

# Foundations of software modelling - opis przedmiotu

Informacje ogólne	
Nazwa przedmiotu	Foundations of software modelling
Kod przedmiotu	11.3-WE-INFP-FounofSM-Er
Wydział	Wydział Nauk Inżynieryjno-Technicznych
Kierunek	Informatyka
Profil	ogólnoakademicki
Rodzaj studiów	Program Erasmus pierwszego stopnia
Semestr rozpoczęcia	semestr zimowy 2021/2022

Informacje o przedmiocie	
Semestr	5
Liczba punktów ECTS do zdobycia	7
Typ przedmiotu	obieralny
Język nauczania	angielski
Sylabus opracował	• dr inż. Łukasz Hładowski

Formy zajęć					
Forma zajęć	Liczba godzin w semestrze (stacjonarne)	Liczba godzin w tygodniu (stacjonarne)	Liczba godzin w semestrze (niestacjonarne)	Liczba godzin w tygodniu (niestacjonarne)	Forma zaliczenia
Wykład	30	2	-	-	Egzamin
Laboratorium	30	2	-	-	Zaliczenie na ocenę
Projekt	15	1	-	-	Zaliczenie na ocenę

## Cel przedmiotu

- obtaining basic knowledge about software modelling
- familiarizing students with practical applications of software modelling for simple software systems
- familiarizing students with proper way of practical implementation of solution to simple problems using software design patterns

## Wymagania wstępne

Object-oriented programming, Software Engineering

## Zakres tematyczny

*Introductory issues.* Background and history of modern modelling techniques. Unified process of application life cycle. System analysis and design. Object paradigm. Object modelling and its role in design of information systems. Class-Responsibility-Collaboration (CRC) diagrams. Software production processes.

*Introduction to Unified Modelling Language (UML) notation and diagrams.* Genesis and purpose of UML. Structural modelling. Basic notions and elements of object architecture: classes, objects, abstractions, encapsulation, inheritance, polymorphism, communication, relations and associations between objects. Static structural diagrams: class and object diagrams. Association modelling: aggregation, composition, generalization, specialization, dependencies and realization. Packages and subsystems. Types, interfaces and implementation classes. Implementation diagrams: component and deployment diagrams. Requirements and their specification. Use case diagrams. Use case analysis: inclusion, extension, grouping and generalization. Behavioural modelling. Sequence and collaboration diagrams. Roles, messages and stimuli. Interactions and collaborations. Analysis of system states. State and activity diagrams. Flow transfer. Decisions. Concurrency. Signals and communication

*Design patterns.* Formulation of programming problems. Overview of most popular construction, structural and behavioural design patterns. Creational and testing patterns.

*Practical issues.* Work with use cases. General overview on design, deployment and testing. Presentation of dedicated UML design tools.

## Metody kształcenia

**lecture:** brainstorm, discussion, practical tasks, conventional lecture

**laboratory:** brainstorm, working with source files, discussion, working in groups, practical tasks, conventional lecture

**project:** brainstorm, working with source files, discussion, working in groups, practical tasks, conventional lecture

## Efekty uczenia się i metody weryfikacji osiągnięcia efektów uczenia się

Opis efektu	Symbole efektów	Metody weryfikacji	Forma zajęć
-------------	-----------------	--------------------	-------------

Opis efektu	Symbole efektów	Metody weryfikacji	Forma zajęć
Understands the need for unit tests and can implement them for simple cases. Can use simple programming tools for testing.		<ul style="list-style-type: none"> <li>projekt</li> </ul>	<ul style="list-style-type: none"> <li>Wykład</li> <li>Projekt</li> </ul>
Can implement a fragment of simple system in chosen programming language using design patterns and proper object-oriented techniques.		<ul style="list-style-type: none"> <li>projekt</li> </ul>	<ul style="list-style-type: none"> <li>Projekt</li> </ul>
Uses UML for description and formulation of solutions to programming problems.		<ul style="list-style-type: none"> <li>projekt</li> <li>sprawdzian</li> <li>egzamin</li> </ul>	<ul style="list-style-type: none"> <li>Wykład</li> <li>Laboratorium</li> <li>Projekt</li> </ul>
Can implement simple design patterns in chosen programming language. Knows disadvantages and advantages of a chosen pattern and can propose alternative solution.		<ul style="list-style-type: none"> <li>projekt</li> <li>sprawdzian</li> <li>test egzaminacyjny z progami punktowymi</li> </ul>	<ul style="list-style-type: none"> <li>Wykład</li> <li>Laboratorium</li> <li>Projekt</li> </ul>
Can notice and describe advantages and disadvantages of proposed solution during the UML modelling phase.		<ul style="list-style-type: none"> <li>projekt</li> </ul>	<ul style="list-style-type: none"> <li>Projekt</li> </ul>
Knows, understands and uses simple rules for software engineering.		<ul style="list-style-type: none"> <li>projekt</li> <li>sprawdzian</li> <li>test egzaminacyjny z progami punktowymi</li> </ul>	<ul style="list-style-type: none"> <li>Wykład</li> <li>Laboratorium</li> <li>Projekt</li> </ul>
Can choose proper tools for software engineering.		<ul style="list-style-type: none"> <li>bieżąca kontrola na zajęciach</li> <li>sprawdzian</li> </ul>	<ul style="list-style-type: none"> <li>Laboratorium</li> </ul>
Can notice and describe advantages and disadvantages of the proposed solution by source code analysis. Can refactor simple code.		<ul style="list-style-type: none"> <li>projekt</li> </ul>	<ul style="list-style-type: none"> <li>Wykład</li> <li>Projekt</li> </ul>

## Warunki zaliczenia

**Lecture** - a credit is given for obtaining a passing grade for all exams administered at least once per semester

**Laboratory** - to receive a final passing grade student has to receive passing grades for all tasks required by the curriculum.

**Project** - to receive a final passing grade student has to receive passing grades for all tasks and projects required by the curriculum..

**Calculation of the final grade** = lecture: 33% + laboratory: 33% + project: 33%

## Literatura podstawowa

1. Martin R.C.: *Clean Code: A Handbook of Agile Software Craftsmanship*, Prentice Hall, 2008,
2. Martin R.C.: *Clean Architecture: A Craftsman's Guide to Software Structure and Design*, Prentice Hall, 2017,
3. Freeman E., Freeman E., Bates B., Sierra K.: *Head First Design Patterns!*, O'Reilly Media, 2004,
4. *UML @ Classroom*, Seidl, M., Scholz, M, Springer International Publishing, 2015,
5. Larman C.: *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, (3rd Edition), Prentice Hall, 2004.

## Literatura uzupełniająca

1. Martin R.C., Martin M.: *Agile Principles, Patterns, and Practices in C#*, Prentice Hall, 2006,
2. Beck K.: *Test Driven Development: By Example*, Addison-Wesley Professional, 2002,
3. Way J.: *Laravel Testing Decoded*, Leanpub 2013.

## Uwagi

Zmodyfikowane przez dr inż. Łukasz Hładowski (ostatnia modyfikacja: 18-07-2021 12:49)