GDAŃSK UNIVERSITY
OF TECHNOLOGY

Subject card

Subject name and code	Software engineering, PG_00064064		
Field of study	Technical Physics		
Date of commencement of studies	October 2024	Academic year of realisation of subject	2026/2027
Education level	first-cycle studies	Subject group	Optional subject group Subject group related to scientific research in the field of study
Mode of study	Full-time studies	Mode of delivery	at the university
Year of study	3	Language of instruction	Polish
Semester of study	6	ECTS credits	7.0
Learning profile	general academic profile	Assessment form	exam
Conducting unit	Division of Theoretical Physics and Quantum Informaton -> Institute of Physics and Applied Computer Science -> Faculty of Applied Physics and Mathematics -> Faculties of Gdańsk University of Technology		
Name and surname of lecturer (lecturers)	Subject supervisor	dr hab. inż. Marta Łabuda	
	Teachers		

Lesson types	Lesson type	Lecture	Tutorial	Laboratory	Project	Seminar	SUM
	Number of study hours	30.0	0.0	0.0	45.0	0.0	75
	E-learning hours included: 0.0						

Learning activity and number of study hours	Learning activity	Participation in didactic classes included in study plan	Participation in consultation hours	Self-study	SUM
	Number of study hours	75	5.0	95.0	175

Subject objectives	The aim of the course is to familiarize students with the principles of planning, architectural design, and software development methods, including scientific software. The objective of the project classes is the practical application of knowledge in software engineering through the implementation of a team-based IT project. Students learn how to plan and manage a project, analyze and specify requirements, select an appropriate software development model, and design the architecture and structure of a system. The classes develop skills in system modeling using UML, teamwork, project documentation, and the use of modern tools and technologies that support the software development process.

Learning outcomes	Course outcome	Subject outcome	Method of verification
	[K6_U03] knows programming languages and can use basic software packages	The student is able to create, compile, and run programs in a selected programming language (e.g., Python), using basic language constructs such as control statements, data structures, and functions/methods. The student is able to use selected software packages and development tools (e.g., IDEs, version control systems, repositories, standard libraries, and frontend frameworks) in the software development process.	[SU1] Assessment of task fulfilment [SU4] Assessment of ability to use methods and tools
	[K6_K05] presents own work results, transfers information in a commonly understandable manner, communicate and self-evaluate, as well as constructively evaluate the effects of other persons' work	The student is able to prepare and present the results of a team IT project in a way that is understandable to audiences with varying levels of technical knowledge. The student is able to perform selfassessment of their own work and provide constructive evaluation of the contributions and outcomes of other team members.	[SK4] Assessment of communication skills, including language correctness [SK3] Assessment of ability to organize work
	[K6_W05] has knowledge of programming methodology and techniques, and the use of selected IT tools in physics and technology	The student is familiar with software development methodologies (e.g., waterfall, iterative, agile) and basic programming techniques used in software engineering. The student is familiar with selected IT tools supporting the software development process (development environments, version control systems, modeling tools such as CASE tools for UML diagrams, and testing tools), is able to use them, and understands their application in problems of physics and engineering.	[SW1] Assessment of factual knowledge
	[K6_K04] cooperate and work in a group, performing different functions	The student is familiar with the basic team roles in IT projects (e.g., analyst, designer, programmer, tester, team leader, Scrum Master, Product Owner) and the principles of effective teamwork in software engineering. The student is able to work effectively in a project team, responsibly fulfilling the assigned role, respecting established collaboration rules, and adapting to the changing needs of the team. The student is able to communicate within a project team, share tasks, report progress, and participate in team decision-making.	[SK1] Assessment of group work skills [SK3] Assessment of ability to organize work

Subject contents	Course content – lecture 1. Introduction to software engineering. Characteristics of computer systems engineering. 2. Planning an IT project: basic characteristics, concepts, project stakeholders; project lifecycle and scope. Task planning. Problem identification; enriched representation. 3. Feasibility study of an IT project. Objectives, evaluation dimensions: technical, economic, organizational, and legal; project risk assessment. 4. Requirements engineering process. Defining and analyzing requirements. Software requirements and their documentation. Characteristics of a good requirement. Methods of requirements elicitation. Classification and categorization of requirements. Requirements approval. Requirements management. 5. Modeling the software development process. Project and software development lifecycle. 6. Strategies and processes for managing IT projects: traditional approaches (waterfall, V-model, prototyping, incremental, spiral). 7. Agile software development methodologies (Agile, reuse, and component-based development). Extreme programming. SCRUM: processes, artifacts, roles. Selection of project management strategies. 8. Architectural design. System structure, control models, modular decomposition, architectures characteristic for different IT products. 9. Object-oriented design. Objects and classes, object-oriented design processes, project evolution. 10. UML language. CASE tools for computer-aided software design. 11. Design of distributed systems. Analysis and design patterns. Classification and examples. 12. Real-time systems design. Hardware architecture. Design of critical systems. Safety and failure analysis. 13. System administration. Containerization, microservices, cloud computing. 14. Data access design and data organization. 15. Prototyping: development environments and technologies. AI in system design. Course content – project PROJECT: Project classes include the identification and analysis of requirements as well as object-oriented modeling using CASE tools and architectural design. The work is carried out in small teams. Each group completes a set of exercises related to an area selected by the team and intended for computerization. The outcome is a complete set of documentation (project assumptions, feasibility report, system requirements specification) and a system architecture design, including UML diagrams, system administration, interface design, data management, as well as a prototype outline of the implementation.	
Prerequisites and co-requisites	None.	
Assessment methods and criteria		

Subject passing criteria	Passing threshold	Percentage of the final grade
Project tasks	50.0%	75.0%
Presentation	50.0%	10.0%
Tests for exam	50.0%	15.0%

Recommended reading	Basic literature	1. Krzysztof Sacha, Inżynieria Oprogramowania, PWN 2022 2. Ian Sommerville, Inżynieria Oprogramowania, wyd.11, PWN 2019 3. Max Kanat-Alexander, Zrozumieć oprogramowanie, Helion 2019 4. Robert C. Martin, Czysta architektura. Struktura i design oprogramowania. Przewodnik dla profesjonalistów, Helion 2022 5. Mike Cohn, Agile. Metodyki zwinne w planowaniu projektów, Helion 2019 6. Robert C. Martin, Zwinne wytwarzanie oprogramowania. Najlepsze zasady, wzorce i praktyki, Helion 2017

	Supplementary literature	1. Bernd Bruegge, Allen H. Dutoit, Inżynieria oprogramowania w ujęciu obiektowym UML, wzorce projektowe i Java Helion 2011 2. Keeling Michael, Zostań architektem oprogramowania, PWN 2019 3. Piotr Gaczkowski, Adrian Ostrowski, Architektura oprogramowania bez tajemnic, Helion 20224. 4. Praca zbiorowa, *Inżynieria oprogramowania w praktyce*, PWN, 2022. 5. Strony domowe do wybranych narzędzi informatycznych. Instrukcje obsługi, przykłady. 6. Roger S. Pressman, Bruce R. Maxim, *Inżynieria oprogramowania. Praktyczne podejście*, Helion, 2021 (wyd. 9)
	eResources addresses	Supplementary http://cleancoder.com/products - Homepage of Robert Martin, software engineer, instructor and author. Expert in software design and development.
Example issues/ example questions/ tasks being completed	1.Planning and feasibility assessment of an IT project: technical, economic, organizational analysis and risk evaluation. 2.Requirements engineering: methods for eliciting, documenting, and managing software requirements. 3.IT project lifecycle and software development models: comparison of traditional and agile approaches. 4.Designing IT system architecture: system structure, modular decomposition, and architectural styles. 5.Object-oriented design and UML modeling: classes, objects, and diagrams supporting system design. 6. Agile IT project management methodologies: SCRUM, Agile, extreme programming, and project strategy selection. 7.Modern technologies in software engineering: containerization, microservices, cloud computing, and AI in system design. Optional: trip to the Tricity Academic Computer Network Center (Centrum Informatyczne Trójmiejskiej Akademickiej Sieci Komputerowej).	
Practical activites within the subject	Not applicable	