STUDY MODULE DESCRIPTION FORM

Name of the module/subject	Code
Software Architecture and Verification	**1010512321010517863**

Field of study	Profile of study (general academic, practical)	Year /Semester
Computing	**general academic**	**1 / 2**
Elective path/specialty: **Software Engineering**	Subject offered in: **English**	Course (compulsory, elective): **obligatory**
Cycle of study: **Second-cycle studies**	Form of study (full-time,part-time): **full-time**	
No. of hours: Lecture: **30** Classes: **-** Laboratory: **30** Project/seminars: **-**		No. of credits: **5**
Status of the course in the study program (Basic, major, other): **major**	(university-wide, from another field): **from field**	
Education areas and fields of science and art: **technical sciences**		ECTS distribution (number and %): **5 100%**

Responsible for subject / lecturer:

Bartosz Walter PhD
email: Bartosz.Walter@cs.put.poznan.pl
tel. 61 6652980
Institute of Computing Science
Piotrowo 2 Str., 60-965 Poznan

Prerequisites in terms of knowledge, skills and social competencies:

1	**Knowledge**	Learning objectives of the first cycle studies defined in the resolution of the PUT Academic Senate, especially K_W1-2, K_W4, K_W6-15 that are verified in the admission process to the second cycle studies ? the learning objectives are available at the website of the faculty www.fc.put.poznan.pl 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	**Skills**	Learning objectives of the first cycle studies defined in the resolution of the PUT Academic Senate, especially K_U1-2, K_U4, K_U7-8, K_U14-20, K_U22-23, K_U26 that are verified in the admission process to the second cycle studies ? the learning objectives are available at the website of the faculty www.fc.put.poznan.pl 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	**Social competencies**	Learning objectives of the first cycle studies defined in the resolution of the PUT Academic Senate, especially K_K1-9 that are verified in the admission process to the second cycle studies ? the learning objectives are available at the website of the faculty www.fc.put.poznan.pl 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.

Assumptions and objectives of the course:

1. Provide students 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

Study outcomes and reference to the educational results for a field of study

Knowledge:

1. has well-established theoretical knowledge of computer systems architecture, software design, software testing and verification, software engineering - [K_W4]

2. has detailed theoretical 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 - [K_W5]

3. has knowledge regarding trends and the most important new developments in computer science and related disciplines - [K_W6]

4. has basic knowledge regarding life-cycle of software or hardware systems - [K_W7]

5. knows the fundamental methods, techniques and tools employed to solve complex engineering tasks in a selected area of software architecture, software modeling and design, software testing and verification - [K_W8]

Skills:

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. - [K_U1]

2. is able to plan and arrange self-education process - [K_U5]

3. has language skills at B2+ level in accordance with the requirements set out for level B2+ Common European Framework of Reference for Languages - [K_U6]

4. is able to employ analytical, simulation, and experiment methods to formulate and solve engineering tasks and basic research problems - [K_U9]

5. 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 - [K_U10]

6. is able to formulate and test hypotheses regarding engineering problems and basic research problems - [K_U12]

7. is able to assess usefulness and possibility of employing new developments (methods and tools) and new IT products - [K_U13]

8. is able to develop an object-oriented model of a simple software system (e.g., in UML notation) - [K_U17]

9. is able to assess software architecture from the perspective of non-functional requirements - [K_U18]

10. is able to effectively participate in software inspections - [K_U19]

11. is able to systematically execute functional tests - [K_U20]

12. is able to evaluate usefulness of methods and tools (also to identify their limitations) used to solve engineering tasks, i.e., building IT systems or their components - [K_U24]

13. is able to choose appropriate programming language and use it to solve a particular task - [K_U26]

14. 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) - [K_U27]

Social competencies:

1. understands that knowledge and skills related to computer science quickly become obsolete - [K_K1]

2. 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 - [K_K4]

3. is able to correctly assign priorities to own tasks and tasks performed by others - [K_K6]

Assessment methods of study outcomes

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
- 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

Course description

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.

Teaching methods:

1. Lectures: multimedia presentation, presentation illustrated with examples presented on black board, solving tasks, multimedia showcase
2. Tutorials: solving tasks, practical exercises, discussion, teamwork, multimedia showcase, workshops, case studies, tutorial

Basic bibliography:

1. L. Bass, P. Clements, R. Kazman, Software architecture in practice, WNT
2. P. Kruchten, The Rational Unified Process-An Introduction, Addison-Wesley
3. P. Kruchten, Rational Unified Process Made Easy-A Practitioner?s Guide to the RUP, Addison-Wesley
4. J. Warmer, A. Kleppe: OCL. Precise modeling with UML, Addison-Wesley
5. R. V. Binder: Testing Object-Oriented Systems: Models, Patterns and Tools, Addison-Wesley

Additional bibliography:

1. D. Spinellis and G. Gousios, Beautiful Architecture, O?Reilly Media
2. B. Meyer: Object-oriented Software Construction, Prentice Hall

Result of average student's workload

Activity	Time (working hours)
1. participating in laboratory classes / tutorials: 15 x 2 hours, 2. consulting issues related to the subject of the course; especially related to t laboratory classes and projects, 3. implementing, running and verifying software application(s) (in addition to laboratory classes) 4. participating in lectures 5. studying literature / learning aids (10 pages = 1 hour), 150 pages 6. discussing the results of the examination 7. preparing to and participating in exams: 15 hours + 2 hours	30 7 20 30 15 1 17

Student's workload

Source of workload	hours	ECTS
Total workload	120	5
Contact hours	70	3
Practical activities	57	2