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

Department of Computer Engineering

Software Engineering Components. Part 1. Introduction to software engineering

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	*1 year (2 semester)*
ECTS workload	*4 credits*
Testing and assessment	*Test*
Course Schedule	*Lectures 18 (36 hours), Laboratory9 (18hours)*
Language of Instruction	*English*
Course Instructors	Lecturer: Prof., Dr.Sc. Bogdan Korniyenko, b.korniyenko@kpi.ua Laboratory: Prof., Dr.Sc. Bogdan Korniyenko, b.korniyenko@kpi.ua
Access to the course	https://classroom.google.com

Program of academic discipline

1 Course description, goals and objectives, and learning outcomes

Educational discipline "Software Engineering Components. Part 1. Introduction to software engineering" is designed to help the student acquire: knowledge of the theoretical and intellectual basis of design presented in the SWEBOK core; basic models and life cycle stages used in software design; understanding of the principles of application of software design and development technologies; the ability to freely navigate the modern market of software products, which is used in the process of designing and developing software; the ability to use the capabilities of domestic and foreign software tools in the design and development of software, in order to ensure a high level of software quality.

The purpose of studying the discipline "Software Engineering Components. Part 1. Introduction to software engineering" is aimed at forming future engineers with a modern level of information and digital culture, mastering the basic principles of creating software products; acquisition of practical skills of independent compilation of professional software and use of modern information technologies to solve various problems of an applied nature. 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), software architect (Software Architect).

The subject of study of the discipline is modern methods, tools and technologies of software development.

According to the requirements of the EP, the discipline "Software Engineering Components" should ensure that applicants acquire competencies and program learning outcomes: PC01-05, PC07, PC08, PC11-13, PLO01-04, PLO06-11, PLO13-20. In particular, after mastering the module "Software Engineering Components. Part 1. Introduction to software engineering" must demonstrate the following competencies and program learning outcomes:

- the ability to formulate and ensure software quality requirements in accordance with customer requirements, specifications and standards;
- the ability to adhere to specifications, standards, rules and recommendations in the professional field when implementing life cycle processes;
- the ability to apply fundamental and interdisciplinary knowledge to successfully solve software engineering tasks;
- the ability to evaluate and take into account economic, social, technological and environmental factors affecting the field of professional activity;
- the ability to accumulate, process and systematize professional knowledge regarding the creation and maintenance of software and recognition of the importance of lifelong learning;
- the ability to implement phases and iterations of the life cycle of software systems and information technologies based on appropriate models and software development approaches.

According to the results of studying the educational discipline "Software Engineering Components. Part 1. Introduction to software engineering", 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 "Software Engineering Components. Part 1. Introduction to software engineering":

- 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 ofsoftware 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", "Algorithms and data structures".

Module "Software Engineering Components. Part 1. Introduction to software engineering" is necessary for studying the following parts of the discipline "Software Engineering Components", and can also be useful when studying the disciplines "Methodologies and technologies of software development", "Risk management and project quality".

3 Structure of the credit module

A list of the main topics included in the study program of the discipline "Software Engineering Components. Part 1. Introduction to software engineering":

Section1. Life cycle of software.

Topic 1.1. Software engineering. Programming technologies in a historical aspect.

Topic 1.2. Software life cycle. Life cycle models.

Topic 1.3. Software development methodology. Flexible application development. Principles of Agile development. Scrum, RAD. XP programming.

Topic 1.4. Software requirements management.

Section2.Software architecture development.

Topic2.1. Software architecture design.

Topic2.2. Models of system structuring.

Topic2.3. Management simulation and decomposition on the module.

Topic2.4. User interface design.

Section3. Software modeling.

Topic3.1. A structural approach to modeling. SADT methodology.

Topic3.2. Modeling data flows.

Topic3.3. Modeling of data structures. Diagram of state transitions.

Topic3.4. Basics of the UML language. Class diagrams.

Section 4. Management of software projects.

Topic4.1. Tasks of project management.

Topic4.2. Project concepts. Software product risk management.

Topic4.3. Planning of software projects. SMART. WBS. PERT. CMP. Gant Chart.

Topic4.3. Formation of a team of developers. Distribution of roles and responsibilities.

Section5. Software quality assurance and control.

Topic 5.1. Metrics and software quality.

Topic5.2. Software verification and testing.

4 Educational resources and materials

Basic:

1. Pressman, Roger (2010) *Software Engineering: A Practitioner's Approach*, McGraw Hill, New York, NY.
2. Sommerville, Ian (2011) *Software Engineering*, Addison-Wesley, Boston, MA.
3. Stephens, Rod (2015) *Beginning Software Engineering*, Wrox.
4. Tsui, Frank , Orlando Karam and Barbara Bernal (2013) *Essentials of Software Engineering*, Jones & Bartlett Learning , Sudbury, MA.

5. Pfleeger, Shari (2001) *Sofwtare Engineering: Theory and Practice*, Prentice Hall, Upper Saddle River, NJ.

Supplementary:

1 Larman, C. (2005) *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and iterative Development*, Pearson
2 Ambler, S. (2002) *AgileModeling: Effective Practices for Extreme Programming and the Unified Process*, NewYork, John Wiley&Sons.
3 Bass, D.L., Clements, D.P. andKazman, D.R. (2012) *Software Architecture in Practice*, 3rd edn, Upper Saddle River, NJ, Addison Wesley
4 Beck, K. (2004) *Extreme Programming Explained: Embrace Change*, Upper Saddle River, NJ, Addison Wesley
5 Clemens Szyperski (2002) Component Software: Beyond object-oriented programming, Addison-Wesley
6 *John Cheesman & John Daniels (2000) UML Components: A simple process for specifying component-based software (The component software series) Addison-Wesley*
7 Rob Pooley, Perdita Stevens (2006) *Using UML Software Engineering with Objects and Components*, second edition. Addison-Wesley
8 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
Section1. Life cycle of software. Topic 1.1. Software engineering. Programming technologies in a historical aspect. Topic 1.2. Software life cycle. Life cycle models. Topic 1.3. Software development methodology. Flexible application development. Principles of Agile development. Scrum, RAD. XP programming. Topic 1.4. Software requirements management.	18	4	4	10
Section2.Software architecture development. Topic 2.1. Software architecture design. Topic 2.2. Models of system structuring. Topic 2.3. Management simulation and decomposition on the module. Topic 2.4. User interface design.	26	6	4	16
Section3. Software modeling. Topic 3.1. A structural approach to modeling. SADT methodology. Topic 3.2. Modeling data flows. Topic 3.3. Modeling of data structures. Diagram of state transitions. Topic 3.4. Basics of the UML language. Class diagrams.	28	8	4	16

Section 4. Management of software projects. Topic 4.1. Tasks of project management. Topic 4.2. Project concepts. Software product risk management. Topic 4.3. Planning of software projects. SMART. WBS. PERT. CMP. Gant Chart. Topic 4.3. Formation of a team of developers. Distribution of roles and responsibilities.	24	6	4	14
Section5. Software quality assurance and control. Topic 5.1. Metrics and software quality. Topic 5.2. Software verification and testing.	26	12	4	10
Total hours in semester	120	36	20	64

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: IT project management methodologies;
- Laboratory work #2: Risk Analysis;
- Laboratory work #3: Comparative analysis of information systems;
- Laboratory work #4: Interim project planning. Gantt Chart;
- Laboratory work # 5: Needs Identification Stage.

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 Professor of the Department of Information Systems and Technologies, Bogdan Korniyenko

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

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