COURSE DESCRIPTION CARD - SYLLABUS

Course name
Software Architecture and Verification [S2Inf1E-IO>SAV]

Course

Field of study
Computing

Year/Semester
1/2

Area of study (specialization)
Software Engineering

Profile of study
general academic

Level of study
second-cycle

Course offered in
English

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
30

Laboratory classes
30

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

5,00

Coordinators

dr hab. inż. Bartosz Walter prof. PP
bartosz.walter@put.poznan.pl

mgr inż. Michał Maćkowiak
michal.mackowiak@put.poznan.pl

Lecturers

Prerequisites

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. 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.Student should understand the need to extend his/her competences and has the willingness to work in a team.

Course objective

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

Course-related learning outcomes

Knowledge:
- has organized and well-formed theoretical general knowledge regarding software architecture, software testing and software modeling
- 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
- has advanced and detailed knowldege regarding software life cycle which involves designing software architecture, modeling, and creating unit tests

Skills:
- 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.
- 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
- is able to assess usefulness and possibility of employing new developments (methods and tools) and new it products
- is able to design and evaluate software architecture of complex software systems, using appropriate methods
- is able to design (according to a provided specification which includes also non-technical aspects) a complex device, an it system and is able implement it (at least partially) using appropriate methods, techniques, and tools (including adjustment of available tools or developing new ones
is able to work in a group, performing different roles, like architect, developer, tester

Social competences:
- student understands that knowledge and skills related to computer science quickly become obsolete
- 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

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Learning outcomes presented above are verified as follows:
1. Formative assessment:
a) lectures: based on the answers to the questions which test understanding of material presented o n the lectures;
b) laboratory classes: based on the assessment of the tasks done during classes and as a homework.
2. Summative assesment:
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:
- 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.

Programme content

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. Overview of testing methods at different levels. Role and structure of tests in a software project.

Course topics

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. Overview of testing methods at different levels. Role and structure of tests in a software project.

Teaching methods

Lectures: multimedia presentation with the examples presented on a whiteboard.
Laboratory classes: multimedia presentation, exmaples presented on a whiteboard, tasks given by the tutor to the students.

Bibliography

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

Breakdown of average student's workload

	Hours	ECTS
Total workload	125	5,00
Classes requiring direct contact with the teacher	60	2,50
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	65	2,50