UNIVERSITY OF GOTHENBURG

COMPUTER SCIENCE AND ENGINEERING

DIT271 Formal Methods in Software Development, 7.5 higher education credits

Formella metoder i mjukvaruutveckling, 7,5 högskolepoäng

Second Cycle

Confirmation

This course syllabus was confirmed by Department of Computer Science and Engineering on 2017-02-10 to be valid from 2017-08-28, autumn semester of 2017.

Field of education: Science 100%
Department: Computer Science and Engineering

Position in the educational system

The course is part of the Computer Science Master's programme and a single subject course at the University of Gothenburg.

The course can be part of the following programmes: 1) Computer Science, Master's Programme (N2COS) and 2) Computer Science, Bachelor´s Programme (N1COS)

Main field of studies	*Specialization*
Computer Science	A1F, Second cycle, has second-cycle course/s as entry requirements

Entry requirements

The requirement for the course is to have successfully completed courses corresponding to 120 hec within the subject Computer Science or equivalent, specifically DIT201 Logic in Computer Science, 7.5 hec, and a 7.5 hec course in object-oriented programming (or equivalent) are required.

Applicants must prove knowledge of English: English 6/English B or the equivalent level of an internationally recognized test, for example TOEFL, IELTS.

Learning outcomes

On successful completion of the course the student will be able to:

Knowledge and understanding

- explain the potential and limitations of using logic based verification methods for assessing and improving software correctness,
- identify what can and what cannot be expressed by certain specification/modeling formalisms,
- identify what can and cannot be analyzed with certain logics and proof methods,

Skills and abilities

- express safety and liveness properties of (concurrent) programs in a formal way,
- describe the basics of verifying safety and liveness properties via model checking,
- successfully employ tools which prove or disprove temporal properties,
- write formal specifications of object-oriented system units, using the concepts of method contracts and class invariants,
- describe how the connection between programs and formal specifications can be represented in a program logic,
- verify functional properties of simple Java programs with a verification tool,

Judgement and approach

- judge and communicate the significance of correctness for software development,
- employ abstraction, modelling, and rigorous reasoning when approaching the development of correctly functioning software.

Course content

The aim of this course is to teach knowledge and skills in, and judgement about, two important styles of formal methods for reasoning about software: model checking and deductive verification. Each style will be introduced in three ways: conceptual, theoretical, and practical, using a particular tool. The course builds on skills in first-order logic and temporal logic, and shows how these formalisms can be applied, and extended, for the verification of software.

On the model checking side, we cover the following topics:

- a specification language for concurrent processes,
- verifying assertions,
- synchronization,
- verifying safety and liveness properties in temporal logic.

On the deductive verification side, we cover the following topics:

- a unit level specification language for Java programs,
- a logic for verification of Java programs,
- verification of Java programs, in the sense that the implementation of a unit fulfils the specification.

Sub-courses

1. **Oral Examination** *(Muntlig Tentamen),* 5 higher education credits
   Grading scale: Pass with Distinction (VG), Pass (G) and Fail (U)
2. **Laboratory** *(Laboration),* 2.5 higher education credits
   Grading scale: Pass (G) and Fail (U)

Form of teaching

There are about two lectures each week, and one exercise class per week. The students perform practical case studies using the tools in the laboratory assignments.

Language of instruction: English

Assessment

The course is examined by an individual oral exam (U-VG), and compulsory laboratory assignments handed in during the course (U-G). The practical laboratory assignments are normally carried out in pairs.

If a student, who has failed the same examined component twice, wishes to change examiner before the next examination, a written application shall be sent to the department responsible for the course and shall be granted unless there are special reasons to the contrary (Chapter 6, Section 22 of Higher Education Ordinance).

In cases where a course has been discontinued or has undergone major changes, the student shall normally be guaranteed at least three examination occasions (including the ordinary examination) during a period of at least one year from the last time the course was given.

Grades

The grading scale comprises: Pass with Distinction (VG), Pass (G) and Fail (U).
In order to be awarded the grade Pass (G) for the whole course, the student must pass both the sub-courses. In order to be awarded the grade Pass with Distinction (VG) , the student needs to get the grade Pass with Distinction on the sub-course Oral examination and pass the sub-course Laboratory.

Course evaluation

The course is evaluated through meetings both during and after the course between teachers and student representatives. Further, an anonymous questionnaire is used to ensure written information. The outcome of the evaluations serves to improve the course by indicating which parts could be added, improved, changed or removed.

Additional information

The course replaces the DIT270 Software Engineering using Formal Methods course. The course cannot be included in a degree which contains DIT270. Neither can the course be included in a masterdegree which is based on a bachelordegree in which the course DIT270 is included.

The course is a joint course together with Chalmers.

Course literature to be announced 8 weeks prior to the start of the course.