This course reviews the architectural principles of cloud computing, describes threats and security controls at each level of abstraction, and addresses cloud management services for trustworthy, secure, and resilient operation.

Mobile Application Security MAP

Mobile devices present distinctive challenges for security, including problems of device association, power constraints, and restricted interfaces. Mobile applications often incorporate both local and remote services, complicating the management and enforcement of security policies. This course presents a range of techniques for the design and implementation of secure mobile applications, balancing the requirements of functionality, security, resource utilisation, and privacy.

MSc in Software Engineering

A postgraduate degree is evidence of individual ability and understanding beyond the expectations of industry training, undergraduate education, and professional experience. It is a demonstration that you have achieved a mastery of the subject: that you can select, adapt, and apply appropriate techniques; that you can evaluate what is, and what is not, working; that you can anticipate, and facilitate, change. This kind of evidence can be invaluable in the workplace: to lend additional authority to your opinions; to better establish your credentials; and to reassure others as to your suitability for new roles and responsibilities.

A postgraduate degree is also an opportunity for personal development: a chance to test your ideas and intuitions, to experiment with new tools and techniques, and to make new connections between theory and practice. It is an environment in which you can take a step back from the immediate demands and compromises of your latest project, and think more strategically about the nature of the problems you encounter, and how they might be solved more efficiently, and more effectively. This kind of opportunity can be invaluable in your personal life, bringing new confidence, skills, and inspiration.

The University of Oxford

The courses take place in a purpose-built teaching facility in Oxford, part of the new building for the Department of Computer Science. Each course is taught by a subject expert: a member of faculty, or an industrial practitioner. The students will bring a varying combination of expertise and experience: some will be developers, managers, or consultants; others will be architects, designers, or testers. Class sizes are kept small to facilitate learning and interaction.

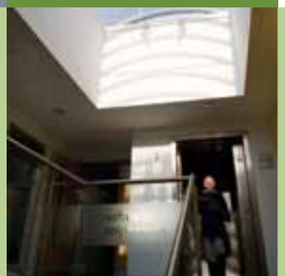
All students are members of the University's Department of Computer Science, a recognised centre of excellence for teaching and research in computing and related disciplines. The University of Oxford was the first university in the English-speaking world, and is consistently ranked among the ten leading universities globally.

Each student will be a member also of one of the Oxford colleges. Several colleges offer places for this course, but most students choose to belong to Kellogg College, a college established specifically to meet the expectations of students on professional and non-residential programmes. If a student on the programme already has a degree from Oxford, and was a member of a different college for their previous period of study, then they may prefer to return to that college. All of the teaching faculty whose teaching is primarily for part-time students are themselves members of Kellogg.

.

UNIVERSITY OF OXFORD

DEPARTMENT OF COMPUTER SCIENCE



Studying on the Programme

Getting Started

All of the courses described above can be taken as individual programmes of professional training. You may book a place on any course on-line, or by calling the Programme Office. One month before the teaching week – or upon payment of the invoice, if later – you will be sent some initial reading material. The teaching week itself runs from 9am to 5pm from Monday to Thursday, and from 9am to 12.30pm on Friday. At the end of the week, you will be given an assignment task: you can take this, and get feedback on your submission, even if you have no plans to use the course as credit towards a postgraduate qualification.

To study for a postgraduate qualification – the MSc in Software Engineering – you need to make a formal application to the University. You can take up to two courses before doing this and still use them as credit, provided that you complete the assignments. If you appear to meet the admission criteria, you will then be invited for an interview, where you will have the opportunity to discuss your expectations, your study plans, and your readiness to take part in a programme of part-time, professional education. If your application is successful, then you may be admitted at the beginning of the next term: in January, April, or October.

The admission criteria are straightforward; relevant industrial experience is valued as highly as previous education. At a minimum, we expect applicants to have either a degree-level qualification in a related discipline, or a substantial record of practical achievement in software development in a professional context. We will ask also for at least two references, one of which will normally come from your current employer.

www.softeng.ox.ac.uk/apply

Course Selection

Each of the courses is designed to work as a separate programme of learning, and courses in related subjects can be taken in any order. Some of the courses assume familiarity with material taught in others: if you are not already familiar with this material, then you may obtain greater value by attending the other courses first. Advice and guidance on course selection can be obtained from the Programme Office.

"I work in IT, but my background is in physics. This was an ideal opportunity to get a formal qualification in the area that I work in."

Academic Awards

To be awarded an MSc in Software Engineering, you will need to attend ten short courses, complete the corresponding assignments, and write a dissertation based upon a research project of your own design. You have four years from the date of admission to do this, although more time will be allowed in exceptional circumstances. Most students take three or four years to complete the MSc; some take two years, which is the minimum period allowed between admission and graduation.

The security courses offered by the Programme can be used for the MSc in Software Engineering. If you prefer to take the majority of your courses on security subjects, and write a dissertation on the same topic, then you can choose to be examined instead for the MSc in Software and Systems Security.

If your plans change while you are studying and you are no longer able to meet the requirements for an MSc, even if more time were allowed, then you may choose to be examined for a lower graduate qualification. Attendance at four (or eight) courses, and the successful completion of the corresponding assignments, can lead to the award of a Postgraduate Certificate (or Postgraduate Diploma) in Software Engineering. Should you later return to study on the Programme, you will be able to use these courses as credit towards an MSc.

Fees

There is a fee for each course attended, which covers materials and lunches during the teaching week, and the assignment, but not accommodation. This is payable strictly in advance.

There is an additional registration fee for students on the MSc in Software Engineering. This may be paid in up to four annual instalments.

These fees are revised each year, typically in line with the rate of inflation in the UK.

Key Facts

- a flexible programme in software engineering leading to an MSc from the University of Oxford
- a choice of over 30 different courses, each based around an intensive teaching week in Oxford
- MSc requires 10 courses and a dissertation, with up to four years allowed for completion
- applications welcome at any time of year, with admissions in October, January, and April.

Contact

Software Engineering
University of Oxford
Department of Computer Science
Wolfson Building
Parks Road
OX1 3QD UK
+44 1865 283525
office@softeng.ox.ac.uk
www.softeng.ox.ac.uk