Swansea University
Prifysgol Abertawe

FACULTY OF SCIENCE AND ENGINEERING

UNDERGRADUATE STUDENT HANDBOOK

YEAR 3 (FHEQ LEVEL 6)

SOFTWARE ENGINEERING

DEGREE PROGRAMMES

SUBJECT SPECIFIC
PART TWO OF TWO
MODULE AND COURSE STRUCTURE
2022-23

DISCLAIMER

The Faculty of Science and Engineering has made all reasonable efforts to ensure that the information contained within this publication is accurate and up-to-date when published but can accept no responsibility for any errors or omissions.

The Faculty of Science and Engineering reserves the right to revise, alter or discontinue degree programmes or modules and to amend regulations and procedures at any time, but every effort will be made to notify interested parties.

It should be noted that not every module listed in this handbook may be available every year, and changes may be made to the details of the modules. You are advised to contact the Faculty of Science and Engineering directly if you require further information.

The 22-23 academic year begins on 26 September 2022

Full term dates can be found here

DATES OF 22-23 TERMS

26 September 2022 – 16 December 2022

9 January 2023 – 31 March 2023

24 April 2023 – 09 June 2023

SEMESTER 1

26 September 2022 – 27 January 2023

SEMESTER 2

30 January 2023 – 09 June 2023

SUMMER

12 June 2023 – 22 September 2023

IMPORTANT

Swansea University and the Faculty of Science of Engineering takes any form of **academic misconduct** very seriously. In order to maintain academic integrity and ensure that the quality of an Award from Swansea University is not diminished, it is important to ensure that all students are judged on their ability. No student should have an unfair advantage over another as a result of academic misconduct - whether this is in the form of **Plagiarism**, **Collusion** or **Commissioning**.

It is important that you are aware of the **guidelines** governing Academic Misconduct within the University/Faculty of Science and Engineering and the possible implications. The Faculty of Science and Engineering will not take intent into consideration and in relation to an allegation of academic misconduct - there can be no defence that the offence was committed unintentionally or accidentally.

Please ensure that you read the University webpages covering the topic – procedural guidance here and further information here. You should also read the Faculty Part One handbook fully, in particular the pages that concern Academic Misconduct/Academic Integrity. You should also refer to the Faculty of Science and Engineering proof-reading policy and this can be found on the Community HUB on Canvas, under Course Documents.

Welcome to the Faculty of Science and Engineering!

Whether you are a new or a returning student, we could not be happier to be on this journey with you.

This has been a challenging period for everyone. The COVID-19 pandemic has prompted a huge change in society as well as how we deliver our programmes at Swansea University and the way in which you study, research, learn and collaborate. We have been working hard to make sure you will have or continue to having an excellent experience with us.

We have further developed some exciting new approaches that I know you will enjoy, both on campus and online, and we cannot wait to share these with you.

At Swansea University and in the Faculty of Science & Engineering, we believe in working in partnership with students. We work hard to break down barriers and value the contribution of everyone. Our goal is an inclusive community where everyone is respected, and everyone's contributions are valued. Always feel free to talk to academic staff, administrators, and your fellow students - I'm sure you will find many friendly helping hands ready to assist you.

We all know this period of change will continue and we will need to adapt and innovate to continue to be supportive and successful. At Swansea we are committed to making sure our students are fully involved in and informed about our response to challenges.

In the meantime, learn, create, collaborate, and most of all – enjoy yourself!

Professor Johann (Hans) Sienz
Interim Pro-Vice Chancellor/Interim Executive Dean
Faculty of Science and Engineering

Faculty of Science and Engineering	
Interim Pro-Vice Chancellor/Interim Executive Dean	Professor Johann Sienz
Head of Operations	Mrs Ruth Bunting
Associate Dean – Student Learning and Experience (SLE)	Professor Paul Holland
School of Mathematics and Computer Science **Head of School: Professor Elaine Crooks**	
School Education Lead	Dr Neal Harman
Head of Computer Science	Professor Xianghua Xie
Computer Science Programme Director	Undergraduate: Dr Liam O'Reilly MSc: Dr Bertie Müller
Year Coordinators	Year 0 – Dr Deepak Sahoo Year 1 – Dr Mike Edwards Year 2 – Dr Giedre Sabaliauskaite and Dr Trang Doan Year 3 – Dr Jens Blanck Year 4 – Dr Tom Owen

STUDENT SUPPORT

The Faculty of Science and Engineering has two **Reception** areas - Engineering Central (Bay Campus) and Wallace 223c (Singleton Park Campus).

Standard Reception opening hours are Monday-Friday 9am-5pm.

The **Student Support Team** provides dedicated and professional support to all studentsin the Faculty of Science and Engineering. Should you require assistance, have any questions, be unsure what to do or are experiencing difficulties with your studies or in your personal life, our team can offer direct help and advice, plus signpost you to further sources of support within the University. There are lots of ways to get information and contact the team:

Email: studentsupport-scienceengineering@swansea.ac.uk (Monday–Friday, 9am–5pm)

Call: +44 (0) 1792 295514 and 01792 6062522 (Monday-Friday, 10am–12pm, 2–4pm).

Zoom: By appointment. Students can email, and if appropriate we will share a link to our Zoom calendar for students to select a date/time to meet.

The current student **webpages** also contain useful information and links to otherresources:

https://myuni.swansea.ac.uk/fse/coe-student-info/

READING LISTS

Reading lists for each module are available on the course Canvas page and are also accessible via http://ifindreading.swan.ac.uk/. We've removed reading lists from the 22-23 handbooks to ensure that you have access to the most up-to-date versions. Access to print material in the library may be limited due to CV-19; your reading lists will link to on-line material whenever possible. We do not expect you to purchase textbooks, unless it is a specified key text for the course.

THE DIFFERENCE BETWEEN COMPULSORY AND CORE MODULES

Compulsory modules must be **pursued** by a student.

Core modules must not only be **pursued**, but also **passed** before a student can proceed to the next level of study or qualify for an award. Failures in core modules must be redeemed.

Further information can be found under "Modular Terminology" on the following link - https://myuni.swansea.ac.uk/academic-life/academic-regulations/taught-guidance/essential-info-taught-students/your-programme-explained/

Year 3 (FHEQ Level 6) 2022/23
Software Engineering

BSc Software Engineering[G600]
BSc Software Engineering with a Year Abroad[C60B]
BSc Software Engineering with a Year in Industry[G60A]

Coordinator: Dr JE Blanck

Compulsory Modules

Semester 1 Modules	Semester 2 Modules
CSC301 **Software Engineering Project Planning and Management** **15 Credits** **Ms L Powell**	**CSC364** **Software Testing** **15 Credits** **Dr E Neumann**
CSP300 **Software Engineering Project Implementation and Dissertation** **15 Credits** **Dr JE Blanck**	
CSP301 **Software Engineering Project Specification and Development** **15 Credits** **Dr JE Blanck**	
Total 120 Credits	

Optional Modules

Choose a maximum of 30 credits
The maximum credit limit applies to the modules in this section and also the CSC306/CSC348 and CSC318/CSC345 sub-sections.
You cannot take both CSC306 and CSC306B in the same Academic Year. The same rule applies to CSC348/CSC348B, CSC318/CSC318B and CSC345/CSC345B.

CSC313	High Integrity Systems	Dr AG Setzer	TB1	15
CSC368	Embedded System Design	Dr H Nguyen/Prof SA Shaikh	TB1	15
CSC372	Optimisation	Dr AAM Rahat	TB1	15
CSC385	Modelling and Verification Techniques	Dr U Berger	TB1	15
CSC390	Teaching Computing via a School Placement	Mr SW Powell	TB1	15

And

Choose a maximum of 15 credits
You may choose to make no selection in this section.

CSC306	Writing Mobile Apps	Dr T Owen/Dr T Owen	TB1	15
CSC348	Web Application Development	Dr SP Walton/Dr SP Walton	TB2	15

And

Choose a maximum of 15 credits
You may choose to make no selection in this section.

CSC318	Cryptography and IT-Security	Dr P Kumar/Dr PD James/Dr P Kumar/..	TB1	15
CSC345	Big Data and Machine Learning	Dr S Sharifzadeh/Dr Z Li	TB2	15

And

Choose a maximum of 30 credits
The maximum credit limit applies to the modules in this section and also the CSC306B/CSC348B and CSC318B/CSC345B sub-sections.
You cannot take both CSC306 and CSC306B in the same Academic Year. The same rule applies to CSC348/CSC348B, CSC318/CSC318B and CSC345/CSC345B.

CSC309	Invention and Innovation in Computing	Prof JV Tucker	TB1+2	15
CSC325	Artificial Intelligence	Dr AZ Wyner/Dr B Muller	TB2	15
CSC337	Data Visualisation	Dr C Wacharamanotham/Dr B Muller	TB2	15
CSC349	User Experience	Dr MI Ahmad	TB2	15
CSC371	Advanced Object-Oriented Programming	Prof LY Wu	TB2	15
CSC375	Logic for Computer Science	Dr U Berger	TB2	15
CSC384	Introduction to Video Games Programming	Dr SP Walton	TB2	15

And

Choose a maximum of 15 credits

You may choose to make no selection in this section.

CSC306B	Writing Mobile Apps	Dr TK Astarte	TB2	15
CSC348B	Web Application Development	Dr A Buzdalova	TB2	15

And

Choose a maximum of 15 credits

You may choose to make no selection in this section.

CSC318B	Cryptography and IT-Security	Dr PD James/Dr P Kumar	TB2	15
CSC345B	Big Data and Machine Learning	Dr S Sharifzadeh	TB2	15

CSC301 Software Engineering Project Planning and Management

Credits: 15 Session: 2022/23 September-January

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Ms L Powell

Format: Lectures and Lab Sessions

Delivery Method: Lectures, Lab Sessions, directed reading, guided project development planning.

Module Aims: Software projects have long had a reputation for cost and time overruns - but they need not, and there are well-established, and emerging, techniques and processes to manage them well and effectively: for example, agile methodologies like Scrum which are becoming a de-facto standard in the industry. Also, many projects have significant legal, social, ethical and professional consequences that a practitioner needs to be aware of and sensitive to. This module develops the fundamental skills of successfully building complex software systems, and the implications, including on wider society, of doing so. It will also prepare students for work on any project by equipping them with the skills to successfully plan them, and to commence that planning process.

Module Content: Project planning and management principles:
- Timescales and dependencies
- Scoping and resources
- Risks: identifying, quantifying, managing, monitoring, and mitigating
- Team management

Requirements and Specifications:
- Heavyweight and lightweight models

Methodologies for developing software:
- Traditional - waterfall, prototyping, spiral
- Rapid Application Development (RAD)
- Iterative and incremental
- Agile development
- Hybrid models - Scrum (agile/incremental)

Legal, Social, Ethical, and Professional Issues

Intended Learning Outcomes: On completion of this module, students will:
*Understand the legal, social, ethical and professional framework, particularly with reference to software engineering.
*Have knowledge of a range of software development models, including agile as well as traditional, and their specific properties, advantages, and disadvantages.
*Make choices between a range of software development models.
*Have a basic understanding of the issues of project planning and management in general, including risk analysis and controls, time scale and resource planning, exception monitoring, and progress monitoring and control.
*Have experience of team work in software projects.
*Have basic knowledge of project planning.
*Have an appreciation of the legal, social, ethical and professional implications of software projects.

Assessment: Report (20%)
Group Work - Coursework (50%)
Coursework 1 (20%)
Laboratory work (10%)

Resit Assessment: Coursework reassessment instrument (100%)

Assessment Description: •Individual Coursework – Requirements, Dependencies and CPA
•Individual Report – Project Proposal – to include:
- description of the project/problem to be solved (requirements);
- outline of proposed solution (specification);
- identification of LSEPI issues;
- identification of pre-project risks (not those arising from choice of tools/methodology etc.).

•Group Work – Project Plan – to include:
- Refinements/changes to proposed solution (since Project Proposal Document);
- Methodology choice;
- Technology choices;
- Project plan and dependencies (if appropriate - dependent on methodology choice);
- Basic risk analysis and management (including risk arising out of methodology and technology choices);
- Testing plan

•Laboratory Work – Weekly 1-hour lab session with tasks to be signed off in lab sessions

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Individual categorised feedback (structured by assessment criteria) on each coursework component. Individual feedback on submissions from lecturer and/or demonstrators in laboratory sessions.
Failure Redemption: Resubmission of failed component(s)
Additional Notes: Only available to BSc Software Engineering and MEng Computing students. Updated September 2016.

CSC306 Writing Mobile Apps

Credits: 15 Session: 2022/23 September-January

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr T Owen

Format: 30 lectures and labs.

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: This module will introduce students to developing well-designed and functional apps for mobile devices. Special emphasis is placed on general design paradigms for mobile devices, taking into account limitations such as battery life, limited memory and low user attention compared with desktop computers.

Module Content: Brief history of mobile apps
Basics of Android Programming
- Kotlin for Android and the Android SDK
- Android development environments

Components of an Android Application
- Activities
- Intents
- Tasks
- Resources
- Broadcast Receivers
- Services

Android User Interfaces and Views
- GUIs in Android
- Event handling
- Graphics

Data Persistence
- Databases

Messaging and Networking
Location Services
- Accessing and using location information
- Maps and working with mapping services

Intended Learning Outcomes: Students will be able to apply the methods and techniques they have learned to design and implement Android apps using the standard APIs, paradigms and frameworks for the platform.

Students will have a systematic understanding of the interface and communications paradigms for mobile applications on small-screen devices with non-traditional IO, in the particular context of Android application development.

Students will be able to develop applications targetted on mobile systems by means of device simulators, and deploy them to the actual hardware.

Students will have an understanding of the history of mobile apps.

Assessment: Coursework 1 (30%)
Coursework 2 (10%)
Coursework 3 (60%)

Resit Assessment: Coursework reassessment instrument (100%)

Assessment Description: Coursework 1 (Design) - 30%
Coursework 2 (Peer Review) - 10%
Coursework 3 (Implementation) - 60%

Coursework 1 is a design prototype for an app with a prototype interface and description + justification of design choices and reflection on design approach. Students submit the full XML of the interface. No functionality is required. Coursework 2 invites students to review the XML submissions of their peers, critiquing the use of Android components and performing a usability analysis. Marks are awarded for quality of feedback given to peers. Coursework 3 requires students to implement an app based on their initial design submission, taking into account feedback from Coursework 2. This is a large piece of software development work.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with group and individual analytical feedback for courseworks. Examination feedback summarising strengths and weaknesses of the class.
Failure Redemption: The resit instrument is an exam.
Additional Notes: Updated July 2017. Available to visiting and exchange students. However, experience of Java programming is required for this module.

CSC306B Writing Mobile Apps

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr TK Astarte

Format: 30 lectures and labs.

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: This module will introduce students to developing well-designed and functional apps for mobile devices. Special emphasis is placed on general design paradigms for mobile devices, taking into account limitations such as battery life, limited memory and low user attention compared with desktop computers.

Module Content: Brief history of mobile apps
Basics of Android Programming
- Kotlin for Android and the Android SDK
- Android development environments

Components of an Android Application
- Activities
- Intents
- Tasks
- Resources
- Broadcast Receivers
- Services

Android User Interfaces and Views
- GUIs in Android
- Event handling
- Graphics

Data Persistence
- Databases

Messaging and Networking
Location Services
- Accessing and using location information
- Maps and working with mapping services

Intended Learning Outcomes: Students will be able to apply the methods and techniques they have learned to design and implement Android apps using the standard APIs, paradigms and frameworks for the platform.

Students will have a systematic understanding of the interface and communications paradigms for mobile applications on small-screen devices with non-traditional IO, in the particular context of Android application development.

Students will be able to develop applications targetted on mobile systems by means of device simulators, and deploy them to the actual hardware.

Students will have an understanding of the history of mobile apps.

Assessment: Coursework 1 (30%)
Coursework 2 (10%)
Coursework 3 (60%)

Resit Assessment: Coursework reassessment instrument (100%)

Assessment Description: Coursework 1 (Design) - 30%
Coursework 2 (Peer Review) - 10%
Coursework 3 (Implementation) - 60%

Coursework 1 is a design prototype for an app with a prototype interface and description + justification of design choices and reflection on design approach. Students submit the full XML of the interface. No functionality is required. Coursework 2 invites students to review the XML submissions of their peers, critiquing the use of Android components and performing a usability analysis. Marks are awarded for quality of feedback given to peers. Coursework 3 requires students to implement an app based on their initial design submission, taking into account feedback from Coursework 2. This is a large piece of software development work.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with group and individual analytical feedback for courseworks.
Examination feedback summarising strengths and weaknesses of the class.

Failure Redemption: The resit instrument is a coursework submission.

Additional Notes:

Updated July 2017. Available to visiting and exchange students. However, experience of Java programming is required for this module.

CSC309 Invention and Innovation in Computing

Credits: 15 Session: 2022/23 September-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Prof JV Tucker

Format: 30 lectures including presentations and consultation hours.

Delivery Method: Lectures, Presentations by students.

Module Aims: The course will introduce the student to the history of contemporary computing. Among themes to be explored are the role of invention and innovation, and their commercialisation; and the impact of computing developments on society. The course will offer the opportunity for the student to investigate computing innovations and their historical development, or to work practically on items in the University's History of Computing Collection.

Module Content: Advanced topics relating to the history of contemporary computing.
- History of computing.
- The role of invention and innovation and their commercialisation.
- Impact of computing developments on society.
- Materials from the University's History of Computing Collection.

Intended Learning Outcomes: The student will be able to investigate contemporary computing subjects and their historical development, and
critically assess the literature on specific topics. He or she will also be able to evaluate the impact of computing developments to society.

The students will be able to give a substantial presentation on an in-depth researched topic.

The students will be able to write a dissertation on a computing topic from a historical point of new, or work practically with material objects from the History of Computing Collection.

Assessment: Presentation (25%)
Report (75%)

Resit Assessment: Coursework reassessment instrument (100%)

Assessment Description: Students will write a dissertation on an appropriate and approved subject, which may record work on the History of Computing Collection. The dissertation must be submitted before the beginning of the examination period in Semester 2. The dissertation will account for 80% of the credit for the course.

Students must prepare a formal proposal for the subject of their dissertation that must be approved before the end of Semester 1. Typically, the sizes of proposals range between 3-5 pages. They must contain
a provisional title;
a brief overview of the subject;
some objectives of the dissertation;
provisional chapter titles;
a good bibliography of the subject.

Students must give a seminar on their chosen subject in Semester 2. The seminar is given to the whole class and normally lasts for 25 minutes. The seminar presentation will account for 20% of the credit for the course.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Individual written feedback.

Failure Redemption: Resubmit coursework as appropriate.

Additional Notes:

Available to visiting and exchange students

CSC313 High Integrity Systems

Credits: 15 Session: 2022/23 September-January

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr AG Setzer

Format: 20 hours lectures, 10 hours lab

Delivery Method: On campus lectures.

Module Aims: This module introduces various techniques for developing high integrity systems.

Module Content: Introduction and Motivation:
What are high integrity and critical systems? Legal and ethical issues. Examples of major failures of high integrity systems. Successes and how/why they worked. Standards for safety-critical software and their shortcomings.

Analysis:
The hazard analysis process. Safety analysis and the safety case. Safety issues related to, but outside software. Human factors: the role of the poor interfaces in software failures.

Specification and Verification:
Languages and tools for formal specification and verification of software. Detailed demonstration of one tool and its underlying theory.

Software Production:
Issues in program language selection to minimise failure. The software engineering process in the production of high-integrity software.

Correctness: Validation and verification: the advantages and disadvantages of testing and formal verification.

Intended Learning Outcomes: Students will be familiar with issues surrounding high-integrity systems, including legal and ethical issues and hazard analysis.
They will know how to apply techniques for specifying and verifying high-integrity software.
They will have experience with using tools for developing critical systems.

Assessment: Examination 1 (70%)
Coursework 1 (15%)
Coursework 2 (15%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard Computer Science format unseen examination, duration 2hrs.
The coursework will consist of various programming tasks.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Examination feedback summarising strengths and weaknesses of the class.

Failure Redemption: Resit exam and/or resubmit assignments as appropriate.

Additional Notes:

Created March 2015. Available to visiting and exchange students.
Updated in July 2015

CSC318 Cryptography and IT-Security

Credits: 15 Session: 2022/23 September-January

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr P Kumar, Dr PD James

Format: 30 hours lectures and labs

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: The aim of this course is to examine theoretical and practical aspects of computer and network security.

Module Content: Security threats and their causes.
Security criteria and models.
Cryptography: including basic encryption, DES, AES, hash functions.
Access Control.
Security tools and frameworks: including IPSec, TLS, SSL, SSH and related tools.
Vulnerabilities and attacks: including port scanning, packet sniffing, SQL injection.
Security issues in wireless networks.
Security on the cloud..
Block Chain Technology and Bitcoin
Penetration Testing.
Tor Network.

Intended Learning Outcomes: Students will have the ability to identify security threats and their causes in today's computing infrastructures.
Students will be able to understand and apply techniques from Crytography and Cryptanalysis.
Students will gain experience in building secure systems.
Students will be able to apply techniques to enhance the security of existing systems, and gain a critical awareness of the limits of these techniques.

Assessment: Examination 1 (70%)
Coursework 1 (10%)
Coursework 2 (10%)
Laboratory work (10%)
Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard Computer Science format unseen examination (70%).
1 Coursework and work done in a lab.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with group and individual analytical feedback for courseworks.
Examination feedback summarising strengths and weaknesses of the class.

Failure Redemption: Resit exam.

Additional Notes:

Available to visiting and exchange students.
Updated july 2019.

CSC318B Cryptography and IT-Security

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr PD James, Dr P Kumar

Format: 30 hours lectures and labs

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: The aim of this course is to examine theoretical and practical aspects of computer and network security.

Module Content: Security threats and their causes.
Security criteria and models.
Cryptography: including basic encryption, DES, AES, hash functions.
Access Control.
Security tools and frameworks: including IPSec, TLS, SSL, SSH and related tools.
Vulnerabilities and attacks: including port scanning, packet sniffing, SQL injection.
Security issues in wireless networks.
Security on the cloud..
Block Chain Technology and Bitcoin
Penetration Testing.
Tor Network.

Intended Learning Outcomes: Students will have the ability to identify security threats and their causes in today's computing infrastructures.
Students will be able to demonstrate understanding and be able to apply techniques from Cryptography and Cryptanalysis.
Students will be able to demonstrate experience in building secure systems.
Students will be able to apply techniques to enhance the security of existing systems, and gain a critical awareness of the limits of these techniques.

Assessment: Examination 1 (70%)
Coursework 1 (10%)
Coursework 2 (10%)
Laboratory work (10%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard Computer Science format unseen examination.
1 Coursework and work done in a lab.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with group and individual analytical feedback for courseworks.
Examination feedback summarising strengths and weaknesses of the class.

Failure Redemption: Resit exam.

Additional Notes:

Available to visiting and exchange students.

CSC325 Artificial Intelligence

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr AZ Wyner, Dr B Muller

Format: 20 hours lectures, 10 hours lab.

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: CSC325 is an introduction to Artificial Intelligence, focusing primarily on reasoning and problem solving as a search for a solution rather than on statistical techniques for classification. The course may cover topics from amongst: search techniques; knowledge representation and expert systems; planning; scheduling; qualitative reasoning; language processing with grammar rules; and meta-programming, as well as agents, multi-agent systems, and agent collaboration.

Module Content: • Fundamental Issues in AI
• Basic Search Strategies
• Advanced Search
• Reasoning Under Uncertainty
• Programming for AI
• Basic Knowledge Representation and Reasoning
• Advanced Representation and Reasoning
• Natural Language Processing
• Advanced: Application of NLP and Explainable AI
• Concept of rational agent
• Multi-Agent Systems
• Agent communication and collaboration

Intended Learning Outcomes: On completion of this module, students will
1. be able to demonstrate a systematic knowledge of the fundamental concepts in AI.
2. be able to apply a wider range of AI techniques and to evaluate their advantages and disadvantages.
3. be able to identify problems that are amenable to solution by AI methods and methods which may be suited to solve a given problem.
4. be able to demonstrate competency in developing programs to address problems in AI automatically.

Assessment: Examination 1 (60%)
Coursework 1 (15%)
Coursework 2 (15%)
Laboratory work (10%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard format Computer Science exam.
Practical programming assignments.
Laboratory work.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with analytical individual feedback for assignment.
Examination feedback summarising strengths and weaknesses of the class.

Failure Redemption: Resit examination

Additional Notes:

Created July 2019.
Updated September 2021

CSC337 Data Visualisation

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr C Wacharamanotham, Dr B Muller

Format: 20 Lectures and 10 hours practical classes

Delivery Method: On campus lectures

Module Aims: Data Visualization is concerned with the automatic or semi-automatic generation of digital images that depict data in a meaningful way(s). It is a relatively new field of computer science that is rapidly evolving and expanding. It is also very application oriented, i.e., real tools are built in order to help scientists from other disciplines.

Module Content: Introductory topics include: purposes and goals of visualisation, applications, challenges, the visualisation pipeline, sources of data: data dimensionality, data types, and grid types.

Information visualisation topics include: abstract data, hierarchical data, tree maps, cone trees, focus and context techniques, hyperbolic trees graphs and graph layouts, multi-dimensional data, scatter plots, scatter plot matrices, icons, parallel coordinates, interaction techniques, linking and brushing.

Volume visualisation topics include: slicing, surface vs. volume rendering, transfer functions, interpolation schemes, direct volume visualisation, ray casting, shear-warp factorisation, image order vs. object order algorithms, gradients, filtering, interpolation, and isosurfacing.

Flow visualisation topics include: simulation, measured, and analytical data, steady and time-dependent (unsteady) flow, direct and indirect flow visualisation, applications, hedgehog plots, vector glyphs, numerical integration schemes, streamlines, streamline placement, geometric flow visualisation techniques, line integral convolution (LIC), texture-based techniques, feature-based flow visualisation.

Intended Learning Outcomes: Students will be able to:
- identify problems that can be addressed with visualisation.
- understand data visualisation techniques and be able to (critically) appraise their suitability to particular situations.
- choose and apply visualisation techniques to effectively reveal insights into data.

Assessment: Examination 1 (60%)
Coursework 1 (20%)
Coursework 2 (20%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard Computer Science format unseen examination, duration 2hrs.
Practical Programming Coursework.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with group and individual analytical feedback for courseworks.
Examination feedback summarising strengths and weaknesses of the class.

Failure Redemption: Resit exam and/or resubmit assignments as appropriate.

Additional Notes:

Updated June 2016. Available to visiting and exchange students

CSC345 Big Data and Machine Learning

Credits: 15 Session: 2022/23 September-January

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr Z Li

Format: 20 hours lectures, 10 hours lab.

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: This module provides a broad introduction to artificial intelligence, machine learning, pattern recognition, and their applications to big data problems. The students will gain understanding and knowledge of the theoretical foundations of learning, learn effective machine learning techniques, and acquire practical know-how in applying some of those theories and techniques to real world problems. Topics include big data concept, data mining, learning theories, supervised and unsupervised learning, and reinforcement learning.

Module Content: This module covers three parts: introduction to big data and learning, data analysis techniques, and learning concepts and methods.

Introduction to big data and data mining;
Data clustering;
Dimensionality reduction: linear techniques;
Dimensionality reduction: nonlinear techniques;
Discriminative analysis;
Learning theory, including bias and variance theory, innovation process in machine learning;
Expert systems;
Unsupervised learning;
Supervised learning, including parametric and nonparametric methods, neural network, kernels, support vector machine, randomised decision trees;
Reinforcement and adaptive control;
Example applications to bioinformatics, health informatics, and web data processing.

Intended Learning Outcomes: Upon completion of this module students will be able to:
- Describe, explain, and critique the fundamental techniques of analysing complex and heterogeneous data.
- Describe and explain machine learning techniques and their applications to big data problems.
- Discuss and contrast both conventional and state-of-the-art machine learning techniques.
- Implement and apply machine learning techniques to synthesise solutions.
- Analyse big data problems and evaluate and devise potential solutions.

Assessment: Examination 1 (60%)
Coursework 1 (20%)
Coursework 2 (20%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard format Computer Science exam.
Essay based practical programming assignment.
Laboratory work.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with analytical individual feedback for assignment.
Examination feedback summarising strengths and weaknesses of the class.

Failure Redemption: Redemption of failure via the resit instrument.

Additional Notes: Created March 2015; updated July 2016. Available to visiting and exchange students.

CSC345B Big Data and Machine Learning

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr S Sharifzadeh

Format: 20 hours lectures, 10 hours lab.

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: This module provides a broad introduction to artificial intelligence, machine learning, pattern recognition, and their applications to big data problems. The students will gain understanding and knowledge of the theoretical foundations of learning, learn effective machine learning techniques, and acquire practical know-how in applying some of those theories and techniques to real world problems. Topics include big data concept, data mining, learning theories, supervised and unsupervised learning, and reinforcement learning.

Module Content: This module covers three parts: introduction to big data and learning, data analysis techniques, and learning concepts and methods.

Introduction to big data and data mining;
Data clustering;
Dimensionality reduction: linear techniques;
Dimensionality reduction: nonlinear techniques;
Discriminative analysis;
Learning theory, including bias and variance theory, innovation process in machine learning;
Expert systems;
Unsupervised learning;
Supervised learning, including parametric and nonparametric methods, neural network, kernels, support vector machine, randomised decision trees;
Reinforcement and adaptive control;
Example applications to bioinformatics, health informatics, and web data processing.

Intended Learning Outcomes: Upon completion of this module students will be able to:
- Describe, explain, and critique the fundamental techniques of analysing complex and heterogeneous data.
- Describe and explain machine learning techniques and their applications to big data problems.
- Discuss and contrast both conventional and state-of-the-art machine learning techniques.
- Implement and apply machine learning techniques to synthesise solutions.
- Analyse big data problems and evaluate and devise potential solutions.

Assessment: Examination 1 (60%)
Coursework 1 (20%)
Coursework 2 (20%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard format Computer Science exam.
Essay based practical programming assignment.
Laboratory work.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with analytical individual feedback for assignment.
Examination feedback summarising strengths and weaknesses of the class.

Failure Redemption: Redemption of failure via the resit instrument.

Additional Notes: Available to visiting and exchange students.

CSC348 Web Application Development

Credits: 15 Session: 2022/23 September-January

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr SP Walton

Format: 18 hours lectures, 12 hours labs

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: The module will develop the principles and technologies used for building web-based systems. Practical experience of building web systems will be gained via laboratories and coursework. Existing high programming skill and experience is essential for this module.

Existing programming experience is essential for this module.

Module Content: The history of web application development.
HTML and CSS: Introduction and Good Practices.
Web Application Design.
MVC driven web applications
Security and identity in web applications
Web development using Javascript and AJAX

Intended Learning Outcomes: Students will have a systematic understanding of the key aspects of current web programming principles and technologies.
Students will be able to plan and deliver a web application to a deadline.
Students will be able to create web applications following methodological good practice.
Students will be able to design secure web applications and evaluate their effectiveness.

Assessment: Coursework 1 (10%)
Coursework 2 (20%)
Coursework 3 (70%)

Resit Assessment: Coursework reassessment instrument (100%)

Assessment Description: Coursework 1 and 2 – 30% - Code Review and code submission.

An important part of working in a software engineering organisation is code reviews. In this process engineers look at each other's code to spot bugs and ensure standards are being adhered to. You will submit a source file from your project to be reviews by another student and review another student's source file. You will be assessed both on your adherence to standards with your source code and the quality of your code review.

Coursework 2 – 70% - Implementation.

You will submit the implementation of a small web application. You will be asked to evaluate this by answering a series of questions referencing your implementation. This will assess both your knowledge of the theory and ability to apply that in practice.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Feedback during presentation of implementation. Individual analytical feedback for courseworks.

Failure Redemption: Resit exam and/or resubmit assignments as appropriate.

Additional Notes: Students taking this module must have good programming skills (i.e., be a competent programmer in any standard programming language) as this module requires significant programming ability.

Created March 2015, updated July 2016. Available to visiting and exchange students.

CSC348B Web Application Development

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr A Buzdalova

Format: 18 hours lectures, 12 hours labs

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: The module will develop the principles and technologies used for building web-based systems. Practical experience of building web systems will be gained via laboratories and coursework. Existing high programming skill and experience is essential for this module.

Existing programming experience is essential for this module.

Module Content: The history of web application development.
HTML and CSS: Introduction and Good Practices.
Web Application Design.
MVC driven web applications
Security and identity in web applications
Web development using Javascript and AJAX

Intended Learning Outcomes: Students will be able to demonstrate a systematic understanding of the key aspects of current web programming principles and technologies.
Students will be able to plan and deliver a web application to a deadline.
Students will be able to create web applications following methodological good practice.
Students will be able to design secure web applications and evaluate their effectiveness.

Assessment: Coursework 1 (20%)
Coursework 2 (10%)
Coursework 3 (70%)

Resit Assessment: Coursework reassessment instrument (100%)

Assessment Description: Coursework 1 and 2 – Code Review and code submission.

An important part of working in a software engineering organisation is code reviews. In this process engineers look at each other's code to spot bugs and ensure standards are being adhered to. You will submit a source file from your project to be reviews by another student and review another student's source file. You will be assessed both on your adherence to standards with your source code and the quality of your code review.

Coursework 2 – Implementation.

You will submit the implementation of a small web application. You will be asked to evaluate this by answering a series of questions referencing your implementation. This will assess both your knowledge of the theory and ability to apply that in practice.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Feedback during presentation of implementation. Individual analytical feedback for courseworks.

Failure Redemption: Resit exam and/or resubmit assignments as appropriate.

Additional Notes: Students taking this module must have good programming skills (i.e., be a competent programmer in any standard programming language) as this module requires significant programming ability.

Available to visiting and exchange students.

CSC349 User Experience

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr MI Ahmad

Format: 14 hours of lectures, 15 hours of practical exercises (5 x 3 hour sessions).

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: This module presents an overview of concepts, design for and evaluation of User Experience (UX). Students will come to understand what UX is, why the concept has become so prominent, what differentiates UX from aesthetic design, and what UX practitioners do in their work. The module will examine specific design approaches, and the student will gain some practical skills applying these techniques. Techniques that will be covered include brainstorming, personas and participatory design. The module will also cover the research methods that are employed to understand specific experiences with technology and impart the beginnings of the skills needed to employ these methods and evaluate the results. This module develops practical skills through the coursework and theoretical understanding is demonstrated in the exam. The module builds on skills developed in the second year HCI module but this is not a pre-requisite for it.

Module Content: - Various UX definitions
- Examples of the application of UX focused design to popular systems
- Scoping techniques ((personas, scenarios, use cases)
- UX design techniques (focus groups, brainstorming, Low fidelity prototyping)
- Interface design for UX (aesthetic design techniques, storyboarding)
- Experimental design (participant selection
- Quantitative evaluation (descriptive stats, significance testing, choosing an appropriate test)
- Qualitative evaluation (thematic transcript analysis) techniques

Intended Learning Outcomes: By the end of this module the student will be able to demonstrate the ability to:
- Explain what UX is and is not and what a UX practitioner's remit is.
- Identify and apply appropriate techniques to produce a UX focused design.
- Identify and apply appropriate techniques to evaluate a UX focused design.
- Construct, organise and run evaluations of technologies.

Assessment: Examination (60%)
Coursework 1 (40%)
Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard format Computer Science Exam (2 hours).
Coursework 1: UX design and evaluation.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Student feedback will be given through 4 main avenues:
- Prior to coursework completion feedback will be given verbally through practical sessions
- Post completion each student will receive individual written feedback
- Post completion the class will be given a lecture discussing any pervasive issues in the coursework
- If students are not satisfied with this they will be encouraged to meet with the lecturer to further discuss their problems

Failure Redemption: A supplementary coursework exercise will be made available for the failed assessment, or resit exam as appropriate.

Additional Notes:

Created Mar 2015. Available to exchange students in the Department of Computer Science.

CSC364 Software Testing

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr E Neumann

Format: 20 hours lectures, 8 hours practicals, 2 hours consultation hours.

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: Testing is the process of systematically experimenting with an object (the SUT = System Under Test) in order to establish its quality, where quality means the degree of accordance to the intention or specification. This module will cover various test scenarios; practical exercises will allow the students to gain hands-on experience.

Module Content: The module provides a profound overview on industrially relevant methods in software testing and points out current research directions.
Functional Testing: Boundary Value Testing, Equivalence Class Testing, Decision Table- Based Testing.
Structural Testing: Path Testing, Data Flow Testing.
Integration and System Testing: Levels of Testing, Approaches to Integration Testing.
Object-Oriented Testing: Issues, Class Testing, Object-Oriented Integration Testing.
Possibly selected Research Topics: e.g. Testing Hybrid Systems.

Intended Learning Outcomes: Thorough understanding of testing as a method to validate software systems; critically evaluate and select software test scenarios; problem analysis.

Assessment: Examination 1 (70%)
Coursework 1 (10%)
Coursework 2 (10%)
Laboratory work (10%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard Computer Science format unseen examination, duration 2hrs
Coursework 1: Development of Blackbox-Test suites following a number of different approaches
Coursework 2: Development of Whitebox-Test suites following a number of different approaches
Lab work - weekly.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with group and individual analytical feedback for courseworks.
Examination feedback summarising strengths and weaknesses of the class.

Failure Redemption: Resit examination and/or resubmit coursework(s) as appropriate.

Additional Notes:

This module is compulsory for BSc Software Engineering, and is available to visiting and exchange students.

CSC368 Embedded System Design

Credits: 15 Session: 2022/23 September-January

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr H Nguyen, Prof SA Shaikh

Format: 18 hours lab sessions, 12 hours lectures.

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: Embedded systems are information processing systems embedded into enclosing products such as cars, telecommunication or fabrication equipment. They are essential for providing ubiquitous information, one of the key goals of modern information technology.

The aim of this module is to provide an overview of embedded system design, to relate the most important topics in embedded system design to each other, and to obtain an appreciation of the model-based approach to embedded systems design.

The lab provides hands-on experience in the design of embedded systems.

Due to the labs' hardware requirements, the number of places available for this module is limited.

Module Content: The lectures discuss selected techniques in their specialisation to the design of embedded systems such as:

- Common characteristics, Requirements, Specification and Modeling
- Programming-language-level description techniques
- Hardware (Sensors, actuators, processors)
- Operating systems, middleware, scheduling
- Model-driven design process
- Hardware/software partitioning and codesign
- Simulation, testing and verification techniques

The labs consist of a series of experiments that give the students hands-on experience in developing real embedded systems where they have to pay attention to constraints such as power and latency. Possible topics include examples from

- control theory
- real-time systems
- discrete control
- fault tolerance
- distributed algorithms.

Intended Learning Outcomes: Students will understand and be able to apply engineering principles for system design and their specific application in embedded systems. They will be able to cope with various methods for specification/modelling, analysis, design, implementation and verification.

Assessment: Laboratory work (20%)
Examination 1 (50%)
Coursework 1 (30%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard Computer Science format unseen examination, duration 2hrs.
Lab exercises plus one coursework.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Examination feedback summarising strengths and weaknesses of the class. Individual feedback on submissions from lecturer and/or demonstrators in laboratory sessions.

Failure Redemption: Resit examination.

Additional Notes: The module has a limited capacity.

CSC371 Advanced Object-Oriented Programming

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Prof LY Wu

Format: 30 hours lectures including lab sessions and support/feedback sessions.

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: This module will give an advanced look at object-oriented programming (OOP) languages. It will investigate how OOP languages and libraries evolved to meet the needs of developers, overcoming the challenges involved in application development using non-OOP languages. The module introduces the programming languages C and C++, including low-level aspects of programming that are usually abstracted away in languages like Java. By starting with C, students will gain an in-depth understanding of the need for OOP concepts before moving on to studying programming in C++ and these concepts.

Module Content: - Introduction to C and memory management.
- The C++ object model: basics of the C++ class, constructors, destructors, memory allocation, operator overloading.
- OO-concepts in C++ - a deeper look at Polymorphism, Encapsulation and Inheritance.
- Friend access, virtual methods and multiple inheritance.
- Generic Programming - Templates and partial specialization, techniques for improving type safety.
- The Standard Library, the STL - its use and design: namespaces, streams, strings, vectors, lists, iterators, maps and algorithms.
- Design patterns - an in-depth look at the concept of design patterns, including several well known patterns such as Singleton.

Intended Learning Outcomes: - Students will be exposed to an industry standard and develop C and C++ oriented solutions to a specific problem.
- Students will be able to critically apply object orientated design to large/complex programming problems.
- Students will have an understanding of the ideas and techniques of generic programming, especially a thorough awareness of how to use the ideas and techniques to write efficient and useful libraries.
- Students will acquire a systematic understanding of the main design patterns and idioms.

Assessment: Examination 1 (70%)
Laboratory work (10%)
Coursework 1 (20%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard Computer Science format unseen examination, duration 2hrs. All questions should be attempted.

Coursework: practical programming assignment.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with group and individual analytical feedback for courseworks.
Examination feedback summarising strengths and weaknesses of the class.

Failure Redemption: Resit examination and/or resubmit coursework(s) as appropriate.

Additional Notes:

Updated July 2017. Available to visiting and exchange students. Experience in an Object-Oriented programming language such as Java is assumed.

CSC372 Optimisation

Credits: 15 Session: 2022/23 September-January

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr AAM Rahat

Format: 30 hours (20 lectures, 10 laboratory)

Delivery Method: On Campus Lectures and Labs.

Module Aims: Optimisation is at the core of many disciplines. Whether we want to improve the performance of a machine learning model,
increase the efficiency of an aircraft design, or simply reduce the costs of productions in a business operation, we must deploy computational optimisation methods for achieving the best results. In this module, we will cover mathematical and
algorithmic fundamentals of optimisation, including derivative and derivative-free approaches for both linear and non-linear
problems. We will also discuss advanced topics, such as multi-objective optimisation, handling uncertainty, principled methods when problem evaluations are computationally expensive, and performance comparison between stochastic optimisers, in the context of real-world problems.

Module Content: * Introduction to optimisation.
* Derivatives and related gradient descent methods.
* Bracketing methods.
* Direct methods.
* Stochastic and evolutionary methods.
* Constrained problems.
* Multi-objective optimisation and decision making.
* Model-based methods.
* Optimisation under uncertainty.
* Performance comparison for stochastic optimisers.
The abs will programmatically explore optimisation problems and algorithms.

Intended Learning Outcomes: On completion of this module, students will be able to:
* Demonstrate systematic understanding of fundamental concepts of optimisation problems and algorithms.
* Analyse an unseen optimisation problem, and formulate a mathematical description.
* Propose an appropriate method to solve an optimisation problem, and justify their selection.
* Develop appropriate software for solving optimisation problems.
* Critically evaluate performance of multiple competing optimisers, and communicate analysis to specialist and nonspecialist
audiences

Assessment: Examination (70%)
Coursework 1 (20%)
Online Multiple Choice Questions (10%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Examination. Standard unseen 2 hour Computer Science examination.
Coursework. A practical programming assignment on solving an optimisation problem.
3 Quizzes. Overall 30 multiple choice questions.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Individual feedback on coursework and report..

Failure Redemption: 100% Examination Resit Instrument.

Additional Notes:

This module will be open to visiting and exchange students.

CSC375 Logic for Computer Science

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr U Berger

Format: 20 hours lectures,
2 x 3 hours practicals,
4 problem consultation hours.

Delivery Method: On campus.

Module Aims: This module provides an introduction to logic and its applications to computer science, in particular to the formal specification and verification of computer programs.

Module Content: Propositional logic (syntax, semantics, proof system).
Predicate logic (syntax, semantics, proof system).
Applications of logic to program specification and verification.
Application of tools to carry out formal proofs.

Intended Learning Outcomes: At the end of this module students will understand the syntax, semantics and proof rules of first-order
predicate logic, be aware of other logics that serve special purpose in computer science (e.g. modal logic, process logic), understand the importance of logic for computer science, be able to express informal statements as formulas in predicate logic and to understand formal proofs.
Students will have used an interactive logic tool to carry out formal proofs of varying difficulty.

Assessment: Examination (70%)
Coursework 1 (10%)
Laboratory work (10%)
Coursework 2 (10%)

Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard Computer Science format unseen examination, duration 2hrs.
Coursework consist of 1 assignment and work in computer labs:
Assignment: Syntax and semantics of propositional logic.
Lab: Formal proofs in natural deduction using an interactive proof tool.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solutions provided along with group and individual analytical feedback for courseworks.
Examination feedback summarising strengths and weaknesses of the class.
Individual feedback on submissions from lecturer and/or demonstrators in laboratory sessions.

Failure Redemption: Resit examination and/or resubmit coursework(s) as appropriate.

Additional Notes:

Available to visiting and exchange students.

CSC384 Introduction to Video Games Programming

Credits: 15 Session: 2022/23 January-June

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr SP Walton

Format: 30 hours

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: Developing video games is a challenging area in software engineering. In these highly interactive applications, the software developer has nowhere to hide from users who demand performance. In this module, we will explore advanced software engineering techniques to help us create games with clean, efficient code which is easy to understand. This will be achieved by learning several advanced game programming patterns. Students will also gain experience developing applications in C# using the game engine Unity3D.

Module Content: - Introduction to C#
- Introduction to Unity3D
- General Programming Patterns and their application to video games
- Game Specific Programming Patterns
 * Sequencing Patterns
 * Behavioral Patterns
 * Decoupling Patterns
 * Optimization Patterns

Intended Learning Outcomes: - Students will have the ability to create prototype games using C# and Unity3D.
- Students will apply knowledge of software engineering to understand and evaluate video game software design patterns.
- Students will be able to analyse a games programming task and apply the correct software design pattern to the task, evaluating and justifying that decision.

Assessment: Coursework 1 (20%)
Coursework 2 (40%)
Coursework 3 (40%)

Resit Assessment: Coursework reassessment instrument (100%)

Assessment Description: A programming task to illustrate the students meet the learning objectives related to Behavioural, Decoupling and Optimisation. Students submit a video and written answers to show they have applied all the concepts correctly.

Assessment 1– Peer code review

Assessment 2 – Submission of a video showing prototype game running. This will be assessed on quality of the product in terms of performance and how well it met the brief.

Assessment 3 – Submission of source code and a set of short answer questions where students highlight in their implementation where they have used the software design patterns covered in the course and why.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Analytical individual feedback on coursework. Individual feedback on submissions from the lecturer and/or demonstrators in laboratory sessions.

Failure Redemption: Resit coursework(s) as appropriate.

Additional Notes: Students taking this module must have good programming skills (i.e., be a competent programmer in any standard programming language) as this module requires significant programming ability.

Created Feb 2020.

CSC385 Modelling and Verification Techniques

Credits: 15 Session: 2022/23 September-January

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Dr U Berger

Format: 20 lectures,
2 x 3 practicals,
4 problem consultation hours.

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: This module will give an overview of the landscape and the state of the art of current modelling and verification techniques. Students will gain hands-on experience in using a tool for modelling and verification.

Module Content: Overview of techniques for formal verification.

Interactive theorem proving, automated theorem proving and model checking.

Introduction to one specific logic for modelling and verification.

Techniques for modelling of software using verification tools.

Practical verification of software examples.

Intended Learning Outcomes: After completing this module a student will be able to:
- Explain the current state of the art of modelling and verification techniques;
- Use a verification tool and translate mathematical notation into the input language of that tool;
- Apply a verification tool to practical examples.

Assessment: Examination 1 (70%)
Coursework 1 (15%)
Laboratory work (15%)
Resit Assessment: Examination (Resit instrument) (100%)

Assessment Description: Standard format Computer Science exam (2 hours).
Coursework consists of one assignment and lab work.
Assignment: Mathematical and logical foundations of concurrent processes.
Lab: Modelling and verification in CSP using the process tools ProBE and FDR.

Moderation approach to main assessment: Second marking as sampling or moderation

Assessment Feedback: Outline solution provided along with group and individual analytical feedback for coursework.

Examination feedback summarising strengths and weaknesses of the class.

Individual feedback on submissions from lecturer and/or demonstrators in laboratory sessions.

Failure Redemption: Resit examination and/or resubmit coursework(s) and /or redo lab exercise as appropriate

Additional Notes:

Available to visiting students

CSC390 Teaching Computing via a School Placement

Credits: 15 Session: 2022/23 September-January

Pre-requisite Modules:

Co-requisite Modules:

Lecturer(s): Mr SW Powell

Format: Contact Hours: 7 hours training day, 3 hours midterm session.
Placement Hours: 10 days (5 hours each) in school

Delivery Method: Contact Hours: training session, personal tutorials, school placement days.

Module Aims: This module is for students with an interest in entering teaching, and involves a weekly placement in a local school or college under the mentorship of a Computing/ICT teacher. The student will engage both in observation and in various teaching activities. The module will be assessed on the basis of the mentor's report and on written project work.

Module Content: There is no formal syllabus. The students will have an introductory training day to provide basic information and practical advice. Each student will then spend 10 days in a school or college under the supervision of a teacher-mentor, firstly mainly observing, and then progressing to small-scale teaching activities.

Intended Learning Outcomes: After studying this module students will gain:
First hand experience of teaching in a school or college environment.
Awareness of, and practice in, skills needed to deliver technical material to secondary school children.

Assessment: Practical (30%)
Report (30%)
Assignment 1 (40%)
Resit Assessment: Coursework reassessment instrument (100%)

Assessment Description: Written assessment by teacher-mentor.
Continuous assessment based on student log of activities within school or college.
Preparation of learning materials and final report.

Moderation approach to main assessment: Universal non-blind double marking

Assessment Feedback: Direct written and oral feedback from the Lecturer.

Failure Redemption: Resubmission of project work.

Additional Notes:

Requires an enhanced Disclosure and Baring Service (DBS) check. Not available to visiting and exchange students. Number of places on the module contingent on the availability of school and/or college placements.

Anyone choosing this module must do so during preselection and if they do not already have an enhanced DBS check acquired through the University they must contact Prof Moller (f.g.moller@swan.ac.uk) immediately. An enchanced DBS takes some weeks at minimum, and so it's essential that anyone choosing this module does so in pre-selection before the summer vacation.

Anyone choosing this module must also to commit to attend a full-day training event in the department on Thursday during "Induction Week" (immediately preceding the first week of teaching).

Anyone undertaking this module must abide by all instructions given by their host school. If a disagreement arises between a student and their placement school and the school decides to no longer host the student, then the student will be required to withdraw from the module.

CSP300 Software Engineering Project Implementation and Dissertation

Credits: 15 Session: 2022/23 September-June

Pre-requisite Modules:

Co-requisite Modules: CSP301

Lecturer(s): Dr JE Blanck

Format: 15 hours meetings with supervisor. Project module final assessment via dissertation and regular supervisory meetings.

Delivery Method: On-campus/virtual lectures and lab sessions.

Module Aims: This module forms the second part of the Level 6 project (together with CSP301) for BSc Software Engineering and MEng Computing students. It consists of the implementation of a software system, and a substantial written dissertation.

Module Content: Students must develop and document a software (or software/hardware) system that would normally be a well-defined application. They must use a well-defined software life-cycle model (SLC). The deliverables are the application itself, and the dissertation (with comprehensive appendix - see 'Component Descriptions' below).

This module forms the second part of a pair (see CSP301) that together form a complete Level 6 Software Engineering project.

The dissertation deadline is defined on the College of Science Intranet, in the Department Teaching Diary and in the Project Handbook (links to both of these are on the Computer Science Student Information page on Blackboard). Both dissertation and the software developed must be submitted.

Intended Learning Outcomes: By the end of this module, students will
(1) have defined the requirements for, specify, designed and implemented a complete software system,
(2) have applied the major phases of the life-cycle of a software engineering project,
(3) have managed a substantial project and have carried out a risk analysis,
(4) have evaluated and reflected on the management of their project,
(5) have researched and presented the background material, and documented their project.

Assessment: Project (100%)

Assessment Description: The dissertation (30-50 pages) is a comprehensive and self-contained report on the work done on the project (see CSP301). The document should include a comprehensive appendix not part of the above page count. This appendix contains all deliverables as prescribed by the initial document and appropriate for the chosen SLC.

For internal students the module CSP301 must have been completed before a dissertation will be accepted. External students must viva / demonstrate their project before it will be considered by the Department. The document can address the following topics:

- the main technical problem(s) that have been solved;

- a brief survey on similar work;

- a study and survey of relevant literature;

- description of the chosen software life-cycle model (SLC);

- a clear and precise narrative description of the project suppported by references to the deliverables (requirements, design, implementation, testing, user manual, etc.) as prescribed by the initial document and appropriate for the chosen SLC;

- a table or tables linking statements made in the dissertation to the evidence provided in the appendix;

- consideration of the legal, social, professional and ethical issues related to the system if any (or explicitly addressing why they do not apply);

- discussion of tools that were used;

- reflection on the project results (e.g. reliability, functionality, compliance with specification);

- suggestions for further work.

This list is for guidance and is not a checklist - not all sections will be appropriate in all cases, and in some cases other topics not listed should be included.

Submission:
Information on submission (process and deliverables) can be found in the Software Engineering Project Handbook on the Computer Science Student Information page on Blackboard. Information on submission deadlines can be found on the College of Science Intranet, the Computer Science Student Information page; and in the Software Engineering Project Handbook.

Moderation approach to main assessment: Universal double-blind marking

Assessment Feedback: Individual written feedback and grades.

Failure Redemption: Resubmission of dissertation and/or software system.

Additional Notes:

Available only to BSc Software Engineering and MEng Computing majors.

CSP301 Software Engineering Project Specification and Development

Credits: 15 Session: 2022/23 September-June

Pre-requisite Modules:

Co-requisite Modules: CSP300

Lecturer(s): Dr JE Blanck

Format: 15 hours project meetings, presentation seminars, project fair.

Delivery Method: Project supervision meetings plus attendance at Annual Colloquium and Project Demonstration Fair

Module Aims: The aims of this module in conjunction with CSP300 are:
• to provide BSc Software Engineering students the opportunity of specifying,
designing and implementing a complete system and experiencing the major phases of the life-cycle of a computing project;
• to enhance students' competence in system design, risk analysis and management, and their fluency in using programming languages and tools;
• to give students an intellectual challenge to their abilities to learn new subjects without instruction, and to further develop their abilities in literature searching, report writing, verbal presentation, project planning and time management.

Module Content: Level 6 Software Engineering/MEng Computing projects are about the production of a substantial, high-quality piece of software built to a set of requirements and a specification. The Department produces an annual list of proposed projects, and students should approach members of staff for detailed information on those projects that interest them or suggestions for alternatives. Each student will be supervised by a member of staff and will be required to attend regular (at least every two weeks) meetings. This is a project preliminaries module - with CSP300 as the follow on module. This modules involves a number of milestones including the production of one document, a short public presentation and a demo/viva at the departmental Project Fair. Precise instructions on project deliverables (content, length, submission process) can be found in the Software Engineering Project Handbook, available on Blackboard.

As well as about the final deliverable, Software Engineering/MEng Computing projects are about the project process. Projects are expected to follow current good practice in software development and project management, relevant to the specific nature of the work undertaken. It is expected that projects will include appropriate background research, planning, scheduling, periodic or ongoing review of progress and the project development process, and risk management. It is also expected that they use current software tools and methodology (e.g. some form of agile or other development process), suited to the nature of the project (and that this choice is explained and justified); that version management tools are used; and that appropriate deployment tools are used.

Deadlines for the submission of documents can be found on the College of Science Coursework Schedule, in the Departmental Teaching Diary and in the Software Engineering Project Handbook (links to all of these can be found on the Computer Science Student Information page on Blackboard). The time and date of the Project Fair will be announced to students in good time, and will be published on the Computer Science Student Information page on Blackboard.

Intended Learning Outcomes: By the end of this module, students will
(1) have defined the requirements for, specify, designed and implemented a complete software system,
(2) have applied the major phases of the life-cycle of a software engineering project,
(3) have managed a substantial project and have carried out a risk analysis,
(4) have evaluated and reflected on the management of their project,
(5) have researched and presented the background material, and documented their project.

Assessment:	Report (35%) Presentation (25%) Presentation (30%) Coursework 1 (10%)
Resit Assessment:	Coursework reassessment instrument (100%)

Assessment Description: Initial Project Document.
Length: Approx 15 pages.

This document should give the title and introduction to the project area. It should detail the scientific and technical background of the project, and present a rigorous discussion of the project. It also should include an informed choice of an appropriate software life-cycle model (SLC) and specify which of its deliverables will be included in the project.

The document should cover the following topics:
• the main technical problem(s) to be solved;
• a brief survey on similar work;
• a study and survey of relevant literature;
• the software and hardware constraints if appropriate;
• selection of an appropriate software life-cycle model (SLC)
• specification and brief explanation of the deliverables to be included in the project (requirements, design, implementation, testing, user manual, etc.)
• brief reflection on how to provide evidence of these deliverables in later reports
• demonstration of first, appropriate steps dependent on the SLC;
• a detailed project plan specifying the deadlines according to the SLC;
• anticipated problems and further areas of study or influence.

The Initial Project Document is normally submitted approximately 3 weeks after start of term.

Public Presentation.
Duration: About 15 minutes.

The presentation of the aims and background of the project, choice of SLC and the progress to date, will be given to an audience of about 20 students and staff. It will be held during November.

Demonstration.

The Department will organise a Project Demonstration Fair, to which will be invited contacts from industry and students from other levels. All lecturing staff will attend at various times. Students will produce a poster and will be expected to explain and demonstrate their project whenever a large enough audience is gathered. Students will be assessed on the quality of their poster and demonstration.

Moderation approach to main assessment: Universal double-blind marking

Assessment Feedback: Individual feedback on all documents from both supervisor and second marker, together with an agreed grade. Collective feedback and agreed grade for public presentation and demonstration.

Failure Redemption: Resubmission of written components as appropriate.

Additional Notes:

Only available to BSc Software Engineering and MEng Computing majors.
Last updated 17/12/2016