POZNAN UNIVERSITY OF TECHNOLOGY

EUROPEAN CREDIT TRANSFER AND ACCUMULATION SYSTEM (ECTS)

pl. M. Skłodowskiej-Curie 5, 60-965 Poznań

COURSE DESCRIPTION CARD - SYLLABUS

Course name

Software development process organization

Course

Field of study	Year/Semester
Computer Science	1/1
Area of study (specialization)	Profile of study
Intelligent Information Technologies	general academic
Level of study	Course offered in
Second-cycle studies	polish
Form of study	Requirements
full-time	compulsory

Number of hours

Lecture	Laboratory classes	Other (e.g. online)
15	15	
Tutorials	Projects/seminars	

Number of credit points

2

Lecturers

Responsible for the course/lecturer:

dr inż. Marcin Szeląg

e-mail: marcin.szelag@cs.put.poznan.pl

tel. 61 665 3023

Instytut Informatyki

ul. Piotrowo 2, 60-965 Poznań

Responsible for the course/lecturer:

Prerequisites

Learning outcomes of the 1st level studies as defined in the resolution 42 of the senate of the Poznań University of Technology from April 24, 2017. Especially effects K1st_W1-8, verified during enrolment for the 2nd level studies - these effects are presented on the webpage www.cat.put.poznan.pl.

Course objective

1. Dissemination of knowledge concerning: version control systems, agile software cost assesment methods, automation of software build, continuous integration, and software deployment.

2. Development of skills of using proper techniques and tools during software development process

3. Formation of competence concerning conscious inclusion in developed software of external modules and libraries, respecting their licensing constraints.

Course-related learning outcomes

Knowledge

1. student has advaced and deepened knowledge concerning chosen tools for continuous integration, build, and version control computer systems - [K2st_W1]

2. student has advanced detailed knowledge concerning software development workflows employing Git versioning system - [K2st_W3]

3. student has advanced detailed knowledge concerning continuous integration and continuous delivery/deployment of software - [K2st_W3]

4. student has advanced detailed knowledge concerning automation of software build using Gradle [K2st_W3]

5. student has advanced and detailed knowledge concerning preparation of software release using Git version cotrol system - [K2st_W5]

6. student has advanced and detailed knowledge concerning software deployment using lightweight software containers of the Docker platform - [K2st_W5]

Skills

1. student can effectively use Git version control system to manage software configuration - [K2st_U2]

2. student can build application using communicating microservices, realized using Docker software containers - [K2st_U5]

3. student can assess usefulness and fitness of new tools to continuous integration and automated build of software - [K2st_U6]

4. student can properly use Delphi effort estimation method in the software development process - [K2st_U7]

5. student can assess usefulness of tools used to build, continuously integrate, and deploy software - [K2st_U9]

6. student can automate the process of software build and preparation of its distributable version using Gradle system - [K2st_U11]

Social competences

1. student understands that knowledge and skills concerning software build and deployment age quickly - [K2st_K1]

2. student understands the necessity of tracking trends concerning software versioning, build, continuous integration, and continuous delivery/deployment - [K2st_K2]

3. student is aware of the rules concerning licenses of external modules and libraries employed in developed software - [K2st_K4]

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

Fomative assessment:

a) concerning lectures:

- based on answers to questions concerning previous lectures,

b) concening laboratories:

- based on current progress in realization of tasks.

Summary assessment:

a) concerning lectures:

- verification of knowledge and skills by written test during the last lecture, composed of closed questions (multiple choice), governing entire material; to pass (3.0 evaluation) a student has to obtain at least 6 out of 12 points,

- discussion of test results,

b) concening laboratories:

- continuous evaluation during classes - verifcation of sufficient realization degree of assigned tasks or note concerning insufficient realization due to lack of commitment

- evaluation of an extra task verifying acquired practical skills, realized partially in lab and partially at home

- verification of knowledge and skills using written test during the last laboratory, composed of closed questions (multiple choice), governing entire material; to pass (3.0 evaluation) a student has to obtain at least 18 out of 35 points.

Optionally, it is possible to gain additional points for activity during classes, and especially for:

- efficient application of acquired knowledge to solve given task,

- remarks concerning improvement of learning material.

Programme content

Lectures cover the following subjects:

Agile effort estmation methods in software development process.

Management of software versions - Git system, chosen Git workflows.

Automation of software build - Gradle system.

Software continuous integration and continuous delivery/deployment.

Building applications using communicating microservices .

Application deployment using software containers - Docker system.

Laboratories cover the following subjects:

Distributed versioning system Git - basic commands, advanced topics, Gitflow workflow.

Gradle system - analysis and extension of exemplary building scripts, dependency management, writing of own tasks and plugins.

Continuous software integration - configuration of Jenkins server.

Development of applications using Docker software containers - building images, running containers, docker-compose tool, docker stack, application scaling, orchestration.

Teaching methods

1. lectures: multimedia presentation, demonstration, discussion,

2. laboratories: verbal introduction, electronic notes, tutorials, open practical tasks.

Bibliography

Basic
1. Pro Git, 2nd edition, Scott Chacon, Ben Straub, 2014 (https://git-scm.com/book/en/v2).

2. Agile Estimating and Planning, Mike Cohn, Pearson, 2005.

3. Gradle User Manual (https://docs.gradle.org/current/userguide/userguide.html).

4. Jenkins User Documentation (https://jenkins.io/doc/).

5. Docker: Up & Running, 2nd Edition, Sean P. Kane, Karl Matthias, O'Reilly Media, 2018.

Additional
1. Continuous Integration, M. Fowler, 2006, http://www.martinfowler.com/articles/continuousIntegration.html.

2. Building and Testing with Gradle, T. Berglund, M. McCullough, O'Reilly Media, 2011.

3. Docker Documentation (https://docs.docker.com/).

Breakdown of average student's workload

	Hours	ECTS
Total workload	50	2,0
Classes requiring direct contact with the teacher	32	1,3
Student's own work (preparation for laboratory classes, homework (mini-project), preparation for tests, literature study) [1]	18	0,7

[1] delete or add other activities as appropriate