GHENT UNIVERSITY

Course Specifications

Valid as from the academic year 2020-2021

Software Engineering (E761035)

Due to Covid 19, the education and evaluation methods may vary from the information displayed in the schedules and course details. Any changes will be communicated on Ufora.

Course size (nominal values; actual values may depend on programme)

Credits	Study time	Contact hrs
6.0	180 h	60.0 h

Course offerings and teaching methods in academic year 2020-2021

A (semester 2)	Dutch	Gent	seminar: practical PC room classes	36.0 h
			lecture	24.0 h

Lecturers in academic year 2020-2021

Ongenae, Veerle	TW05	lecturer-in-charge

Offered in the following programmes in 2020-2021	crdts	offering
Bachelor of Science in Engineering Technology (main subject Information Engineering Technology)	6	A
Linking Course Master of Science in Information Engineering Technology	6	A
Preparatory Course Master of Science in Information Engineering Technology	6	A

Teaching languages

Dutch

Keywords

UML, Systeemanalyse, Modellering, C#, Design Patterns, Dependency Injection, Computerwetenschappen (P170), Informatica (P175), Computertechnologie (T120)

Position of the course

The purpose of this course is to teach students advanced object oriented programming and design.
In the first part of this course, the software development is treated. The aim is to enable the student to bring small projects to a successful conclusion. Methods are thaught to produce high-quality software.
In addition, this course aims to provide students with insight into the available "design patterns" for software design and for typical software problems.

Contents

Part 1: Systems Analysis and Design
• Basics of good programming practice: characteristisation of good software and a good development.
• Reuse: how to reuse existing software and write code that can be reused.
• The different phases of the development process.
• Basics of UML.
• Requirements analysis and modelling: how to define the system to be developed.
• Design and realisation: converting a formal model to code.

Part 2: Design patterns
An overview of the most used "design patterns" and object oriented design principles: Strategy, Observer, Decorator,
Factory Method, Abstract Factory, Singleton, Command, Adapter, Facade, Template Method, Iterator, Composite, State, Proxy, MVC, Bridge, Builder, Chain of Responsibility, Flyweight, Interpreter, Mediator, Memento, Prototype, Visitor, inversion of control, dependency injection, ...
In addition, a number of advanced programming concepts are introduced: GUI programming, delegates, extension methods, asynchronous methods, ...

Initial competences

Being able to program and design in an object oriented way on an advanced level

Final competences

1 Being able to apply principles of software design to the practice of production, maintenance and quality
2 Being able to analyse, structure and translate a relatively complex problem into an object oriented design
3 Being able to convert an object oriented design to a working computer program in Java and to test this program critically
4 To have insight into the available "design patterns " for software design and for typical software problems
5 To be able to know when which pattern is useful
6 Being able to develop programs using patterns in a suitable way
7 To be able to refactor programs according to some patterns

Conditions for credit contract

Access to this course unit via a credit contract is determined after successful competences assessment

Conditions for exam contract

This course unit cannot be taken via an exam contract

Teaching methods

Lecture, seminar: practical PC room classes

Extra information on the teaching methods

- Lectures (24 hrs)
- Labs (36 hrs): individual work on PC; presence required

Learning materials and price

"C# 3.0 Design Patterns", Judith Bishop, O'Reilly, 2008, completed with teacher's course (Dutch), slides and examples of programming
Software: Visual Studio 2019 Community Edition

References

“Head First Design Patterns”, Eric Freeman, Elisabeth Robson, Bert Bates & Kathy Sierra, O'Reilly Media
"Design Patterns: Elements of Reusable Object-Oriented Software", Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Addison-Wesley
“Praktisch UML”, 5e editie, Jos Warmer en Anneke Kleppe, ISBN 9789043020558

Course content-related study coaching

The student can always make an appointment with the teachers

Evaluation methods

end-of-term evaluation and continuous assessment

Examination methods in case of periodic evaluation during the first examination period

Written examination, skills test

Examination methods in case of periodic evaluation during the second examination period

Written examination, skills test

Examination methods in case of permanent evaluation

Skills test

Possibilities of retake in case of permanent evaluation

examination during the second examination period is not possible

Extra information on the examination methods

Several computer tests on PC and same tasks during the labs.

Calculation of the examination mark

Exam: 60% (written examination (60%) and computer exercises(40%))
Exercises/Labs: 40% (tests en tasks)
In the second examination period: score = maximum (E, 40% L + 60% E), where L is the score of the lab and E the score of the exam in the second examination period