National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

Department of Computer Engineering

Agile Programming Techniques

The program of the academic discipline (Syllabus)

Details of the academic discipline	
Cycle of Higher Education	***First cycle of higher education (Bachelor's degree)***
Field of Study	*12 Information technologies*
Specialty	*121 Software Engineering*
Education Program	*Computer Systems Software Engineering*
Type of Course	*Normative*
Mode of Studies	*Full-time education*
Year of studies, semester	*3 year (6 semester)*
ECTS workload	*4 credits*
Testing and assessment	*Exam*
Course Schedule	*Lectures 18 (36 hours), Laboratory 9 (18 hours)*
Language of Instruction	*English*
Course Instructors	Lecturer / Laboratory: Senior lecturer of the Department of IST, PhD, Serhii Orlenko, orlenko_sergey@ukr.net
Access to the course	In the Telegram group of disciplines and in Campus

Program of academic discipline

1 Course description, goals and objectives, and learning outcomes

The educational discipline "Agile Programming Techniques" provides students with thorough training in theoretical, methodological, and practical foundations in software development methodologies, teamwork, requirements analysis, design, development, and testing of information technologies for solving applied and scientific tasks in the field of information systems and technologies

The purpose of studying the discipline "Agile Programming Techniques" is aimed at forming future engineers with a modern level of information and digital culture, mastering the basic principles of creating software products; mastery of algorithmic thinking; acquiring of practical skills in the independent compilation of professional software and use of modern information technologies to solve various problems of an applied nature taking considering of the requirements for its quality, reliability, production characteristics. The formation of learning goals and students' understanding of various aspects of the future profession is a necessary component of training a qualified software engineer (Software Engineer), system architect (System Architect), and software architect (Software Architect).

The subject of study of the discipline is modern methods, tools, and technologies of software development used in teamwork, requirements analysis, design, implementation, testing, implementation, and operation of information systems and technologies, information processing systems based on modern processing technologies.

According to the requirements of the EP, the discipline "Agile Programming Techniques" should ensure that applicants acquire competencies and program learning outcomes: PC11, PLO22.

After mastering the module "Agile Programming Techniques" applicants must demonstrate the following competencies and program learning outcomes:

- Ability to implement Phases and Iterations of the Life Cycle of Software Systems and information technologies based on appropriate Software Development Models and Approaches
- Know and be able to apply Project Management Methods and Tools
- Ability to evaluate and ensure the quality of the work performed
- Ability to develop business solutions and evaluate new technology offerings
- Ability to apply standards in the field of information systems and technologies when developing functional profiles, building and integrating systems, products, services and elements of the organization's infrastructure

According to the results of studying the educational discipline "Agile Programming Techniques", the following **knowledge** should be obtained:

- basic concepts of software engineering;
- approaches to managing the software development process;
- principles of architectural and object-oriented software design;
- main types of tools for software development;
- principles and models of software development, programming methodology;
- software development requirements management tools;
- basic methods of software quality assurance and testing.

Skills that should be acquired as part of studying the academic discipline "Agile Programming Techniques":

- formulate requirements for the software product;
- solve problems using decomposition;
- create diagrams of various types;
- develop the structure of the software project;
- design and implement a convenient user interface;
- draw up documentation for the software project;
- work with several versions of the software project;
- perform various types of software testing;
- determine the technical and economic indicators of the software product;
- organize and support teamwork.

Such a combination of general and special competences, theoretical and practical knowledge, skills and abilities helps to increase the professional level of bachelor's degree holders in order to carry out effective activities in the field of development of software engineering.

2 Pre-requisites and post-requisites of the discipline (place in the structural and logical scheme of training according to the relevant educational program)

Necessary disciplines: "Programming Fundamentals", "Software Engineering Components", “Group Dynamics & Communications”

Module "Agile Programming Techniques" is necessary for studying the disciplines "Risk management and project quality", “Complex Systems Design”

3 Structure of the credit module

A list of the main topics included in the study program of the discipline "Agile Programming Techniques":

Section 1. Team work

Topic 1.1 Types and technologies of communication

Topic 1.2 Work in a team

Section 2. Software development methodologies (software)

Topic 2.1 Types of software development methodologies

Topic 2.2 Flexible software development methodologies

Topic 2.3 Comparative characteristics of traditional and flexible development methodologies

Section 3. Life cycle of software.

Topic 3.1. Software engineering. Programming technologies in a historical aspect.

Topic 3.2. Software life cycle. Life cycle models.

Topic 3.3. Software development methodology. Flexible application development. Principles of Agile development. Scrum, RAD. XP programming.

Topic 1.4. Software requirements management.

Section 4. Software Requirements Engineering

Topic 4.1 Basic requirements engineering processes

Topic 4.2 Definition and characteristics of types of software requirements. Levels of software requirements

Topic 4.3 Identification and formation of software requirements

Topic 4.4 Documentation of requirements. Methods of writing quality requirements. Documentation standards

Topic 4.5 Analysis and coordination of requirements. Inspection, certification, completeness, identification of conflicts and inconsistencies in requirements. Basics of risk management when creating software

Topic 4.6 Requirements management. Requirements tracing and instrumental support of the requirements management process

Topic 4.7 Integration of requirements analysis and software development processes

Section 5. Software architecture development.

Topic 5.1. Software architecture design.

Topic 5.2. Models of system structuring.

Topic 5.3. Management simulation and decomposition on the module.

Topic 5.4. User interface design.

Section 6. Fundamentals of software design methodology

Topic 6.1 Software design methodologies and technologies

Topic 6.2 Structural approach to software design

Topic 6.3 Object-oriented approach to software design

Section 7. Software modeling.

Topic 7.1. A structural approach to modeling. SADT methodology.

Topic 7.2. Modeling data flows.

Topic 7.3. Modeling of data structures. Diagram of state transitions.

Topic 7.4. Basics of the UML language. Class diagrams.

Section 8. Management of software projects.

Topic 8.1. Tasks of project management.

Topic 8.2. Project concepts. Software product risk management.

Topic 8.3. Planning of software projects. SMART. WBS. PERT. CMP. Gant Chart.

Topic 8.4. Formation of a team of developers. Distribution of roles and responsibilities.

Section 9. Software quality assurance and control.

Topic 9.1 Definition of basic concepts. Concept of testing

Topic 9.2. Metrics and software quality.

Topic 9.3 PP development technology through testing. TDD technology

Topic 9.4. Software verification and testing.

4 Educational resources and materials

Basic:

1. P. Laplante, "Remember the human element in IT project management," in IT Professional, vol. 5, no. 1, pp. 46-50, Jan. 2003, doi: 10.1109/MITP.2003.1176490.
2. Pressman, Roger (2010) *Software Engineering: A Practitioner's Approach*, McGraw Hill, New York, NY.
3. Carmichael A., Haywood D. (2002) *Better Software Faster*, Prentice Hall.
4. Sommerville, Ian (2011) *Software Engineering*, Addison-Wesley, Boston, MA.
5. Stephens, Rod (2015) *Beginning Software Engineering*, Wrox.
6. Tsui, Frank , Orlando Karam and Barbara Bernal (2013) *Essentials of Software Engineering*, Jones & Bartlett Learning , Sudbury, MA.
7. Pfleeger, Shari (2001) *Sofwtare Engineering: Theory and Practice*, Prentice Hall, Upper Saddle River, NJ.

Supplementary:

1 Cohn Mike.(2005) *Agile Estimating and Planning*.: Pearson; 1st edition.. – 360 p.

2 Larman, C. (2005) *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and iterative Development*, Pearson

3 Ambler, S. (2002) *AgileModeling: Effective Practices for Extreme Programming and the Unified Process*, NewYork, John Wiley&Sons.

4 Bass, D.L., Clements, D.P. andKazman, D.R. (2012) *Software Architecture in Practice*, 3rd edn, Upper Saddle River, NJ, Addison Wesley

5 Beck, K. (2004) *Extreme Programming Explained: Embrace Change*, Upper Saddle River, NJ, Addison Wesley

6 Clemens Szyperski (2002) Component Software: Beyond object-oriented programming, Addison-Wesley

7 John Cheesman & John Daniels (2000) UML Components: A simple process for specifying component-based software (The component software series) Addison-Wesley

8 Rob Pooley, Perdita Stevens (2006) *Using UML Software Engineering with Objects and Components*, second edition. Addison-Wesley

9 Christopher Fox (2006) *Introduction to Software Engineering Design*. Addison Wesley

Educational content

5 Methodology

Sections and topics	Hours			
	Total	including		
		Lectures	Practical work	Self-study
Section 1. Team work *Topic 1.1 Types and technologies of communication* *Topic 1.2 Work in a team*	4	2		10
Section 2. Software development methodologies (software) *Topic 2.1 Types of software development methodologies* *Topic 2.2 Flexible software development methodologies* *Topic 2.3 Comparative characteristics of traditional and flexible development methodologies*	10	2	2	6
Section 3. Life cycle of software. *Topic 3.1. Software engineering. Programming technologies in a historical aspect.* *Topic 3.2. Software life cycle. Life cycle models.* *Topic 3.3. Software development methodology. Flexible application development. Principles of Agile development. Scrum, RAD. XP programming.* *Topic 1.4. Software requirements management.*	16	4	4	8
Section 4. Software Requirements Engineering *Topic 4.1 Basic requirements engineering processes* *Topic 4.2 Definition and characteristics of types of software requirements. Levels of software requirements* *Topic 4.3 Identification and formation of software requirements* *Topic 4.4 Documentation of requirements. Methods of writing quality requirements. Documentation standards* *Topic 4.5 Analysis and coordination of requirements. Inspection, certification, completeness, identification of conflicts and inconsistencies in requirements. Basics of risk management when creating software* *Topic 4.6 Requirements management. Requirements tracing and instrumental support of the requirements management process* *Topic 4.7 Integration of requirements analysis and software development processes*	22	6	4	12
Section 5. Software architecture development. *Topic 5.1. Software architecture design.* *Topic 5.2. Models of system structuring.* *Topic 5.3. Management simulation and decomposition on the module.* *Topic 5.4. User interface design.*	18	4	4	10

Section 6 Fundamentals of software design methodology *Topic 6.1 Software design methodologies and technologies* *Topic 6.2 Structural approach to software design*	10	4		6
Section 7. Software modeling. *Topic 7.1. A structural approach to modeling. SADT methodology.* *Topic 7.2. Modeling data flows.* *Topic 7.3. Modeling of data structures. Diagram of state transitions.* *Topic 7.4. Basics of the UML language. Class diagrams.*	12	4		8
Section 8. Management of software projects. *Topic 8.1. Tasks of project management.* *Topic 8.2. Project concepts. Software product risk management.* *Topic 8.3. Planning of software projects. SMART. WBS. PERT. CMP. Gant Chart.* *Topic 8.4. Formation of a team of developers. Distribution of roles and responsibilities.*	14	6		8
Section 9. Software quality assurance and control. *Topic 9.1 Definition of basic concepts. Concept of testing* *Topic 9.2. Metrics and software quality.* *Topic 9.3 PP development technology through testing. TDD technology* *Topic 9.4. Software verification and testing.*	14	4	4	6
Total hours in semester	120	36	18	66

Laboratory works:

The purpose of conducting laboratory classes is for students to consolidate theoretical knowledge and acquire the necessary practical skills for working with modern technologies for software engineering.

- Laboratory work #1: Software development methodologies;
- Laboratory work #2: Life cycle of software;
- Laboratory work #3: Software Requirements Engineering;
- Laboratory work #4: Software architecture development;
- Laboratory work # 5: Software quality assurance and control.

6 Self-study

- preparation for lectures by studying the previous lecture material;
- preparation for laboratory work with the study of the theory of laboratory work with an oral answer to the given questions of the section;
- preparation of results of laboratory work in the form of a protocol.

Attendance Policy and Assessment

7 Attendance Policy

During classes in an academic discipline, students must adhere to certain disciplinary rules:
• extraneous conversations or other noise that interferes with classes are not allowed;
• the use of mobile phones and other technical means is not allowed without the teacher's permission.

Laboratory works are submitted personally with a preliminary check of theoretical knowledge, which is necessary for the performance of laboratory work. Validation of practical results includes code review and execution of test tasks.

8 Monitoring and grading policy

Current control: survey on the subject of the lesson
Calendar control: conducted twice a semester as a monitoring of the current status of meeting the syllabus requirements.
Semester control: test
Conditions for admission to semester control: enrollment of all laboratory work

System of rating points and evaluation criteria
The student's rating in the discipline consists of the points he receives for:
1. performance and defense of 5 laboratory works;
2. execution of 2 modular control works (MCW).

Laboratory works:
"excellent", a complete answer to the questions during the defense (at least 90% of the required information) and a properly prepared protocol for laboratory work - 10 points;
"good", a sufficiently complete answer to the questions during the defense (at least 75% of the required information) and a properly prepared protocol for laboratory work - 8 points;
"satisfactory", incomplete answer to the questions during the defense (at least 60% of the required information), minor errors and a properly prepared protocol for laboratory work - 6 points;
"unsatisfactory", an unsatisfactory answer and/or an improperly prepared protocol for laboratory work - 0 points.

Modular Control Works:
"Excellent", full answer (not less than 90% of the information you need) - 25 points;
"Good", a full answer (not less than 75% of the information you need), or a complete answer with minor mistakes - 20 points;
"Satisfactory", incomplete answer (but not less than 60% of the information you need) and minor mistakes - 16 points;
"Unsatisfactory", unsatisfactory response (incorrect problem solution), requires mandatory re - writing at the end of the semester - 0 points.

The maximum sum of weighted points of control measures during the semester is:

$$R=5*Rlab+2*Rmcw=5*10+2*25=100.$$

Table1 — Correspondence of rating points to grades on the university scale

Score	*Grade*
100-95	Exellent
94-85	Very good
84-75	Good
74-65	Satisfactory
64-60	Sufficient
below 60	Fail
Course requirements are not met	Not graded

Syllabus of the course:

designed by Senior Lecturer of the Department of IST, PhD, Serhii Petrovych Orlenko

adopted by Department of Computer Engineering (protocol № 10, 25.05.2022)

approved by the methodical commission of FICT (protocol № 10, 09.06.2022)