ID2207 Modern Methods in Software Engineering 7.5 credits

Moderna metoder inom programvaruutveckling (software engineering)

This is a translation of the Swedish, legally binding, course syllabus.

If the course is discontinued, students may request to be examined during the following two academic years

Establishment

Course syllabus for ID2207 valid from Spring 2019

Grading scale

A, B, C, D, E, FX, F

Education cycle

Second cycle

Main field of study

Computer Science and Engineering

Specific prerequisites

- Computer Science courses 30 hp
- Operating Systems courses 7,5 hp
- Computer Programming courses 7,5 hp
- English "level B" (Swedish Gymnasium)

Language of instruction

The language of instruction is specified in the course offering information in the course catalogue.

Intended learning outcomes

The course aims both in giving students knowledge about modern software development methods and developing skills in usage the methods.

Our goal is to present a variety of approaches to software development and discuss their applicability boundaries, benefits, restriction and complementariness.

During the course students should learn about Software Engineering methods. In particular, they:

1. Learn methods for dealing with complexity and changes in software construction. This means that students should get understanding of main approaches to abstraction, models, decomposition and software life-cycle.
2. Understand basic components of software development process. This means that students should learn main methods and approaches to requirement elicitation and analysis, system and object design.
3. Learn some modern approaches to software development. This means that students should learn about agile methods of software development and have experience in applying them to desing a software system.
4. To get experience in evaluating different methods for producing of a high quality software system within time. This means that students should get practice in applying and comparison of different approaches to software development.
5. To get experience in reporting and discussing results of the course homework and project both in oral and written forms.
6. Understand ethical aspects and importance of sustainability in software development.

The course also includes a seminar as a part of the Software Engineering of Distributed Systems master program. The intention of the seminar is to put the course into the context of the software engineering research in general and into the context of the master program in particular.

Course contents

Introduction and basic concepts of Software Engineering (SE). Abstraction/Models and Decomposition. Software Life-Cycle. Unified process. Software Modeling language. Unified Modeling Language (UML). Requirements elicitation and analysis. System design. Object design. Applying patterns. Refactoring. Mapping models to code. Testing. Agile software development and agile modeling. Basics of Extreme Programming. Software project management.

Practical part of the course includes exercises and a small software development project applying SE methods.

Course literature

Textbook for the course:
Object-Oriented Software Engineering: Using UML, Patterns and Java: International Edition, 3/E, Bernd Bruegge, Allen H. Dutoit, ISBN: 0136061257, Publisher: Prentice Hall, Copyright: 2010, Format: Paper; 800 pp Published: 29 July 2009 (available in the Kista Electrum book store

Lecture notes

Recommended Reading:

The following sources are recommended to obtain a deeper understanding of the subject.

- E. Gamma et al. Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995
- M. Shaw, D. Garlan. Software Architecture. Perspectives on an Emerging Discipline. Prentice-Hall, 1996
- http://www.extremeprogramming.org/start.html
- Kent Beck. Extreme Programming Explained: Embrace Change, Publisher: Addison-Wesley Professional; 1st edition (October 5, 1999)
- Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts. Refactoring: Improving the Design of Existing Code, Addison-Wesley Professional; 1st edition (June 28, 1999)
- M. Matskin and E. Tyugu. Structural Synthesis of Programs and Its Extensions. Computer and Informatics Journal, v. 20, 2001, pp. 1-25

Additional articles in the curriculum may be added during the course.

Examination

- ANN1 - Assignment, 3.0 credits, grading scale: P, F
- TEN1 - Examination, 4.5 credits, grading scale: A, B, C, D, E, FX, F

Based on recommendation from KTH’s coordinator for disabilities, the examiner will decide how to adapt an examination for students with documented disability.

The examiner may apply another examination format when re-examining individual students.

Written examination (TEN1 4,5 hp)

Ethical approach

- All members of a group are responsible for the group's work.
- In any assessment, every student shall honestly disclose any help received and sources used.

- In an oral assessment, every student shall be able to present and answer questions about the entire assignment and solution.