SOFTWARE DEVELOPMENT TOOLS

Faculty of Computer Science				
Study programme:	**Computer Science**		Degree level:	**Engineer's degree full-time programme**
Specialization	**---**		Diploma path:	**2026/2027W - 2026/2027S**
Module name:	**Software Development Tools** (Narzędzia procesu tworzenia oprogramowania)			
Module type:	**obligatory**	**Semester: 2**	ECTS:**2**	Module ID:**FCS-00071**
No. of hrs in semester:	Lecture (L) - **15** Classes(C) - **0** Specialization workshop (SW) - **15** Project (P) - **0** Laboratory classes (LC) - **0** Seminar (S) - **0**			
Prerequisites	Programming Basics (FCS-00031),			

Aims and objectives:	The purpose of the lecture is to present engineering methods and tools that support the software development process. These methods and tools consider different stages of software development from requirements gathering, through implementation and testing. The purpose of the specialization workshop is to practically present tools that support the process of software development from the moment of defining requirements, project implementation and deployment.		
Forms of teching activities::	lecture, specialization workshop,	Assessment:	Evaluation must be relevant to the intended learning outcomes:
			Lectures: written examination, Practical classes: two written tests.
Module content:	Lecture: Joel's test, integrated development environments (IDEs), code and changes management (software versioning and revision control systems), dynamic (run-time) software testing, software profiling/software performance testing, source code documentation, functional tests, bug management, requirement management, GUI prototyping tools, software distribution - instalators Specialization workshop: integrated development environments (IDEs) (e.g., MS VS, Eclipse, NetBeans, Jupyter), revision control systems (SVN, GIT), debugging, run-time software testing in unmanaged and managed code, time and memory profiling, source code documention, bug manangement, requirement management, GUI prototyping tools, software distribution - instalators		
Teaching methods:	programming, lecture problem,		

Learning outcomes

Symbol	Specify min. 4, max. 8 learning outcomes in the following order: knowledge – skills – competence. Each learning outcome must be verifiable	Reference to the programme learning outcomes of education
LO1	Knows the selected tools using to specify, design, develop and testing software applications.	INF1_W06 INF1_W09 INF1_W12
LO2	Knows and understands the role of software engineering tools. Knows the categories of tools supporting software application design. Has basic knowledge about the current tools supporting software delopment.	INF1_W11
LO3	Is able to choose and use properly integrated development tools, revision control systems and tools for run time testing of software applications. Is able to compare this kind of software development tools in accordance with their usability and enonomic issues (speed, cost, functionality).	INF1_U06 INF1_U13
LO4	Is able to choose and use properly tools for software profiling, documentation generation, use case testing, bug management and distribution. Is able to compare this kind of tools in accordance with their usability and enonomic issues (speed, cost, functionality).	INF1_U07 INF1_U11 INF1_U13

No. of learning outcome	Methods of assessing the learning outcome	Type of teaching activities (if more than one) during which the outcome is assessed
LO1	written examination	L
LO2	written examination	L
LO3	first written test, work during classes	Sw
LO4	second written test, work during classes	Sw

Student's workload (in hours)			
	1 - Attendance at lectures		15
	2 - Attendance at classes		15
	3 - Report preparation and homeworks	None	8
	4 - Participation in student-teacher sessions	None	4
	5 - Preparation for the exam	None	4
	6 - Preparation for classes	None	4
		TOTAL:	**50**

Quantitative indicators			
	Student's workload - activities that require direct teacher participation: (4)+(2)+(1)+(5)	38	**ECTS** 1.5
	Student's workload connected with practical classes (6)+(2)+(3)	27	1.1

Basic references:	
	1. B. Collins-Sussman, B.W. Fitzpatrick, C.M. Pilato, Version Control with Subversion, http://svnbook.red-bean.com/en/1.7/svn-book.pdf 2. J. Spolsky, The Best Software Writing, Apress, 2005 3. Microsoft, Patterns & Practices: Performance Testing Guidance for Web Applications, 2007, https://perftestingguide.codeplex.com

	4. I. Sommerville, Software engineering, Pearson Education, Boston, 2004. 5. P. Glavich, C. Farrell, .NET Performance Testing and Optimization The Complete Guide, Simple Talk Publishing, 2010
Further reading	1. H. van Vliet, Software engineering :principles and practice, John Wiley and Sons, 2008. 2. D. Spinellis, Code Reading: The Open Source Perspective, Addison-Wesley Professional, 2003 3. M. Fowler, K. Beck, D. Roberts, E. Gamma, Refactoring, Improving the Design of Existing Code, Addison-Wesley Professional, 1999 4. G. J. Myers, C. Sandler, T. Badgett, The Art of Software Testing Hardcover, Wiley, 2011

Unit:	Software Department	Lecturer/ instructor	
Date of issuing the programme:	17th Feb. 2022	Author of the programme:	dr inż. Krzysztof Jurczuk

L - lecture, C - classes, LC - laboratory classes, P-project, SW - specialization workshop, S - seminar