COURSE DESCRIPTION CARD - SYLLABUS

Course name
Software Evolution and Maintenance

Course

Field of study
Computing

Year/Semester
1/2

Area of study (specialization)
Software Engineering

Profile of study
general academic

Level of study
Second-cycle studies

Course offered in
English

Form of study
full-time

Requirements
elective

Number of hours

Lecture
30

Laboratory classes
30

Other (e.g. online)

Tutorials

Projects/seminars
15

Number of credit points

6

Lecturers

Responsible for the course/lecturer:
dr hab. inż. Bartosz Walter
email: bartosz.walter@put.poznan.pl
tel. +48 61 655 2980
Faculty of Computing and Telecommunications
ul. Piotrowo 2, 60-965 Poznań

Responsible for the course/lecturer:

Prerequisites

Student should have knowledge concerning software development processes and models, and basic skills in programming (at least in reading the code). They should also be capable of continuous learning and knowledge acquisition from selected sources, as well as express the readiness for collaborating in small teams.

Course objective

The objective for this course is to provide the students with knowledge on the processes of evolution of software systems, the types of evolutionary changes, and reacting to the evolution by planned and conscious maintenance activities. Students, upon completing the course, are expected to evaluate maintainability of a software system, apply changes and verify their correctness, as well as perform code reviews and apply refactorings.

Course-related learning outcomes

Knowledge

1. Students posesses well-grounded knowledge on the software system's life cycle.

2. Student posessess knowledge on selected methods, languages and notations used for developing software.

3. Student posesses knowledge on design patterns and best practices in software design

4. Student knows selected metrics and measurement methods for software quality characteristics (concerning the size, complexity, etc.)

Skills

1. Student can re-design, fix or update a software system.

2. Student can evaluate the design quality and analyze its impact on a software system.

Social competences

1. Student can effectively collaborate in small teams.

2. Student enhances their knowledge, based on commonly available source, making a conscious selection of them.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

The knowledge presented during the lecture will be verified two-fold: (i) by solving during the lectures in small teams two design case studies and discussing their pros and cons, and (ii) during the final examination (multilple-choice test that verifies the understading of the lectures). The two forms would be weighted 30:70, and the passing score is 50%. The list of examination problems will be provided during the last lecture within the course.

The skills acquired during laboratory classes will be verified by 3-4 group assignments, concerning the issues presented and discussed during the classes. The passing score is also 50%.

Programme content

1. Lecture: Overview of models of software evolution. Measurement and metrics for evolution and maintenance of software artifacts. Types of maintenance activities. Approaches to maintainability evaluation. Methods of restructuring and refactoring legacy systems. Observation and analysis of changes in software repositories.

2. Laboratory classes: Observation of software evolution. Collecting and analyzing evolution metrics. Flaws in software maintenance. Maintaining a software system in an iterative software development lifecycle. Techniques of refactoring.

Teaching methods

1. Lecture: multimedia presentation, discussion

2. Laboratory classes: presentation supported by provided examples, programming the software and design assignments in groups, discussion

Bibliography

Basic

1. T. Mens, S. Demeyer: Software Evolution. Springer Science and Business Media, 2008

2. R. C. Martin: Czysty kod. Podręcznik dobrego programisty. Helion, 2010

3. J. Visser et al.: Building Maintainable SOftware. Java Edition. Ten Guidelines for Future-Proof Code. O'Reilly Media, 2016.

Additional

1. M. Fowler: Refactoring. Improving the design of existing code. Addison-Wesley, 2018.

2. Priyadarshi Tripathy, Kshirasagar Naik: Software evolution and maintenance. A practitioner's approach. Addison Wiley, 2015

Breakdown of average student's workload

	Hours	ECTS
Total workload	150	6,0
Classes requiring direct contact with the teacher	75	3,0
Student's own work (literature studies, preparation for laboratory classes/tutorials, preparation for tests/exam, project preparation) [1]	75	3,0

[1] delete or add other activities as appropriate