Track Outline

Computer Science

Department of Science

Track Outline Computer Science
Department of Science

Track coordinator: Dr Andrew Brooks
Email: a.brooks@ucr.nl
Tel: 0118-655534
Office no. & location: Eleanor 1.09

Other instructors in track: None.

Date of last update of this track outline: 10th January 2019

1. List of courses in track

SCICOMP102 Introduction to Computer Science
Brief Description:
This is an introductory course in problem solving and computer programming in Java. Although Java is an object oriented programming language, the course begins by introducing traditional structured programming and data constructs (i.e. selections, loops, methods, primitive types, and arrays). Then consideration is given to the object-oriented programming constructs (i.e. encapsulation, composition, inheritance, polymorphism, abstract classes, and interfaces). The second meeting of the class each week is entirely devoted to laboratory work where students tackle programming exercises and demonstrate their work. Two larger programming projects are also undertaken. There is a written midterm exam and a written final exam. By the end of the course a student will have obtained a reasonable familiarity with the Java API (Application Programming Interface) and a Java IDE (Integrated Development Environment).
Prerequisites: None

SCICOMP201 Database Management
Brief Description:
Database management systems are one of the foundations upon which a modern economy is built. The course begins by introducing SQL, a special-purpose language designed for managing data in a relational database management system (RDBMS). Then consideration is given to the theory underpinning relational databases, data storage and querying, and transaction management. The second meeting of the class each week is entirely devoted to laboratory work where students tackle exercises and demonstrate their work. Projects are undertaken to provide practical experience of the design, building, and evaluation of database management systems. An individual project provides practical experience of data cleansing. There is a written midterm exam and a written final exam. By the end of the course a student will have obtained a reasonable familiarity with both relational and non-relational database management systems.
Prerequisites: C or better grade in SCICOMP102 or by permission of the instructor

SCICOMP202 Networks and Communications
Brief Description:
Computer networks are the foundations on which the modern commercial, entertainment, industrial, and social world is built. The course begins with the history and development of the modern Internet and associated transmission media. Then consideration is given to the issues of packet structure, protocols that govern packet transmission, routing protocols, and

error control. An introduction is given to network security. The second meeting of the class each week is entirely devoted to laboratory work where students tackle exercises and demonstrate their work. Projects are undertaken to provide practical experience of analysing Internet traffic, network simulation, and network security. An individual project provides practical experience of researching in-depth a specialist tool or protocol. There is a written midterm exam and a written final exam. By the end of the course a student will have obtained a reasonable familiarity with how the Internet works and how the Internet can be monitored and controlled.
Prerequisites: C or better grade in SCICOMP102 or by permission of the instructor

SCICOMP301 Topics in Computer Science (currently Operating Systems)
Brief Description:
An operating system (OS) is system software that is the interface between hardware and applications. The course begins by introducing the typical structure of an OS, by distinguishing between processes and threads, and by explaining how an OS schedules work. Then consideration is given to the issues of concurrency, memory management, and file systems. The second meeting of the class each week is entirely devoted to laboratory work where students tackle exercises and demonstrate their work. Projects are undertaken to provide practical experience of modelling concurrency with Petri Nets, evaluating scheduling algorithms, and investigating part of an OS. An individual project provides practical experience of writing concurrent code. There is a written midterm exam and a written final exam. By the end of the course a student will have obtained a reasonable familiarity with how an OS manages resources and provides services to applications.
Prerequisites: C or better grade in SCICOMP102 or by permission of the instructor

SCICOMP302 Algorithms and Data Structures
Brief Description:
Algorithms define the steps necessary to solve a particular problem. An inappropriate choice of algorithm and associated data structure can seriously impact on the performance of an application. The study of algorithm design and analysis provides techniques which help minimize the execution time of an algorithm. An emphasis is on the experimental performance analysis of algorithms. An introduction is given to NP-completeness. The second meeting of the class each week is entirely devoted to laboratory work where students tackle exercises and demonstrate their work. Projects are undertaken to provide practical experience of algorithm performance measurement and the empirical investigation of algorithms in real applications. There is a written midterm exam and a written final exam. By the end of the course a student will have obtained a reasonable familiarity with sorting and searching algorithms, breadth-first and depth-first search algorithms, and minimum spanning tree algorithms.
Prerequisites: C or better grade in SCICOMP102 or by permission of the instructor

Scheduling Information (subject to change)
SCICOMP102 will presented once or twice per year.
SCICOMP201 and SCICOMP202 will have deliveries alternated on spring semesters.
SCICOMP301 and SCICOMP302 will have deliveries alternated on fall semesters.

2. Overview of the track

The introductory course is a programming course where the emphasis is on the individual learning the craft of programming. Success in this course is required in order to take further computer science courses. This arrangement is typical of other degrees or tracks in

computer science at other institutions worldwide. In the introductory course, copying and collusion is routinely discouraged by the instructor and students are quizzed on their comprehension of their code submissions for laboratory exercises and projects.

Algorithms, databases, networks, and operating systems are considered the cornerstones of computer science and for this reason, the track has a separate course on each of these.

The track's focus is more applied than theoretical as such a focus strongly supports active learning. Half of all scheduled class time in each course is devoted to laboratory work.

Each non-introductory course in part comprises a combination of group-based and individual projects. The group-based projects develop teamwork skills which are considered important by the computing profession. At least one project in each non-introductory course is investigative, with the goal of performing and analysing the results of an experiment, as investigative skills are considered important by the computing profession.

3. Relations between the 11 UCR measurable program outcomes and the student learning outcomes per course

Table A) UCR program outcomes and their translation to individual track

		UCR program outcomes Student can:	**UCR Computer Science track outcomes Student can:**
Demonstrate mastery of disciplinary knowledge			
	Definition of the discipline	Distinguish what phenomena are studied and what types of questions scholars hope to answer via what methods. (This includes an awareness of assumptions and limitations, and the understanding that multiple paradigms exist in a single discipline.)	- analyze and solve problems by writing computer programs. - design, execute, and report on experimental investigations. - compare, contrast, and make use of various database technologies (databases). - compare, contrast, and make use of various algorithms (algorithms). - compare, contrast, and make use of various operating systems (operating systems). - compare, contrast, and make use of various networks (networks).
	Theories	Demonstrate understanding of the most prominent theories of the discipline.	- demonstrate an understanding of the relational algebra theory underpinning the Structured Query Language (databases). - demonstrate an understanding of complexity theory (algorithms). (the theory of Turing machines is not currently taught) - demonstrate an understanding of concurrency theory (operating systems). - demonstrate an understanding of communication theory (networks).
	Methodologies	Apply common analytical methods and tools of the	- use Integrated Development Environments (IDEs) and Application

		discipline and assess work of others	Programming Interfaces (APIs). - adopt and use new tools and techniques as the need arises. (peer assessment is not currently made use of)
Demonstrate mastery of disciplinary skills			
	Critical Thinking	Independently formulate and critically review problem formulations, arguments and results (critical thinking, problem solving)	- analyze and solve problems by writing computer programs. - perform empirical investigations involving hypothesis formulation.
	Research	Apply aspects of the main research methodologies of the discipline	- design, execute, and report on an experimental investigation (e.g. algorithm performance, database performance, network performance, concurrent program performance).
	Communication	Communicate effectively (orally and in writing) with both scholarly- and lay-audiences	- communicate effectively with peers. (human-computer interaction and software engineering which would cover communication with a lay-audience are not currently taught)
	Learning	Independently acquire and evaluate relevant academic information, reflect on one's own progress and identify one's knowledge gaps, and master new topics in discipline	- perform empirical or proof-of-concept investigations involving the learning and use of unfamiliar computer languages, platforms, or tools. - assimilate information in articles of moderate complexity.
Understand and exercise academic attitudes and values			
	Academic and Professional standards	Understand and adopt standards for academic integrity and relevant professional standards	- demonstrate an understanding of, and follow, one or more professional codes of conduct. (not currently taught – see open issues)
	Discipline's role in the world	Reflect in logical, social and/or ethical terms on interaction between discipline and the natural world, society and/or self.	- demonstrate an understanding of the cultural, social, legal, and ethical issues inherent in the discipline of computing. (not currently taught – see open issues)
Understand connections with other disciplines			
	Related fields	Transfer knowledge and/or skills from other related disciplines	- perform empirical or proof-of-concept investigations on real-world applications.
	Complex problems	Present analysis of (and possibly first steps towards) the solution of complex multi-facetted problems requiring knowledge and/or skills from different disciplines	- be part of a team that builds a complex software application involving requirements analysis, design, implementation, and testing. (software engineering is not currently taught – see open issues)

Table B) Link of Program Outcomes in track Computer Science to Student Learning Outcomes per course

		SCICOMP102 Introduction to Computer Science	SCICOMP201 Database Management	SCICOMP202 Networks & Comms.	SCICOMP302 Algorithms and Data Structures	SCICOMP301 Topics (Operating Systems)
Demonstrate mastery of disciplinary knowledge						
	Definition of the discipline	*Laboratories. Projects. Homework. Exams.*	*Laboratories. Projects. Homework. Exams.*	*Laboratories. Projects. Homework. Exams.*	*Laboratories. Projects. Homework. Exams.*	*Laboratories. Projects. Homework. Exams.*
	Theories		*Exam section.*	*Exam Section.*	*Exam section.*	*Exam Section.*
	Methodologies	*Laboratories. Projects.*	*Various laboratories and projects.*	*Various laboratories and projects.*	*Various laboratories and projects.*	*Various laboratories and projects.*
Demonstrate mastery of disciplinary skills						
	Critical Thinking	*Laboratories. Projects.*	*Project.*	*Project.*	*Project.*	*Project.*
	Research		*Project.*	*Project.*	*Project.*	*Project.*
	Communication		*Group-based Projects and associated presentations.*	*Group-based Projects and associated presentations.*	*Group-based Projects and associated presentations.*	*Group-based Projects and associated presentations.*
	Learning		*Investigative project. Various homework.*	*Investigative project. Various homework.*	*Investigative project. Various homework.*	*Investigative project. Various homework.*
Understand and exercise academic attitudes and values						
	Academic and Professional standards					
	Discipline's role in the world					
Understand connections with other disciplines						
	Related fields		*Investigative project.*	*Investigative project.*	*Investigate project.*	*Investigative project.*
	Complex problems					

4. Role of the track in a student's course program

Students wishing to pursue a computing-related Master are advised to take as many computer science, mathematics, and statistics courses as possible.

5. Related graduate studies

Students completing the track have gone on to pursue a computing-related Master at Utrecht University or Eindhoven University of Technology, though this usually means taking a pre-Master programme.

At Utrecht University, a pre-Master programme takes approximately six months and is usually only feasible for Dutch-speaking students as courses are selected from the Bachelor programme which is taught in Dutch:
https://www.uu.nl/masters/en/computing-science/pre-masters-programme.

Several universities throughout the Netherlands offer pre-Master programmes for computing-related Masters. Some of these programmes are taught in English, some in Dutch. Some have a fixed set of courses that must be undertaken, others offer flexibility depending on the subjects already studied at Bachelor level. In some cases, the pre-Master programme could be less than 30 ECTS, depending on the subjects already studied at Bachelor level. A student interested in taking a computing-related Master is advised to examine programmes and programme requirements at several universities.

Students completing the track have also gone on to pursue a computing-related Master (with varying requirements on the taking of additional computer science, mathematics, or statistics courses) at universities in the United Kingdom, Norway, Germany, Sweden, and Canada.

6. Additional information

Students are advised to use their own laptops for laboratory and project work. The following laptop specification is recommended: at least 8GB of RAM, at least 256GB of solid state hard drive, a 6th Generation Intel i5 processor or better, with a graphic processing unit (GPU), with wireless and Bluetooth connectivity, with USB and HDMI ports, and with a 15 inch screen size or larger. (Note: program code editors (IDEs) use a lot of screen space and working with very small screens can be awkward.) This site provides guidance on understanding various laptop specifications:
https://www.dell.com/en-us/work/shop/dell-laptops-and-notebooks/sc/laptops.

Open source or otherwise free to use software will be made use of. Such software is usually readily installable on Windows or Ubuntu platforms. There may, however, be occasions when installers are not available for Mac OS X platforms and software such as Parallels Desktop will required to be purchased which allows Windows to be run on a Mac. There is a student discount for Parallels Desktop at https://www.surfspot.nl.

Students considering running Ubuntu on their laptops should visit here:
https://www.ubuntu.com/desktop.

Excursions are not an important part of the track. Visits to computing companies may be undertaken as the opportunity arises, but such visits will not be compulsory.

There are a growing number of online computing courses that can be useful to take and some universities recognize the credits earned from these courses. Check the following sites for online computing courses:
https://www.edx.org/
https://www.coursera.org/
https://eu.udacity.com
https://www.khanacademy.org/
https://www.futurelearn.com/
https://www.udemy.com

Useful playlists can be found on YouTube for all kinds of computing topics. At YouTube, simply perform a search using the keyword playlist and the topic of interest.

One useful place to look for internships is on LinkedIn (https://www.linkedin.com).

The track instructor is available to supervise internships and projects during semester and over the summer months.

7. Open Issues

Departments of Computer Science at other institutions typically offer an entire course to cover computer ethics. The knowledge area of *Social Issues and Professional Practice* is described in the ACM's 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. The track instructor's current view is not to provide an entire course on computer ethics at the expense of one of the other courses.

Departments of Computer Science at other institutions typically offer an entire course on software engineering which involves the construction of a complex system by groups of students and which requires students to seek knowledge outside the discipline of computer science itself. The knowledge area of *Software Engineering* is described in the ACM's 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. The track instructor's current view is not to provide an entire course on software engineering at the expense of one of the other courses.

Departments of Computer Science at other institutions typically offer an entire *elective* course on the theory of computation. The knowledge area of *Algorithms and Complexity* is described in the ACM's 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. The track instructor's current view is not to provide an entire course on the theory of computation at the expense of one of the other courses.

* See http://www.acm.org/education/curricula-recommendations for details.