COURSE DESCRIPTION CARD - SYLLABUS

Course name
Software Engineering [S1Teleinf1>IO]

Course

Field of study
Teleinformatics

Year/Semester
3/6

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
30

Laboratory classes
0

Other
0

Tutorials
0

Projects/seminars
30

Number of credit points

5,00

Coordinators

dr inż. Adam Wojciechowski
adam.wojciechowski@put.poznan.pl

Lecturers

Prerequisites

Algorithms and Data Structures I and II Object-Oriented Programming Languages

Course objective

none

Course-related learning outcomes

1 129 / 5 000
Knowledge:
1. Has general knowledge of software engineering
2. Has basic knowledge of the software development process phases
3. Knows and understands the principles of requirements definition, system modeling, software design, software testing and performance improvement, and software documentation.
4. Knows the basic methods, techniques, and tools used to solve simple IT tasks in software engineering.
5. Has knowledge of using CASE tools
6. Has knowledge of development trends and the most important new developments in software engineering
Skills:

1. Defines and describes requirements for simple IT systems
2. Models IT systems (e.g., in UML notation)
3. Designs and implements a simple IT system using appropriate methods, techniques, and tools
4. Tests software and improves its performance
5. Uses at least one popular version management system
6. Documents software
Social Competencies:
1. Is able to work in a team and understands the principles of social collaboration

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

The learning outcomes presented above are assessed as follows:
a) for lectures:
• Written examination
or
• Assessment of knowledge, skills, and innovation presented in laboratory and homework assignments
b) for project exercises:
• Assessment of the project based on presentations by project team members

Programme content

The course curriculum covers the following topics:
• Traditional lifecycle models and basic software development methodologies
• The requirements definition phase and the IEEE 830 standard
• System analysis/modeling using UML
• Software design and implementation
• Software reliability – error avoidance, fault tolerance, testing
• Software evolution, software commercialization models
• Software engineering tools, change and configuration management
• IT project management
• Software development methodologies
• Risk management in IT projects
• The innovation factor in software development.
The laboratory program includes students completing a project in which they will practically apply the following concepts:
• Requirements modeling (problem description, functional and non-functional requirements specification)
• System modeling in UML
• Software design
• Code implementation based on UML diagrams
• Software testing, creating unit tests
• Code documentation
• Version management tools: SVN, GIT

Course topics

The course curriculum covers the following topics:
• Traditional lifecycle models and basic software development methodologies
• The requirements definition phase and the IEEE 830 standard
• System analysis/modeling using UML
• Software design and implementation
• Software reliability – error avoidance, fault tolerance, testing
• Software evolution, software commercialization models
• Software engineering tools, change and configuration management
• IT project management
• Software development methodologies
• Risk management in IT projects
• The innovation factor in software development.
The laboratory program includes students completing a project in which they will practically apply the following concepts:

• Requirements modeling (problem description, functional and non-functional requirements specification)
• System modeling in UML
• Software design
• Code implementation based on UML diagrams
• Software testing, creating unit tests
• Code documentation
• Version management tools: SVN, GIT

Teaching methods

1. lecture: multimedia presentation, presentation illustrated with examples given on the board, solving tasks, case studies,

Bibliography

Basic:
1. A. Jaszkiewicz, Software Engineering, Helion, 1997.
2. G. Booch, J. Rambaugh, I. Jacobson, UML User's Guide, WNT, 2000.
3. M. Fowler, K. Scott, UML in a Drop, Oficyna Wydawnicza LTP, 2002.
4. Ian Sommerville, Software Engineering, WNT, 2003.
5. UML. Software Engineering. 2nd Edition, P. Stevens, Helion, Gliwice, 2007.
6. E. Gamma, R. Helm, R. Johnson, J. Vlissides, Design Patterns. Elements of Reusable Object-Oriented Software, WNT, 2008.

7. Sacha, Software Engineering, PWN, 2010.

Supplementary:

1. S. Maguire, Software Reliability, Helion, 2002.

2. J. W. Cooper, Java. Design Patterns, Helion, 2001.

3. R. S. Pressman, A Practical Approach to Software Engineering, WNT, 2004.

4. J. Warmer, A. Kleppe, OCL Software Engineering: Precise Modeling in UML, WNT, 2003.

5. W. B. Kernighan, R. Pike, Software Engineering: A Software Lesson, WNT, 2002.

6. R. C. Martin, M. Martin, Agile, Agile Programming, Helion, 2008.

Breakdown of average student's workload

	Hours	ECTS
Total workload	120	5,00
Classes requiring direct contact with the teacher	64	3,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	56	2,00