COURSE DESCRIPTION CARD - SYLLABUS

Course name
Software engineering [S2Elmob1-SPE>IO2]

Course

Field of study
Electromobility

Year/Semester
2/3

Area of study (specialization)
Energy Processing Systems

Profile of study
general academic

Level of study
second-cycle

Course offered in
Polish

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
0

Laboratory classes
15

Other
0

Tutorials
0

Projects/seminars
15

Number of credit points

2,00

Coordinators

mgr inż. Dominik Matecki
dominik.matecki@put.poznan.pl

Lecturers

Prerequisites

A student starting this subject should have basic knowledge of the basics of programming and mathematics. He should also have the ability to obtain information from indicated sources and be ready to cooperate within a team.

Course objective

Providing students with basic knowledge and good practices when creating software. Familiarization with popular tools for managing programming projects.

Course-related learning outcomes

Knowledge:
1. Has extended knowledge in the field of programming techniques and the use of modern IT tools for the analysis and synthesis of electrical systems of hybrid and electric vehicles, including traction vehicles.
2. Has theoretically based knowledge of modern methods of data collection, processing and analysis, also in the field of machine learning.
3. Has extended and systematized knowledge in the field of designing algorithms and programming

microcontrollers used in vehicles, as well as standards and the use of communication interfaces for exchanging data with vehicle components.
4. basic knowledge of data protection, IT system security, risk analysis and threat modeling in vehicle IT systems.

Skills:
1. Is able to formulate and test hypotheses related to complex engineering problems and simple research problems in the area of electromobility, as well as interpret the obtained results and draw critical conclusions.

Social competences:
1. Understands that in the field of technology, knowledge and skills devalue quickly, which requires constant supplementation.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

The skills acquired during laboratory classes are verified on the basis of a final test, consisting of 10-15 tasks with different points depending on their degree of difficulty, and on the basis of reports from laboratory classes. Passing threshold: 50% of points. The skills acquired during project classes are verified on the basis of cleared projects. Passing threshold: 50% of points.

Programme content

Software paradigms, design rules, component principle, architecture
software, "Clean code, project creation, error detection and handling,
Version control systems, functional security.

Course topics

Topics covered during the laboratory: discussion of software engineering issues, software paradigms (programming: structured, object-oriented and functional), design rules (SRP, OCP, Liskov substitution principle, ISP, DIP), component principle, software architecture, "Clean Code" (Creating names: variables, classes, methods, functions, etc.; Comments; Formatting; Objects and structures; Classes; Systems;), project creation, error detection and handling, Unified Modeling Language (UML), version control systems (SVN, Git), functional safety (ISO 26262 and ASIL), automotive Software Process Improvement and Capability (ASPICE), MISRA-C standard. Project topics refer to the content presented during the lecture and laboratory. They are focused on analyzing, solving and developing results for a complex issue composed of several laboratory topics.

Teaching methods

Laboratory exercises: multimedia presentation, presentation illustrated with examples given on the blackboard and carrying out tasks given by the teacher - practical exercises.
Projects: multimedia presentation, presentation illustrated with examples given on the board and development of tasks given by the teacher - own work.

Bibliography

Basic:
1. K.Beck, A.Cynthia, Wydajne programowanie - Extreme Programming, Mikom, 2005.
2. A. Cockburn, Jak pisać efektywne przypadki użycia, WNT, Warszawa 2004.
3. E. Gamma, R. Helm, R. Johnson, J. Vlissides, Wzorce projektowe, Elementy oprogramowania obiektowego wielokrotnego użytku, WNT, Warszawa, 2005
4. Shalloway A.,Trott James R., Projektowanie zorientowane obiektowo. Wzorce projektowe. Gliwice, Helion, 2005
5. S.Covey, 7 nawyków skutecznego działania REBIS, 2002.
6. M.Fowler, K.Scott, UML w kropelce, LTP, 2002.
7. R. Pressman, Software Engineering, McGraw-Hill, New York 1997.

Additional:

1. W. Humphrey, A Discipline for Software Engineering, Addison-Wesley, Reading 1995.
2. W. Humphery, Introduction to the Team Software Process, Addison-Wesley, Reading 2000.
3. Robert C. Martin, Czysty kod. Podręcznik dobrego programisty, Helion.
4. M. Świderski, A. Gąsiorek, Application of a control algorithm for the master unit of a distributed photovoltaic system, International Journal of Electronics and Telecommunications no. 4 vol. 68 2022.

Breakdown of average student's workload

	Hours	ECTS
Total workload	55	2,00
Classes requiring direct contact with the teacher	30	1,00
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	25	1,00