COURSE DESCRIPTION CARD - SYLLABUS

Course name
Software engineering [S1Bioinf1>IO]

Course

Field of study
Bioinformatics

Year/Semester
3/5

Area of study (specialization)
–

Profile of study
general academic

Level of study
first-cycle

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
15

Laboratory classes
30

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

4,00

Coordinators

dr hab. inż. Piotr Zielniewicz
piotr.zielniewicz@put.poznan.pl

dr hab. inż. Mirosław Ochodek prof. PP
miroslaw.ochodek@put.poznan.pl

Lecturers

Prerequisites

The student starting this course should have basic knowledge of the fundamental of programming, tools used in computer science, algorithms and data structures, object-oriented programming. In addition, s/he should have the ability to solve basic problems in the field of programming and the ability to obtain information from the indicated sources.

Course objective

1) Provide students with basic knowledge of software engineering regarding: managing software projects, defining requirements, system modeling, software design, quality assurance (including software testing), software development support tools (including version control tools) 2) Developing students" skills in solving simple problems in the field of design, construction and testing of software, the use of software support tools, modifying and using programming components. 3) Developing students" skills so they can fulfill the roles of analysts / designers / programmers in a sofwtare development teams (also using agile methodologies).

Course-related learning outcomes

Knowledge:
1. Has a basic knowledge regarding IT project management.
2. Has basic knowledge of requirements engineering (functional requirements, use cases, non-functional requirements).
3. Has basic knowledge of modeling and software design.
4. Has a basic knowledge of software verification and validation methods.

Skills:
1. Is able to specify functional and non-functional requirements.
2. Is able to create a prototype of an application user interface.
3. Is able to create object models in the UML notation (classes, state machine, sequence).
4. Is able to create test cases and automate them (unit tests).
5. Is able to organize work in a software development team.
6. Is able to use the integrated programming environment.

Social competences:
1. Is able to interact and work in a software development team.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Learning outcomes presented above are verified as follows:
Forming Assessment:
a) lectures: on the basis of answers to questions and participation in quizzes during the classes
b) laboratory: based on the evaluation of the progress in the implementation of tasks during the classes
Summative assessment:
In terms of the assessment of the learning outcomes concerning the acquired skills and social competences (mainly the assessment of laboratory classes):
a) the final grade consists of two components: the evaluation of a mini-project implementation (50%) and a test (50%). The test consists of both test questions (multiple choice) and open-ended tasks (developing a UML diagram, writing a code fragment for unit testing).
In terms of learning outcomes related to the acquired knowledge:
a) during the lectures, students solve quizzes and short problem tasks. For providing an acceptable solution (depending on its form and nature), a student receives 2%.
b) multiple choice test including single-choice questions (one correct answer) and questions with possibly one or more correct answers (the question type is explicitly indicated in the test). The student receives 1 point for answering the question correctly. Points are converted to a percentage scale.
Based on the percentage points obtained (from the election test and during the lecture), the final grade is determined according to the scale:
- >= 90% - 5,0
- <80%, 90%) - 4,5
- <70%, 80%) - 4,0
- <60%, 70%) - 3,5
- <50%, 60%) - 3,0
- mniej niż 50% - 2,0

Programme content

The course program covers the following topics:
- Introduction to software engineering and the course, including the discussion of importance and role of software development in the modern world, vision of an IT project, consequences of software failures, the scope of software engineering
- Version management systems (Git)
- Functional and non-functional requirements and business process modeling
- Software modeling and analysis (including UML notation)
- Software prototyping
- Software design
- Project management methodologies (Scrum)
- Software testing (unit and acceptance)

The laboratory program covers the following topics:
- Software configuration management tools (Git)
- User interface prototyping
- Documenting requirements (use cases, user stories)
- Modeling of systems in the UML notation
- Software testing - unit testing
- Practical implementation of a mini-project
- Project patterns

Course topics

Lecture program includes the following topics:

Introduction to software engineering and Scrum methodology
Version control systems
Requirements in IT projects
Software design (UML notation)
Design patterns
Software testing

Laboratory program includes the following topics:

Organizational classes - presentation of course rules, organization of project teams
Project goal, context, user stories, use case diagram
Requirements specification
User interface prototyping
System modeling using UML notation (part I)
System modeling using UML notation (part II)
Work on own project
Design patterns - selected creational and structural patterns
Design patterns - selected behavioral patterns and practical implementation
Unit testing of software

Teaching methods

Teaching methods include:
a) lecture: multimedia presentation, presentation illustrated with examples given on the board, solving problems, case studies.
b) laboratory exercises: problem solving, practical exercises, discussion, team work, multimedia show, workshops, demonstration and implementation of a mini-project.

Bibliography

Basic
1. I. Sommerville, Inżynieria oprogramowania, Wydawnictwo Naukowe PWN, 2020
2. A. Jaszkiewicz, Inżynieria oprogramowania, Helion, 1997.
3. K. Schwaber, J. Sutherland, The Scrum Guide: Przewodnik po Scrumie: Reguły Gry, http://www.scrumguides.org, (dostępny online), 2017.
Additional
1. Wzorce projektowe w języku Java: https://www.tutorialspoint.com/design_pattern
2. Kopczyńska, Sylwia, Jerzy Nawrocki, and Mirosław Ochodek. An Empirical Study on Catalog of Non-functional Requirement Templates: Usefulness and Maintenance Issues. Information and Software Technology (2018).
3. Nawrocki, Jerzy, et al. Agile requirements engineering: A research perspective. International Conference on Current Trends in Theory and Practice of Informatics. Springer, Cham, 2014.

Breakdown of average student's workload

	Hours	ECTS
Total workload	100	4,00
Classes requiring direct contact with the teacher	45	2,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	55	2,00