COURSE DESCRIPTION CARD - SYLLABUS

Course name
Programming engineering tools [S1MiKC1E>NIP]

Course

Field of study
Microelectronics and Digital Communication

Year/Semester
1/2

Area of study (specialization)
–

Profile of study
general academic

Level of study
first-cycle

Course offered in
English

Form of study
full-time

Requirements
compulsory

Number of hours

Lecture
0

Laboratory classes
20

Other
0

Tutorials
0

Projects/seminars
0

Number of credit points

2,00

Coordinators

dr inż. Łukasz Kułacz
lukasz.kulacz@put.poznan.pl

Lecturers

Prerequisites

The student should have a basic knowledge of programming, especially the concepts of variables, classes and functions. He or she should have the ability to implement simple programs and recognize the risks associated with creating underdeveloped software.

Course objective

The purpose of the course is to provide students with basic knowledge of the tools used during software development. These tools are primarily concerned with storing and version control of code in local and remote repositories, collaboration on the same code, quality assurance techniques through tests of varying levels of detail and complexity, the basics of continuous integration and continuous deployment methods, and extensions to code editors that facilitate the creation of good quality software. The course also covers the basics of using a profiler and a code debugger.

Course-related learning outcomes

Knowledge:
The student has basic theoretical and practical knowledge of software repositories, software testing, automation of the testing process and principles of creating correct and readable code. The student is

familiar with basic tools to facilitate software development.

Skills:
The student is able to effectively use a local software repository for his/her work, as well as sync efef his/her work through a remote repository. The student is able to collaborate with other software developers through remote repositories and link their software to them. The student is able to prepare basic tests for code and test their code both locally and through automation linked to a remote software repository. The student is also able to use basic tools to facilitate the work of software development and thus improve the quality of the code being developed. The student is able to use a profiler to evaluate code performance and a debugger to analyze code performance.

Social competences:
The student is aware of the possibilities and limitations during software development, especially the need to ensure good code quality. Understands the potential impact of incorrectly prepared software. Is aware of the challenges and risks associated with multiple programmers working together on the same code.

Methods for verifying learning outcomes and assessment criteria

Learning outcomes presented above are verified as follows:

In terms of the laboratory, verification of the established learning outcomes is carried out by: substantive evaluation of individual or group tasks performed in class or in the form of assignments to be completed after class, activity in class. Tasks in the form of commands to be carried out by implementing the functionality specified in the task, for which there is an assigned number of points to be obtained. The adopted grading scale:
2.0 for <0%; 50%>
3.0 for (50%; 60%>
3.5 for (60%; 70%>
4.0 for (70%; 80%>
4.5 for (80%; 90%>
5.0 for (90%; 100%>

Programme content

As part of the course, the student will become familiar with the various tools available to support code development and conduct ICT projects in accordance with best practices. These tools are intended to both enable multi-platform and multi-team work. In particular, the student will learn about version control tools and selected repositories, automated testing systems, the concept of microservices and CI/CD, code quality testing and code coverage with tests.

Course topics

Labs:
1. The basics of the git system. (1 class unit)
2. Collaboration using the git system. (2 class units)
3. Preparation of unit and integration tests. (2 class units)
4. Preparation of functional and performance tests. (2 class units)
5. Code quality control and code coverage with tests. (1 class unit)
6. Tools to facilitate work during software development. (2 class units)

Teaching methods

Laboratories will be conducted in the form of practical classes at the computer, during which students will have the opportunity to independently implement solutions in accordance with the provided instructions. Some of the tasks are scheduled for implementation in groups of several people. During the implementation, consultations with the tutor are also foreseen, which will allow students to get guidance and discuss more difficult implementation issues. Outside of class, students have the opportunity to take advantage of additional consultations with the tutor.

Bibliography

Basic:
1. Jakość oprogramowania. Podręcznik dla profesjonalistów. Michał Sobczak
2. TDD w praktyce. Niezawodny kod w języku Python. Harry Percival
3. Git i GitHub. Kontrola wersji, zarządzanie projektami i zasady pracy zespołowej. Mariot Tsitoara

Additional:
1. Testowanie i jakość oprogramowania. Modele, techniki, narzędzia. Adam Roman

Breakdown of average student's workload

	Hours	ECTS
Total workload	60	2,00
Classes requiring direct contact with the teacher	20	0,50
Student's own work (literature studies, preparation for laboratory classes/ tutorials, preparation for tests/exam, project preparation)	40	1,50