Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра програмного забезпечення комп’ютерних систем

AGILE SOFTWARE DEVELOPMENT

Working program of the academic discipline (Syllabus)

Course requisites

Cycle of Higher Education	***First (bachelor)***
Study area	*12 Information technologies*
Speciality	*121 Software engineering*
Educational program	*Software Engineering of Multimedia and Information Retrieval Systems*
Discipline status	*Elective*
Study form	*Daytime*
Year of study, semester	*4 year of training, 7 semester*
Discipline volume	*Lectures: 36 hours, computer workshop: 18 hours, self work: 66 hours.*
Semester control/ control measures	*Final test, modular control work, calendar control*
Course schedule	*According to the schedule for the autumn semester of the current academic year (rozklad.kpi.ua)*
Language	*Ukrainian*
Information about head of the course / teachers	*Lecturer: PhD, Associate Professor, Iana Khitsko, iana.khitsko@gmail.com* *Teacher of computer workshop: PhD, Associate Professor, Iana Khitsko, iana.khitsko@gmail.com*
Access to course	Google classroom: https://classroom.google.com/c/NDE3OTQxMzE5NjU5?cjc=ne2zis4

Outline of the Course

1. Course description, goals, objectives, and learning outcomes

The purpose *of studying the discipline "Agile Software Development" is to allow students to develop the competencies necessary for solving practical problems of professional activity related to software development according to modern agile methodologies.*

The ***goal*** *of studying the discipline "Agile Software Development" is the formation of students' ability to choose a software development methodology in accordance with the specified requirements and design and construction environment; apply various Agile practices in software design and construction; conduct an analysis of the effectiveness of the application of practices and sub-processes from the Agile family.*

The ***subject*** *of the discipline "Agile Software Development" is the algorithmic support of the processes of software analysis, design, monitoring and construction.*

*The discipline "Agile Software Development" strengthens the following **professional competencies** of the educational and professional program:*

PC01 *- Ability to identify, classify and formulate software requirements;*
PC02 *- Ability to participate in software design, including its structure, behavior and functioning processes modeling (formal description);*
PC03 *- Ability to develop software systems architectures, modules and components;*
PC04 *- Ability to formulate and ensure software quality requirements in accordance with customer requirements, specifications and standards;*
PC05 *- Ability to follow specifications, standards, rules and recommendations in the professional field during the life cycle processes implementation;*
PC08 *- Ability to apply fundamental and interdisciplinary knowledge to successfully solve software engineering problems;*
PC10 *- Ability to accumulate, process and systematize professional knowledge about software creation and maintenance, and determination of the importance of lifelong learning;*
PC11 *- Ability to implement phases and iterations of the life cycle of the software systems and information technology based on appropriate models and approaches to software development;*
PC12 *- Ability to carry out the system integration process, apply change management standards and procedures to maintain software integrity, overall functionality and reliability.*

*The discipline «Agile Software Development» contributes to the formation of **program learning outcomes**:*

PLO03 *- To know the software life cycle basic processes, phases and iterations;*
PLO04 *- To know and apply professional standards and other regulatory documents in the field of software engineering;*
PLO05 *- To know and apply relevant mathematical concepts, domain methods, system and object-oriented analysis and mathematical modeling for software development;*
PLO06 *- Ability to select and use the appropriate task of software development methodology;*
PLO09 *- To be able to use collecting, formulating and analyzing software requirements methods and tools;*
PLO10 *- To conduct a pre-project survey of the subject area, system analysis of the design object;*
PLO12 *- To apply effective approaches to software design in practice;*
PLO20 *- To know approaches to evaluation and quality assurance of software;*
PLO23 *- To be able to document and present the software development results.*

2. Discipline prerequisites and postrequisites (place in the structural and logical education scheme according to the relevant educational program)

The successful study of the discipline «Agile Software Development» preceded by the study of the disciplines "Software engineering components. Part 4. Software quality and testing", "Software engineering components. Part 2. Software modeling. Analysis of software requirements", "Software engineering components. Part 3. Software architecture" of the curriculum for bachelor's training in the specialty 121 Software engineering.

The theoretical knowledge and practical skills acquired during the mastering of the discipline "Agile Software Development" ensure the successful implementation of course and diploma projects in the specialty 121 Software Engineering.

3. Content of the course

The discipline «Agile Software Development» involves the study of such topics:

Topic 1. Software Development lifecycle

Topic 2. Agile methodologies

Topic 3. Agile best practices

Modular test

Test

4. Coursebooks and teaching resources

Basis reference:

1. *Electronic campus of NTUU "KPI by Igor Sikorsky". Discipline materials for "Agile Software Development". – http://login.kpi.ua*
2. *Web-portal of Applied Mathematics Faculty. Materials archive. «Хіцко» folder. – Режим доступу : http://fpm.kpi.ua/archive/dir.do?sys_id=obj_2*

Additional reference:

3. *Kent Beck. "Manifesto for Agile Software Development" [електронний ресурс] / Kent Beck, James Grenning, Robert C. Martin, Mike Beedle, Jim Highsmith, Steve Mellor, Arie van Bennekum, Andrew Hunt, Ken Schwaber, Alistair Cockburn, Ron Jeffries, Jeff Sutherland, Ward Cunningham, Jon Kern, Dave Thomas, Martin Fowler, Brian Marick // Agile Alliance, 2001. – режим доступу - http://agilemanifesto.org/ .*
4. *Steve McConnell (1996). Rapid Development: Taming Wild Software Schedules, Microsoft Press Books, ISBN 978-1-55615-900-8*
5. *DSDM Consortium. DSDM Atern: the Handbook [електронний ресурс] / DSDM Consortium, 2008. – режим доступу - https://www.agilebusiness.org/resources/dsdm-handbooks/dsdm-atern-handbook-2008 .*
6. *Schwaber, Ken (2004). "SCRUM Development Process"(PDF). Advanced Development Methods - режим доступу - http://www.jeffsutherland.org/oopsla/schwapub.pdf*
7. *"Extreme Programming Rules"*. extremeprogramming.org.
8. *Palmer, S.R., & Felsing, J.M. (2002). A Practical Guide to Feature-Driven Development. Prentice Hall. (ISBN 0-13-067615-2)*
9. *Cockburn, Alistair. Crystal Clear, A Human-Powered Methodology for Small Teams [Text] / Alistair Cockburn // Addison-Wesley Professional, 2004. – pp.336. - ISBN 0-201-69947-8.*
10. *Kanban: Successful Evolutionary Change for Your Technology Business, David J. Anderson. (United States, Blue Hole Press, 2010. ISBN 978-0984521401*
11. *Scrumban: Essays on Kanban Systems for Lean Software Development, Corey Ladas. (United States, Modus Cooperandi Press, 2009. ISBN 9780578002149*
12. *Mary Poppendieck; Tom Poppendieck (2003). Lean Software Development: An Agile Toolkit. Addison-Wesley Professional. ISBN 978-0-321-15078-3.*
13. *Cohn, Mike. "Planning Poker Cards: Effective Agile Planning and Estimation". Mountain Goat Software, 30 March 2016. - режим доступу - https://www.mountaingoatsoftware.com/tools/planning-poker*
14. *Eric Evans, 2015 Domain Driven Design, Definitions and Pattern Summaries. Режим доступу - https://domainlanguage.com/wp-content/uploads/2016/05/DDD_Reference_2015-03.pdf*
15. *Leybourn, E. (2013). Directing the Agile Organisation: A Lean Approach to Business Management. London: IT Governance Publishing: 71–79.*

Educational content

5. Methods of mastering the discipline (educational component)

№	*Training session type*	*Lesson description*

Topic 1. Software development life cycle		
1	Lecture 1. Software development life cycle	Phases of the project according to the PMI methodology and software development process. Life cycle models. Waterfall model. Prototyping model. The big bang model. V-shaped. Iteration. Incremental and iterative models. Spiral model.
Topic 2. Методології Agile		
3	Lecture 2. Agile priciples	Basic software development methodologies. Basic principles of Agile.
4	Lecture 3. RAD & DSDM	Application of RAD methodology. The main phases of DSDM and its application.
	Lecture 4. SCRUM (Part 1)	Scrum: main principles and development cycle.
5	Lecture 5. SCRUM (Part 2)	Product backlog, user stories and meetings in SCRUM.
6	Computer Workshop 1. SCRUM	Objective: plan a project and develop its three iterations using SCRUM.
7	Lecture 6. Extreme programming	Main XP principles and lifecycle.
8	Lecture 7. FDD	Feature driven development basic concepts and practices.
9	Lecture 8. Crystal methodologies and Lean	The main aspects of Crystal methodologies, their varieties and differences. Basic concepts and principles of Lean.
10	Lecture 9. Kanban and Scrumban	Kanban and Scrumban methodologies, main differences and conditions of application.
Topic 3. Agile best practices		
11	Lecture 10. Sprint and product backlog	Iterative and incremental development. Sprint and product backlog.
12	Computer Workshop 2. TRELLO	Objective: register in the TRELLO system, create multiple iterations and tasks for each iteration, make changes to the task and change the status of the task.
13	Lecture 11. User stories and story points	User stories and story points, estimations.
14	Computer Workshop 3. User stories and their estimation	Objective: develop user stories and evaluate them using planning poker.
15	Lecture 12. Agile planning and velocity	Planning Poker and Velocity in Agile Development.

16	*Lecture 13. Domain driven modelling*	*Domain driven modelling, its phases and features*
17	*Lecture 14. Story modelling and testing in Agile*	*Story modelling and testing in Agile*
18	*Lecture 15. Agile other practices*	*Continuous integration. Cross-functional team*
Modular test		

6. Self-study

The discipline "Agile Software Development" is based on independent preparations for classroom classes on theoretical and practical topics.

№	*The name of the topic that is submitted for independent study*	*Hours of study*	*References*
1	*Preparing for lecture 1*	*2*	*3, pp. 3-28; 4, pp. 25-28*
2	*Preparing for lecture 2*	*2*	*5*
3	*Preparing for lecture 3*	*2*	*6, pp. 10-25; 6*
4	*Preparing for lecture 4*	*2*	*7, pp. 1-25*
5	*Preparing for lecture 5*	*2*	*7, pp. 1-25*
6	*Preparing for computer workshop 1*	*2*	*7, pp. 1-25*
7	*Preparing for lecture 6*	*2*	*8*
8	*Preparing for lecture 7*	*2*	*9, pp. 5-13*
9	*Preparing for lecture 8*	*2*	*10, pp. 127-132*
10	*Preparing for lecture 9*	*2*	*11, pp. 13-20*
11	*Preparing for lecture 10*	*2*	*12, pp. 67-75*
12	*Preparing for computer workshop 2*	*2*	*11, pp. 13-20; 12, pp. 67-75*
13	*Preparing for lecture 11*	*2*	*13, pp. 161-164*
14	*Preparing for computer workshop 3*	*2*	*13, pp. 161-164; 12, pp. 67-75*

15	*Preparing for lecture 12*	*2*	*14*
16	*Preparing for lecture 13*	*2*	*15*
17	*Preparing for lecture 14*	*2*	*15*
18	*Preparing for lecture 15*	*2*	*16, pp. 71-79*
19	*Preparing for modular test*	*6*	*1, pp. 3-28; 2, pp. 25-28; 7, pp. 1-25; 9, pp. 5-13; 10, pp. 127-132; 11, pp. 13-20; 12, pp. 67-75; 14-15; 16, pp. 71-79*
20	*Preparing for the test*	*6*	*1, pp. 3-28; 2, pp. 25-28; 7, pp. 1-25; 9, pp. 5-13; 10, pp. 127-132; 11, pp. 13-20; 12, pp. 67-75; 14-15; 16, pp. 71-79*
21	*Rapid Application Development*	*2*	*5, pp. 10-25*
22	*Ping-pong programming*	*2*	*8*
23	*Crystal Orange application*	*2*	*10, pp. 127-132*
24	*Kanban board creation in Trello*	*4*	*11, pp. 13-20*
25	*Planning Pocker task planning method*	*4*	*14, pp. 1-8*
26	*Story-boarding*	*4*	*15, pp. 71-79*

Policy and Assessment

7. Course policy

- *Attending lectures is mandatory.*
- *Attending computer workshop classes may be occasional and as needed to protect computer workshop work.*
- *Rules of behavior in classes: activity, respect for those present, turning off phones.*
- *Adherence to the policy of academic integrity.*
- *Rules for protecting the works of the computer workshop: the works must be done according to the option of the student, which is determined by his number in the group list.*

- *The rules for assigning incentive and penalty points are as follows.*

Incentive points are awarded for a creative approach in the performance of computer workshop works (the maximum number of points for all works is 2 points).
Penalty points are calculated for:

- plagiarism (the program code does not correspond to the task variant, the identity of the program code among different works) in the works of the computer workshop: -5 points for each attempt.

8. Types of control and rating system for evaluating learning outcomes (ELO)

During the semester, students perform 3 computer practicals. The maximum number of points for each computer workshop: 16 points.

Points are awarded for:
- quality of laboratory work (computer workshop): 0-6 points;
- answer during the defense of laboratory work (computer workshop): 0-6 points;
- timely submission of work for defense: 0-4 points.

Performance evaluation criteria:
6 points – the work is done qualitatively, in full;
4-5 points – the work is done qualitatively, in full, but has shortcomings;
1-3 points – the work is completed in full, but contains minor errors;
0 points – the work is incomplete or contains significant errors.

Answer evaluation criteria:
6 points – the answer is complete, well-argued;
4-5 points – in general, the answer is correct, but has flaws or minor errors;
1-3 points – there are significant errors in the answer;
0 points - there is no answer or the answer is incorrect.

Criteria for evaluating the timeliness of work submission for defense:
4 points – the work is presented for defense no later than the specified deadline;
3 points – the work is submitted for defense within 1 week after the specified deadline;
2 points – the work is submitted for defense within 2 weeks after the specified deadline;
2 points – the work is submitted for defense within 3 weeks after the specified deadline;
0 points – the work is submitted for defense within 4 or more weeks after the specified deadline.

The maximum number of points for performing and defending computer practicals:
16 points × 3 lab. works (comp. practical) = 48 points.

*The task for the **modular test** consists of 6 questions, the answer to 4 of which is evaluated by 8 points, by 2 - 10 points.*

Evaluation criteria for more difficult test questions:
9-10 points – the answer is correct, complete, well-argued;
7-8 points – the answer is correct, detailed, but not very well argued;
5-6 points - in general, the answer is correct, but has shortcomings;
3-4 points – there are minor errors in the answer;
1-2 points – there are significant errors in the answer;
0 points - there is no answer or the answer is incorrect.

Evaluation criteria for simpler test questions:
7-8 points – the answer is correct, complete, well-argued;
5-6 points - in general, the answer is correct, but has shortcomings;
3-4 points – there are minor errors in the answer;

1-2 points – there are significant errors in the answer;
0 points - there is no answer or the answer is incorrect.

The maximum number of points for a modular control work:
10 points × 2 questions + 8 points × 4 questions = 52 points.

The rating scale for the discipline is equal to:
R = RS = 48 points + 52 points = 100 points.

Calendar control: is carried out twice a semester as a monitoring of the current state of fulfillment of the syllabus requirements.
At the first certification (8th week), the student receives "credited" if his current rating is at least 8 points (50% of the maximum number of points a student can receive before the first certification).
At the second certification (14th week), the student receives "passed" if his current rating is at least 16 points (50% of the maximum number of points a student can receive before the second certification).

Semester control: test.

Conditions for admission to semester control:

With a semester rating (Rc) of not less than 60 points and the enrollment of all computer workshop, the student receives credit "automatically" according to the table (Table of correspondence of rating points to grades on the university scale). Otherwise, he has to perform the final test.

Completion and protection of a computer workshop is a necessary condition for admission to the credit control work.

If the student does not agree with the "automatic" grade, he can try to improve his grade by writing a credit test, while his points received for the semester are kept, and the better of the two grades received by the student is assigned ("soft" grading system) .

Table of correspondence of rating points to grades on the university scale:

Points	*Grade*
100-95	Excellent
94-85	Very good
84-75	Good
74-65	Satisfactorily
64-60	Enough
< 60	Unsatisfactorily
Admission conditions are not met	Not admitted

9. Additional information about the course

The list of questions to be submitted for semester control is given in Appendix 1.

Course syllabus:

Is created by PhD, Associate Professor Iana Khitsko**.**

Adopted by Computer Systems Software Department (protocol № 8 from 25.01.23)

Approved by the Faculty Board of Methodology (protocol № 6 from 27.01.23)

Appendix 1. List of questions to be submitted for semester control

1. *Name the main practices in Feature Driven Design (minimum 3).*
2. *How does a daily Scrum meeting go and what issues are discussed?*
3. *Are intermediate product releases distributed in cascade and incremental software development models?*
4. *What scale of projects can use Crystal Orange?*
5. *What are the conditions for applying the RAD methodology?*
6. *Is there a sprint finish in Scrumban?*
7. *Is there a sprint finish in Kanban?*
8. *How does a user story differ from a use case?*
9. *How is team performance evaluated in Scrum?*
10. *What is the anti-pattern 'watch the master' in pair programming?*
11. *How can requirements change in cascade and incremental software development models?*
12. *Can an architect or a project manager pre-determine task estimates in the planning poker process?*
13. *What are story points used for?*
14. *What is the MoSCoW principle?*