



COMPONENTS OF SOFTWARE ENGINEERING.

PART 3. SOFTWARE ARCHITECTURE

Syllabus

Requisites of the Course

Cycle of Higher Education	<i>First cycle of higher education (Bachelor's degree)</i>
Field of Study	<i>12 Information Technologies</i>
Specialty	<i>121 Software engineering</i>
Education Program	<i>Software Engineering of Multimedia and Information Retrieval Systems</i>
Type of Course	<i>Normative</i>
Mode of Studies	<i>full-time</i>
Year of studies, semester	<i>2nd year, 4th semester</i>
ECTS workload	<i>36 hours for lectures, 18 hours for laboratory work, 96 hours for self-study.</i>
Testing and assessment	<i>Exam, modular control work, calendar control</i>
Course Schedule	<i>According to http://roz.kpi.ua/</i>
Language of Instruction	<i>English</i>
Course Instructors	<i>Inna Saiapina, PhD, Assoc. Prof., saiapina@pzks.fpm.kpi.ua</i>
Access to the course	<i>Google classroom. To be provided to registered students.</i>

Outline of the Course

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

The study of the "Components of software engineering. Part 3. Software architecture" allows students to develop the competencies necessary for solving practical tasks of professional activities related to the design, modeling and development of software for multimedia and information-search systems.

***The purpose** to study the course "Components of software engineering. Part 3. Software architecture" is the formation of students' understanding of the basic principles and mechanisms and the ability to design, model and program implementation of multimedia and information-search systems and applications working independently or in a team.*

***The subject** of the "Components of software engineering. Part 3. Software architecture" course are methods, methodologies and approaches to building a software architecture, its component modules and relationships between them.*

The study of the "Components of software engineering. Part 3. Software architecture» 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

Studying the course "Components of software engineering. Part 3. Software architecture" 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.

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.

PLO38 To be able to apply programming technologies for multimedia and information retrieval systems software development.

2. Prerequisites and post-requisites of the course (the place of the course in the scheme of studies in accordance with curriculum)

To the successful study of the course "Components of software engineering. Part 3. Software architecture" precedes the study of the course "Components of Software Engineering. Part 2. Software Modeling. Analysis of Software Requirements" 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. Content of the course

Course "Components of software engineering. Part 3. Software architecture" involves the study of the following topics:

Topic 1. Introduction. Basic definitions and concepts.

Topic 2. Architecture planning and design.

Topic 3. Object analysis and modeling.

Topic 4. Architectural styles and tools for their implementation.

Modular control work.

Exam

4. Coursebooks and teaching resources

Main literature:

1. Educational and methodological materials on the subject "Components of software engineering. Part 3. Software architecture".

Use to master the practical skills of the discipline. The materials are in Google classroom; access is to be provided to registered students.

Additional literature:

2. L. Bass, P. Clements, R. Kazman. *Software Architecture in Practice*. Addison-Wesley, 2021. 497 p. URL: <https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=30659> or <https://people.ece.ubc.ca/~matei/EECE417/BASS/>)
3. M. Richards, N. Ford. *Fundamentals of Software Architecture*, O'Reilly Media, 2021.
4. M. Richards. *Software Architecture Patterns*. O'Reilly Media, 2015. 55 p.
5. R. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson, 2017. 432 p.
6. B. Rump. *Agile Modeling with UML*. Springer, 2017, 388 p. DOI 10.1007/978-3-319-58862-9
7. E. Freeman, E. Robson. *Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software*. O'Reilly Media, 2021, 669 p.
8. C4 model – URL: <https://c4model.com/>
9. *Architectural Blueprints—The "4+1" View Model of Software Architecture*, by Philippe Kruchten – URL: <https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf>

Educational content

Methodology for mastering the course (educational component)

No.	Type of class	Description of the class
		Topic 1: Introduction. Basic definitions and concepts

1	<i>Lecture 1: Introduction. Basic definitions and concepts (2 ac.h.)</i>	<i>Introduction. Information on the organisation of the educational process. Useful resources. Academic integrity. Assessment system. What this course will teach you. The life cycle of software development. The concept of software architecture. Task for the self-study work: p. 6, No. 1.</i>
<i>Topic 2. Architecture planning and design</i>		
2	<i>Lecture 2. Life Cycle Standards and Models (2 ac. h.)</i>	<i>Waterfall model. Incremental model. Spiral model. Evolutionary model. Basic software development methodologies. Agile SCRUM. Extreme Programming, SAFE, Kanban. Task for the self-study work: p. 5, No 2.</i>
3	<i>Laboratory session 1. Laboratory work 1. Part 1. (2 ac. h.)</i>	<i>Software requirements analysis and formulation. Part 1. Task for the self-study work: p. 5, No 3.</i>
4	<i>Lecture 3: System architecture and quality attributes (2 ac. h.)</i>	<i>Scenarios of quality attributes. Reliability. Modifiability, Performance, Security. Testability. Task for the self-study work: p. 5, No 4.</i>
<i>Topic 3: Object analysis and modelling</i>		
5	<i>Lecture 4. Object-oriented analysis and design (2 ac. h.)</i>	<i>Types of software documentation notation. Tools for building diagrams. Use-case Diagram. Object-oriented approach and basic principles. Trade-offs in requirements and design. Task for the self-study work: p. 5, No 5.</i>
6	<i>Laboratory session 2. Laboratory work 1. Part 2. (2 ac.h.)</i>	<i>Software requirements analysis and formulation. Part 2. Task for the self-study work: p. 5, No 6.</i>
7	<i>Lecture 5. Object-oriented modelling (2 ac.h.)</i>	<i>UML class diagrams. Specialised UML diagrams. C4 notation. Task for the self-study work: p. 5, No 7.</i>
8	<i>Lecture 6. Cohesion and coupling (2 ac.h.)</i>	<i>Cohesion as an internal characteristic of a module. Coupling as an external characteristic of a module. Their impact on architecture. Monolithic vs Distributed architectures. Misconceptions about the organisation of distributed architectures. Task for the self-study work: p. 5, No 8.</i>
9	<i>Laboratory session 3. Laboratory work 2. (2 ac. h.)</i>	<i>Information system modelling and documentation. Task for the self-study work: p. 5, No 9</i>
<i>Topic 4. Architectural styles and tools for their implementation</i>		
10	<i>Lecture 7. Layered architecture (2 ac.h.)</i>	<i>Layered Architecture: features of application, topology and purpose of layers, examples, design principles, closed/open layers, anti-pattern "architectural funnel", main characteristics of use. Task for the self-study work: p. 5, No 10.</i>
11	<i>Lecture 8. Microservice architecture. Part 1. (2 ac.h.)</i>	<i>Microservice Architecture. The history of microservices. The disadvantages of monolithic and service-oriented architecture as predecessors of microservices. Microservice architecture, main characteristics.</i>

		<i>Task for the self-study work: p. 5, No 11.</i>
12	<i>Laboratory session 4. Laboratory work 3. (2 ac. hrs.)</i>	<i>Creating a prototype of an information system. Task for the self-study work: p. 5, No 12.</i>
13	<i>Лекція 9. Мікросервісна архітектура. Частина 2 (2 ac.h.)</i>	<i>Мікросервісна. 9 характеристик гарної архітектури мікросервісів. Процес проектування архітектури: визначення компонентів. Task for the self-study work: p. 5, No 13.</i>
14	<i>Лекція 10. Мікросервісна архітектура. Частина 3 (2 ac.h.)</i>	<i>Мікросервісна архітектура. Процес проектування архітектури: визначення шаблонів зв'язку між компонентами та визначення технологічного стеку. Task for the self-study work: p. 5, No 14.</i>
15	<i>Laboratory session 5. Laboratory work 4. Part 1 (2 ac.h.)</i>	<i>An overview of how a message broker works. Part 1. Task for the self-study work: p. 5, No 15</i>
16	<i>Lecture 11. Event-driven architecture. Part 1 (2 ac.h.)</i>	<i>Event Driven Architecture (EDA). Events. Types. Main characteristics. Purpose and advantages. Push and Pull methods. Orchestration and choreography. Task for the self-study work: p. 5, No 16.</i>
17	<i>Lecture 12. Event-driven architecture. Part 2 (2 ac.h.)</i>	<i>Event-driven architecture. Event Sourcing and CQRS. The scope of EDA. Streaming & EDA. Logging and monitoring. Implementation of EDA. Task for the self-study work: p. 5, No 17.</i>
18	<i>Laboratory session 6. Laboratory work 4. Частина 2 (2 ак. год.)</i>	<i>An overview of how a message broker works. Part 2 Task for the self-study work: p. 5, No 18</i>
19	<i>Lecture 13. API implementation technologies (2 ac. h.)</i>	<i>Technologies for the implementation of Application Programming Interface (API). REST API. GraphQL. gRPC. History of creation. Basic principles of implementation. Advantages and disadvantages. Task for the self-study work: p. 5, No 19</i>
20	<i>Lecture 14. Case study: Developing an application architecture. Case 1. (2 ac. h.)</i>	<i>Case 1: development of the application architecture. Identification of functional and non-functional requirements. Analysis and design of system components. Choosing a technology stack. Designing the architecture of services and APIs. Method of messaging. Task for the self-study work: p. 5, No 20.</i>
21	<i>Laboratory session 7. Laboratory work 5. (2 ac. h.)</i>	<i>Introduction to REST architectural style and GraphQL query language. Task for the self-study work: p. 5, No. 21.</i>
22	<i>Lecture 15. Case study: Developing an application architecture. Case 2. (2 ac. h.)</i>	<i>The main types of applications. Web applications. Web APIs. Mobile applications. Console applications. Service applications. Desktop applications. Case 2: development of a system architecture with an intensive flow of telemetry data. Task for the self-study work: p. 5, No. 22.</i>

23	Lecture 16. Case study: Developing an application architecture. Case 3. (2 ac. h.)	Case 3: development of an e-commerce application architecture. Task for the self-study work: p. 5, No. 23.
24	Laboratory session 8. Laboratory work 6. (2 ac. h.)	Development of an information system architecture. Task for the self-study work: p. 5, No. 24.
25	Lecture 17: Virtualisation and containers (2 ac. h)	Types of virtualization. Virtual machines and virtual containers. Advantages and disadvantages. Docker: a tool for containerisation. Task for the self-study work: p. 5, No. 25.
26	Lecture 18. Module control work (2 ac.h.)	Modular control work Task for the self-study work: p. 5, No. 26.
27	Laboratory session 9.	The final lesson. Finalisation of laboratory work results. Task for the self-study work: p. 5, No. 26.

5. Self-study work of a student

The course "Components of software engineering. Part 3. Software architecture " is based on self-study preparations for classroom classes on theoretical and practical topics.

No.	Name of the topic for the self-study work	Number of hours	Literature
1	Preparation for lecture 1	2	1
2	Preparation for lecture 2	2	1,4-6,,8,9, 11,13
3	Підготовка до лабораторної роботи 1. Частина 1	2	1-5,8,9
4	Preparation for lecture 3	2	1-6, 8
5	Preparation for lecture 4	2	1,4,5,12, 15
6	Preparing for the laboratory work 1. Частина 2	3	1,4- 6,8,9,11,13
7	Preparation for lecture 5	2	1,4,5,12, 15
8	Preparation for lecture 6	2	1-5, 7, 10
9	Preparing for the laboratory work 2	4	1,4,5,12, 15
10	Preparation for lecture 7	2	1-6, 8-11
11	Preparation for lecture 8	2	1-6, 8-11
12	Preparing for the laboratory work 3	5	1
13	Preparation for lecture 9	2	1-6, 8-11
14	Preparation for lecture 10	2	1-6, 8-11
15	Preparing for the laboratory work 4. Частина 1	2	1, 4,6, 8-10
16	Preparation for lecture 11	2	1-6, 8-11
17	Preparation for lecture 12	2	1-6, 8-11
18	Preparing for the laboratory work 4. Частина 2	3	1, 4,6, 8-10
19	Preparation for lecture 13	2	1-6
20	Preparation for lecture 14	2	1-6, 8-11

21	<i>Preparing for the laboratory work 5</i>	4	1-6
22	<i>Preparation for lecture 15</i>	2	1-6, 8-11
23	<i>Preparation for lecture 16</i>	2	1-6, 8-11
24	<i>Preparing for the laboratory work 6</i>	5	1-6, 8-11
25	<i>Preparation for lecture 17</i>	1	1,3,6,11,15
26	<i>Підготовка до модульної контрольної роботи</i>	8	1-15
27	<i>Підготовка до екзамену</i>	30	1-15

Policy and Assessment

6. Course policy

- *Attending lectures is mandatory.*
- *Attending laboratory classes can be occasional and as needed to defend laboratory work.*
- *Rules of behavior in classes: activity, respect to the others, turning off phones.*
- *Adherence to the policy of academic integrity.*
- *Rules for the laboratory work defence: the work must be done according to the student's variant, which is determined by his number in the group list, or the topic and subject area approved by the teacher.*

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

Bonus points are awarded for:

- activity in lectures and laboratory classes. The maximum number of points for all classes is 5 points.

Penalty points are calculated for:

- plagiarism The performed laboratory work does not correspond to the task option, the identity of laboratory work reports among different works (number of points: 5 points).

7. Monitoring and grading policy

During the semester, students perform 6 laboratory works. The maximum number of points for laboratory work: 5 points.

Points are awarded for:

- quality of laboratory work (report): 0-2 points;

- survey (test) during the defense of laboratory work: 0-2 points;

- timely submission of work for defense: 0-1 point.

Criteria for evaluating the quality of laboratory work (report):

2 points – the work is done qualitatively, in full;

0-1 point – the work is incomplete or contains errors.

Evaluation criteria for the survey on the protection of laboratory work:

2 points – the answer is complete, well-argued;

1 point – there are errors in the answer;

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

If the laboratory work is re-submitted for examination, the total maximum grade for the laboratory work is reduced by 1 point.

The maximum number of points for performing and defending laboratory work:

$R_L = 6 \text{ laboratory works} \times 5 \text{ points} = 30 \text{ points.}$

The task for the modular control work consists of 14 test questions - 8 questions with one correct answer and 6 questions with several correct answers. Each question with one correct answer is valued at 1 point, each question with several correct answers are valued at 2 points.

Evaluation criteria for each test question with one correct answer:

1 point – the answer is correct;

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

Evaluation criteria for each multiple-choice test question:

2 points – all correct answers and no incorrect answers are selected;

1 point – at least 50% of all correct answers are chosen;

0 points – no answer or all answers are incorrect.

The maximum number of points for a modular control work:

$R_{MKR} = 1 \text{ point} \times 8 \text{ test questions with one correct answer} + 2 \text{ points} \times 6 \text{ questions with several correct answers} = 20 \text{ points.}$

The rating scale for the course is equal to:

$R = R_c + R_{exam} = R_L + R_{MKR} + R_{exam} = 30 \text{ points} + 20 \text{ points} + 50 \text{ points} = 100 \text{ points.}$

Calendar control: is conducted 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 15 points (50% of the maximum number of points a student can receive before the second certification).

Semester control: exam

Conditions for admission to semester control:

With a semester rating (R_c) of at least 30 points and the enrollment of all laboratory work, the student is admitted to the exam. After passing the exam, a grade is assigned according to the table (Table of correspondence of rating points to grades on the university scale).

A necessary condition for admission to the exam is the performance and defense of laboratory work.

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

<i>Scores</i>	<i>Rating</i>
100-95	Perfectly
94-85	Very good
84-75	Fine
74-65	Satisfactorily
64-60	Enough
Less than 60	Unsatisfactorily
Admission conditions not met	Not allowed

Syllabus of the course

Is designed by Inna Saiapina, Ph.D., Assoc. Prof.

Approved by Computer Systems Software Department (protocol № 11, April 30 2025)

Approved by the Faculty Board of Methodology (protocol № 11, May 23, 2025)