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

Department of Computer Systems Software

Components of Software Engineering. Part 1. Introduction to Software Engineering

Syllabus

Subject details	
HE level	***First (bachelor's)***
Knowledge area	*12 Information Technologies*
Specialty	*121 Software engineering*
Educational program	*Software Engineering of Multimedia and Information Retrieval Systems*
Subject status	*Normative*
Study mode	*Internal (full-time)*
Year of studies, semester	*1st year, spring semester*
Subject volume	*Total number of academic hours - 120, lectures - 36 hours, lab practicum - 18 hours,* *student's unsupervised work – 66 hours.*
Semestrial control/ control measures	*Credit, control works, modular test*
Class schedule	*According to the University spring semester schedule, http://rozklad.kpi.ua/*
Instruction language	*English*
Information about course head/professor	*Lector: PhD student, assistant, Andrii I. Severin, severinandrey97@gmail.com* *Lab. Practicum: PhD student, assistant, Andrii I. Severin*
Course material location	Google classroom: https://classroom.google.com/u/0/c/NTkxMzI1ODQ5NTA3?hl=en

Study program

1. Subject description, objective, matter and outcome

The purpose of studying the credit module "Components of Software Engineering. Part 1. Introduction to Software Engineering" is to provide students with practical skills in designing and building programs for computer systems:
- study of the principles of design and construction of software for computer systems
- study of the template approach to the design of software systems and UML language
- study of methods and tools of software development
- study of object-oriented approach to software development.

***The subject** matter of the discipline "Components of Software Engineering. Part 1. Introduction to Software Engineering" are methods, techniques, templates, tools and systems for research, automated and automatic design, debugging, production and operation, design documentation, standards, procedures and tools to support the management of software lifecycle.*

The study of the «Components of Software Engineering. Part 1. Introduction to Software Engineering» course contributes for students to the formation of general (GC) and professional competences (PC),

necessary for solving practical problems of professional activity related to modeling, designing and software development:
***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.*
***PC07** Knowledge of information data models, the ability to create software for data storage, retrieval and processing.*
***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.*
***PC13** Ability to reasonably select and master software development and maintenance tools.*
***PC21**. Ability to identify, analyze and document software requirements for multimedia and information retrieval systems*
***PC22**. Ability to create innovative startup projects, calculate basic technical and economic indicators and develop business models of multimedia software and information retrieval systems innovative startup projects that have commercial potential for investment.*

Studying the course «Components of Software Engineering. Part 1. Introduction to Software Engineering» contributes to students' formation of the following program learning outcomes (PLO) according to the educational program:
***PLO01** To analyze, purposefully search and select the necessary information and reference resources and knowledge to solve professional problems, taking into account modern advances in science and technology.*
***PLO02** To know the professional ethics code, understand the social significance and cultural aspects of software engineering and adhere to them in professional activities.*
***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.*
***PLO06** To know and apply relevant mathematical concepts, domain methods, system and object-oriented analysis and mathematical modeling for software development.*
***PLO07** To know and to apply in practice the fundamental concepts, paradigms and basic principles of the functioning of language, instrumental and computational tools of software engineering.*
***PLO08** To know and to be able to develop a human-machine interface.*
***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.*
***PLO11** To select initial data for design, guided by formal methods of describing requirements and modeling.*
***PLO13** To know and apply methods of developing algorithms, designing software and data and knowledge structures.*
***PLO14** To apply in practice instrumental software tools for domain analysis, design, testing, visualization, measurement and documentation of software.*

PLO15 *To choose programming languages and development technologies to solve the problems of creating and maintaining software.*
PLO16 *To have the software development, design approval and all types of software documentation release skills.*
PLO17 *To be able to apply methods of component software development.*
PLO18 *To know and be able to apply information technology of processing, storage and transmission of data.*
PLO19 *To know and be able to apply software verification and validation methods.*
PLO20 *To know approaches to evaluation and quality assurance of software.*
PLO23 *To be able to document and present the software development results.*
PLO31 *To be able to identify, analyze and document software requirements for multimedia and information retrieval systems*
PLO32 *To be able to develop and analyze full cycle models for multimedia and information retrieval systems software creation.*
PLO33 *To be able to organize a software product management complete cycle.*
PLO34 *To be able to create innovative startup projects of designing multimedia and information-search systems software that have commercial potential for investment.*
PLO35 *To be able to develop and analyze business models of innovative startup projects of developing multimedia and information retrieval systems software that have commercial potential for investment.*
PLO36 *To be able to manage the creation and implementation of software projects in accordance with international standards.*
PLO38 *To be able to apply programming technologies for multimedia and information retrieval systems software development.*

2. Prerequisites and postrequisites of the subject (its place in the structural and logical scheme of education according to the corresponding educational program)

To the successful study of the course «Components of Software Engineering. Part 1. Introduction to Software Engineering» precedes the study of the course "Fundamentals of Programming" of the bachelor's study curriculum in the specialty 121 Software engineering.

Theoretical knowledge and practical skills, received during the study of the course "Components of software engineering. Part 3. Software architecture" contribute to the assimilation of material from the courses "Components of software engineering. Part 4. Software quality and testing", "Components of software engineering. Course work", "Software security", "Bachelor Thesis" and "Pre-diploma practice" of the bachelor's study curriculum in the specialty 121 Software engineering.

3. The content of the discipline

List of sections and topics of the whole discipline:
Section 1*. Basic terms of Software Engineering, software development lifecycle, SDLC models:*
- Topic 1.1. Basic concepts and problems of software development.
- Topic 1.2. Software life cycle; international software life cycle standards.
- Topic 1.3. Models and methodologies of software development.
Section 2*. Software requirement engineering*
- Topic 2.1. Basic terms. Consideration of the requirement engineering process.
- Topic 2.2. Techniques for requirements gathering.
- Topic 2.3. Functional and non-functional requirements. FURPS+ software classification model.
- Topic 2.4. Use case diagram.
- Topic 2.5. Quality of software requirements.
Section 3*. Software design*
- Topic 3.1. Software architecture design. Design model and its components.
- Topic 3.2. Fundamental concepts of software design.

- Topic 3.3. Entity-relationship diagram, UML diagrams.
- Topic 3.4. Client-server architecture.
- Topic 3.5. Parallelism.
- Topic 3.6. Exception handling.
***Section 4**. Development*
- Topic 4.1. Goals of Coding.
- Topic 4.2. Coding standards. Code style.
- Topic 4.3. Development Tools.
- Topic 4.4. Version control systems (VCS). Git
- Topic 4.5. Unit testing.
***Section 5**. Testing, Deployment and Maintenance*
- Topic 5.1. Software testing, benefits of it.
- Topic 5.2. Verification and validation.
- Topic 5.3. Testing approaches, manual vs automated testing.
- Topic 5.4. Testing levels.
- Topic 5.5. Testing documentation. Bug report elements.
- Topic 5.6. Deployment and support processes, best practices.
- Topic 5.7. Fixed Price vs Time-and-Materials approaches for software projects.

4. Training materials and resources

Basic references:

1. Langer, A., 2012. Guide to software development. London: Springer.

2. Maheshawri, M. and Ch. Jain, P., 2012. A Comparative Analysis of Different types of Models in Software Development Life Cycle. International Journal of Advanced Research in Computer Science and Software Engineering, 2(5), pp.285-288.

3. Fowler, M., Beck, K. (2013). Refactoring : Improving the Design of Existing Code. Addison-Wesley. ISBN: 0201485672.

4. Gamma, E., Helm, R., Johnson, R. E., Vlissides, J. (2016). Design patterns : elements of reusable object-oriented software. Addison-Wesley. ISBN: 9780201633610 0201633612.

5. Martin, R. C., Coplien, J. O. (2009). Clean code: a handbook of agile software craftsmanship. Upper Saddle River, NJ [etc.]: Prentice Hall. ISBN: 9780132350884 0132350882.

6. Everatt, G.D.; McLeod, R Jr (2007). "Chapter 2: The Software Development Life Cycle". Software Testing: Testing Across the Entire Software Development Life Cycle. John Wiley & Sons. pp. 29–58. ISBN 9780470146347.

Additional references:

7. Bidgoli Hossein. 2003. Encyclopedia of Information Systems. Amsterdam: Academic Press.

8. Glinz M, Wieringa R (2007) Stakeholders in requirements engineering. IEEE Softw, IEEE Comput Soc 24(2):18–21.

9. Beck, K., et al. (2001) The Agile Manifesto. Agile Alliance. http://agilemanifesto.org/.
10. James Rumbaugh, Ivar Jacobson, and Grady Booch. 2010. Unified Modeling Language Reference Manual (2nd. ed.). Addison-Wesley Professional.

11. Beck, K., Andres, C. (2004). Extreme Programming Explained : Embrace Change (2nd Edition). Addison-Wesley Professional. ISBN: 0321278658.

12. Sharma, S., Sarkar, D. and Gupta, D., 2012. Agile Processes and Methodologies: A Conceptual Study. International Journal on Computer Science and Engineering (IJCSE), 4(5), pp.892-894.

13. Boyde, J., 2012. A down-to-earth guide to SDLC project management. [Place of publication not identified]: [CreateSpace Independent Pub. Platform].

14. Tatnall, A. (eds) Encyclopedia of Education and Information Technologies. Springer, Cham. https://doi.org/10.1007/978-3-030-10576-1_300606.

15. Lindstrom, L. & Jeffries, R. (2004). Extreme Programming and Agile Software Development Methodologies.. IS Management, 21, 41-52.

16. Ashbacher, C. (2003). "Test-Driven Development: By Example" by Kent Beck (Review).. J. Object Technol., 2, 203-204.

Educational content

5. Methods of mastering the discipline (educational component)

Lecture №	*Type of education lesson*	*Description*
Section 1. Basic terms of Software Engineering, software development lifecycle, SDLC models.		
1	*Lecture 1. Basic terms of Software Engineering, software development lifecycle*	*Overview of the course content, introduction to the discipline: the basic terms of software engineering, stages of the software development lifecycle*
2	*Lecture 2. SDLC models and methodologies*	*Overview of the most famous models and methodologies of software development (waterfall, V-model, prototyping, incremental, iterative, spiral, Agile models)*
Test 1		
Section 2. Software Requirement Engineering		
3	*Lecture 3. Requirement gathering and analysis*	*Basic terms. Consideration of the requirement engineering process. Techniques for requirements gathering. Functional and non-functional requirements. FURPS+ software classification model. Use case diagram.*
4	*Practice 1. Requirements gathering and analysis*	*Defining actors of the system, wring the functional requirements for a specific software system. Drawing the use case diagrams.*
5	*Lecture 4. Quality of software requirements*	*Overview of characteristics of a good requirement and criteria for a good set of requirements. Practical cases.*
6	*Laboratory work 1. Software Requirements Engineering*	*Defining actors of the system, wring the functionalvand non-functional requirements for a specific software system. Drawing the use case diagrams.*
Test 2		
Section 3. Software design		

7	*Lecture 5. Software Design*	*Design model and its components. Fundamental concepts of software design.*
8	*Practice 2. Entity-relationship diagram, UML diagrams*	*Creation of an entity-relationship diagram, and UML diagrams for a specific software system.*
9	*Lecture 6. Important Questions in Software Design*	*Client-server architecture. Parallelism. Exception handling.*
10	*Laboratory work 2. Software Design*	*Creation of an entity-relationship diagram, and UML diagrams for a specific software system.*
Section 4. Development		
11	*Lecture 7. Development phase*	*Goals of Coding. Coding standards. Code Style. Development Tools. Version control systems (VCS).*
12	*Lecture 8. Unit testing and Git*	*Unit testing. Structure of unit tests. Common git commands*
13	*Practice 3. Git, Flowchart, Unit testing*	*Overview of common git commands. Solving the algorithmic problem. Drawing the flowchart diagram. Writing unit tests.*
14	*Laboratory work 3. Development*	*Solving the algorithmic problem. Drawing the flowchart diagram. Writing unit tests. Working with Git repository.*
Test 3		
Section 5. Testing, Deployment and Maintenance		
14	*Lecture 9. Software Testing*	*Software testing, benefits of it. Verification and validation. Testing approaches, manual vs automated testing. Testing levels.*
15	*Lecture 10. Testing, Deployment and Maintenance*	*Testing documentation. Bug report elements. Deployment and support processes, best practices. Fixed Price vs Time-and-Materials approaches for software projects.*
Test 4		

Laboratory classes:
The purpose of the cycle of laboratory works is for students to acquire the necessary practical skills of software design and development.
Laboratory work includes:

1. *Development and analysis of a mathematical algorithm for solving the problem.*
2. *Decomposition of the problem.*
3. *Drawing UML diagrams.*
4. *Printout of the program.*
5. *The results of the program.*
6. *Documentation of laboratory work and realized program*
7. *Conclusions on mastering the topic of laboratory work.*

6. Independent student work

№	Topic given for independent student work	Number of hours	References
1	Preparation for lecture 1	1	1; 2; 8
2	Preparation for lecture 2	1	1; 4, 7
3	Preparation for test 1	2	1; 3; 10
4	Preparation for lecture 3	1	5; 11-13
5	Preparation for practice 1	5	3; 6; 10
6	Preparation for lecture 4	1	3; 8; 10
7	Preparation for test 2	3	11-13
8	Preparation for laboratory work 1	10	2; 5
9	Preparation for lecture 5	1	2; 9
10	Preparation for test 3	3	11-13
11	Preparation for lecture 6	1	2; 6
12	Preparation for practice 2	5	1; 2; 3
13	Preparation for laboratory work 2	10	11-13
14	Preparation for lecture 7	1	4; 5
15	Preparation for lecture 8	1	4; 6
16	Preparation for practice 3	5	9-10
17	Preparation for laboratory work 3	10	4, 12
18	Preparation for lecture 9	1	4, 10
19	Preparation for lecture 10	1	3, 10-12
20	Preparation for test 4	3	1-16

Policy and control

7. Course policy (educational component)

- The student should be present at each lesson. A student may be absent from class if he / she has passed the relevant laboratory work or for a clear reason: illness, etc.

- During the lesson, students can talk to each other only with the permission of the teacher.

- If it is necessary, the student can use the means of communication to search for information on the Internet, etc.

- If it is necessary, student can defend the work online, for this the teacher appoints the time.

- Each laboratory work has its own deadline. Violation of the deadline is punishable by a fine.

- Demonstration of someone else's work is punishable by a fine.

8. Types of control and rating system for assessing learning outcomes (RSO)

During the first lecture, students become acquainted with the rating system of evaluation, which is based on the Regulations on the system of evaluation of learning outcomes https://document.kpi.ua/files/2020_1-273.pdf.

*The semester credit module rating of a student is calculated based on 100-point scale. Points are given for **3** laboratory works (R_l) and **4** tests (R_t). The maximum value of points for one work is equal to 20. The maximum value of points for one test is equal 10 points. The maximum value of a rating for a semester (Rs) makes $Rs = R_l + R_t = 100$ points.*

For laboratory work points are awarded for:
- timeliness of preparation of the protocol for laboratory class, completeness of performance of the theoretical task: 0-5 points;
- written lab report on the topic of laboratory work: 0-5 points;
- correct functioning of the developed programs: 0-10 points.

Evaluation criteria of tests:

- correct and meaningful answer – 9-10 points;

- correct answer, incomplete explanations - 5-8 points;

- the answer contains errors - 1-4 points;

- no answer or the answer is incorrect - 0 points.

Calendar attestation of students (8 and 14 weeks of semesters) for the discipline is carried out according to the value of the current rating of the student at the time of attestation. If the value of this rating is not less than 50% of the maximum possible at the time of certification, the student is considered satisfactorily certified. Otherwise, in the attestation statement is set "unsatisfactory".

A necessary condition for obtaining a test by a student is the performance and defense of all laboratory works with a sum of at least 60 %. Otherwise, the student should perform a test. Students who do not have academic debt can also increase their grades by taking a test. When performing the test, the student's semester rating (Rs) is reset. An assessment based on the results of the test (R_K) is entered in the statement.

The paper for the test consists of 10 tasks on the topics of lectures and laboratory works performed in the semester. Each task is evaluated from 0 to 10 points.

Criteria for evaluating each task on four levels:

- correct and meaningful answer – 9-10 points;

- correct answer, incomplete explanations - 5-8 points;

- the answer contains errors - 2-4 points;

- no answer or the answer is incorrect - 0 points.

The maximum score for the test is $R_K = 100$ points. It becomes a semester grade.

Taking into account the received sum of points the final estimation is defined by the following table on a university scale

Number of points	*Grade*
100-95	Excellent
94-85	Very good
84-75	Good
74-65	Satisfactory
64-60	Sufficient
Less than 60	Unsatisfactory

Requirements not met	Not permitted

9. Additional information on the discipline (educational component)

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

Syllabus of the subject is:

Prepared by, PhD student, assistant, Andrii I. Severin.

Adopted by Computer Systems Software Department (protocol № 12 from 26.04.23)

Approved by the Faculty Board of Methodology (protocol № 10 from 26.05.23)

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

1. Basic terms of software engineering, stages of the software development lifecycle.

2. Comparison of SDLC models.

3. Agile model. Values and principles.

4. Defining which requirements from the list are "bad" (don't meet criteria of good requirements) and correcting them.

5. Use case diagram. Relationships.

6. Definition of UML. Types of diagrams. Usage.

7. Characteristics of a Good Requirements.

8. Centralized and distributed version control systems.

9. Characteristic for the development/coding process from the SDLC (what is it; purpose/goals; coding standards, tools).

10. Fundamental concepts of Software Design.

11. Code style.

12. Software design. Diagrams that can be used to describe the system design.

13. Software Maintenance (definition, purpose, need for maintenance). Best practices which can be used during this stage.

14. Black-box and White-box testing.

15. Characteristics of a good bug report. Elements of a bug report.

16. Fixed Price vs Time-and-Materials pricing models for the project.

17. Characteristic for the testing process from the SDLC (what is it; purpose; differences between validation and verification, testing types, testing levels).

18. Tools can be used during the development process.

19. Creation of the physical ER diagram.