ARCHITECTURE AND DESIGN OF SOFTWARE (PART 2)

COURSE SYLLABUS

Code and name of specialty	121 – Software Engineering	**Institute / faculty**	Faculty of Computer Science and Software Engineering
Program name	“Software Engineering”	**Department**	Software Engineering and Management Information Technologies
Type of program	Educational and Professional	**Language of instruction**	Ukrainian, English

LECTURER

Full name, *e-mail*	**DVUKHHLAVOV Dmytro**, ***Dmytro.Dvukhhlavov@khpi.edu.ua***

Ph.D., Associate Professor, Associate Professor of Software Engineering and Information Technology Management.
Has prepared and published more than 40 publications, 1 article in publications indexed in Scopus, 2 textbooks, 2 guidelines for making practice tasks.
h-index = 3, i10-index = 0 y Google Academy-https://scholar.google.com/citations?user= OAzyFg8AAAAJ&hl=ru; identifier ORCID-https://orcid.org/ 0000-0002-3361-3212).
Leading lecturer of the courses: Advanced Java programming course *(Bachelors) (Ukrainian)*, Java-based web applications *(Bachelors) (Ukrainian)*, Architecture and Design of Software (part 1) *(Bachelors) (English and Ukrainian)*, Architecture and Design of Software (part 2) *(Bachelors) (English and Ukrainian).*

GENERAL DESCRIPTION OF THE COURSE

Summary	The discipline «Architecture and Design of Software (Part 2)» is study compulsory discipline in the cycle of professional training of preparation level "bachelor" on specialty 121 “Software Engineering”. It is taught in the fifth semester in the amount of 90 hours (3 ECTS credits), in particular: lectures - 32 hours, laboratory classes - 32 hours, independent work - 26 hours. The study of the discipline involves the preparation and defense of a course project, during which students design software. Final reporting on the discipline - an exam. Teaching the discipline provides students with an understanding of the content of the software development process in IT companies, as well as the development of basic skills in software design and collaboration with the project code required to work in a team of software developers.
Course objectives	The course aims to deepen the knowledge system to deepen knowledge about the software life cycle and standards governing the implementation of its stages, the principles of organizing software development at the present stage of development of the IT industry, software development technologies, design strategies and methodologies and software development, on the presentation of software design results, on the principles of user interface design, as well as the development of software development skills based on a defined list of software requirements and skills of git version control system and github repository during software development. Mastering the discipline is an important element for the making diploma work.

Types of classes and control	Lectures, laboratory classes, consultations, course project. Final control – exam.						
Terms	6						
Student workload (credits) / Type of course	3 / Mandatory	**Lectures (hours)**	32	**Workshops (hours)**	32	**Independent work (hours)**	26
Program competencies	GC 2. Ability to apply knowledge in practical situations. GC 05. Ability to learn and master modern knowledge. GC 06. Ability to search, process and analyze information from various sources. PC13. Ability to identify, classify and formulate software requirements. PC14. Ability to participate in software design, including modelling (formal description) of its structure, behavior and functioning processes. PC15. Ability to develop architectures, modules and components of software systems. PC17. Ability to adhere to specifications, standards, rules and recommendations in the professional field in the implementation of life cycle processes. PC19. Knowledge of information data models, the ability to create software for data storage, retrieval and processing. PC23. Ability to implement phases and iterations of the life cycle of software systems and information technology based on appropriate models and approaches to software development. PC24. Ability to carry out the system integration process, apply change management standards and procedures to maintain the integrity, overall functionality and reliability of the software. PC25. Ability to reasonably select and master software development and maintenance tools.						
Learning outcomes	PO01. Analyze, purposefully search for and select the necessary information and reference resources and knowledge to solve professional problems, taking into account modern advances in science and technology. PO03. Know the basic processes, phases and iterations of the software life cycle. PO06. Ability to select and use the appropriate task methodology of software development. PO07. Know and apply in practice the fundamental concepts, paradigms and basic principles of operation of language, tools and computing software engineering. PO10. Conduct a pre-project survey of the subject area, systematic analysis of the design object. PO11. Choose source data for design, guided by formal methods of describing requirements and modelling. PO12. Put effective approaches to software design into practice. PO13. Know and apply methods of algorithm development, software design and data and knowledge structures. PO14. Put into practice the tools of domain analysis, design, testing, visualization, measurement and documentation of software. PO15. Being motivated to choose programming languages and development technologies to solve problems of software design and maintenance. PO16. Have the skills of team development, approval, design and release of all types of software documentation. PO17. Be able to apply methods of component software development. PO19. Know and be able to apply methods of software verification and validation. PO20. Know the approaches to evaluating and ensuring the quality of software PO23. Be able to document and present the results of software development.						

Teaching and learning methods	The main method of teaching during lectures is the explanatory-illustrative method. To intensify cognitive activity, students' speeches and organization of discussions on certain issues of lectures are provided. During laboratory works there is a training of skills in two directions - development of skills of work with the system of control of versions and development of skills of designing of the software on the basis of system of the certain requirements to it. Execution of laboratory work on software design actually corresponds to the stages of execution of the course project, which must be defended in the credit week of the semester. The subject area for design is individual. When developing a course project, emphasis is placed on the fact that the acquired skills are important for diploma design.
Forms of assessment (continuous assessment CAS, final assessment FAS)	Assimilation of the theory is tested in the form of a rapid survey during lectures (CAS), a survey or automated testing at the beginning of laboratory work (CAS). Control of mastering the material for self-study involves the preparation and defense of abstracts on individual topics (2 abstracts) (CAS). Checking the level of practical skills is tested in laboratory work performed on individual options (CAS). Final control is carried out on the defense of the course project and during the semester exam (FAS). The exam provides written experience on theoretical and ethical issues and the creation of a specific model for an individual task in a limited time.

ASSESSMENT AND GRADING

	0-34	F	Unsatisfactory (with mandatory repetition of the course)			

Ranges of points corresponding to grades						
	Total score (points) for all types of learning activities	**ECTS grading scale**	**The national grading scale**	**Allocation of grade points**	Assimilation of theory	15 points
	90-100	A	excellent			
	82-89	B	good		Practice Git	15 points
	74-81	C				
	64-73	D	satisfactory		Practice on projecting SW and passing course project	50 points
	60-63	E				
	35-59	FX	Unsatisfactory (with the exam retake option)		Exam	20 points
	0-34	F	Unsatisfactory (with mandatory repetition of the course)		Summary	100 points

Course policy	Students are required to attend classes according to the schedule. In the absence of a student at the lecture, he works out a syllabus of lectures before the next lesson. Participation in laboratory work involves the need to repeat the lecture material and self-study of recommended sources. At the beginning of the laboratory there is an experience of students for the materials of lectures and independent work. Performing laboratory tasks requires prior preparation and advance processing of all necessary materials for productive discussions during the lesson and their operational implementation. All laboratory work is required to obtain a final grade in the discipline. An important element of training is the need to adhere to the schedule of presentation of laboratory results

	and abstracts. For delay in execution without an officially confirmed reason, the score is reduced.

COURSE STRUCTURE AND CONTENT

Topic 1	A modern point of view at the software development process. SCRUM framework: purpose and principles of application during development. Contents CD / CI (continuous integration / continuous delivery). Software to automate software development (6 hours)	**Laboratory class 1**	Installing and configuring the git system. Work in git-bash (4 hours) (PT64)	**Independent work**	Course project. Choice of task for automation and definition of subject area. Description of business processes of the subject area. Definition of functional and non-functional requirements to the designed software. Development of SRS for the created software. Designing a domain model of subject area objects.
Topic 2	Design as a component of the software development life cycle. Basis of definition and regulations. Notations and presentation of design results. (6 years)	**Laboratory class 2**	Branch management using git (4 hours) (PT64)		Course project. Development of use case diagrams and their refinement by developing scenarios or interaction diagrams.
Topic 3	UML as a notation to represent design results. Classification of UML diagrams. Use UML diagrams to describe the behavior and structure of software. UML charting software. (6 years)	**Laboratory class 3**	Using the GitHub repository (2 hours) (PT64)		Course project. Entity-Relation diagram design. Reasonable choice of database management system.
Topic 4	Strategies and methodologies for software development (10 hours)	**Laboratory class 4**	Formalized representation of business processes and software requirements for the use of CASE-systems (6 hours) (PT59,PT60)		Course project. Reasonable choice of software architecture. Development and description of the structure of software components and schemes for its deployment. Development of components and deployment diagram.
Topic 5	User interface design (4 hours)	**Laboratory class 5**	Designing system behavior for different categories of users based on UML diagrams (6 hours) (PT61)		
		Laboratory class 6	Design of the scheme for data storage (6 hours) (PT62)		
		Laboratory class 7	Designing the structure of software components and its deployment scheme (4 hours) (PT62)		

RECOMMENDED READING

Compulsory

1 Sommerville Ian. Software Engineering (6th ed.) Retrieved from https://www.academia.edu/6826193/Ian_Sommerville_Software_Engineering_6th_Edition.
2 Guide to Software Engineering Body of Knowledge (SWEBOK), version 3.0. (2014). IEEE Computer Society.
3 Лаврищева, К. М. (2013). Software Engineering комп'ютерних систем. Парадигми, технології та CASE засоби програмування. Київ: Наукова думка.
4 Git-Documentation (git-scm.com). Retrieved from https://git-scm.com/doc.
5 Табунщік, Г. В., Каплієнко, Т. І., Петрова, О. А. (2016). Проектування та моделювання програмного забезпечення сучасних інформаційних систем. Запоріжжя.
6 Петрик, М. Р., Петрик, О. Ю. (2015). Моделювання програмного забезпечення. Тернопіль: Вид-во ТНТУ.
7 Rumpe Bernhard. (2016).
Modeling with UML. Language, Concepts, Methods. Springer. Retrieved from https://nibmehub.com/opac-service/pdf/read/Modeling%20with%20UML_%20Language-%20Concepts-%20Methods.pdf .
8 Barry, W, Boehm, Jo, Ann Lane, Richard Turner. (2014).Supannika Koolmanojwong. The Incremental Commitment Spiral Model: Principles and Practices for Successful Systems and Software. Addison-Wesley.
9 Kruchten Philippe. (2004). The Rational Unified Process: An Introduction. (3rd ed.). Pearson.
10 Wiegers Karl, Beatty Joy. (2013). Software Requirements (3nd ed). Microsoft Press.
11 Грицюк, Ю. І. (2018). Аналіз вимог до програмного забезпечення.
12 Піхлер, Р. (2019). Agile продукт-менеджмент за допомогою. Scrum. Фабула.
13 Мартін Роберт. (2019). Чистий код. Фабула.
14 Clements Paul, Bachmann Felix, Bass Len, Garlan David, Ivers James, Little Reed… Stafford Judith. (2010). Documenting Software Architectures: Views and Beyond. (2nd ed.). Addison-Wesley Professional.

Recommended

Стандарти

1 ISO/IEC/IEEE 12207 Systems and software engineering. Software life cycle processes.
2 IEEE 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology.
3 ISO/IEC 9126-1. Software engineering-Product quality-Part 1: Quality model.
4 IEEE 1471. Recommended Practice for Architectural Description of Software-Intensive Systems.
5 ISO/IEC/IEEE 42010. Systems and software engineering-Architecture description.
6 IEEE Std 830. IEEE Recommended Practice for Software Requirements Specifications.
7 IEEE 1016. IEEE Recommended Practice for Software Design Descriptions.

Ресурси інтернет

1 Retrieved from https://refactoring.guru/ua/design-patterns.
2 Dr. Winston W. Rovce. Managing the development of large software systems. Retrieved from http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf.
3 Microsoft Solutions Framework (MSF) Team Model. Retrieved from https://www.microsoft.com/en-us/download/details.aspx?id=3214.
4 Manifesto for Agile Software Development. Retrieved from http://agilemanifesto.org/
5 UML modeling tools for Business, Software, Systems and Architecture. Retrieved from https://www.sparxsystems.com/
6 Retrieved from https://www.smart-it.com/ru/2021/08/12-best-software-development-methodologies-with-pros-and-cons/

ACADEMIC INTEGRITY

The student must adhere to the Code of Ethics of Academic Relations and Integrity of NTU "KhPI": to show discipline, politeness, friendliness, honesty, responsibility

The content of this syllabus is consistent with the course program.