ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

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

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Моделювання та оцінка 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 –5 кредитів 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 дисципліни	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	3	5	5	150	34	34			82				+
Всього			**5**	**150**	**34**	**34**			**82**				**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 факультету _____ Олег САВЕНКО
Підпис

Хмельницький 2023

МОДЕЛЮВАННЯ ТА ОЦІНКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тип дисципліни	Обов'язкова
Рівень вищої освіти	Перший(бакалаврський)
Мова викладання	Українська, Англійська
Семестр	П'ятий
Обсяг кредитів ЄКТС	5
Форма здобуття освіти	Очна(денна)

Результати навчання. Студент, який успішно завершив вивчення дисципліни, має: Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізу та математичного моделювання для розробки програмного забезпечення. Вміти вибирати та використовувати відповідну задачі методологію створення програмного забезпечення. Проводити передпроєктне обстеження предметної області, системний аналіз об'єкта проєктування. Вибирати вихідні дані для проєктування, керуючись формальними методами опису вимог та моделювання. Застосовувати на практиці ефективні підходи щодо проєктування програмного забезпечення. Застосовувати на практиці інструментальні програмні засоби доменного аналізу проєктування, тестування, візуалізації, вимірювань та документування програмного забезпечення. Вміти застосовувати методи компонентної розробки програмного забезпечення. Знати та вміти застосовувати методи верифікації та валідації програмного забезпечення. Знати підходи щодо оцінки та забезпечення якості програмного забезпечення.

Пререквізити – Основи інженерії програмного забезпечення, Аналіз вимог та якість програмного забезпечення, Алгоритми та структури даних

Кореквізити – Архітектура та проєктування програмного забезпечення, Конструювання програмного забезпечення

Зміст навчальної дисципліни. Загальні підходи до моделювання, Принципи моделювання. Основи структурного моделювання програмного забезпечення, Основи моделювання поведінки програмного забезпечення. Основи моделювання подій. Моделювання архітектури програмного забезпечення.

Запланована навчальна діяльність: лекцій 34 год., лабораторних занять 34 год., самостійної роботи 82 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, письмовий іспит

Вид семестрового контролю: іспит

Навчальні ресурси:

1. Вігерс, Карл І. Джой Бітті (2016) Розробка вимог до програмного забезпечення. - . Retryeved from: http://www.twirpx.com/file/1073169/
2. Табунщик, Т.І., Каплієнко, Г. В., Петрова О.А. (2016) Проектування та моделювання програмного забезпечення сучасних інформаційних систем Запоріжжя: Дике Поле,
3. Введення в програмну інженерію і управління життєвим циклом програмного забезпечення Guide to Software Engineering Base of Knowledge (SWEBOK): Пер. з англ. С.Орлик Retryeved from:sorlik.blogspot.com/
4. Модульне середовище для навчання. Доступ до ресурсу: https://msn.khmnu.edu.ua.
5. Електронна бібліотека університету. Доступ до ресурсу: http://library.khmnu.edu.ua

Викладач: кандидат педагогічних наук, доцент Оксана ОНИШКО

3. ПОЯСНЮВАЛЬНА ЗАПИСКА

Дисципліна “Моделювання та оцінка 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, Аналіз вимог та якість програмного забезпечення, Алгоритми та структури даних
Кореквізити – Архітектура та проєктування програмного забезпечення, Конструювання програмного забезпечення

Мета дисципліни полягає в наданні студентам комплексних теоретичних знань та практичних навичок у сфері моделювання та оцінки програмного забезпечення, зокрема в розробці моделей систем, їх оцінці та оптимізації.
Предмет дисципліни охоплює вивчення основних принципів та методологій моделювання програмного забезпечення, включаючи об'єктне моделювання, використання UML, структурне моделювання, а також практики моделювання архітектури програмних систем.
Завдання дисципліни включають:Освоєння принципів та методик створення моделей програмних систем.Аналіз та використання UML для концептуалізації програмних систем.Моделювання взаємодій, поведінки, архітектури програмного забезпечення. Набуття практичних навичок в реалізації моделей, включаючи оцінку та тестування. Розуміння ролі моделювання в процесах розробки та управління вимогами до програмного продукту.
Курс надає студентам знання та інструменти для глибокого розуміння аспектів створення та аналізу програмного забезпечення, від моделювання вимог до оцінки готових програмних продуктів.

Відповідно до ***Стандарту вищої освіти*** із та освітньої програми дисципліна сприяє забезпеченню:
компетентностей:
ФК1. Здатність ідентифікувати, класифікувати таформулювати вимоги до програмного забезпечення. ФК2. Здатність брати участь у проєктуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його струк-тури, поведінки та процесів функціонування. ФК3. Здатність розробляти архітектури, модулі та компоненти програм-них систем 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. ФК11. Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення. ФК12. Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення.
програмних результатів навчання:
ПРН5. Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об’єктно-орієнтованого аналізу та математичного моделювання для розробки програмного забезпечення. ПРН6 Уміння вибирати та використовувати відповідну задачі методологію створення програмного забезпечення. ПРН10 Проводити передпроєктне обстеження предметної області, системний аналіз об’єкта проєктування. ПРН11 Вибирати вихідні дані для проєктування, керуючись формальними методами опису вимог та моделювання. ПРН12 Застосовувати на практиці ефективні підходи щодо проєктування

програмного забезпечення. ПРН14 Застосовувати на практиці інструментальні програмні засоби доменного аналізу проєктування, тестування, візуалізації, вимірювань та документування програмного забезпечення. ПРН17 Вміти застосовувати методи компонентної розробки програмного забезпечення. ПРН19 Знати та вміти застосовувати методи верифікації та валідації програмного забезпечення ПРН20. Знати підходи щодо оцінки та забезпечення якості програмного забезпечення.

Результати навчання. Студент, який успішно завершив вивчення дисципліни, має: Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізу та математичного моделювання для розробки програмного забезпечення. Вміти вибирати та використовувати відповідну задачі методологію створення програмного забезпечення. Проводити передпроєктне обстеження предметної області, системний аналіз об'єкта проєктування. Вибирати вихідні дані для проєктування, керуючись формальними методами опису вимог та моделювання. Застосовувати на практиці ефективні підходи щодо проєктування програмного забезпечення. Застосовувати на практиці інструментальні програмні засоби доменного аналізу проєктування, тестування, візуалізації, вимірювань та документування програмного забезпечення. Вміти застосовувати методи компонентної розробки програмного забезпечення. Знати та вміти застосовувати методи верифікації та валідації програмного забезпечення. 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

“Моделювання та оцінка програмного забезпечення”

Теми	Лекції	Лаб. роб.	Самостійна робота здобувачів
Тема 1.Вступ. Характеристика підходів при моделюванні програмного забезпечення	6	6	20
Тема 2. Основні поняття структурного моделювання програмного забезпечення	8	8	15
Тема 3. Базові поняття для моделювання поведінки програмного забезпечення.	8	8	15
Тема 4. Основи при моделюванні подій в програмному забезпеченні	6	6	16
Тема 5. Моделювання архітектури програмного забезпечення	6	6	16
Разом за 7-й семестр	**34**	**34**	**82**

5. ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

“ Моделювання та оцінка програмного забезпечення ”

5.1. Зміст лекційного курсу*

№ лекції	Перелік тем лекцій, їх анотація	Кількість годин
Тема 1. .Вступ. Характеристика підходів при моделюванні програмного забезпечення		6
1	**Лекція 1. Місце моделювання та його значення.** Огляд методик розробки програмного забезпечення. Генерування правильного коду з мінімальним обсягом. Моделювання програмного забезпечення. Поняття моделі систем. Неформальне та формальне моделювання .Література: [1-3; 7; 11].	2
2	**Лекція2**. **Принципи моделювання**. Аналіз більш вживаних принципів. Поняття об’єктного моделювання. Існуючі підходи об’єктного моделювання. Характеистика мови моделюванняUML Література: [2; 5; 12].	2
3	**Лекція 3**. **Концептуальна модель UML.** Будівельні блоки та їх складові. Складові варіантів використання. Базові структурні сутності. Зв’язки в UML. Діаграми UM та їх опис .Ознаки добре розробленої UML моделі. Література: [3-4; 10].	2
Тема 2. Основні поняття структурного моделювання програмного забезпечення		
4	**Лекція 4. Моделювання класів, об’єктів та екземплярів.** Класи. Вимоги до класів. Атрибути. Вимоги до атрибутів. Створення атрибутів та операцій. Обов’язки класів. Стереотипи, присвоєні значення, обмеження. Моделювання коментарів. Класифікатор. Розширені класи та їх властивості. Види класифікаторів. Шаблонні класи. Екземпляр. Види екземплярів. Автомат. Активні і пасивні елементи. Література: [4; 7; 14].	2
5	**Лекція 5**. **Моделювання зв’язків та відношень**	2

	Види зв'язків. Моделювання простих залежностей та одиничного спадкування. Розширення асоціацій. Моделювання зв'язків залежності. Моделювання одиничного успадкування. Множинне успадкування. Моделювання структурних зв'язків. Створення мереж зв'язків. Види розширених зв'язків. Моделювання зв'язків різних рівнів. Стереотипи зв'язків узагальнення між класами. Видимість класів і об'єктів. Асоціації-класи. Поняття інтерфейсу. Види інтерфейсів. Моделювання статичних і динамічних типів. . Література: [2; 4-6; 9].	
6	**Лекція 6. Діаграми для моделювання статичних характеристик системи** Поняття діаграми. Базові поняття моделювання. Принципи використання діаграм. Види діаграм. Основні властивості діаграм класів. Пряме проектування. Зворотнє проектування. Характеристика діаграм об'єктів. Процес моделювання структур об'єктів. Поняття компоненти. Види компонент. Зв'язок компоненти та інтерфейсу. Внутрішня структура. Моделювання структурованих класів. Моделювання інтерфейсу. Література: [1-3; 7; 11]	2
7	**Лекція 7.Компоненти та інтерфейси. Діаграми компонентів.** Компонента. Інтерфейс. Внутрішня структура. Коннектор. Зв'язки між компонентою та інтерфейсом. Заміщувана компонента. Порт .Внутрішня структура. Моделювання структурованих класів.Моделювання програмного інтерфейсу. . Література: [4-5; 9-12]	2
Тема 3. Базові поняття для моделювання поведінки програмного забезпечення.		
8	**Лекція 8. Моделювання варіантів використання** Елементи діаграми моделювання варіантів використання. Моделювання поведінки системи. Реалізація варіантів використання з використанням кооперації. Діаграми варіантів використання. Моделювання контексту системи . Моделювання вимог до системи . Пряме й зворотне проектування. Література: [1; 4; 12].	2
9	**Лекція 9. Діаграми варіантів використання** Опис діаграми використання. Складові діаграми використання. Суб'єкт. Діаграми варіантів використання у задачах проектування. Моделювання контексту системи. Моделювання вимог до системи. Пряме та зворотне проектування. Варіанти використання і тестування. Література: [3-4; 6; 13].	2
10	**Лекція 10. Моделювання взаємодії об'єктів Стратегія моделювання взаємодії як обміну поведінками між об'єктами** Елементи взаємодії об'єктів. Прототипні екземпляри класів. Потоки керування.Фази поведінки. Діаграми взаємодії при моделювання рпотоків керування. Контекст взаємодії. Об'єкти і ролі у взаємодії. Прототипні об'єкти та прототипні зв'язки або посилання. Типи та екземпляри повідомлень.	2

	Література: [3; 5; 9].	
11	**Лекція 11. Діаграми взаємодії. Діаграми діяльності** Види діаграм взаємодії. Стратегії розкадрування сценаріїв. Ознаки діаграм послідовностей. Зміст діаграми послідовностей. Структуроване високорівневе керування на діаграмах послідовностей. Оператори керування. Види керування. Вкладені діаграми послідовності і діяльності. Діаграми комунікацій. Моделювання складних потоків. Діаграма діяльності. Дії і вузли діяльності. Потоки керування. Моделювання паралельних потоків у системі. Області розширення. Моделювання потоку робіт. Моделювання операцій. Література: [1-2; 7; 11].	2
Тема 4. Основи при моделюванні подій в програмному забезпеченні		
12	**Лекція 12. . Моделювання подій сигналів. Кінцеві автомати. Моделювання поведінки спільно об'єктів.** Події сигналу, виклику, часу і зміни. Види подій. Моделювання серії сигналів. Моделювання сигналів класами. Синхронність події виклику. Подія часу. Подія зміни. Моделювання винятків. Опрацювання подій активних і пасивних об'єктах. Стани, переходи, діяльності. Моделювання ЖЦ об'єкта. Створення добре структурованих алгоритмів. Література: [5-6, 10, 13].	2
13	**Лекція 13. Процеси і потоки керування.** Моделювання систем реального часу. Час, тривалість і місце розташування. Моделювання тимчасових обмежень. Моделювання розподілу об'єктів. Моделювання мігруючих об'єктів. Моделювання мігруючих об'єктів.. Системи реального часу і розподілені системи. Література: [8-9, 13, 15].	2
14	**Лекція 14. Діаграми станів** Моделювання реактивних об'єктів. Реактивний або керований подіями об'єкт. Стабільний стан. Пряме і зворотне проектування. Література: [7-8, 10].	2
Тема 5. Моделювання архітектури програмного забезпечення		
15	**Лекція 15. . Поняття про архітектуру та раціональний уніфікований процес**. Система архітектури. Архітектура. Архітектурні вигляди. Життевий цикл розроблення програмного забезпечення.Аналіз та планування вимог. Упровадження. Література: [9, 12, 14].	2
16	**Лекція 16**. **Моделювання архітектурних зразків** Механізми і каркаси як основа архітектури системи. Шаблони та параметризовані кооперації. Пакет із стереотипом. Моделювання структурного аспекту зразка проектування. Моделювання поведінкового аспекту зразка проектуання. Моделювання архітектурного зразка.	2

	Література: [10-11, 13].	
17	**Лекція17. Моделювання кооперації. Моделювання пакетів як спосіб організації елементів моделі.** Кооперації, реалізації та взаємодії. Структурна складова кооперації. Відмінність кооперації і класів. Моделювання реалізації варіанта використання. Моделювання реалізації операції. Моделювання механізму. Матеріалізація взаємодій. Пакети, видимість, імпорт і експорт. Моделювання груп елементів. Моделювання архітектурних виглядів. Масштабування великих систем. Література: [5, 10-11].	2
Разом за 7-й семестр		**34**

5.2 Зміст лабораторних занять

№ з/п	Теми лабораторних занять	Години
1	Розробка функціональної та об'єктної модельної області	4
2	Розробка варіантів використання (use case)	4
3	Розробка діаграми класів	4
4	Розробка діаграми об'єктів	4
5	Побудова діаграми взаємодії	4
6	Побудова діаграми діяльності	4
7	Побудова діаграми станів	4
8	Розробка патернів проектування	4
9	Підсумкове заняття	**2**
Разом за 7-й семестр		**34**

5.3 Зміст самостійної роботи

Об'єм самостійної роботи з дисципліни **"Моделювання та оцінка програмного забезпечення"** становить 82 години. 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. Характеристика підходів при моделюванні програмного забезпечення.** Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №1. Література: [1-5; 7; 11].	6
2	**Тема 1. .Вступ. Характеристика підходів при моделюванні програмного забезпечення.** Опрацювання лекційного матеріалу, захист лабораторної роботи № 1. Література: [4; 7; 11].	6
3	**Тема 1. .Вступ. Характеристика підходів при моделюванні програмного забезпечення.** Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №2. Література: [3-4; 10].	6

4	**Тема 2. Основні поняття структурного моделювання програмного забезпечення** Опрацювання лекційного матеріалу. Захист лабораторної роботи №2. Література: [4; 7; 11].	6
5	**Тема 2. Основні поняття структурного моделювання програмного забезпечення** Опрацювання лекційного матеріалу. Підготовка до виконання лабораторної роботи №3. Література: [4; 7; 11].	6
6	**Тема 2. Основні поняття структурного моделювання програмного забезпечення** Опрацювання лекційного матеріалу, захист лабораторної роботи №3. Література: [1; 4; 12].	6
7	**Тема 2. Основні поняття структурного моделювання програмного забезпечення.** Опрацювання лекційного матеріалу. Підготовка до виконання лабораторної роботи №4. Література: [1; 4; 12].	3
8	**Тема 3. Базові поняття для моделювання поведінки програмного забезпечення .**Опрацювання лекційного матеріалу, захист лабораторної роботи №4. Підготовка до проходження тесту. Література: [5-6; 10].	3
9	**Тема 3. Базові поняття для моделювання поведінки програмного забезпечення.** Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №5. Література: [2-4; 8-9]	3
10	**Тема 3. Базові поняття для моделювання поведінки програмного забезпечення.**Опрацювання лекційного матеріалу, захист лабораторної роботи №5. Література: [4; 6; 13]	3
11	**Тема 3. Базові поняття для моделювання поведінки програмного забезпечення.** Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №6. Література: [1; 3; 6]	4
12	**Тема 4. Основи при моделюванні подій в програмному забезпеченні.**Опрацювання лекційного матеріалу, захист лабораторної роботи №6. Література: [2; 13].	3
13	**Тема 4. Основи при моделюванні подій в програмному забезпеченні.**Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №7. Література: [4; 7; 13].	3
14	**Тема 4. Основи при моделюванні подій в програмному забезпеченні** Опрацювання лекційного матеріалу. Захист лабораторної роботи №7. Література: [4; 7; 13].	6
15	**Тема 5. Моделювання архітектури програмного забезпечення** Опрацювання лекційного матеріалу, підготовка до виконання лабораторної роботи №8. Література: [2; 5; 11].	6
16	**Тема 5. Моделювання архітектури програмного забезпечення** Опрацювання лекційного матеріалу, підготовка до виконання та захист лабораторної роботи №8. Література: [2; 5; 11].	6
17	**Тема 5. Моделювання архітектури програмного забезпечення** Опрацювання лекційного матеріалу. Підготовка до іспиту.	6
Разом за 7-й семестр		**82**

8. 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 практичних навичок реалізації програмних систем, основ моделювання і аналізу програмних систем, аналізу розробки, специфікації та управління вимогами

8. 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.

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="10">Аудиторна робота</td><td rowspan="2">Семестровий контроль, іспит (І)</td><td rowspan="2">Підсумковий бал</td></tr>
<tr><td colspan="9">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>9</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="9">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

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

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. Назвіть складові частини об’єктно-орієнтованого підходу
2. Опишіть класи і відношення між ними
3. Які є категорії класів
4. Які є Ознаки складної системи
5. Опишіть структуру та проектування складних систем.
6. Дайте поняття декомпозиції системи
7. Структура системи
8. Структура системи
9. Які є специфікації
10. Дайте опис об’єктної моделі
11. З чого складатюься діаграми станів і переходів
12. Які складові діаграми об’єктів
13. Системна архітектура ПЗ
14. Об’єктно-орієнтований аналіз, проектування та програмування
15. Дайте опис діаграми взаємодій
16. Класифікація систем
17. Управління проектами
18. Ідентифікація компонентів системи
19. Механізми абстракції
20. Системна архітектура програмної системи
21. Варіанти використання
22. UML як мова моделювання ПЗ
23. UML-диаграми класів
24. Універсальний підхід моделювання ПЗ
25. Декомпозиція системи
26. UML-діаграми об’єктів
27. UML-диаграми дій
28. Складові частини об’єктно-орієнтованого підходу
29. UML- діаграми сценаріїв реалізації
30. Об’єктна модель
31. Складові частини об’єктно-орієнтованого підходу
32. Об’єктно-орієнтований аналіз, проектування та програмування
33. Об’єктна модель
34. Класифікація систем
35. Об’єктно-орієнтований аналіз, проектування та програмування
36. UML-діаграми варіантів використання
37. Об’єктна модель системи.
38. Складові частини об’єктного підходу.
39. Застосування об’єктних моделей.
40. Класифікація моделей та їх ідентифікація.

9. МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ

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

10. РЕКОМЕНДОВАНА ЛІТЕРАТУРА

ОСНОВНА:

1. Вігерс, Карл І. Джой Бітті (2016) Розробка вимог до програмного забезпечення. - . Retryeved from: http://www.twirpx.com/file/1073169/
2. Табунщик, Т.І., Каплієнко, Г. В., Петрова О.А. (2016) Проектування та моделювання програмного забезпечення сучасних інформаційних систем Запоріжжя: Дике Поле,
3. Введення в програмну інженерію і управління життєвим циклом програмного забезпечення Guide to Software Engineering Base of Knowledge (SWEBOK): Пер. з англ. С.Орлик Retryeved from:sorlik.blogspot.com/
4. Г.В.Табунщік., Каплієнко, Т.І. Петрова. О.А. (2016) Проектування та моделювання програмного забезпечення сучасних інформаційних систем. Запоріжжя
5. Shishkov B. (2020) Designing Enterprise Information Systems: Merging Enterprise Modeling And Software Specification. New York: Springer
6. Dwyer Barry. Morgan Kaufmann, (2016) Systems Analysis and Synthesis: Bridging Computer Science and Information Technology.
7. Петрик М.Р., Петрик О.Ю. Моделювання програмного забезпечення : науково методичний посібник. Тернопіль : Вид-во ТНТУ імені Івана Палія, 2015. 200 с.

ДОПОМІЖНА:

8. Елізабет Халл, Кен Джексон, Дік Джеремі, (2017), «Інженерія вимог» (Requirements Engineering), пров. ДМК Прес
9. I Jacobson I., Lawson H. Bud, Ng P.-W., McMahon P, E., Goedicke M.(2019) The Essentials of Modern Software Engineering: Free the Practices from the Method Prisons! Association for Computing Machinery and Morgan & Claypool Publishers
10. А. Бубнов, С. Бубнов, К. Майков,(2018) «Розробка і аналіз вимог до програмного забезпечення «, КУРС
11. Mejia J., Muñoz M., Rocha A., Quiñonez Y. (Eds.) (2021) New Perspectives in Software Engineering: Proceedings of the 9th International Conference on Software Process Improvement (CIMPS 2020) Springer,
12. Rosen C. (2020) Guide to Software Systems Development: Connecting Novel Theory and Current Practice. New York: Springer
13. Standard for Software Verification and Validation Plans (ANSI / IEEE standard 1012-1986)
14. D’Andrade Brian. (2021) Software Engineering: Artificial Intelligence, Compliance, and Security. Nova Science Publishers,

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 IT Faculty
_____ Savenko O.S.
"_1_" _____ 2023.

COURSE PROGRAM
Software Modelling and Evaluation

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)	3	5	5	150	34	34			82				+
Total			**5**	**150**	**34**	**34**			**82**				**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 MODELING AND EVALUATION

Type of discipline	Compulsory
Level of higher education	First (Bachelor's)
Language of Instruction	English
Semester	5
ECTS Credits	5
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 . General approaches to modeling, Principles of modeling. Basics of software structural modeling, Basics of software behavior modeling. Basics of event modeling. Software architecture modeling.

Planned educational activity: 34 hours of lectures, 34 hours of laboratory classes, 82 hours of independent work; together 150 hours

Teaching methods: methods of problem-based teaching, verbal, visual (lectures); explanatory - 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.

Educational resources:

1. Wiegers , Carl I. Joy Beatty (2016) Development software requirements _ provision _ - Retrieved from : http://www.twirpx.com/file/1073169/
2. Tabunshchik, T.I., Kaplienko , H.V., Petrova O.A. (2016) Design and Modeling software software modern of information systems of Zaporizhzhia : Dyke Pole,
3. Introduction to software engineering and management software life cycle providing Guide to Software Engineering Base of Knowledge (SWEBOK): Trans. From English S. Orlyk Retrieved from:sorlik.blogspot.com /
4. 1. MOODLE Learning Platform. Access to the resource https://msn.khmnu.edu.ua.
5. University Electronic Library. Access to the resource: http://library.khmnu.edu.ua.
6. University Repository. Access to the resource https://elar.khmnu.edu.ua/home.

Lecturer: Associate Professor, PhD Onyshko O.H.

3. EXPLANATORY NOTE

The discipline " Software modeling and evaluation " is a discipline of professional and practical training.
The **objective** of the course is to provide students with a comprehensive theoretical understanding and practical skills in the field of software modeling and assessment, specifically in the development of system models, their evaluation, and optimization.

The **subject** of the course encompasses the study of fundamental principles and methodologies of software modeling, including object modeling, the use of UML, structural modeling, as well as practices of modeling software system architecture.

The **course tasks** include: mastering the principles and methods of creating software system models; analyzing and utilizing UML for system conceptualization; modeling interactions, behavior, and software architecture; acquiring practical skills in model implementation, including assessment and testing; and understanding the role of modeling in software development and requirements management processes.
The course equips students with the knowledge and tools for a deep understanding of the aspects of creating and analyzing software, from modeling requirements to evaluating the finished software products.

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.

PC2. Ability to participate in software design, including modelling (formal description) of its structure, behaviour, and operational processes.
PC3. Ability to develop architectures, modules, and components of software systems.
PC5. Ability to adhere to specifications, standards, rules, and recommendations in the professional field during the implementation of lifecycle processes.
PC10. Ability to accumulate, process, and systematise professional knowledge regarding the creation and maintenance of software and recognise the importance of lifelong learning.
PC11. Ability to implement phases and iterations of the life cycle of software systems and information technologies based on relevant software development models and approaches.
PC12. Ability to execute the system integration process and apply standards and change management procedures to maintain the integrity, overall functionality, and reliability of the software.
Program learning outcomes: PLO5 To understand and apply relevant mathematical concepts, domain and system methods, object-oriented analysis, and mathematical modelling for software development.
PLO6 To select and 17isuali a software development methodology appropriate for the task.
PLO10To conduct a pre-project survey of the subject area and system analysis of the design object.
PLO11To select initial data for design, guided by formal methods of requirement descriptions and modelling.
PLO12To apply effective software design approaches in practice.
PLO14To use instrumental software tools in practice for domain analysis, design, testing, 17isualization, measurement, and software documentation.
PLO17To be skilled in applying methods of component software development.
PLO19To know and apply methods for software verification and validation.
PLO20To know approaches to software quality evaluation and assurance..

4. COURSE CREDIT STRUCTURE

Topics	Lectures	lab.works	Individual works
Topic 1. Introduction. Characteristics of software modeling approaches	6	6	20
Topic 2. Basic concepts of software structural modeling	8	8	15
Topic 3. Basic concepts for modeling software behavior.	8	8	15
Topic 4. Basics of event modeling in software	6	6	16
Topic 5. Software architecture modeling	6	6	16
Together for the 7th semester	**34**	**34**	**82**

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. Characteristics of software modeling approaches		6
1	**Lecture 1. The place of modeling and its meaning.** Overview of software development techniques . Generating the correct code with the minimum amount. Software modeling. Concept of systems model. Informal and formal modeling. Literature: [1-3; 7; 11].	2
2	**Lecture 2** . **Principles of modeling** . Analysis of more commonly used principles. The concept of object modeling. Existing object modeling approaches. Characteristics of the UML modeling language Literature: [2; 5; 12].	2
3	**Lecture 3** . **UML conceptual model .** Building blocks and their components. Components of use options. Basic structural entities. Connections in UML . UM diagrams and their description. Signs of a well-developed UML model. Literature: [3-4; 10].	2
Topic 2. Basic concepts of software structural modeling		
4	**Lecture 4. Modeling classes , objects and instances .** Classes. Class requirements. Attributes. Attribute requirements. Creation of attributes and operations. Class duties. Stereotypes, assigned values, limitations. Modeling comments. Classifier. Extended classes and their properties. Types of classifiers. Template classes. Copy Types of specimens. Automaton. Active and passive elements. Literature: [4; 7; 14].	2
5	**Lecture 5** . **Modeling connections and relations** Types of connections . Modeling simple dependencies and single inheritance. Expansion of associations. Modeling of dependency	2

	relationships . Modeling single inheritance. Multiple inheritance. Modeling of structural connections . Creating networks of connections . Types of extended connections . Modeling of connections of different levels. Stereotypes of relations of generalization between classes. Visibility of classes and objects. Associations-classes. Concept of interface. Types of interfaces. Modeling of static and dynamic types. . Literature: [2; 4-6; 9].	
6	**Lecture 6. Diagrams for modeling static characteristics of the system** The concept of a diagram. Basic concepts of modeling. Principles of using diagrams. Types of diagrams. Basic properties of class diagrams. Direct design. Reverse engineering. Characteristics of diagrams of objects. The process of modeling object structures. The concept of components. Types of components. Communication of components and interfaces. Internal structure. Modeling structured classes. Interface modeling. Literature: [1-3; 7; 11]	2
7	**Lecture 7. Components and interfaces. Component diagrams.** Component. Interface. Internal structure. Connector . Connections between the component and the interface. Replaceable component. Port .Internal structure. Modeling of structured classes. Modeling of the software interface. . Literature: [4-5; 9-12]	2
Topic 3. Basic concepts for modeling software behavior .		
8	**Lecture 8. Modeling of use cases** Elements of the use case modeling diagram . Modeling of system behavior. Implementation of use cases using cooperation. Use case diagrams. Modeling the context of the system . Modeling system requirements _ . Direct and reverse designing . Literature: [1; 4; 12].	2
9	**Lecture 9. Diagrams of use cases** Description of the usage chart. Component diagrams of use. Subject. Diagrams of use cases in design tasks. Modeling the context of the system. Modeling system requirements. Forward and reverse engineering. Options for use and testing. Literature: [3-4; 6; 13].	2
10	**Lecture 10. Modeling interaction objects Strategy modeling interaction as an exchange behaviors between objects** Elements of object interaction. Prototype instances of classes. Management flows . Phases of behavior. Interaction diagrams when modeling control flows . Context of interaction. Objects and roles in interaction. Prototype	2

	objects and prototypic relationships or references. Message types and instances. Literature: [3; 5; 9].	
11	**Lecture 11. Interaction diagrams. Activity charts** Types of interaction diagrams. Storyboarding strategies. Signs of sequence diagrams. Sequence diagram content. Structured high-level control on sequence diagrams. Management operators. Types of management. Sequence and activity diagrams are attached. Communication diagrams. Modeling complex flows. Activity chart. Actions and nodes of activity. Control flows. Modeling parallel flows in the system. Areas of expansion. Workflow modeling. Simulation of operations. Literature: [1-2; 7; 11].	2
Topic 4. Basics of event modeling in software		
12	**Lecture 12 . Modeling events signals . Final automatic machines Modeling behavior together objects in** Signal, call, time and change events. Types of events. Modeling a series of signals. Signal modeling by classes. Call event synchronicity. Time event. Change event. Modeling exceptions. Processing of events of active and passive objects. States, transitions, activities. Modeling of the residential complex of the object. Creation of well-structured algorithms. Literature: [5-6, 10, 13].	2
13	**Lecture 13. Management processes and flows .** Simulation of real-time systems . Time, duration and location. Modeling of temporal constraints. Modeling the distribution of objects. Modeling of migrating objects. Modeling of migrating objects. Real-time systems and distributed systems. Literature: [8-9, 13].	2
14	**Lecture 14. Diagrams became** Modeling of reactive objects. A reactive or event-driven object. Stable condition. Direct and reverse engineering. Literature: [7-8, 10].	2
Topic 5. Software architecture modeling		
15	**Lecture 15. Concept of architecture and rational unified process** ... Architecture system. Architecture. Architectural views. Life cycle of software development. Requirements analysis and planning. Implementation . Literature: [9, 12].	2

16	**Lecture 16** . **Modeling architectural samples** Mechanisms and frameworks as the basis of system architecture. Templates and parameterized cooperations. Package with a stereotype. Modeling the structural aspect of the design sample. Modeling the behavioral aspect of the design sample . Modeling of an architectural model. Literature: [10-11, 13].	2
17	**Lecture 17. Modeling cooperation . Modeling packages as a method organizations elements models** ... Cooperation, implementation and interaction. Structural component of cooperation. The difference between cooperation and classes. Modeling the implementation of the use case. Simulation of operation implementation. Mechanism modeling. Materialization of interactions. Packages, visibility, import and export. Modeling of groups of elements. Modeling of architectural forms. Scaling large systems. Literature: [5, 10-11].	2
Together for the 7th semester		**34**

5.2 Content of laboratory classes

No. z/p	Topics of laboratory classes	hours
1	Development of functional and object model area	**4**
2	Development of use cases (use case)	**4**
3	Development of a class diagram	**4**
4	Development of a diagram of objects	**4**
5	Building an interaction diagram	**4**
6	Construction of activity diagram	**4**
7	Construction of a state diagram	**4**
8	Development of design patterns	**4**
9	Software development using pattern design “ Abstract “Factory “	**2**
Together for the 7th semester		**34**

5,3 Content of independent work

The volume of independent work in the discipline **“ Software modeling and evaluation “** is 99 hours. They include study of lecture material, theoretical and laboratory tasks, preparation for laboratory works, their defense and ongoing testing.

Number of the week	Topic name	hours
1	**Topic 1 . Introduction. Characteristics of software modeling approaches .** Elaboration of lecture material, preparation for laboratory work No. 1. Literature: [1-5; 7; 11].	6

2	**Topic 1. .Introduction. Characteristics of software modeling approaches .** Development of lecture material , defense of laboratory work No. 1. Literature: [4; 7; 11].	6
3	**Topic 1. Introduction. Characteristics of software modeling approaches .** Elaboration of lecture material, preparation for laboratory work No. 2. Literature: [3-4; 10].	6
4	**Topic 2. Basic concepts of structural software modeling** Processing of lecture material. Protection of laboratory work #2. Literature: [4; 7; 11].	6
5	**Topic 2. Basic concepts of structural software modeling** Processing of lecture material. Preparation for laboratory work No. 3. Literature: [4; 7; 11].	6
6	**Topic 2. Basic concepts of structural software modeling** Processing of lecture material, defense of laboratory work #3. Literature: [1; 4; 12].	6
7	**Topic 2. Basic concepts of software structural modeling.** Processing of lecture material. Preparation for laboratory work No. 4. Literature: [1; 4; 12].	3
8	**Topic 3. Basic concepts for modeling software behavior .** Elaboration of lecture material, defense of laboratory work #4. Preparation for passing the test. Literature: [5-6; 10].	3
9	**Topic 3. Basic concepts for modeling software behavior .** Elaboration of lecture material, preparation for laboratory work No. 5. Literature: [2-4; 8-9]	3
10	**Topic 3. Basic concepts for modeling software behavior .** Elaboration of lecture material, defense of laboratory work No. 5. Literature: [4; 6; 13]	3
11	**Topic 3. Basic concepts for modeling software behavior .** Elaboration of lecture material, preparation for laboratory work No. 6. Literature: [1; 3; 6]	4
12	**Topic 4. Basics of event modeling in software .** Elaboration of lecture material, defense of laboratory work #6. Literature: [2; 13].	3
13	**Topic 4. Basics of event modeling in software .** Elaboration of lecture material, preparation for laboratory work No. 7. Literature: [4; 7; 13].	3
14	**Topic 4. Basics of event modeling in software Processing** of lecture material. Protection of laboratory work #7. Literature: [4; 7; 13].	6
15	**Topic 5. Software architecture modeling** Processing of lecture material, preparation for laboratory work No. 8. Literature: [2; 5; 11].	6
16	**Topic 5. Software architecture modeling** Processing of lecture material, preparation for performance and defense of laboratory work #8. Literature: [2; 5; 11].	6
17	**Topic 5. Modeling of software architecture** Processing of lecture material. Preparation for the exam.	6
Together for the 7th semester		**82**

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 -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. FORMS AND METHODS OF EVALUATING LEARNING OUTCOMES

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:

Familiar -orientated (OO) – a person has an approximate idea of the concepts being studied, is able to: program the main elements of software systems in various programming languages, choose modern methodologies and technologies of software analysis , 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 synthesize, generate new ideas and concepts, 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 applicant 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, has 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.

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

<table>
<tr><td colspan="9">Auditory work</td><td rowspan="2">Semester control, exam (I)</td><td rowspan="2">Final score</td></tr>
<tr><td colspan="9">Laboratory work (LR)</td><td>Test</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>9</td><td>T</td><td rowspan="2">0.4</td><td rowspan="2">LR*0.3+T*0.3+I*0.4</td></tr>
<tr><td colspan="9">WC = 0.3</td><td>WC = 0.3</td></tr>
</table>

Conventional designations: WC – 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

<table>
<tr><th>Evaluation of ECTS</th><th>Institutional interval scoring scale</th><th colspan="3">Domestic assessment, criteria</th></tr>
<tr><td>A</td><td>4.75–5.00</td><td>5</td><td>Excellent – deep and complete mastery of the educational material and identification of relevant abilities and skills</td><td rowspan="4">Enrolled</td></tr>
<tr><td>B</td><td>4.25–4.74</td><td>4</td><td>Good – complete knowledge of the educational material with a few minor errors</td></tr>
<tr><td>C</td><td>3.75–4.24</td><td>4</td><td>Good – a generally correct answer with two or three significant errors</td></tr>
<tr><td>D</td><td>3.25–3.74</td><td>3</td><td>Satisfactory – incomplete mastery of the program material,</td></tr>
</table>

			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.

9. QUESTIONS FOR SELF-CONTROL

1. Name c pantries parts object-oriented approach
2. Describe classes and relations between them
3. What are the categories classes
4. What are the Signs complex systems
5. Describe the structure and design complex systems.
6. Give the concept of decomposition systems
7. System structure
8. System structure
9. What are the specifications
10. Give a description of the object model and
11. What are the diagrams made of states and transitions
12. What are the components of the diagram objects
13. Software system architecture
14. Object-oriented analysis , design and programming
15. Give a description of the diagram interactions
16. Classification of systems
17. Project management
18. Identification components systems
19. Mechanisms abstractions
20. System architecture software systems
21. Options using
22. UML as a language software modeling
23. UML diagrams classes
24. Universal approach software modeling
25. Decomposition systems
26. UML diagrams objects
27. UML diagrams actions
28. Constituents parts object-oriented approach
29. UML diagrams scenarios implementation
30. Object model

31. Constituents parts object-oriented approach
32. Object-oriented analysis , design and programming
33. Object model
34. Classification of systems
35. Object-oriented analysis , design and programming
36. UML diagrams options using
37. Object model of the system .
38. Constituents parts object approach _
39. Application object models.
40. Classification models and their identification .

9. TEACHING AND LEARNING MATERIALS

Educational discipline process _ provided necessary educational and methodical developments in a modular environment .

10. RECOMMENDED BOOKS

MAIN:

1. Wiegers , Carl I. Joy Beatty (2016) Development software requirements _ provision _ - Retrieved from : http://www.twirpx.com/file/1073169/
2. Tabunshchik, T.I., Kaplienko , G.V., Petrova O.A. (2016) Design and Modeling software software modern of information systems of Zaporizhzhia : Dyke Pole,
3. Introduction to software engineering and management software life cycle providing Guide to Software Engineering Base of Knowledge (SWEBOK): Trans. from English S. Orlyk Retrieved from:sorlik.blogspot.com /
4. H.V. Tabunshchik , Kaplienko , T.I. Petrova. O.A. (2016) Design and Modeling software software modern information systems. Zaporizhzhia
5. Shishkov B. (2020) Designing Enterprise Information Systems: Merging Enterprise Modeling And Software Specification . New York: Springer
6. Dwyer Barry . Morgan Kaufmann , (2016) Systems Analysis and Synthesis : Bridging Computer Science and Information Technology.
7. Petryk M.R., Petryk O.Yu. Modeling software provision : scientific methodical guide . Ternopil : Publication of the TNTU named after him Ivan Palia , 2015. 200 p.

AUXILIARY:

8. Elizabeth Hull , Ken Jackson, Dick Jeremy , (2017), " Engineering requirements " (Requirements Engineering), prov . DMK Pres
9. And Jacobson I., Lawson H. Bud , Ng P.-W., McMahon P, E., Goedicke M. (2019) The Essentials of Modern Software Engineering: Free the Practices from the Method Prisons ! Association for Computing Machinery and Morgan & Claypool Publishers
10. A. Bubnov, S. Bubnov, K. Maikov, (2018) " Development and analysis software requirements _ provision ", COURSE
11. Mejia J., Muñoz M., Rocha A., Quiñonez Y. (Eds .) (2021) New Perspectives in Software Engineering: Proceedings of the 9th International Conference on Software Process Improvement (CIMPS 2020) Springer,

12. Rosen C. (2020) Guide to Software Systems Development: Connecting Novel Theory and Current Practice . New York: Springer
13. Standard for Software Verification and Validation Plans (ANSI / IEEE standard 1012-1986)
14. D'Andrade Brian . (2021) Software Engineering: Artificial Intelligence, Compliance , and Security. Nova Science Publishers

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.