ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ЗАТВЕРДЖУЮ
Декан факультету
інформаційних технологій
_____ Олег САВЕНКО
_____ 2023 р.

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Аналіз вимог та якість програмного забезпечення

Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Рівень вищої освіти – Перший бакалаврський
Освітньо-професійна програма – Інженерія програмного забезпечення
Обсяг дисципліни – 7кредитів ЄКТС, ***Шифр дисципліни*** – ОПП.09
Статус дисципліни: обов'язкова, ***Мова навчання*** Англійська, українська
Факультет – Інформаційних технологій
Кафедра – Інженерії програмного забезпечення

Форма навчання	Курс	Семестр	Обсяг дисциплін и	Кількість годин						Курсовий проект	Курсова робота	Форма семестрового контролю	
				Аудиторні заняття					Самостійна робота, у т.ч. ІРС				
			Кредити ЄКТС	Разом	Лекції	Лабораторні роботи	Практичні заняття					Залік	Іспит
Очна (денна	2	3	7	210	34	51			125				+
Всього			**7**	**210**	**34**	**51**			**125**				**1**

Робоча програма складена на основі Стандарту вищої освіти, освітньо-професійної програми підготовки бакалаврів 2023 року та навчального плану

Програма складена _____ Оксана ОНИШКО
Схвалено на засіданні кафедри ІПЗ
протокол № 1 від 31.08.2023

Зав. кафедри інженерії програмного забезпечення _____ Леонід БЕДРАТЮК

Робоча програма розглянута та схвалена Вченою радою факультету інформаційних технологій

Голова Вченої ради _____ Олег САВЕНКО
Підпис

Хмельницький 2023

АНАЛІЗ ВИМОГ ТА ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

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

Результати навчання. Студент, який успішно завершив вивчення дисципліни, має: знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення. Знати і застосовувати професійні стандарти і інші нормативно-правові документи в галузі інженерії програмного забезпечення. Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення. Вибирати вихідні дані для проєктування, керуючись формальними методами опису вимог та моделювання. Знати підходи щодо оцінки та забезпечення якості програмного забезпечення.

Пререквізити – Основи інженерії програмного забезпечення
Кореквізити – Моделювання та оцінка програмного забезпечення, Організація комп'ютерних мереж

Зміст навчальної дисципліни. Класифікація програмних систем. Вимоги до програмного забезпечення. Вплив аналізу вимог по ПЗ на ітеративні цикли розробки проекту. Концепції проектування програмного забезпечення. Розробка вимог. Нормативи та стандарти. Верифікація та тестування програмного забезпечення.**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. тестування, письмовий іспит

Вид семестрового контролю: іспит.

Навчальні ресурси:

1. Laporte, C. Y., April, A. (2018). Software Quality Assurance. Wiley. – 720p
2. Постіл. С.Д. UML. Уніфікована мова моделювання інформаційних систем: Навч. посіб., 2019. - 321 с.
3. Galin, D. (2018). Software Quality: Concepts and Practice. Wiley. – 720p
4. Smith, H. T. G. (2020). Software Quality Assurance: A Guide for Developers and Auditors. CRC Press. – 480 p.
5. Chopra, R. (2018). Software Quality Assurance: A Self-Teaching Introduction. Mercury Learning and Information. – 660 p.
6. Ian Sommerville. (2021) Software Engineering, 10th edition. Published by Pearson – 816 p.
7. Модульне середовище для навчання. Доступ до ресурсу: https://msn.khmnu.edu.ua.

Викладач: кандидат педагогічних наук, доцент Оксана ОНИШКО

3. ПОЯСНЮВАЛЬНА ЗАПИСКА

Дисципліна “Аналіз вимог та якість до програмного забезпечення” є однією із дисциплін професійної підготовки і займає провідне місце у підготовці фахівців освітнього рівня "бакалавр” за освітньо-професійною програмою "Інженерія програмного забезпечення".

Пререквізити – Основи інженерії програмного забезпечення
Кореквізити – Моделювання та оцінка програмного забезпечення, Організація комп’ютерних мереж

Мета дисципліни – полягає в забезпеченні теоретичної та практичної підготовки студентів, що має забезпечити отримання студентами основних знань у галузі сучасних технологій проектування, інженерії вимог до програмного забезпечення, отримання ними практичних навичок реалізації програмних систем, основ моделювання і аналізу програмних систем, аналізу розробки, специфікації та управління вимогами

Предмет дисципліни. Предмет дисципліни «Аналіз вимог до програмного забезпечення» охоплює теоретичні знання, задачі, методи та вимоги до програмного забезпечення, процесів його проектування та конструювання.

Завдання дисципліни: Основними завданнями вивчення дисципліни «Аналіз вимог до програмного забезпечення» є знання про розробку та аналіз вимог, які висуваються до програмного продукту. Проводиться класифікація вимог, аналізуються властивості вимог, розглядаються методології, стандарти, нотації роботи з вимогами. Аналізуються складові аналізу вимог: виявлення, специфікація та документування, верифікація. Розглядається роль моделей, інструментальних засобів, процесів керування вимогами

Відповідно до ***Стандарту вищої освіти*** із та освітньої програми дисципліна сприяє забезпеченню:

компетентностей:

ФК1. Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення. ФК4. Здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням, стандартами. ФК5. Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу. ФК10. Здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя ФК12. Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення.

програмних результатів навчання:

ПРН3. Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення. ПРН4 Знати і застосовувати професійні стандарти і інші нормативно-правові документи в галузі інженерії програмного забезпечення. ПРН9. Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення. ПРН11 Вибирати вихідні дані для проєктування, керуючись формальними методами опису вимог та моделювання. ПРН20. Знати підходи щодо оцінки та забезпечення якості програмного забезпечення.

Результати навчання. Студент, який успішно завершив вивчення дисципліни, має: знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення. Знати і застосовувати професійні стандарти і інші нормативно-правові документи в галузі інженерії програмного забезпечення. Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення. Вибирати вихідні дані для

проєктування, керуючись формальними методами опису вимог та моделювання. Знати підходи щодо оцінки та забезпечення якості програмного забезпечення.

Політика дисципліни Організація освітнього процесу з дисципліни відповідає вимогам положень про організаційне і навчально-методичне забезпечення освітнього процесу, освітній програмі та навчальному плану. Студент зобов'язаний відвідувати лекції, практичні заняття, лабораторні роботи, тощо, згідно з розкладом, не запізнюватися на заняття, виконувати усі завдання та контрольні точки відповідно до графіка. Пропущені практичні заняття і лабораторні роботи студент зобов'язаний опрацювати самостійно у повному обсязі і відзвітувати перед викладачем не пізніше, ніж за тиждень до чергової атестації. До практичних занять і лабораторних робіт студент має підготуватися за відповідною темою і проявляти активність. Набутті особою знання з дисципліни або її окремих розділів у неформальній освіті зараховуються відповідно до Положення про порядок перезарахування результатів навчання та визначення академічної різниці у ХНУ.

4. Структура залікових кредитів дисципліни

Теми	Лекції	Лаб. роб.	Самостійна робота здобувачів
Тема 1.Вступ. Класифікація програмних систем	2	8	21
Тема 2. Вимоги до програмного забезпечення	6	8	21
Тема 3. Вплив аналізу вимог по ПЗ на ітеративні цикли розробки проекту.	4	8	21
Тема 4. Концепції проектування програмного забезпечення	14	8	21
Тема 5. . Розробка вимог. Нормативи та стандарти. ЕСПД	6	8	21
Тема 6. Верифікація та тестування програмного забезпечення	2	11	20
Разом за 7-й семестр	**34**	**51**	**125**

5. ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

5.1. Зміст лекційного курсу*

№ лекції	Перелік тем лекцій, їх анотація	Кількість годин
Тема 1. Вступ. Види програмних систем		2
1	**Лекція 1.** Етапи розробки ПЗ. Роль вимог у плануванні програмних проектів. Предмет та задачі дисципліни «Аналіз вимог до програмного забезпечення». Класифікація програмних систем. Огляд стандартів та технологій для розробки програмних систем (MRPII, CRM, ERPII та ін.). Методологія структурного аналізу та проектування SADT. Методологія розробки програмного забезпечення RUP. Приклади вдалого і невдалого формування вимог до програмних проектів. Література: [1-3; 7; 11].	2
Тема 2. Вимоги до програмного забезпечення.		4
2	**Лекція2.** Класифікація вимог. Користувацькі вимоги. Методи виявлення та опрацювання вимог у технологіях MSF, RUP, XP. Типи вимог до програмного забезпечення. Класифікація вимог. Джерела вимог. Якість вимог. Користувацькі вимоги. Формалізація вимог. Функціональні та не функціональні вимоги. Методи виявлення та опрацювання вимог: інтерв‘ювання, анкетування, активне і пасивне спостереження, опитування, вивчення технічної документації. Технологій повторного використання проектних рішень. Виявлення вимог за технологіями MSF, RUP, XP. Користувацькі історії, прототипування, сценарії використання, мозковий штурм. Література: [2; 5; 12].	2
3	**Лекція 3**. . Функціональні й нефункціональні вимоги, особливості реалізації у програмних проектах. Функціональні вимоги. Вплив на структуру та реалізацію програмного проекту. Відображення нефукнціональних вимог у структурі програмного проекту. Рекомендації до структури й методів опису програмних вимог – стандарт IEEE 830 (“Recommended Practice for Software Requirements Specifications”). 7 Специфікація вимог до програмного проекту. Типи специфікацій та їх застосування. Програмні засоби для ведення специфікацій та управління вимогами. Література: [3-4; 10].	2

4	**Лекція 4.** Перевірка вимог. Змінення та модифікація вимог до програмного проекту. Моделі розробки програмного забезпечення. Перевірка вимог до програмного продукту: верифікація, валідація та тестування. Моделі розробки програмного забезпечення. Методології сімейства IDEF. Сучасні методології розробки програмного забезпечення SWAT, JAD, RAD. Гнучкі (agile) методології розробки програмного забезпечення. Література: [4; 7; 14].	2
Тема 3. Вплив аналізу вимог по ПЗ на ітеративні цикли розробки проекту.		8
5	**Лекція 5.** Управління вимогами. Управління якістю. Управління рішеннями. Вплив аналізу вимог до ПЗ на ітеративні цикли розробки проекту. Управління вимогами (Requirement Management). Управління якістю розробки ПП. Стандарт ISO 9000. Управління рішеннями. Стандарт COBIT. Бізнес-рішення і управління рішеннями. Бібліотека інфраструктури інформаційних технологій ITIL. Література: [2; 4-6; 9].	2
6	**Лекція 6.** Технологія RUP. Уніфікована мова моделювання UML 2.0. Три групи діаграм моделювання програмних систем. Шаблони проектування. Програмне середовище Rational Rose. Література: [1-3; 7; 11]	2
Тема 4. Концепції проектування програмного забезпечення.		2
7	**Лекція 7.** Технологія RUP. Процес розробки ПЗ. Методи виявлення та документування вимог. Уніфікована мова моделювання UML, історія створення та основні концепції. Три групи діаграм моделювання програмних систем у UML 2.0. Типи класифікаторів та відношень. Література: [4-5; 9-12]	2
8	**Лекція 8.** Засоби автоматизації проектування ПЗ компанії Rational Software. Програмне середовище Rational Rose. Шаблони проектування: абстрактнеконкретне (abstraction-occurrence), компоновщик (composite), гравець-роль (player-role), одинак (singleton), спостерігач (observer), делегування (delegation), фасад (facade), адаптер (adapter). Література: [1; 4; 12].	2
9	**Лекція 9.** Проектування, орієнтоване на користувача. Діаграми поведінки. Аналіз та коректування користувацьких вимог. Користувацькі інтерфейси. Проектування, орієнтоване на користувача. Модель аналізу вимог до програмного забезпечення. Діаграми поведінки при представленні моделей програмної системи: діаграма прецедентів (Use Case), діаграма діяльності (Activity), діаграма автомата (State Machine). Нотації та ідеологія. Діаграми бізнес-прецедентів та прецедентів, побудова, аналіз та деталізація. Мова обмежень OCL. Узгодження користувацьких вимог. Користувацькі інтерфейси. Клієнтський досвід. Стабілізація користувацьких вимог. Література: [3-4; 6; 13].	2
10	**Лекція 10.** Реалізація підсистем. Діаграми структур. Розробка архітектури програмної системи. Діаграми структур при моделюванні програмних систем: діаграма класів (Class), діаграма об‘єктів (ObjecT), діаграма компонентів (Component), діаграма композитних структур (Composite structure), діаграма розвертування (Deployment), діаграма пакетів (Package). Аналітична та програмна моделі програмної системи. Література: [3; 5; 9].	2
11	**Лекція 11.** Розробка діаграм бізнес-класів та класів. Типи зв‘язків у діаграмі класів. Визначення операцій у класах. Документування операцій. Створення атрибутів та методів. Діаграма об‘єктів. Деталізація при розробці моделі ПЗ.	2

	Розробка архітектури програмної системи: розподілені архітектури, «канали-та-фільтри» (pipe-and-filter), «модель-вигляд-контролер» (model-viewcontrol Література: [1-2; 7; 11].	
12	**Лекція 12.** Принципи розробки програмного забезпечення. Діаграми взаємодії. Загальні принципи розробки програмного забезпечення: декомпозиція (decomposition), незв‘язність (decoupling), зачеплення (cohesion), повторне використання (reuse), можливість повторного використання (reusability), переносимість (portability), тестованість (testability), гнучкість (flexibility). Література: [5-6, 10, 13].	2
13	**Лекція 13.** Діаграми взаємодії при моделюванні програмних систем: діаграми послідовностей (Sequence) та комунікації (Communication), діаграма огляду взаємодії (Interaction Overview) та синхронізації (Timing). Література: [8-9, 13, 15].	2
Тема 5. . Розробка вимог. Нормативи та стандарти. ЕСПД		
14	**Лекція 14.** Етапи розробки вимог до програмних систем. Специфікація вимог. Стандарти і методології проектування бізнес-процесів. Єдина система програмної документації (ЕСПД). Стадії і етапи розробки. Оформлення текстових програмних документів. Технічне завдання. Оформлення експлуатаційних документів. Література: [7-8, 10, 17].	2
15	**Лекція 15.** Технічне завдання на програмний проект. Складові та їх зміст. Технічне завдання, етапи розробки та зміст. Етапи реалізації технічного завдання на стадіях ескізного, технічного та робочого проекту. Керування пріоритетами. Технічне завдання на розробку програмної системи, роль глосарію у стабілізації технічних вимог. Література: [9, 12, 15].	2
Тема 6. Верифікація та тестування програмного забезпечення		
16	**Лекція 16.** Розробка програмного забезпечення через тестування. Впровадження програмної системи. Верифікація та тестування програмного забезпечення. Керування якістю розробки програмних продуктів. Технологія XP. Література: [10-11, 13, 16].	2
17	**Лекція17.** Розробка програмного забезпечення через тестування (test-first programming). Впровадження програмних систем. Документація, що розробляється для впровадження програмного проектуЛітература: [5, 10-11, 15].	2
Разом за 7-й семестр		**34**

5.2 Зміст лабораторних занять

№ з/п	Теми лабораторних занять	Години
1	Аналіз проблеми. Постановка. Робота з реальними замовниками, ідентифікація заінтересованих осіб та інтерв’ю з ними, аналіз отриманого матеріалу, формулювання проблеми, її актуальності і потреб зацікавленості.	**6**

2	Нотація моделювання бізнес-процесів (BPMN). Використання іх в моделюванні та аналізі бізнес- процесів..	**6**
3	Використання UML для опису, візуалізації та документування різних артефактів програмно-інтенсивної системи..	**6**
4	Візуальне представлення етапів бізнес-процесів через техніку блок-схем..	**6**
5	Представлення потоку даних в системі через діаграму потоку даних (DFD) та виявлення потенційних проблем і можливостей в системі.	**6**
6	Виявлення ролей та діяльності зацікавлених сторін в системі через діаграми рольової активності (RAD).	**6**
7	Планування та відслідковування ходу проєкту через діаграми Ганта.	**6**
8	Моделювання та аналіз функціональних вимог за допомогою IDEF діаграм.	**6**
9	Оформлення технічного завдання	**3**
Разом за 7-й семестр		**51**

5.3 Зміст самостійної роботи

Об'єм самостійної роботи з дисципліни **"Аналіз вимог та якість програмного забезпечення"** становить 125 годин. Вони включають опрацювання лекційного матеріалу, теоретичних і лабораторних завдань, підготовку до виконання лабораторних робіт, їх захисту і поточне тестування.

Номер тижня	Назва теми	Години
1	**Тема 1. Вступ. Види програмних систем.** Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №1. Література: [1-5; 7; 11].	17
2	**Тема 2. Вимоги до програмного забезпечення.** Опрацювання лекційного матеріалу, захист лабораторної роботи № 1. Література: [4; 7; 11].	17
3	**Тема 2 Вимоги до програмного забезпечення.** Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №2. Література: [3-4; 10].	**17**
4	**Тема 2. Вимоги до програмного забезпечення** Опрацювання лекційного матеріалу. Захист лабораторної роботи №2. Література: [4; 7; 11].	**17**
5	**Тема 3. Вплив аналізу вимог по ПЗ на ітеративні цикли розробки проекту.** Опрацювання лекційного матеріалу. Підготовка до виконання лабораторної роботи №3. Література: [4; 7; 11].	**17**
6	**Тема 3. Вплив аналізу вимог по ПЗ на ітеративні цикли розробки проекту.**Опрацювання лекційного матеріалу, захист лабораторної роботи №3. Література: [1; 4; 12].	**17**
7	**Тема 4. Концепції проектування програмного забезпечення.** Опрацювання лекційного матеріалу. Підготовка до виконання лабораторної роботи №4. Література: [1; 4; 12].	**17**
8	**Тема 4. Концепції проектування програмного забезпечення.**Опрацювання лекційного матеріалу, захист лабораторної роботи №4. Підготовка до проходження тесту. Література: [5-6; 10].	**17**
9	**Тема 4. Концепції проектування програмного забезпечення.** Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №5.	**17**

	Література: [2-4; 8-9]	
10	**Тема 4. Концепції проектування програмного забезпечення.**Опрацювання лекційного матеріалу, захист лабораторної роботи №5. Література: [4; 6; 13]	**17**
11	**Тема 4. Концепції проектування програмного забезпечення.**Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №6. Література: [1; 3; 6]	**17**
12	**Тема 4 Концепції проектування програмного забезпечення.**Опрацювання лекційного матеріалу, захист лабораторної роботи №6. Література: [2; 13].	**18**
13	**Тема 4. Концепції проектування програмного забезпечення.**Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №7. Література: [4; 7; 13].	**18**
14	**Тема 5. . Розробка вимог. Нормативи та стандарти. ЕСПД** Опрацювання лекційного матеріалу. Захист лабораторної роботи №7. Література: [4; 7; 13].	**18**
15	**Тема 5. . Розробка вимог. Нормативи та стандарти. ЕСПД** Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №8. Література: [2; 5; 11].	**18**
16	**Тема 6. Верифікація та тестування програмного забезпечення** Опрацювання лекційного матеріалу, підготовка до виконання та захист лабораторної роботи №8. Література: [2; 5; 11].	**18**
17	**Тема 6. Верифікація та тестування програмного забезпечення** Опрацювання лекційного матеріалу. Підготовка до іспиту.	**18**
Разом за 7-й семестр		**125**

6. МЕТОДИ НАВЧАННЯ

Процес навчання з дисципліни ґрунтується на використанні традиційних та сучасних методів: методи проблемного викладання, словесні, наочні (лекції); пояснювально-ілюстративні, проблемного викладання, дослідницькі, частково-пошукові (лабораторні заняття), проблемного викладання, дослідницькі, частково-пошукові (самостійна робота: індивідуальні завдання). Всі заняття проводяться з використанням інформаційних технологій і мають за мету – набуття здобувачами першого (бакалаврського) рівня вищої освіти практичних навичок з конструювання програмного забезпечення, тестування та налагодження відладки, використовувати метрики для оцінки розробленого ПЗ.

7. ФОРМИ І МЕТОДИ ОЦІНЮВАННЯ РЕЗУЛЬТАТІВ НАВЧАННЯ

Поточний контроль здійснюється під час лекційних та лабораторних занять. Семестровий контроль проводиться у формі іспиту. При виведенні остаточної оцінки враховуються результати поточного контролю та письмового іспиту.

Процес оцінювання підготовленості здобувача першого (бакалаврського) рівня вищої освіти можна розділити на етапи:

Перший етап оцінювання направлений на визначення знань інформаційного мінімуму. Якщо здобувач твердо засвоїв визначену навчальним планом суму формальних знань, то це означає, що він вміє використати їх при вирішенні різних питань при проектуванні програмних систем, вміє розширити їх.

Перед вивченням дисципліни, як правило, проводиться вхідний контроль знань з дисциплін, що їй передують і забезпечують. При цьому необхідно встановити рівні та критерії сформованості знань щодо змісту навчальних елементів. Такими рівнями є:

Ознайомчо-орієнтовний (ОО) – особа має орієнтовне уявлення щодо понять, які вивчаються, здатна: програмувати основні елементи програмних систем різними мовами програмування, обирати сучасні методології та технології аналізу програмного забезпечення, обґрунтовано використовувати сучасні середовища розроблення програмного забезпечення для розроблення програмних систем.

Понятійно-аналітичний (ПА) – особа має чітке уявлення щодо навчального об'єкту, здатна перенести раніше засвоєнні знання на типові ситуації.

Продуктивно-синтетичний (ПС) – особа має глибоке розуміння щодо навчального об'єкту, здатна здійснювати синтез, генерувати нові ідеї та уявлення, переносити раніше засвоєні знання на нетипові, нестандартні ситуації.

Кожний вид роботи з дисципліни оцінюється за *чотирибальною* шкалою. Семестрова підсумкова оцінка визначається як середньозважена з усіх видів навчальної роботи, виконаних і зданих *позитивно* з врахуванням коефіцієнта вагомості. Вагові коефіцієнти змінюються залежно від структури дисципліни і важливості окремих її видів робіт. Здобувач, який набрав позитивний середньозважений бал за поточну роботу і не здав контрольний захід (іспит), вважається невстигаючим.

При оцінюванні знань здобувачів першого (бакалаврського) рівня вищої освіти використовуються різні засоби контролю, зокрема: усне опитування перед допуском до виконання лабораторної роботи – здійснюється на її початку; засвоєння теоретичного матеріалу з тем перевіряється тестовим контролем; якість виконання, набуття теоретичних знань і практичних навичок перевіряється шляхом захисту кожної лабораторної роботи згідно з робочою програмою дисципліни і робочим навчальним планом.

Оцінка, яка виставляється за *лабораторне заняття*, складається з таких елементів: усне опитування здобувачів перед допуском до виконання лабораторної роботи; знання теоретичного матеріалу з теми; якість оформлення протоколу і графічної частини; вміння здобувача обґрунтувати прийняті конструктивні рішення; своєчасний захист лабораторної роботи. Для виконання програми дисципліни здобувач повинен отримати 7 оцінок за лабораторні роботи.

Термін захисту лабораторної роботи вважається своєчасним, якщо здобувач захистив її на наступному після виконання роботи занятті.

Пропущене лабораторне заняття здобувач повинен відпрацювати не пізніше, ніж за два тижні до кінця теоретичних занять у семестрі.

При *оцінюванні знань* здобувачів першого (бакалаврського) рівня вищої освіти викладач керується такими критеріями.

Оцінку «відмінно» отримує здобувач за глибоке і повне опанування змісту навчального матеріалу, в якому він легко орієнтується, понятійного апарату, за уміння зв’язувати теорію з практикою, вирішувати практичні завдання, висловлювати і обґрунтовувати свої судження. Відмінна оцінка передбачає грамотний, логічний виклад відповіді (як в усній, так і в письмовій формі), якісне зовнішнє оформлення. Здобувач повинен набути практичних навичок із проектування та програмної реалізації програмних систем.

Оцінка «відмінно» виставляється здобувачу, який глибоко засвоїв основні принципи проектування програмних систем та вміє їх раціонально застосувати, знає методики та вміє ними користуватися при розробленні програмного забезпечення. Здобувач не повинен вагатися при видозміні запитання, повинен робити детальні та узагальнюючі висновки.

Оцінку «добре» отримує здобувач за повне засвоєння навчального матеріалу, володіння понятійним апаратом, орієнтування у вивченому матеріалі, свідоме використання знань для вирішення практичних завдань, грамотний виклад відповіді, але у змісті і формі відповіді мали місце окремі неточності (похибки), нечіткі формулювання закономірностей тощо. Відповідь здобувача повинна будуватись на основі самостійного мислення.

Оцінку «добре» отримує здобувач за правильну відповідь з однією-двома суттєвими помилками.

Оцінки «задовільно» заслуговує здобувач, який виявив знання основного навчально-програмного матеріалу в обсязі, необхідному для подальшого навчання та практичної діяльності за професією, що справляється з виконанням практичних завдань, передбачених програмою. Як правило, відповідь здобувача будується на рівні репродуктивного мислення, здобувач слабо знає структуру курсу, допускає помилки у відповіді, засвоїв і набув практичних навичок у проектуванні та реалізації програмних систем, але припустився неточностей. Вагається при відповіді на видозмінене запитання, разом з тим здобувач володіє знаннями, що дозволяють йому під керівництвом викладача усунути неточності у відповіді.

Оцінки «задовільно» заслуговує здобувач за неповне опанування програмного матеріалу, але отримані знання і набуті практичні навички із проектування та розроблення програмного забезпечення.

Оцінка «незадовільно» виставляється, коли здобувач має розрізнені, безсистемні знання, не вміє виділяти головне і другорядне, допускається помилок у визначенні понять, перекручує їх зміст,

хаотично і невпевнено викладає матеріал, не може використовувати знання при вирішенні практичних завдань. Як правило, оцінка "незадовільно" виставляється здобувачу, який не може продовжити навчання без додаткових знань з курсу.

На основі результатів поточного контролю і іспиту виставляється підсумкова семестрова оцінка.

Залік виставляється при отриманні здобувачем з дисципліни від 3,00 до 5,00 балів. При цьому за вітчизняною шкалою ставиться «зараховано», а за шкалою ECTS – набрана здобувачем кількість балів та відповідна їй оцінка.

При викладанні дисципліни використовуються такі види навчальних занять, як лекції, лабораторні роботи, індивідуальне консультування і керівництво самостійною роботою здобувача, в т.ч. за індивідуальним завданням.

При оцінюванні знань здобувачів використовуються різні засоби контролю, зокрема: допуск до виконання лабораторної роботи здійснюється на її початку усним опитуванням кожного здобувача; засвоєння теоретичного матеріалу змістових модулів перевіряється змістовим контролем; якість виконання, набуття теоретичних знань і практичних навичок перевіряється шляхом захисту кожної лабораторної роботи та індивідуального завдання згідно з робочим планом.

Оцінка, яка виставляється за лабораторне заняття, складається з таких елементів: усне опитування здобувачів перед допуском до виконання лабораторної роботи; знання теоретичного матеріалу з теми; якість оформлення протоколу і графічної частини; вміння здобувача обґрунтувати прийняті конструктивні рішення; своєчасний захист лабораторної роботи.

Термін захисту лабораторної роботи вважається своєчасним, якщо здобувач захистив її на наступному після виконання роботи занятті.

Пропущене з поважної причини лабораторне заняття здобувач повинен відпрацювати в лабораторіях кафедри у встановлений викладачем термін.

Система поточного контролю знань, умінь, навичок

Формою поточного контролю, що проводиться на кожному лабораторному занятті, є коротке усне та письмове опитування викладеного лекційного матеріалу, а також письмове виконання прикладів розв’язування задач. Формою підсумкового контролю є письмова контрольна робота, яка включає одне питання з переліку питань для підсумкового контролю і задачу.

Структурування дисципліни за видами робіт і оцінювання результатів навчання здобувачів у семестрі за ваговими коефіцієнтами

<table>
<tr><td colspan="8">Аудиторна робота</td><td rowspan="2">Семестровий контроль, іспит (І)</td><td rowspan="2">Підсумковий бал</td></tr>
<tr><td colspan="7">Лабораторні роботи (ЛР)</td><td>Тест</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>Т</td><td rowspan="2">0,4</td><td rowspan="2">ЛР*0,3+Т*0,3+І*0,4</td></tr>
<tr><td colspan="8">ВК = 0,3</td><td>ВК = 0,3</td></tr>
</table>

Умовні позначення: ВК – ваговий коефіцієнт, ЛР – лабораторна робота, Т – тест, І – іспит.

Оцінювання тестових завдань. Тематичний тест для кожного здобувача складається з двадцяти тестових завдань, кожне з яких оцінюється одним балом. Максимальна сума балів, яку може набрати здобувач, складає 20.

Оцінювання здійснюється за чотирибальною шкалою.

Відповідність набраних балів за тестове завдання оцінці, що виставляється здобувачу, представлена у нижченаведеній таблиці.

Сума балів за тестове завдання	1–11	12–14	15–18	19-20
Оцінка	2	3	4	5

На тестування відводиться 30 хвилин. Тестування проводиться з використанням модульного середовища для навчання MOODLE. Правильні відповіді здобувач реєструє в он-лайн режимі в модульному середовищі MOODLE. Через 30 хвилин здобувачі завершують тестування та надсилають свої відповіді на сервер. Викладач оголошує результати тестування згідно журналу оцінок модульного середовища MOODLE.

Якщо здобувач отримав негативну оцінку, то він має перездати її в установленому порядку, але обов'язково до терміну наступного контролю.

Підсумкова семестрова оцінка за національною шкалою і шкалою ЄКТС встановлюється в автоматизованому режимі після внесення усіх оцінок до електронного журналу. Співвідношення вітчизняної шкали оцінювання і шкали оцінювання ЄКТС наведені у наступній таблиці.

Співвідношення вітчизняної шкали оцінювання і шкали оцінювання ЄКТС

Оцінка ЄКТС	Інституційна інтервальна шкала балів	Вітчизняна оцінка, критерії		
A	4,75–5,00	5	***Відмінно*** – глибоке і повне опанування навчального матеріалу і виявлення відповідних умінь та навиків	Зараховано
B	4,25–4,74	4	***Добре*** – повне знання навчального матеріалу з кількома незначними помилками	
C	3,75–4,24	4	***Добре*** – в загальному правильна відповідь з двома-трьома суттєвими помилками	
D	3,25–3,74	3	***Задовільно*** – неповне опанування програмного матеріалу, але достатнє для практичної діяльності за професією	
E	3,00–3,24	3	***Задовільно*** – неповне опанування програмного матеріалу, що задовольняє мінімальні критерії оцінювання	
FX	2,00–2,99	2	***Незадовільно*** – безсистемність одержаних знань і неможливість продовжити навчання без додаткових знань з дисципліни	Незараховано
F	0,00–1,99	2	***Незадовільно*** – необхідна серйозна подальша робота і повторне вивчення дисципліни	

Залік виставляється, якщо середньозважений бал, який отримав студент з дисципліни, знаходиться у межах від 3,00 до 5,00 балів. При цьому за вітчизняною шкалою ставиться оцінка «зараховано», а за шкалою ЄКТС – буквене позначення оцінки, що відповідає набраній студентом кількості балів відповідно до таблиці Співвідношення.

8. ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Дайте означення програмного забезпечення
2. Дайте означення життєвого циклу
3. Назвіть основні процеси та етапи життєвого циклу
4. Яка мета розробки програмного забезпечення.
5. На які типи за успішністю поділяються проекти за термінологією The Standish Group
6. Назвіть основні причини провалів та успіхів проектів за The Standish Group
7. Дайте означення рівням вимог до програмного забезпечення.
8. Які рівні вимог ви знаєте?
9. Які етапи включає розробка вимог
10. Які дії відносяться до управління вимог?
11. Назвіть ризики при розробці програмного забезпечення
12. Які є переваги високоякісного процесу розробки вимог?.
13. Назвіть характеристики положень специфікації вимог
14. Які ви знаєте джерела вимог?
15. Охарактеризуйте стратегії вимог?
16. Які є способи представлення вимог?
17. Дайте характеристику акторам та варіантам використання вимог.
18. Назвіть основні стилі опису специфікацій варіантів використання
19. З чого складається шаблон повного опису варіанту використання по А.Коберну
20. Охарактеризуйте розділи шаблону варіанту використання RUP
21. Специфікація нефункціональних вимог
22. Які моделі UML пояснюють функціональність системи?

23. Опишіть діаграму варіантів використання
24. Які основні компоненти діаграми дій?
25. Коли використовують діаграму станів?
26. Які діаграми UML пояснюють внутрішній устрій системи?
27. Які характеристики продукту відносяться до атрибутів якості програмного забезпечення?
28. Що розуміється під управлінням вимогами до програмного забезпечення?
29. Які є базові версії вимог?
30. Які є техніки управління вимогами?
31. Назвіть атрибути вимог до програмного забезпечення
32. Дайте означення якості програмного забезпечення
33. Дайте означення тестування для програмного забезпечення
34. Які є види тестування?
35. Які види тестів ви знаєте?
36. Дайте характеристику комбінованим методам тестування
37. Що означає верифікація програм?
38. Дайте характеристику гнучкому розробленню програмного забезпечення на основі Agile.
39. Які ви знаєте стандарти , що регламентують процес розробки програмного забезпечення?
40. Дайте характеристику міжнародним стандартам ISO

9. МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ

Освітній процес з дисципліни забезпечений необхідними навчально-методичними розробками в модульному середовищі

10. РЕКОМЕНДОВАНА ЛІТЕРАТУРА

ОСНОВНА:

1. Laporte, C. Y., April, A. (2018). Software Quality Assurance. Wiley. – 720p
2. Постіл. С.Д. UML. Уніфікована мова моделювання інформаційних систем: Навч. посіб., 2019. - 321 с.
3. Galin, D. (2018). Software Quality: Concepts and Practice. Wiley. – 720p
4. Smith, H. T. G. (2020). Software Quality Assurance: A Guide for Developers and Auditors. CRC Press. – 480 p.
5. Chopra, R. (2018). Software Quality Assurance: A Self-Teaching Introduction. Mercury Learning and Information. – 660 p.
6. Ian Sommerville. (2021) Software Engineering, 10th edition. Published by Pearson – 816 p.
7. Laplante, P. A., Kassab, M. H. (2022). Requirements Engineering for Software and Systems. : CRC Press. – 428 p

ДОПОМІЖНА:

8. Shyam R Chidamber,Chris F Kemerer (Author), Sloan School of Management. A Metrics Suite for Object Oriented Design. Franklin Classics. - 2018 – 44 p.
9. Heath, F. (2020). Managing Software Requirements the Agile Way: Bridge the Gap Between Software Requirements and Executable Specifications to Deliver Successful Projects. Packt Publishing.- 417p.
10. Beck Kent. Extreme Programming Explained: EmbraceChange (The XP Series). Addison-WesleyProfessional. – 2004 - 224
11. Dave Nicolette. Software Development Metrics 1st Edition - Manning; 1st edition – 2015 – 192 p.
12. Eric J. Braude; Michael E. Bernstein. Software Engineering. Waveland Press, 2016 – 802 p. Blokdyk, G. (2020). Software Requirements Specification a Complete Guide - 2020 Edition. (n.p.): Emereo Pty Limited. – 312 p.
13. Rath, A. K., Mohapatra, H. (2020). Fundamentals of Software Engineering Designed to Provide an Insight Into the Software Engineering Concepts. – 816 p.
14. Requirements Engineering: Foundation for Software Quality: 26th International Working Conference, REFSQ 2020, Pisa, Italy, March 24–27, 2020, Proceedings. (2020). Springer International Publishing. – 308 p

15. Mahfuz, A. (2020). Software Quality Management: Reducing Defect Through Root Cause Analysis. (n.p.): Amazon Digital Services LLC - Kdp. – 200 p
16. Роберт Мартін. Чистий код: Чистий код: створення, аналіз, рефакторинг.- Фабула.2019 – 416 с

11. ІНФОРМАЦІЙНІ РЕСУРСИ

1. Модульне середовище для навчання. Доступ до ресурсу: https://msn.khmnu.edu.ua.

2. Електронна бібліотека університету. Доступ до ресурсу: http://library.khmnu.edu.ua

3. Репозитарій ХНУ:https://elar.khmnu.edu.ua/home

KHMELNYTSKYI NATIONAL UNIVERSITY

APPROVED
Dean of the Faculty
_____ Savenko O.S.
"__" _____ 2023.

COURSE PROGRAM
Software Requirement Analysis and Quality

Field of study: 12 - Information Technologies
Major: 121 – Software Engineering
Level of Higher Education: First Level (Bachelor)
Educational program: Software Engineering
Discipline status: Compulsory
Faculty: Information Technologies
Department: Software Engineering

Study mode	Year	Semester	Total Credits	Number of hours						Course project	Coursework	Semester control form	
				Classwork hours				Seminar classes	Independent work, including individual			pass/ fail test	Exam
			ECTS credits	Total	Lectures	Laboratory works	Practical classes						
Full-time (Daytime)	2	3	7	210	34	51			125				+
Total			**7**	**210**	**34**	**51**			**125**				**1**

The course program is based on the Higher Education Standard, the 2023 Bachelor's degree educational program, and the curriculum.

Program's author _____ O. Onyshko

Approved at the staff meeting of the Software Engineering Department

Minutes from 31.08.2023 No. 1

Head of the Software Engineering Department _____ L. Bedratyuk

The course program is approved by the Academic Board of the Faculty of Information technologies

Head of the Academic Board _____ O.S. Savenko

Khmelnytskyi 2023

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. Laporte, C. Y., April, A. (2018). Software Quality Assurance. Wiley. – 720p
2. Постіл. С.Д. UML. Уніфікована мова моделювання інформаційних систем: Навч. посіб., 2019. - 321 с.
3. Galin, D. (2018). Software Quality: Concepts and Practice. Wiley. – 720p
4. Smith, H. T. G. (2020). Software Quality Assurance: A Guide for Developers and Auditors. CRC Press. – 480 p.
5. Chopra, R. (2018). Software Quality Assurance: A Self-Teaching Introduction. Mercury Learning and Information. – 660 p.
6. Ian Sommerville. (2021) Software Engineering, 10th edition. Published by Pearson – 816 p.
7. MOODLE modular learning environment. Access to the resource https://msn.khmnu.edu.ua..

Lecturer: Associate Professor, PhD Onyshko O.H.

3. EXPLANATORY NOTE

The discipline “Software Requirement Analysis and Quality " is a discipline of professional and practical training.

The purpose of discipline - consists in providing theoretical and practical training preparation and students who have _ to provide receiving students basic knowledge in the field modern technologies design , engineering software requirements _ providing , receiving they are practical skills implementation software systems, basics of modeling and analysis software systems, analysis development , specifications and management requirements

Subject of discipline. The subject of the discipline "Analysis of software requirements" covers theoretical knowledge, tasks, methods and requirements for software, its design and construction processes.

Tasks of the discipline: The main ones tasks study disciplines " Analysis software requirements _ provision » is knowledge about development and analysis requirements which _ are advanced to the software product. Classification is carried out requirements are analyzed properties requirements are considered methodologies, standards , notations work with requirements . They are analyzed components analysis requirements : detection , specification and documentation , verification . The role of instrumental models is considered means, processes management requirements

According to the Standard of higher education in the specified specialty and educational program, the discipline must ensure:

competences: PC1. Ability to identify, classify and formulate software requirements. PC4. Ability to formulate and ensure software quality requirements in accordance with customer requirements, specifications, standards. PC5. Ability to adhere to specifications, standards, rules and recommendations in the professional field when implementing life cycle processes. PC10. The ability to accumulate, process and systematize professional knowledge regarding the creation and maintenance of software and the recognition of the importance of lifelong learning PC12. 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: PLO3. Know the main processes, phases and iterations of the software life cycle. PLO4 Know and apply professional standards and other legal documents in the field of software engineering. PLO9. Know and be able to use methods and tools for gathering, formulating and analyzing software requirements. PLO11 Select initial data for design, guided by formal methods of description of requirements and modeling. PLO20. Know approaches to evaluation and quality assurance of software.

Discipline Policy. The organization of the educational process for the discipline complies with the requirements of the provisions on organizational and instructional-methodological support of the educational process, the educational program, and the curriculum. Students are required to attend lectures, practical classes, laboratory work, etc., according to the schedule, not to be late for classes, and to complete all tasks and checkpoints according to the schedule. Missed practical classes and laboratory work must be independently completed by the student in full and reported to the instructor no later than one week before the next assessment. For practical classes and laboratory work, students must prepare on the relevant topic and demonstrate active participation. Knowledge acquired by an individual in the discipline or its specific sections through informal education is credited according to the Regulation on the procedure for transferring learning outcomes and determining academic differences at KhNU.

4. The structure of credit credits of the discipline

Topics	Lectures	Lab . do	Independent work of acquirers
Topic 1. Introduction. Classification of software systems	2	8	21
Topic 2. Software requirements	6	8	21
Topic 3. Impact of software requirements analysis on iterative project development cycles.	4	8	21
Topic 4. Concepts designing software software	14	8	21
Topic 5. development requirements . Norms and standards . ESPD	6	8	21
Topic 6. Verification and testing software software	2	11	20
Together for the 7th semester	**34**	**51**	**125**

5. PROGRAM OF EDUCATIONAL DISCIPLINE

5.1. Content of the lecture course *

No lectures	List of lecture topics, their abstract	Number of hours
Topic 1. Introduction. Types of software systems		2
1	**Lecture 1.** Stages software development . The role of requirements in planning software projects . Subject and tasks disciplines " Analysis software requirements _ security ". Classification software systems. Overview of standards and technologists and for developments software systems (MRPII, CRM, ERPII, etc.). SADT structural analysis and design methodology . Methodology developments software provision of RUP. Examples successful and unsuccessful formation software requirements _ projects . Literature: [1-3; 7; 11].	2
Topic 2. Software requirements provision _		4
2	**Lecture 2.** Classification requirements . Custom requirements . Methods detection and processing requirements in MSF, RUP, XP technologies . Types of software requirements provision _ Classification requirements . Sources requirements . Quality requirements . Custom requirements . Formalization requirements . Functional and non- functional requirements . Methods detection and processing requirements : interview , questionnaire , active and passive observation , survey , study technical documentation . Technologies of reuse design decisions . Detection requirements according to MSF, RUP, XP technologies . Custom stories , prototyping , scenarios use , brainstorming . Literature: [2; 5; 12].	2
3	**Lecture 3** . . Functional and non-functional requirements , features implementation in software projects. Functional requirements . Impact on structure and implementation software project. Reflection non-functional requirements in the structure software project. Recommendations for the structure and methods description software requirements - standard IEEE 830 (" Recommended Practice for Software Requirements Specifications "). 7 Specification requirements for the software project. Types of specifications and their application _ Software driving aids _ specifications and management requirements . Literature: [3-4; 10].	2
4	**Lecture 4.** Audit requirements . Change and modification requirements for the software project. Models developments software provision _ Audit requirements	2

	for the software product: verification , validation and testing . Models developments software provision _ Methodologies IDEF family . Modern methodology developments software providing SWAT, JAD, RAD. Flexible (agile) methodologies developments software provision _ Literature: [4; 7; 14].	
Topic 3. Impact of software requirements analysis on iterative project development cycles.		8
5	**Lecture 5.** Management requirements . Management quality _ Management decisions . Influence analysis software requirements for iterative project development cycles . Management requirements (Requirement Management). Management quality development of PP. ISO 9000 standard. Management decisions . The COBIT standard. Business solutions and management decisions . Library infrastructure informative ITIL technologies . Literature: [2; 4-6; 9].	2
6	**Lecture 6.** RUP technology . Unified language modeling UML 2.0. Three groups diagrams modeling software systems. Templates designing . Software environment Rational Rose. Literature: [1-3; 7; 11]	2
Topic 4. Concepts designing software provision _		2
7	**Lecture 7. RUP** technology . Process software development . Methods detection and documentation requirements . Unified language UML modeling , history creation and main concepts . Three groups diagrams modeling software systems in UML 2.0. Types of classifiers and relations . Literature: [4-5; 9-12]	2
8	**Lecture 8.** Means automation designing the company's software Rational Software . Software environment Rational Rose. Templates design : abstraction - occurrence , composite , player - role , singleton , observer , delegation , facade , adapter . _ _ _ _ _ _ _ _ Literature: [1; 4; 12].	2
9	**Lecture** 9. User - oriented design . Diagrams behavior _ Analysis and correction custom requirements . Custom interfaces . User - centered design . _ Analysis model software requirements _ provision _ Diagrams behavior when presenting software models systems : diagram precedents (Use Case), diagram activity (Activity), diagram of the machine (State Machine). Notations and ideology . Diagrams business precedents and precedents , construction , analysis and detailing . Language OCL restrictions . Harmonization custom requirements . Custom interfaces . Client's experience _ Stabilization custom requirements . Literature: [3-4; 6; 13].	2
10	**Lecture 10.** Realization subsystems . Diagrams of structures. development architecture software systems . Diagrams of structures during modeling software systems: diagram classes (Class), diagram objects (ObjecT), diagram components (Component), diagram composite structures (Composite structure), diagram deployment (Deployment), diagram packages (Package). Analytical and programmatic models software systems . Literature: [3; 5; 9].	2
11	**Lecture 11.** Development diagrams business classes and classes . Types of connections in the diagram classes _ Definition operations in classes . Documentation operations . Creation attributes and methods . Chart objects . Detailing during development software models . development architecture software systems : distributed architecture , "pipe - and - filter " , "model- view -controller" (model-viewcontrol) Literature: [1-2; 7; 11].	2

12	**Lecture 12.** Principles developments software provision _ Diagrams interaction . general principles developments software support : decomposition (decomposition) , decoupling (cohesion) , repeated _ _ reuse , reusability , portability , testability , flexibility . _ _ _ _ _ _ _ _ _ _ _ _ _ Literature: [5-6, 10, 13].	2
13	**Lecture 13.** Diagrams interactions during modeling software systems: diagrams of sequences (Sequence) and communication (Communication), diagram review interactions (Interaction Overview) and synchronization (Timing). Literature: [8-9, 13, 15].	2
Topic 5 . development requirements . Norms and standards . ESPD		
14	**Lecture 14.** Stages developments requirements for software systems. Specification requirements . Standards and methodologies designing business processes . A single software system documentation (ESPD). Stages and stages developments . Design textual software documents . Technical tasks _ Design operational documents . Literature: [7-8, 10, 17].	2
15	**Lecture 15.** Technical tasks for a software project. Components and their content _ Technical tasks , stages development and content . Stages implementation technical tasks in stages sketch , technical and working project. Management priorities . Technical development task _ software systems , the role of the glossary in stabilization technical requirements . Literature: [9, 12, 15].	2
Topic 6. Verification and testing software software		
16	**Lecture 16.** Development software assurance through testing . Implementation software systems . Verification and testing software provision _ Management quality developments software products . XP technology . Literature: [10-11, 13, 16].	2
17	**Lecture 17.** development software provision through testing (test-first programming). Implementation software systems. Documentation that _ is being developed for implementation of the software project Literature: [5, 10-11, 15].	2
Together for the 7th semester		**34**

5.2 Content of laboratory classes

No. z/p	Topics of laboratory classes	hours
1	Analysis of the problem. Setting. Work with real customers, identification of interested persons and interviews with them, analysis of the received material, formulation of the problem, its relevance and interest needs.	**6**
2	Business process modeling notation (BPMN). Their use in modeling and analysis of business processes.	**6**
3	Using UML to describe, visualize, and document various artifacts of a software-intensive system.	**6**
4	Visual representation of the stages of business processes through the technique of block diagrams.	**6**
5	Representing the flow of data in the system through a data flow diagram (DFD) and identifying potential problems and opportunities in the system.	**6**

6	Identification of roles and activities of stakeholders in the system through role activity diagrams (RAD) .	**6**
7	Planning and monitoring the progress of the project through Gantt charts .	**6**
8	Modeling and analysis of functional requirements using IDEF diagrams.	**6**
9	Drawing up the technical task	**3**
Together for the 7th semester		**51**

5.3 Content of independent work

The volume of independent work in the discipline **" Analysis of requirements and software quality "** is 125 hours. They include study of lecture material, theoretical and laboratory tasks, preparation for laboratory works, their defense and current testing.

Number of the week	Topic name	hours
1	**Topic 1. Introduction. Types of software systems .** Elaboration of lecture material, preparation for laboratory work No. 1. Literature: [1-5; 7; 11].	17
2	**Topic 2. Software requirements provision _** Development of lecture material , defense of laboratory work No. 1. Literature: [4; 7; 11].	17
3	**Topic 2 Software requirements _ provision _** Elaboration of lecture material, preparation for laboratory work No. 2. Literature: [3-4; 10].	**17**
4	**Topic 2. Software requirements ensuring** Processing of lecture material. Protection of laboratory work #2. Literature: [4; 7; 11].	**17**
5	**Topic 3. Impact of software requirements analysis on iterative project development cycles.** Processing of lecture material. Preparation for laboratory work No. 3. Literature: [4; 7; 11].	**17**
6	**Topic 3. Impact of software requirements analysis on iterative project development cycles.** Elaboration of lecture material, defense of laboratory work #3. Literature: [1; 4; 12].	**17**
7	**Topic 4. Concepts designing software provision _** Processing of lecture material. Preparation for laboratory work No. 4. Literature: [1; 4; 12].	**17**
8	**Topic 4. Concepts designing software provision _** Elaboration of lecture material, defense of laboratory work #4. Preparation for passing the test. Literature: [5-6; 10].	**17**
9	**Topic 4. Concepts designing software provision _** Elaboration of lecture material, preparation for laboratory work No. 5. Literature: [2-4; 8-9]	**17**
10	**Topic 4. Concepts designing software provision _** Elaboration of lecture material, defense of laboratory work No. 5. Literature: [4; 6; 13]	**17**
11	**Topic 4. Concepts designing software provision _** Elaboration of lecture material, preparation for laboratory work No. 6. Literature: [1; 3; 6]	**17**
12	**Topic 4 Concepts designing software provision _** Elaboration of lecture material, defense of laboratory work #6. Literature: [2; 13].	**18**
13	**Topic 4. Concepts designing software provision _** Elaboration of lecture material, preparation for laboratory work No. 7. Literature: [4; 7; 13].	**18**
14	**Topic 5 . development requirements . Norms and standards . ESPD** Processing of lecture material. Protection of laboratory work #7. Literature: [4; 7; 13].	**18**

15	**Topic 5 . development requirements . Norms and standards . ESPD** Processing of lecture material, preparation for laboratory work No. 8. Literature: [2; 5; 11].	**18**
16	**Topic 6. Verification and testing software ensuring** processing of lecture material, preparation for performance and defense of laboratory work #8. Literature: [2; 5; 11].	**18**
17	**Topic 6. Verification and testing software ensuring** Processing of lecture material. Preparation for the exam.	**18**
Together for the 7th semester		**125**

6. TEACHING METHODS

The teaching process in the discipline is based on the use of traditional and modern 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). All classes are held using information technologies and have the goal of acquiring practical skills in software design, testing and debugging by students of the first (bachelor) level of higher education , using metrics to evaluate the developed software.

7. ASSESSMENT FORMS AND METHODS

Current control is carried out during lectures and laboratory classes. Semester control is conducted in the form of an exam. When deriving the final grade, the results of the current control and the written exam are taken into account.

The process of assessing the preparedness of the applicant of the first (bachelor) level of higher education can be divided into stages:

The first stage of assessment is aimed at determining the knowledge of the information minimum. If the student has firmly mastered the amount of formal knowledge determined by the curriculum, it means that he knows how to use it in solving various issues in the design of software systems, knows how to expand it.

Before studying a discipline, as a rule, there is an input control of knowledge from the disciplines that precede and provide it. At the same time, it is necessary to establish the levels and criteria of the formation of knowledge regarding the content of educational elements. These levels are:

Knowledgeable (OO) - a person has a rough understanding of the concepts being studied, is able to: program the main elements of software systems in various programming languages, choose modern software analysis methodologies and technologies , reasonably use modern software development environments for the development of software systems.

Conceptual-analytical (PA) - a person has a clear idea about the educational object, is able to transfer previously acquired knowledge to typical situations.

Productive-synthetic (PS) - a person has a deep understanding of the educational object, is able to carry out synthesis, generate new ideas and ideas, transfer previously acquired knowledge to atypical, non-standard situations.

Each type of work in the discipline is evaluated on *a four-point* scale. The semester final grade is defined as a weighted average of all types of academic work completed and passed *positively* , taking into account the weighting factor. The weighting factors change depending on the structure of the discipline and the importance of its individual types of work. An applicant who scored a positive weighted average score for the current work and did not pass the control measure (exam) is considered to have failed.

When assessing the knowledge of first (bachelor) level higher education students, various means of control are used, in particular: an oral survey before admission to the performance of laboratory work - it is carried out at the beginning of it; assimilation of theoretical material from topics is checked by test control; the quality of performance, acquisition of theoretical knowledge and practical skills is checked by defending each laboratory work in accordance with the work program of the discipline and the work curriculum.

The grade given for *the laboratory session* consists of the following elements: an oral interview of the test takers before admission to the laboratory work; knowledge of theoretical material on the topic; the quality of the design of the protocol and the graphic part; the acquirer's ability to justify the adopted constructive

decisions; timely protection of laboratory work. To complete the discipline program, the applicant must obtain 7 grades for laboratory work.

The deadline for the defense of laboratory work is considered timely if the applicant defended it at the next class after the completion of the work.

The student must complete the missed laboratory class no later than two weeks before the end of theoretical classes in the semester.

When *evaluating the knowledge* of first (bachelor) level higher education students, the teacher is guided by the following criteria.

The student receives an "excellent" grade for deep and complete mastery of the content of the educational material, in which he can easily navigate, conceptual apparatus, for the ability to connect theory with practice, solve practical tasks, express and justify his judgments. An excellent assessment implies a competent, logical presentation of the answer (both orally and in writing), high-quality external design. The applicant must acquire practical skills in the design and software implementation of software systems.

The grade "excellent" is awarded to the applicant who has thoroughly mastered the basic principles of designing software systems and is able to apply them rationally, knows the methods and knows how to use them in the development of software. The applicant should not hesitate when changing the question, should make detailed and general conclusions.

The applicant receives a grade of "good" for complete assimilation of the educational material, mastery of the conceptual apparatus, orientation in the studied material, conscious use of knowledge to solve practical tasks, competent presentation of the answer, but there were some inaccuracies (errors) in the content and form of the answer, unclear formulations of regularities etc. The applicant's answer should be based on independent thinking.

The winner receives a "good" grade for a correct answer with one or two significant errors.

The grade "satisfactory" is deserved by the applicant who has demonstrated knowledge of the main educational and program material in the amount necessary for further training and practical activity in the profession, which copes with the implementation of practical tasks provided for by the program. As a rule, the applicant's answer is built on the level of reproductive thinking, the applicant has little knowledge of the structure of the course, makes mistakes in the answer, learned and acquired practical skills in the design and implementation of software systems, but made inaccuracies. Hesitates when answering a modified question, at the same time, the applicant has knowledge that allows him to eliminate inaccuracies in the answer under the guidance of the teacher.

The winner deserves a "satisfactory" grade for incomplete mastery of software material, but acquired knowledge and acquired practical skills in software design and development.

The grade "unsatisfactory" is assigned when the student has scattered, unsystematic knowledge, does not know how to distinguish the main from the secondary, makes mistakes in defining concepts, distorts their meaning, presents the material chaotically and uncertainly, cannot use knowledge when solving practical tasks. As a rule, an "unsatisfactory" grade is assigned to a student who cannot continue his studies without additional knowledge of the course.

Based on the results of the current control and exam, a final semester grade is issued.

Credit is issued when the student receives from 3.00 to 5.00 points in the discipline. At the same time, according to the national scale, "credited" is given, and according to the ECTS scale, the number of points scored by the student and the corresponding grade.

When teaching the discipline, such types of training are used as lectures, laboratory work, individual counseling and guidance of the applicant's independent work, including by individual task.

When evaluating the knowledge of applicants, various control tools are used, in particular: admission to the performance of laboratory work is carried out at its beginning by an oral interview of each applicant; assimilation of the theoretical material of content modules is checked by content control; the quality of performance, acquisition of theoretical knowledge and practical skills is checked by defending each laboratory work and individual task according to the work plan.

The grade given for the laboratory session consists of the following elements: an oral interview of the test takers before admission to the laboratory work; knowledge of theoretical material on the topic; the quality of the design of the protocol and the graphic part; the acquirer's ability to justify the adopted constructive decisions; timely protection of laboratory work.

The deadline for the defense of laboratory work is considered timely if the applicant defended it at the next class after the completion of the work.

A candidate who missed a laboratory session for a good reason must complete it in the department's

laboratories within the time limit set by the teacher.

System of current control of knowledge, abilities, skills

The form of current control conducted at each laboratory session is a short oral and written survey of the lecture material, as well as written examples of problem solving. The form of final control is a written control work, which includes one question from the list of questions for final control and a task.

Structuring of the discipline by types of work and evaluation of the study results of applicants in the semester by weighting coefficients

Auditory work								Semester control, exam (I)	Final score
Laboratory work (LR)								Test	
1	2	3	4	5	6	7	8	T	0.4	LR*0.3+T*0.3+I*0.4
VC = 0.3								VC = 0.3	

Conventional designations: VC - weighting factor, LR - laboratory work, T - test, I - exam.

Evaluation of test tasks. The thematic test for each applicant consists of twenty test tasks, each of which is evaluated by one point. The maximum amount of points that the winner can score is 20.

Evaluation is carried out on a four-point scale.

Correspondence of the scored points for the test task to the grade given to the applicant is presented in the table below.

The sum of points for the test task	1–11	12–14	15–18	19-20
Rating	2	3	4	5

30 minutes are allotted for testing. Testing is conducted using the MOODLE modular learning environment. The winner registers the correct answers online in the MOODLE modular environment. After 30 minutes, test takers complete the test and send their answers to the server. The teacher announces the test results according to the evaluation log of the MOODLE modular environment.

If the applicant received a negative grade, he must resubmit it in accordance with the established procedure, but necessarily before the next control period.

The final semester grade according to the national scale and the ECTS scale is set in an automated mode after entering all grades into the electronic journal. The ratio of the domestic assessment scale and the ECTS assessment scale is shown in the following table.

Correlation of the domestic evaluation scale and the ECTS evaluation scale

Evaluation of ECTS	Institutional interval scoring scale	Domestic assessment, criteria		
A	4.75–5.00	5	***Excellent*** - deep and complete mastery of the educational material and identification of relevant abilities and skills	Enrolled
B	4.25–4.74	4	***Good*** - complete knowledge of the educational material with a few minor errors	
C	3.75–4.24	4	***Good*** - a generally correct answer with two or three significant errors	
D	3.25–3.74	3	***Satisfactory*** - incomplete mastery of the program material, but sufficient for practical activities in the profession	
E	3.00–3.24	3	***Satisfactory*** - incomplete mastery of the program material that meets the minimum evaluation criteria	
FX	2.00–2.99	2	***Unsatisfactory*** – unsystematic knowledge acquired and the impossibility of continuing education without additional knowledge of the discipline	Not counted

F	0.00–1.99	2	***Unsatisfactory*** - serious further work and re-study of the discipline is necessary	

Credit is given if the weighted average score received by the student in the discipline is between 3.00 and 5.00 points. At the same time, according to the national scale, the grade "credited" is given , and according to the ECTS scale, the letter designation of the grade corresponds to the number of points scored by the student according to the Ratio table.

8. QUESTIONS FOR STUDENTS' SELF-CONTROL

1. Define software
2. Define the life cycle
3. Name the main processes and stages of the life cycle
4. What is the purpose of software development.
5. The Standish Group , what types of projects are divided into by success
6. Name the main reasons for project failures and successes according to The Standish Group
7. Define the levels of software requirements.
8. What levels of requirements do you know?
9. What stages does requirements development include?
10. What actions are related to requirements management?
11. Name the risks in software development
12. What are the benefits of a high-quality requirements development process?
13. Name the characteristics of the provisions of the requirements specification
14. What sources of requirements do you know?
15. Describe the requirements strategies?
16. What are the ways to submit requirements?
17. Describe the actors and use cases of requirements.
18. Name the main styles for describing use case specifications
19. What does the template of a full description of the use case according to A. Coburn consist of?
20. RUP use case template
21. Specification of non-functional requirements
22. What UML models explain the functionality of the system?
23. Describe the use case diagram
24. What are the main components of an action diagram?
25. When is a flow chart used?
26. UML diagrams explain the internals of a system?
27. What product characteristics are software quality attributes?
28. What is software requirements management?
29. What are the base versions of the requirements?
30. What are requirements management techniques?
31. Name the attributes of software requirements
32. Define software quality
33. Define software testing
34. What are the types of testing?
35. What types of tests do you know?
36. Describe combined testing methods
37. What does program verification mean?
38. Describe agile software development based on Agile.
39. What standards do you know that regulate the software development process?
40. Describe ISO international standards

9. TEACHING AND LEARNING MATERIALS

The educational process in the discipline " Software Requirement Analysis and Quality " is fully and sufficiently provided with the necessary educational and methodical literature.

10. RECOMMENDED BOOKS

Main

1. Laporte, C. Y., April, A. (2018). Software Quality Assurance. Wiley. – 720p
2. Постіл. С.Д. UML. Уніфікована мова моделювання інформаційних систем: Навч. посіб., 2019. - 321 с.
3. Galin, D. (2018). Software Quality: Concepts and Practice. Wiley. – 720p
4. Smith, H. T. G. (2020). Software Quality Assurance: A Guide for Developers and Auditors. CRC Press. – 480 p.
5. Chopra, R. (2018). Software Quality Assurance: A Self-Teaching Introduction. Mercury Learning and Information. – 660 p.
6. Ian Sommerville. (2021) Software Engineering, 10th edition. Published by Pearson – 816 p.
7. Laplante, P. A., Kassab, M. H. (2022). Requirements Engineering for Software and Systems. : CRC Press. – 428 p

Auxiliary:

8. Shyam R Chidamber,Chris F Kemerer (Author), Sloan School of Management. A Metrics Suite for Object Oriented Design. Franklin Classics. - 2018 – 44 p.
9. Heath, F. (2020). Managing Software Requirements the Agile Way: Bridge the Gap Between Software Requirements and Executable Specifications to Deliver Successful Projects. Packt Publishing.- 417p.
10. Beck Kent. Extreme Programming Explained: EmbraceChange (The XP Series). Addison-WesleyProfessional. – 2004 - 224
11. Dave Nicolette. Software Development Metrics 1st Edition - Manning; 1st edition – 2015 – 192 p.
12. Eric J. Braude; Michael E. Bernstein. Software Engineering. Waveland Press, 2016 – 802 p.
13. Blokdyk, G. (2020). Software Requirements Specification a Complete Guide - 2020 Edition. (n.p.): Emereo Pty Limited. – 312 p.
14. Rath, A. K., Mohapatra, H. (2020). Fundamentals of Software Engineering Designed to Provide an Insight Into the Software Engineering Concepts. – 816 p.
15. Requirements Engineering: Foundation for Software Quality: 26th International Working Conference, REFSQ 2020, Pisa, Italy, March 24–27, 2020, Proceedings. (2020). Springer International Publishing. – 308 p
16. Mahfuz, A. (2020). Software Quality Management: Reducing Defect Through Root Cause Analysis. (n.p.): Amazon Digital Services LLC - Kdp. – 200 p
17. Роберт Мартін. Чистий код: Чистий код: створення, аналіз, рефакторинг.- Фабула.2019 – 416 с

11. INFORMATION RESOURCES

1. MOODLE Learning Platform. Access to the resource https://msn.khmnu.edu.ua.
2. University Electronic Library. Access to the resource: http://library.khmnu.edu.ua.
3. University Repository. Access to the resource https://elar.khmnu.edu.ua/home.