KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu **Architektura i weryfikacja oprogramowania**		Kod **1010512321010517863**
Kierunek studiów **Informatyka**	Profil kształcenia (ogólnoakademicki, praktyczny) **ogólnoakademicki**	Rok / Semestr **1 / 2**
Ścieżka obieralności/specjalność **Software Engineering (Inżynieria**	Przedmiot oferowany w języku: **polski**	Kurs (obligatoryjny/obieralny) **obligatoryjny**

Stopień studiów: **II stopień**	**Forma studiów** (stacjonarna/niestacjonarna) **stacjonarna**
Godziny: Wykłady: **30** Ćwiczenia: **-** Laboratoria: **30** Projekty/seminaria: **-**	Liczba punktów **5**
Status przedmiotu w programie studiów (podstawowy, kierunkowy, inny) **inny**	(ogólnouczelniany, z innego kierunku) **ogólnouczelniany**

Obszar(y) kształcenia i dziedzina(y) nauki i sztuki	**Podział ECTS (liczba i %)**
nauki techniczne	**5 100%**
nauki techniczne	**5 100%**

Odpowiedzialny za przedmiot / wykładowca:

Bartosz Walter
email: bartosz.walter@cs.put.poznan.pl
tel. 616652980
Wydział Informatyki
ul. Piotrowo 3 60-965 Poznań

Odpowiedzialny za przedmiot / wykładowca:

Michał Maćkowiak
email: michal.mackowiak@cs.put.poznan.pl
tel. 616652944
Wydział Informatyki
ul. Piotrowo 3 60-965 Poznań

Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

1	**Wiedza:**	Student starting this module should have a basic knowledge regarding basic algorithms and computational complexity, object-oriented programming, design patterns, databases, software testing and web applications.
2	**Umiejętności:**	Should have skills allowing solving basic problems related to requirements analysis, creating software specification, designing systems and skills that are necessary to acquire information from given sources of information.
3	**Kompetencje społeczne**	Student should understand the need to extend his/her competences / has the willingness to work in a team. In addition, with respect to the social skills, the student should demonstrate such attitudes as honesty, responsibility, perseverance, curiosity, creativity, manners, and respect for other people.

Cel przedmiotu:

1. Provide students with knowledge regarding software architecture, within the following scope of understanding what is software architecture, how it should be documented and evaluated

2. Introduce students to component- and service-oriented architectures

3. Develop students' teamwork skills in the context of designing software systems

Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia

Wiedza:

1. has organized and well-formed theoretical general knowledge regarding software architecture, software testing and software modeling - [K2st_W2]

2. has advanced and detailed knowledge related to selected areas of computer science creating software architecture, documenting system architecture, evaluation of architecture, modeling software, designing software, testing and verifying software - [K2st_W3]

3. has advanced and detailed knowldege regarding software life cycle which involves designing software architecture, modeling, and creating unit tests - [K2st_W5]

Umiejętności:

1. is able to acquire, combine, interpret and evaluate information from literature, databases and other information sources (in mother tongue and English); draw conclusions, and formulate opinions based on it. - [K2st_U1]

2. is able to combine knowledge from different areas of computer science (and if necessary from other scientific disciplines) to formulate and solve engineering tasks; and use system approach that also incorporates nontechnical aspects - [K2st_U5]

3. is able to assess usefulness and possibility of employing new developments (methods and tools) and new IT products - [K2st_U6]

4. is able to design and evaluate software architecture of complex software systems, using appropriate methods - [K2st_U10]

5. is able to design (according to a provided specification which includes also non-technical aspects) a complex device, an IT system, or a process; and is able implement it (at least partially) using appropriate methods, techniques, and tools (including adjustment of available tools or developing new ones - [K2st_U11]

6. is able to work in a group, performing different roles, like architect, developer, tester - [K2st_U15]

Kompetencje społeczne:

1. student understands that knowledge and skills related to computer science quickly become obsolete - [K2st_K1]

2. student knows examples and understands the causes of the failures of IT systems that have led to major financial or social losses, or caused damage to health or even death - [K2st_K2]

Sposoby sprawdzenia efektów kształcenia

Formative assessment:

a) lectures:

* based on the answers to the questions which test understanding of material presented on the lectures

b) laboratory classes / tutorials / projects / seminars:

* based on the assessment of the tasks done during classes and as a homework

Summative assessment:

a) verification of assumed learning objectives related to lectures:

* assessment of knowledge and skills, examined by a written test with multiple choices and problem questions. Student can gain 100 points, to pass minimum 50 points are needed

* the final grade is determined using the following scale:

- (90%, 100%] ? 5.0
- (80%, 90%] ? 4.5
- (70%, 80%] ? 4.0
- (60%, 70%] ? 3.5
- (50%, 60%] ? 3.0
- (0%, 50%] ? 2.0

* discussing the results of the examination

b) verification of assumed learning objectives related to laboratory classes / tutorials / projects / seminars:

* assessment of student's preparation to particular laboratory classes and assessment of student's skills needed to realize tasks on these classes
* continuous assessment of student's work during classes - rewarding ability to use learned principles and methods
* assessment of projects realization, including ability to work in team

Possibility to gain additional points by activity on classes:

* elaboration of additional aspects regarding the subject
* effectiveness of applying acquired knowledge to solve problems
* ability to cooperate with the team during solving problems
* providing additional remarks for the lecturer how to improve teaching materials
* highlighting the problems with students' perception to improve the teaching process

Treści programowe

The program of the lecture:

Definition of software architecture. Role of the architect. Process of creating software architecture. Types of software architects. How and what should be documented in description of software architecture. Why the architecture should be evaluated. Description of ATAM (Architecture Tradeoff Analysis Method). Principles of good diagrams. Definition of component-based architecture. Properties of a component. Inversion of control. Dependency injection methods. Role of a component container. Review of component container technologies. Definition of service-oriented architecture. Implementations of service-oriented architecture: web services and REST approach. Modeling constraints for UML models with OCL. Defining pre-and post-conditions for operations. Validation of OCL expressions. The Design by Contract concept as a semin-formal method for specifying functionality. Modeling with Eclipse Modeling Framework. Overview of testing methods at different levels. Role and structure of tests in a software project.

The course consists of fifteen 2-hour laboratory classes and it starts with an instructional session at the beginning of a semester. Students work individually or in teams of 2-4.

The program of laboratory classes is following:

Creating a software architecture description, including usability tree, design decisions and architectural views. Preparation to an ATAM meeting. Performing an ATAM meeting to evaluate the architecture of a sample system. Realization of software development tasks related to a component-based application using Unity 2.0 Container for .NET framework. Creating a system based on software-oriented architecture using web services for .NET framework. Reconfiguring the system in the way the services conform to REST approach. Defining and interpreting OCL constraints for an existing UML model. Defining pre- and post-conditions for operations and methods. Inheritance of pre- and post-conditions. Defining a model and generating application framework with Eclipse Modeling Framework. Designing test cases on different levels of tests. Computing test quality measures. Implementing acceptance tests in selected technologies.

Literatura podstawowa:

1. L. Bass, P. Clements, R. Kazman, "Software architecture in practice", WNT
2. P. Kruchten, "The Rational Unified Process-An Introduction", Addison-Wesley
3. R. V. Binder: "Testing Object-Oriented Systems: Models, Patterns and Tools", Addison-Wesley

Literatura uzupełniająca:

1. D. Spinellis and G. Gousios, "Beautiful Architecture", O'Reilly Media

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)
1. participating in laboratory classes / tutorials: 15 x 2 hours	30
2. consulting issues related to the subject of the course; especially related to the laboratory classes and projects,	2
3. implementing, running and verifying software application(s) (in addition to laboratory classes)	25
4. participating in lectures	30
5. studying literature / learning aids (10 pages = 1 hour), 150 pages	15
6. discussing the results of the examination	1
7. preparing to and participating in exams: 15 hours + 2 hours	17

Obciążenie pracą studenta

forma aktywności	godzin	ECTS
Łączny nakład pracy	120	5
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	63	3
Zajęcia o charakterze praktycznym	57	2