University of Gdańsk

Subject card

Subject name and code	Software Engineering, PG_00198514		
Field of study	Informatics		
Date of commencement of studies	October 2026	Academic year of realisation of subject	2027/2028
Education level	Bachelor's studies	Subject group	Obligatory subject group in the field of study Subject group related to scientific research in the field of study
Mode of study	part-time studies	Mode of delivery	at the university
Year of study	2	Language of instruction	Polish
Semester of study	4	ECTS credits	5.0
Learning profile	academic	Assessment form	credit
Conducting unit			
Name and surname of lecturer (lecturers)	Subject supervisor	dr Robert Fidytek	
	Teachers		

Lesson types

Lesson type	Lecture	Tutorial	Laboratory	Project	Seminar	SUM
Number of study hours	10.0	0.0	20.0	0.0	0.0	30

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	30	0.0	95.0	125

Subject objectives

The aim of the course is to familiarise students with the basic problems, methods, techniques and tools of quality software production
quality. Introducing various software life cycle models, software development methods, basics of documentation, requirements analysis
design, software testing. Discussion of requirements analysis management, creating specifications with defined metrics.
Topics discussed presented with emphasis on object-oriented approach to design and system modelling using UML.

Learning outcomes

Course outcome	Subject outcome	Method of verification
[INFOL3_W04] knows and understands advanced concepts in the field of software engineering, software specifications, validation and verification, and tools supporting the software development process	Has knowledge of software lifecycle models, requirements elicitation/analysis/specification, verification and validation, and testing; knows the basics of OO analysis and design and of UML modelling (class and dynamic models), and the principles of moving from model to design to program.	[SW4] test/exam - oral or written [SW2] presentation/project/paper/ report
[INFOL3_U06] is able to select and apply appropriate IT methods and tools to solve complex problems, is able to use acquired knowledge by appropriately selecting sources and information derived from them, and perform evaluation, critical analysis, and synthesis of this information	Is able to elicit and specify requirements, model a system in UML (use cases, class model, sequence and state diagrams), and select appropriate methods/ tools for analysis, design and testing, using documentation and other sources.	[SU2] presentation/project/paper/ report [SU4] test/exam - oral or written

<table>
<tr><td>Subject contents</td><td>1. introduction; motivations for systematic software development.
2. software life cycle, phases of the classical cycle, place of analysis and design in the development cycle.
3 Requirements. Categories, elicitation, analysis and specification of requirements. Verification and validation; acceptance testing.
4. modelling; world modelling vs. system modelling, place of the model in analysis and design.
5 Use cases. Diagram, interpretation, structured description, examples.
6. Overview of basic principles and concepts of the object-oriented paradigm: object, classification, aggregation, inheritance, communication. Objectivity vs.
human's natural way of perceiving reality.
7. Introduction to the UML methodology: class model, dynamic model. Relationships between models.
8 Objects, classes, attributes, operations. Relationships between classes (associations). Relationships as classes. Roles. Aggregation as a special case of a relationship.
Semantics of aggregation. Propagation of operations as an aggregation criterion. Examples.
9 Inheritance: specialisation and generalisation. Hierarchy of inheritance. Redefining properties of objects down the inheritance hierarchy,
Obscuring Abstract classes. An object as an instance of a class and all its superclasses.
10. How to create a class model? Example of creating a class model.
11. dynamic model: events, actions, activities. Sequence (interaction) diagrams. Linking to m other models. How to create a sequence diagram.

12 State diagram. States: simple, complex, concurrent. inheritance of states. Transitions between states. Linking to the class model. How to create a
diagram of a dynamic model? Example of creating a dynamic model.
13 Moving from model to design. Basic phases of object-oriented design. Division into subsystems and modules, identification of concurrency,
Processor allocation, resource allocation, priorities.
14 Moving from design to program.</td></tr>
<tr><td>Prerequisites and co-requisites</td><td></td></tr>
<tr><td>Assessment methods and criteria</td><td>
<table>
<tr><th>Subject passing criteria</th><th>Passing threshold</th><th>Percentage of the final grade</th></tr>
<tr><td>test</td><td>51.0%</td><td>50.0%</td></tr>
<tr><td>project</td><td>51.0%</td><td>50.0%</td></tr>
</table>
</td></tr>
<tr><td>Recommended reading</td><td>
<table>
<tr><td>Basic literature</td><td>• I. Sommerville, Inżynieria oprogramowania, WNT/PWN.
• K. Sacha, Inżynieria oprogramowania, PWN.
• G. Booch, J. Rumbaugh, I. Jacobson, UML. Przewodnik użytkownika, WNT.</td></tr>
<tr><td>Supplementary literature</td><td>• M. Fowler, UML w kropelce, Oficyna Wydawnicza LTP.
• S. Wrycza, B. Marcinkowski, K. Wyrzykowski, Język UML 2.0 w modelowaniu systemów informatycznych, Helion.
• C. Larman, Stosowanie UML i wzorców projektowych, Helion.</td></tr>
<tr><td>eResources addresses</td><td></td></tr>
</table>
</td></tr>
<tr><td>Example issues/ example questions/ tasks being completed</td><td>Analyse the requirements for the given case study and develop the corresponding UML/BPMN diagrams.</td></tr>
<tr><td>Work placement</td><td>Not applicable</td></tr>
</table>

Document generated electronically. Does not require a seal or signature.