АНАЛІЗ ВИМОГ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тип дисципліни	Обов'язкова
Рівень вищої освіти	Перший(бакалаврський)
Мова викладання	Українська, Англійська
Семестр	Сьомий
Обсяг кредитів ЄКТС	7
Форма здобуття освіти	Очна(денна)

Результати навчання. Відповідно до Стандарту вищої освіти та освітньої програми дисципліна має забезпечити ***компетентності***: здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення; здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням та стандартами; здатність дотримуватись специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу; здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя; здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення; здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення

програмні результати навчання: аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки; знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення; знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення; знати підходи щодо оцінки та забезпечення якості програмного забезпечення

Зміст навчальної дисципліни. Класифікація програмних систем, вимоги до програмного забезпечення, функціональні та нефункціональні вимоги, валідація, верифікація вимог, технологія RUP, MSF, типи класифікаторів та відношень, проектування , реалізація підсистем, тестування програмних систем

Запланована навчальна діяльність: лекцій 34 год., лабораторних занять 51 год., самостійної роботи 125 год.; разом 150 год.

Методи навчання: методи проблемного викладання, словесні, наочні (лекції); пояснювально-ілюстративні, проблемного викладання, дослідницькі, частково-пошукові (лабораторні заняття), проблемного викладання, дослідницькі, частково-пошукові (самостійна робота: індивідуальні завдання).

Форми і методи оцінювання результатів навчання: усне опитування, захист лабораторних робіт. письмові самостійні та контрольні роботи

Вид семестрового контролю: іспит

Навчальні ресурси:

1. Роберт Мартін. Чистий код: Чистий код: створення, аналіз, рефакторинг. - Фабула. 2019 – 416 с.
2. Постіл. С.Д. UML. Уніфікована мова моделювання інформаційних систем: Навч. посіб., 2019. - 321 с.
3. Martin Fowler. Refactoring: Improving the Design of Existing Code (WebEdition), 2nd Edition/ WebEdition.- 2018
4. Aho, Alfred V. фnd Jeffrey D. Ullman. The Theory of Parsing, Translation, and Compiling (Volume 2: Compiling). -Prentice-Hall. -2018
5. Ian Sommerville. Software Engineering, 10th edition. Published by Pearson . July 14th 2021 – 816 p.
6. Модульне середовище для навчання MOODLE. Доступ до ресурсу: https://msn.khnu.km.ua.

Викладач: кандидат педагогічних наук, доцент Онишко О.Г.

SOFTWARE REQUIREMENT ANALYSIS AND QUALITY

Type of discipline	Compulsory
Level of higher education	First (Bachelor's)
Language of Instruction	English
Semester	7
ECTS Credits	7
Course study mode	Full-time (Daytime)

Learning outcomes. According to the Standard of higher education and the educational program, the discipline must ensure:

competencies: the ability to identify, classify and formulate software requirements; 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 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; the 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

program learning outcomes: to analyze, purposefully search for and choose information and reference resources and knowledge necessary for solving professional tasks, taking into account modern achievements of science and technology; know the main processes, phases and iterations of the software life cycle; know and be able to use methods and means of gathering, formulating and analyzing software requirements; know approaches to evaluation and quality assurance of software

Content of the academic discipline. Classification of software systems, software requirements, functional and non-functional requirements, validation , verification of requirements, RUP technology , MSF , types of classifiers and relations, design, implementation of subsystems, testing of software systems

Planned educational activities: 34 hours of lectures, 17 hours of laboratory classes, 99 hours of independent work; together 150 hours

Teaching methods: methods of problem-based teaching, verbal, visual (lectures); explanatory and illustrative, problem-based teaching, research, partially research-based (laboratory classes), problem-based teaching, research, partially research-based (independent work: individual tasks).

Forms and methods of evaluation of learning results : oral survey, defense of laboratory works. written independent and control works, written exam

Type of semester control : exam - 3rd semester.

Educational resources:

1. Robert Martin. Clean Code: Clean Code: Build, Analyze, Refactor . - A story. 2019 – 416 p.
2. The bed S.D. UML. Unified language of modeling of information systems: Teaching . manual , 2019. - 321 p.
3. Martin Fowler . Refactoring : Improving the Design of Existing Code (WebEdition), 2nd Edition / WebEdition .- 2018
4. Aho , Alfred V. fnd Jeffrey D. Ullman . The Theory of Parsing , Translation , and Compiling (Volume 2: Compiling). -Prentice-Hall. -2018
5. Ian Sommerville . Software Engineering , 10th edition . Published by Pearson . July 14th 2021 - 816
6. MOODLE modular learning environment. Access to the resource: https://msn.khnu.km.ua .

Lecturer: Associate Professor, PhD Onyshko O.H.