Software modelling techniques - course description

General information

Course name	Software modelling techniques
Course ID	11.3-WE-INFD-SoftModellTechn-Er
Faculty	Faculty of Computer Science, Electrical Engineering and Automatics
Field of study	Computer Science
Education profile	academic
Level of studies	Second-cycle Erasmus programme
Beginning semester	winter term 2022/2023

Course information

Semester	1
ECTS credits to win	5
Course type	obligatory
Teaching language	english
Author of syllabus	• dr inż. Grzegorz Bazydło

Classes forms

The class form	Hours per semester (full-time)	Hours per week (full-time)	Hours per semester (part-time)	Hours per week (part-time)	Form of assignment
Lecture	30	2	-	-	Credit with grade
Laboratory	30	2	-	-	Credit with grade

Aim of the course

- Familiarize students with the bases of software engineering and program modelling techniques.
- Shaping skills in business process modelling.
- Familiarize students with object modelling principles.
- Shaping skills in program modeling with the use of Unified Modelling Language (UML).

Prerequisites

Object-oriented programming

Scope

- Elements of software engineering. Software development. The software crisis and countermeasures.
- Conceptual modeling. The role of modeling in software design. Historical illustration of modern modeling techniques. Model-Driven Development approach. Model-Driven Architecture.
- Business analysis. Business process modeling in BPMN notation. Business use cases. Modeling software based on the BPMN model.
- Unified Modeling Language. Origin, definition, and goals of UML. UML diagrams description.
- Model-Driven Development and Model-Driven Architecture.
- Analysis, specification, and documenting of the user requirements. Use case modelling. Solution architecture design.
- Agile methods. Software life cycle.
- Fundamentals of object-oriented design (e.g., classes, inheritance, generalization, specialization, polymorphism), relations between objects. System model development.
- Modeling the user interface.

Teaching methods

Lecture: conventional lecture.
Laboratory: laboratory exercises.

Learning outcomes and methods of theirs verification

Outcome description	Outcome symbols	Methods of verification	The class form
Student knows the basics of object-oriented programming and can design programs using an object-oriented paradigm.		• an ongoing monitoring during classes	• Laboratory
Student can model the software using the appropriate modelling languages.		• an ongoing monitoring during classes	• Laboratory
Student knows the basics of UML, the most important types of UML diagrams and their use.		• a test	• Lecture
Student has the knowledge about languages and techniques of modelling software and business processes.		• a test	• Lecture

Outcome description	Outcome symbols	Methods of verification	The class form
Student understands the need for software modelling to facilitate its design and increase its credibility.		• a test	• Lecture

Assignment conditions

Lecture: the main condition to get a pass are sufficient marks for all written tests conducted during the semester.

Laboratory: a condition of pass is to obtain positive grades from all laboratory exercises that are expected to be performed within the laboratory program.

Composition of the final grade: lecture: 50% + laboratory: 50%

Recommended reading

1. Sommerville I.: Software Engineering (10th Edition), Pearson Education, 2016.
2. Booch G., Rumbaugh J., Jacobson I.: The Unified Modeling Language User Guide, Second Edition, Addison-Wesley, 2005.
3. Pilone D., Pitman N.: UML 2.0 in a Nutshell, A Desktop Quick Reference, O'Reilly Media, 2005.
4. Shapiro R., White S. A., Bock C., Palmer N. et al: BPMN 2.0 Handbook Second Edition, Future Strategies Inc., 2012.
5. Martin R. C.: Agile Software Development, Principles, Patterns, and Practices, Pearson Education, 2013.

Further reading

1. Brookes F. P.: The Mythical Man-Month, Anniversary Edition: Essays On Software Engineering, Addison-Wesley, 2010.
2. Osterwalder A., Pigneur Y.: Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers, John Wiley & Sons, 2010.
3. Rasmusson J.: The Agile Samurai: How Agile Masters Deliver Great Software, The Pragmatic Programmers LLC, 2010.
4. Rumbaugh J., Jacobson I., Booch G.: The Unified Modeling Language Reference Manual, Second Edition, Addison-Wesley, 1999.

Notes

Modified by dr inż. Grzegorz Bazydło (last modification: 19-04-2022 19:04)