МІЖНАРОДНИЙ ЄВРОПЕЙСЬКИЙ УНІВЕРСИТЕТ

Навчально-науковий інститут
«Європейська школа бізнесу»

Кафедра інформаційних технологій

ОСНОВИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Методичні рекомендації
до опанування лекційних занять
з навчальної дисципліни
для здобувачів вищої освіти спеціальності
F2 «Інженерія програмного забезпечення»

Київ – 2025

УДК 37.026:004.4:007

Рекомендовано до друку Вченою радою
Навчально-наукового інституту
«Європейська школа бізнесу»
(протокол № 1 від 26.08.2025)

Рецензент:
Шевчук Б.В., к.пед.н., доцент,
доцент кафедри інформаційних технологій МЄУ

Укладач:
Нестеренко О.В., д.т.н., професор,
професор кафедри інформаційних технологій МЄУ

Основи програмної інженерії: Методичні рекомендації до опанування лекційних занять з навчальної дисципліни для здобувачів вищої освіти спеціальності F2 «Інженерія програмного забезпечення» / Нестеренко О.В. Київ: МЄУ, 2025. 33 с.

Запропоноване видання узгоджене з навчальною програмою з дисципліни «Основи програмної інженерії» та може використовуватися як посібник для студентів, які розпочинають навчання на бакалавраті в галузі інформаційних технологій, зокрема за спеціальністю 121 «Інженерія програмного забезпечення».

Для студентів вищих навчальних закладів, які навчаються за напрямком підготовки „Інформаційні технології".

ЗМІСТ

ВСТУП

Головною ознакою сьогодення є повсюдне застосування інформаційних технологій і систем, динамічною складовою яких виступає програмне забезпечення. Розвиток обчислювальної техніки пов'язаний не тільки з вдосконаленням комп'ютерів та їх складових, але й з розвитком технологій програмування.

Тому в сучасних умовах актуальною є відносно нова для технологічного розвитку проблема, суть якої полягає у створенні усе більшої кількості програм, і, відповідно, новий технологічний процес, спрямований на її вирішення – інженерія програмування. Стрімке зростання кількості і об’ємів програм передусім пов’язане з залученням у технологічний процес їх виробництва значних колективів різноманітних фахівців. У зв’язку із цим в контексті цих робіт особливе значення набуває людський фактор і його роль в розробці програмного забезпечення (ПЗ).

Отже, в теперішній час створення ПЗ – це виключно колективний технологічний процес. Тому питання організації бізнесу в цій галузі, управління групами розробників та виробництвом, а також методи та засоби планування, проєктування, контролю є дуже важливими для досягнення мети отримання якісного програмного продукту. Ці та інші дотичні питання вивчаються в дисципліні «Основи програмної інженерії».

Масштабні і складні зміни, які несе нова промислова революція, пов’язана з масовою роботизацією, злиттям технологій і стиранням граней між фізичними і цифровими сферами і, як наслідок, майбутня інтелектуалізація суспільства потребують переважання високопрофесійних фахівців-програмістів. Вони повинні не лише вміти працювати з комп’ютерним обладнанням та різноманітними програмними засобами, розуміти роль інформаційних технологій у суспільній діяльності, а й бути обізнаними в сучасних технологіях управління колективами та підтримки прийняття рішень в групах, мати

цілісний погляд на основи структурування компаній з розробки ПЗ та їх функціонування з метою створення якісних програмних продуктів.

Саме тому метою навчальної дисципліни «Основи програмної інженерії» є оволодіння студентами знаннями про принципи та процеси розробки програмних продуктів та програмних систем, сучасні методології та практики у розробці програмного забезпечення, набуття умінь і практичних навичок щодо використання основних засобів організації процесу розробки програмного забезпечення, вміння аналізувати, проектувати та реалізовувати програмне забезпечення.

Внаслідок сучасних тенденції розвитку практики викладання у вищій освіті, пов'язаних із студентоцентрованим навчанням та викладанням як колективним процесом і розподілом відповідальності між викладачем та студентом, зростає активна роль студента та фокусування на його самостійності у процесі навчання – з одного боку, а з іншого – більшає значення вдосконалення та зросту якості дидактичних компетентностей викладача.

У зв'язку із цим у відповідь на зростаючі потреби в інноваціях та інтелектуалізації необхідно впроваджувати інноваційні змісти та моделі навчання. Враховуючи до того ж широке різноманіття тем, пов'язаних понять і визначень, теоретичних знань і практичних навичок, передбачених навчальною дисципліною «Основи програмної інженерії», вважається за доцільне надати студентам та викладачам додаткові методичні вказівки до опанування лекційних занять, що має сприяти більш швидкому зануренню у проблемну область і глибшому засвоєнню навчального матеріалу.

1. Завдання та структура навчальної дисципліни

Основними завданнями вивчення дисципліни «Основи програмної інженерії» є:

• ознайомити студентів з основними принципами та концепціями програмної інженерії;

• сформувати у студентів знання щодо основних методологій розробки програмного забезпечення та їх застосування;

• розвинути вміння аналізувати, проектувати та реалізовувати програмне забезпечення;

• вивчити основні етапи розробки програмних продуктів і систем;

• розвинути навички застосовування основних методів та інструментів, що використовуються в програмній інженерії;

• підготувати студентів до подальшого підвищення кваліфікації шляхом вивчення спеціалізованих курсів з програмування та розробки програмного забезпечення.

З завдань випливає мета викладання навчальної дисципліни «Основи програмної інженерії», яка полягає у формуванні у студентів здатностей оволодіння знаннями про принципи та процеси розробки програмних продуктів та програмних систем, сучасні методології та практики у розробці програмного забезпечення, набуття умінь і практичних навичок щодо використання основних засобів організації процесу розробки програмного забезпечення.

На вивчення навчальної дисципліни «Основи програмної інженерії» зазвичай відводиться 120 годин 4 кредити ЄКТС. Обсяг дисципліни поділено на два розділи (рис. 1, 2). Перелік розділів і тем дисципліни наведено в табл. 1.

При викладанні навчальної дисципліни «Основи програмної інженерії» крім інформаційних методів навчання, таких як класичні лекції, навчальним планом передбачено час на самостійну роботу

студентів. Завдання, що виносяться на самостійне опрацювання, наведені в табл. 2. Рекомендовані матеріали розміщено на платформі Distance Learning МЄУ.

Рис. 1. Структура першого розділу навчальної дисципліни

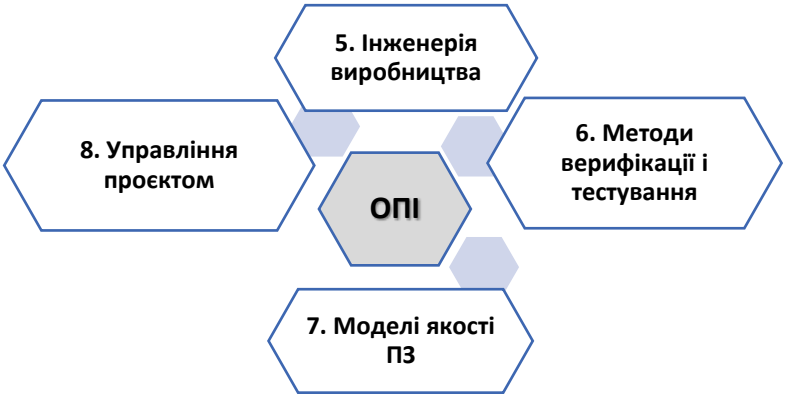

Рис. 2. Структура другого розділу навчальної дисципліни

Таблиця 1

Розділи і теми навчальної дисципліни «Основи програмної інженерії»

№ з/п	Тема	Перелік основних питань
1	2	3
РОЗДІЛ 1		
ЗМІСТОВИЙ РОЗДІЛ 1. ВИЗНАЧЕННЯ ПРОГРАМНОЇ ІНЖЕНЕРІЇ		
1	Тема 1. 1. Загальне визначення програмної інженерії	Роль програмної інженерії в розробці програмного забезпечення. Програмна інженерія як інженерна та наукова дисципліна. Процес розробки програмного забезпечення. Ознайомлення з основними поняттями та термінологією програмної інженерії. Основні міжнародні рекомендації з ПІ. Коротка історична довідка розвитку ПІ. Огляд сучасних тенденцій у програмній інженерії. Інновації та перспективи розвитку програмного забезпечення. Структура дисципліни та організація її вивчення.
2	Тема 1.2. Області знань SWEBOK	Характеристика областей знань з інженерії програмного забезпечення. Інженерія вимог. Проектування програмного забезпечення. Конструювання програмного забезпечення. Тестування програмного забезпечення. Супровід програмного забезпечення. Керування конфігурацією. Керування інженерією програмного забезпечення. Базовий процес ПІ. Методи і інструменти ПІ. Якість програмного забезпечення
3	Тема 1.3. Стандарти і моделі життєвого циклу програмного забезпечення	Характеристика життєвого циклу за стандартом ISO/IEC 12207. Прикладні моделі життєвого циклу. Типи моделей життєвого циклу (каскадна, інкрементна, спіральна, еволюційна та ін.)

Продовження табл. 1

1	2	3
ЗМІСТОВИЙ РОЗДІЛ 2. ВИМОГИ ДО ПРОГРАМНИХ СИСТЕМ		
4	Тема 2.1. Інженерія вимог	Визначення та аналіз вимог до програмного забезпечення. Класифікація вимог. Вимоги до функціональності, надійності, продуктивності та інші характеристики програмного забезпечення. Інженерія вимог. Методи інженерії вимог: ідентифікація, документування, аналіз та верифікація вимог. Аналіз і збирання вимог.. Фіксація вимог. Трасування вимог
5	Тема 2.2. Моделювання та проектування архітектури програмних систем	Принципи та методи проектування програмного забезпечення. Моделювання систем та компонентів програмного забезпечення. Основні поняття об'єктно-орієнтованих методів аналізу. Метод побудови об'єктної моделі предметної області. Проектування архітектури програмних систем. Архітектурні шаблони та стилі. Загальні підходи до проектування програмних систем. Проектування різних видів архітектур програмних систем
6	Тема 2.3. Методи програмування	Прикладне (систематичне) програмування (структурний метод; об'єктно-орієнтований; компонентний; сервісно-орієнтоване; аспектно-орієнтований та ін.).
ЗМІСТОВИЙ РОЗДІЛ 3. ІНЖЕНЕРІЯ ВИРОБНИЦТВА ПРОГРАМНИХ ПРОДУКТІВ		
7	Тема 3.1. Інженерія виробництва	Інженерія компонентів повторного використання. Прикладна інженерія та інженерія предметної області. Інженерія індустріального виробництва програмних продуктів. Оцінювання вартості системи з компонентів
8	Тема 3.2. Методи доведення, верифікації і тестування програм	Тестування програмних систем. Інфраструктура перевірки правильності програмних систем. Мови специфікації програм і їхня класифікація. Методи доведення правильності програм. Верифікація і валідація програм

Продовження табл. 1

1	2	3
9	Тема 3.3. Моделі якості та надійності програмних систем	Модель якості програмних систем. Стандартні показники якості та метрики якості. Ґрунтовні поняття проблематики надійності. Моделі оцінки надійності програмних систем. Сертифікація програмного продукту.
10	Тема 3.4. Управління програмним проектом	Менеджмент проекту. PMBOK. Галузі знань з управління проектами. Модель проектного менеджменту. Інфраструктура програмного проекту. Розподіл робіт і ролей у проекті. Методи планування і керування проектом (графічні, планування і контроль, оцінювання вартості проекту). Керування ризиками у проекті. Керування конфігурацією системи. Інструмент Microsoft Project

Таблиця 2

Загальний перелік завдань, що виносяться на самостійне опрацювання

№ з/п	Теми	Завдання, що виноситься на самостійне опрацювання
1.	1.1	Історичний шлях розвитку програмування
2.	1.2	Міжнародні рекомендацій з програмної інженерії (англ. мова)
3.	1.3	Матеріали щодо моделей життєвого циклу англійською мовою
4.	2.1 2.2	Матеріали на прикладі створення інформаційних системи управління
5.	2.3	Матеріали щодо розвитку теоретичного програмування в Україні
6.	3.1	Матеріали щодо оцінки вартості програмного забезпечення
7.	3.2	Матеріали щодо програмних засобів для реалізації автоматизованого тестування
8.	3.3	Матеріали щодо моделей надійності програмного забезпечення
9.	3.4	Матеріалами з навчання використанню Microsoft Project

2. Результати навчання

Зміст лекційних занять з дисципліни «Основи програмної інженерії містить навчальні елементи, необхідні для реалізації очікуваних результатів навчання, передбачених освітньою програмою спеціальності.

Згідно з вимогами стандарту освіти дисципліна має забезпечувати набуття студентами інтегральних компетентностей (ІК), а також низки загальних (ЗК) та спеціальних (фахових, предметних) (СК) компетентностей .

На основі трансформації цих компетентностей у такі результати навчання, які студент може опанувати на лекції і під час самостійної роботи над лекційним матеріалом та продемонструвати рівень їх сформованості на контрольних заходах формуються дисциплінарні результати навчання (ДР) [1] (рис. 3). Дисциплінарні результати навчання наведено у табл. 3.

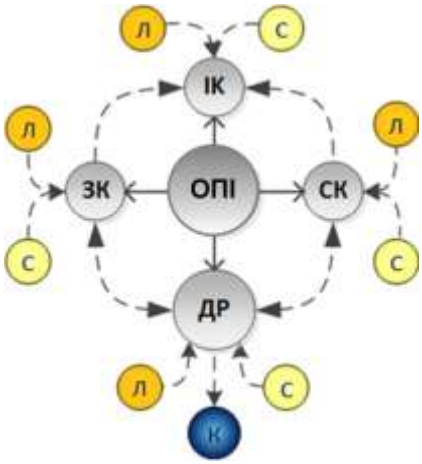

Рис. 3. Трансформація компетентностей і дисциплінарних результатів навчання дисципліни «Основи програмної інженерії» (ОПІ)
(Л – лекційні матеріали, С – самостійна робота, К – контрольні заходи)

Таблиця 3

Дисциплінарні результати навчання

№ з\п	Теми навчання	Знання та навички, що складають дисциплінарні результати навчання
Інтегральна компетентність		
1	2	3
1.	1.1, 1.2	Знати основні терміни та поняття дисципліни. Розуміти значення дисципліни для суспільства та власного розвитку. Вміти аналізувати та інтерпретувати інформацію, отриману з різних джерел, виявляти інтерес до подальшого вивчення дисципліни
Загальні компетентності:		
Здатність застосовувати знання у практичних ситуаціях		
2.	1.3. 2.2. 3.3. 3.4.	Реалізовувати прості програмні рішення відповідно до обраної моделі процесу розробки (Agile, каскадна тощо), використовувати системи контролю версій (наприклад, Git) у процесі командної розробки, впроваджувати базові методи забезпечення якості: тестування, рев'ю коду, автоматизацію збірки, застосовувати на практиці принципи інженерії вимог, моделювання, документації програмного продукту.
Здатність до пошуку, оброблення та аналізу інформації з різних джерел з використанням інформаційних технологій		
3.	1.3. 2.2.	Збирати та систематизувати технічну інформацію з відкритих джерел (стандарти, документація, специфікації) для формування вимог або вибору інструментів розробки, використовувати сучасні інформаційні системи (системи відстеження помилок, Вікі, бази знань, форуми розробників) для розв'язання технічних завдань, аналізувати альтернативні рішення на основі доступної інформації та обґрунтовувати вибір оптимального підходу для конкретного завдання в ПЗ.
Здатність працювати в команді		
4.	3.1. 3.4.	Виконувати командну розробку ПЗ із застосуванням методів колективної роботи, розподіляти ролі в команді, брати на себе відповідальність за конкретні завдання в межах спільного проєкту, ефективно комунікувати з іншими учасниками команди під час планування, реалізації та обговорення програмного продукту, застосовувати інструменти командної роботи для забезпечення прозорості розробки

Продовження табл. 3

1	2	3
Спеціальні (фахові, предметні) компетентності		
Здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням та стандартами		
5.	3.2. 3.3	Вміти ідентифікувати, класифікувати та формалізувати функціональні та нефункціональні вимоги до програмного забезпечення, складати специфікацію вимог відповідно до очікувань замовника та чинних стандартів (наприклад, ISO/IEC/IEEE 29148), складати або адаптувати технічне завдання та плани забезпечення якості відповідно до специфікацій замовника
Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу програмного забезпечення		
6.	1.3. 2.3.	Розуміти основні фази життєвого циклу ПЗ (аналіз вимог, проєктування, реалізація, тестування, впровадження, супровід) та їх взаємозв'язки, орієнтуватися в міжнародних стандартах життєвого циклу ПЗ (ISO/IEC/IEEE 12207 – Стандарт процесів життєвого циклу ПЗ; ISO/IEC 25010 – Модель якості ПЗ; IEEE 830 / ISO/IEC/IEEE 29148 – Специфікації вимог; IEEE 1012 – Верифікація та валідація ПЗ)
Здатність оцінювати і враховувати економічні, соціальні, технологічні та екологічні чинники, що впливають на сферу професійної діяльності		
7.	2.1. 3.1. 3.4.	Знати як аналізувати вартість розробки, супроводу та модернізації ПЗ з урахуванням обсягу вимог, тривалості проєкту та ресурсів, розуміти вплив повторного використання компонентів, застосування open-source, хмарних технологій на економічну ефективність проєкту, враховувати потреби та очікування користувачів з різних соціальних груп при розробці функціоналу ПЗ, адаптувати проєктні рішення до швидкозмінного технологічного середовища (наприклад, DevOps, мікросервіси, low-code/no-code платформи), запропоновувати рішення, орієнтовані на зниження ресурсоспоживання та підвищення екологічної стійкості програмних систем

Продовження табл. 3

1	2	3
Здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя		
8.	1.1. 1.2.	Орієнтуватися в системі знань програмної інженерії (зокрема, SWEBoK), розуміти важливість систематичного оновлення знань через професійні спільноти, літературу, стандарти, вміти знаходити, критично оцінювати та використовувати актуальні джерела технічної інформації (стандарти, документацію, публікації, навчальні ресурси), усвідомлювати динамічний характер сфери розробки ПЗ та необхідність навчання протягом усього життя, виявляти ініціативу в самоосвіті через проходження онлайн-курсів, відвідування технічних конференцій, читання професійної літератури, визначати власні освітні потреби й формувати індивідуальні траєкторії професійного розвитку (learning path).
Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення		
9.	1.3. 2.3. 3.2. 3.4.	Володіти знаннями про моделі життєвого циклу: каскадна (waterfall), інкрементна, спіральна, V-модель, гнучкі (Agile, Scrum, XP), DevOps тощо, здатність обґрунтовано обрати модель розробки відповідно до вимог проєкту, масштабу, рівня ризику, доступних ресурсів, реалізовати етапи розробки ПЗ (від постановки вимог до супроводу) згідно з обраною моделлю або методологією

Продовження табл. 3

1	2	3
Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення		
10.	3.2. 3.3.	Знати підходи до поетапної (інкрементної) та інтеграційно-орієнтованої розробки, включаючи безперервну інтеграцію (CI), орієнтуватися у стандартах, пов'язаних з інтеграцією, наприклад, IEEE 828 (Configuration Management), ISO/IEC/IEEE 24765, в автоматизованій інтеграції з використанням CI-інструментів (Jenkins, GitHub Actions, GitLab CI), в принципах управління змінами (Change Management) та їх ролі у підтримці стабільності ПЗ, знати засоби контролю версій (Git) для підтримки узгодженості вихідного коду під час командної роботи, як аналізувати вплив змін на показники надійності, безпеки, продуктивності
Здатність використовувати інструментарій на базі штучного інтелекту для підтримки процесів на усіх фазах життєвого циклу програмного забезпечення		
11.	1.3. 2.1. 3.1. 3.2. 3.3.	Усвідомлювати можливості використання інструментів штучного інтелекту (ШІ) на різних етапах життєвого циклу ПЗ, знати приклади застосування ШІ для автоматичного аналізу вимог (Natural Language Processing для інтерпретації й узагальнення вимог замовника, автоматичного створення специфікацій або user stories, побудови первинних діаграм на основі текстового опису), генерації коду (AI-асистенти програмування (GitHub Copilot, Amazon CodeWhisperer, ChatGPT, Tabnine)), автоматичного тестування та дебагінгу, прогнозування помилок чи дефектів (використання AI-помічників), підтримки супроводу та рефакторингу (наприклад, інтеграція Copilot у IDE, застосування LLM через API).

3. Методичні рекомендації до лекцій

Опанування лекційного матеріалу є найважливішим етапом навчання. Від того, як студент прослухав матеріал, зрозумів ідеї, які доносить до нього лектор, запам'ятав почуте залежить уся подальша робота з вивчення дисципліни, зокрема проведення самостійної роботи, практичних занять і підготовка до вивчення нових лекційних модулів.

В цьому процесі виняткову роль відіграє конспектування матеріалу лекцій. Важливо зафіксувати принаймні основне, що є в лекції – це визначення, логічні побудови, формулювання основних положень, висновки. В сучасних умовах ведення викладачами електронних ресурсів навчальних матеріалів, у тому числі і конспектів лекцій, які доступні студентству, дослівний запис лекцій втрачає необхідності. Але ведення окремого зошиту для навчальної дисципліни, фіксуючи основне з лекцій, передбачивши поля для подальших доповнень виписками з літературних джерел, власними міркуваннями, новими уявленнями добутими на основі наступних лекцій залишається вельми актуальним і корисним. При цьому варто зазначити, що ведення конспекту перетворює прослуховування лекції у цікаву роботу, а не відбування часу, адже це дисциплінує і сприяє зосередженості на лекції.

Ще одним чинником, який значною мірою впливає на сприйняття матеріалу лекцій є необізнаність і, як наслідок, невизначеність студента про загальний план дисципліни і взаємопов'язаність розділів і тем предмету що розглядається. Важливо з самого початку скласти чітке уявлення щодо зазначених питань.

Нижче на рис. 4 показано структурування матеріалу лекцій з точки зору ієрархії взаємозв'язків, а у табл. 4 наведено основні

елементи лекційних матеріалів, на які необхідно звернути особливої уваги при їх прослуховування та вивченні.

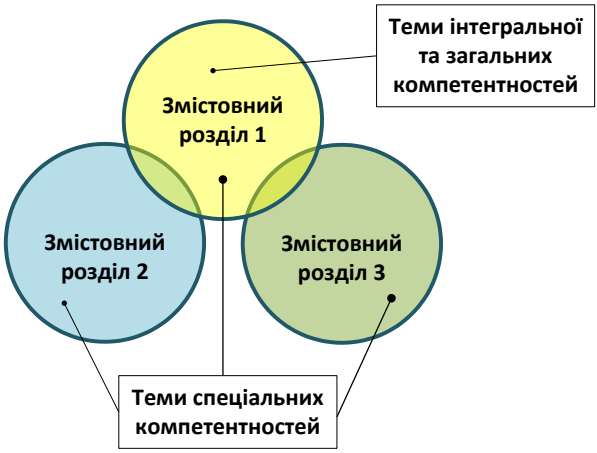

Рис. 4. Ієрархії взаємозв'язків матеріалів лекцій

Таблиця 4

Основні елементи лекцій з навчальної дисципліни

№ лекції	Теми лекції	Основні елементи	Літера-турні джере-ла
1	2	3	4
Лекція 1. Вступ до дисципліни та основні поняття			
1.	1.1, 1.2	Роль програмної інженерії в розробці програмного забезпечення. Програмна інженерія як інженерна та наукова дисципліна. Процес розробки програмного забезпечення. Ознайомлення з основними поняттями та термінологією програмної інженерії. Основні міжнародні рекомендації з ПІ.. Характеристика областей знань з інженерії програмного забезпечення	[2 - 12]

Продовження табл. 4

1	2	3	4
Лекція 2. Стандарти і моделі життєвого циклу			
2.	1.3	Характеристика життєвого циклу за стандартом ISO/IEC 12207. Процесний підхід. Процеси життєвого циклу. Прикладні моделі життєвого циклу. Схема виконання робіт і задач. Типи моделей життєвого циклу	[6, 12, 13, 15, 16]
Лекція 3. Інженерія вимог			
3.	2.1, 2.2	Визначення та аналіз вимог до програмного забезпечення. Класифікація вимог. Принципи та методи проектування програмного забезпечення. Моделювання систем та компонентів програмного забезпечення	[10 – 13, 18]
Лекція 4. Методи програмування			
4.	2.3	Прикладне (систематичне) програмування (структурний метод; об'єктно- орієнтований; компонентний; сервісно-орієнтоване; аспектно-орієнтований та ін.).	[14, 17, 19, 21]
Лекція 5. Інженерія виробництва			
5.	3.1	Інженерія компонентів повторного використання. Прикладна інженерія та інженерія предметної області.	[22 - 26]
Лекція 6. Методи доведення, верифікації і тестування програм			
6.	3.2	Мови специфікації програм і їхня класифікація. Методи доведення правильності програм. Верифікація і валідація програм	[24, 26]
Лекція 7. Моделі якості та надійності програмних систем			
7.	3.3	Модель якості програмних систем. Стандартні показники якості та метрики якості. Ґрунтовні поняття проблематики надійності	[29 - 31]
Лекція 8. Управління програмним проектом			
8.	3.4	Менеджмент проекту. PMBOK. Галузі знань з управління проектами	[28, 32 – 38, 40]

4. Методичні рекомендації до самостійної роботи

Навчальна дисципліна «Основи програмної інженерії» – це жива екосистема знань, заснована на самоорганізації як відповіді на зовнішні зміни. Враховуючи положення, що вища освіта як відповідь на суспільні виклики має постійно актуалізуватись в руслі сучасних трендів (challenge-based learning), ця дисципліна також повинна безперервно вдосконалюватись та оновлюватись [39, 40].

Напрямок програмної інженерії містить прояви різних сфер науки та досліджень, інтегрованих через суспільні потреби. Тому при вивченні предмету до нього має використовуватись міждисциплінарний підхід як до універсального професійного курсу. Необхідно також заначити, що в переході від інноваційної науково-технічної системи навчання STEM (Science, Technology, Engineering, Mathematics) через STEAM (плюс Arts – мистецтво) до STREAM (плюс Reading/wRiting – читання/письмо) саме в самостійній роботі студента проявляються, у доповненні до інжинірингу лекційних занять, творче ставлення до навчання як мистецтва та навички критичного мислення, втілені в читанні і письмі (рис. 5).

Рис. 5. Роль самостійної роботи студента в «потоці» STREAM

В цьому сенсі неабиякого значення набуває хід самостійної роботи студентів, опрацювання матеріалів лекцій та джерел інформації в процесі підготовки до контрольних заходів за результатами вивчення лекційних модулів. Відповідні рекомендації наведено у табл. 5.

Таблиця 5

Рекомендації щодо самостійної роботи студента з навчальної дисципліни

№ лекції	Теми	Завдання, що виноситься на самостійне опрацювання	Рекомендації	Додаткові джерела
1	2	3	4	5
1	1.1	Реферат щодо історичного шляху розвитку програмування	Розглянути процес розвитку програмування в контексті вдосконалення апаратної бази комп'ютерів, їх поширення, зростання кількості, складності і важливості завдань, що вирішуються	[2, с. 48 – 67]
	1.2	Ознайомитись з матеріалами рекомендацій SE2004 та CC2020	Прослідити зміни парадигм комп'ютерного навчання у зв'язку з розвитком інформаційних технологій. Опанування англомовної професійної термінології	[3 – 5, 9]
2	1.3	Ознайомитись з матеріалом Selecting a development approach	Опанування англомовної професійної термінології на прикладах розгляду моделей життєвого циклу	[12]

Продовження табл. 5

1	2	3	4	5
3	2.1	Ознайомитись з матеріалом глави «Побудова систем управління підприємством» з навчального посібника «Інформаційні системи управління підприємствами»	Розглянути питання визначення та аналізу вимог до програмного забезпечення на прикладі створення інформаційних систем управління підприємствами	[13, с. 96 – 105? 118 – 120]
	2.2		Розглянути принципи та методи проектування програмного забезпечення на прикладі створення інформаційних систем управління підприємствами	[13, с. 106 – 120]
4	2.3	Ознайомитись з матеріалом «Історія теоретичного програмування в Україні»	Коріння теоретичного програмування в Україні, викристалізовування основних прийомів програмування і проблем теоретичного програмування стосовно до автоматизації програмування.	[14, с. 45 – 51]
5	3.1	Ознайомитись з матеріалом статті «Оцінка вартості програмного забезпечення»	Оцінка вартості розробки програмних продуктів як важливий елемент забезпечення конкуренто-здатності на сучасному динамічному ринку.	[20]
6	3.2	Ознайомитись з матеріалом статті «Огляд програмних засобів та методологій для реалізації систем автоматизованого тестування»	Основи управління групами у галузі розроблення програмних систем. Чинники, що впливають на групову динаміку команд розробників. Адаптивне тестування знань претендентів для підбору у команди розробників програмних систем	[30]

Продовження табл. 5

1	2	3	4	5
7	3.3	Ознайомитись з матеріалом статті «Огляд і аналіз моделей надійності програмного забезпечення»	Класифікація моделей надійності за різними критеріями. Моделі надійності, що враховують явище недосконалого відлагодження	[31]
8	3.4	Ознайомитись з матеріалами з навчання використанню Microsoft Project, у тому числі відео-уроки на Youtube	Теоретичний матеріал з розділів «Основи теорії управління проектам», «Планування процесу реалізації ІТ-проекту». Керівництво по основним прийомам роботи в середовищі Microsoft Project	[38, с. 5-38, 45-88]

5. Засоби діагностики, критерії та процедури оцінювання лекційних модулів

Лекційні заняття оцінюються шляхом виміру якості виконання конкретизованих завдань.

Конкретизовані завдання формуються на основі узагальнених шляхом конкретизації вихідних даних та способу демонстрації результатів навчання. Узагальнені засоби діагностики розробляються на базі програмних результатів навчання за стандартами вищої освіти та подаються в робочій програмі.

Поточний контроль результатів навчання зручно здійснювати за допомогою тестів, а також бального оцінювання практичних робіт (рис. 6). За результатами поточних контролів з усіх видів навчальних занять максимальне оцінювання досягає 90 балів.

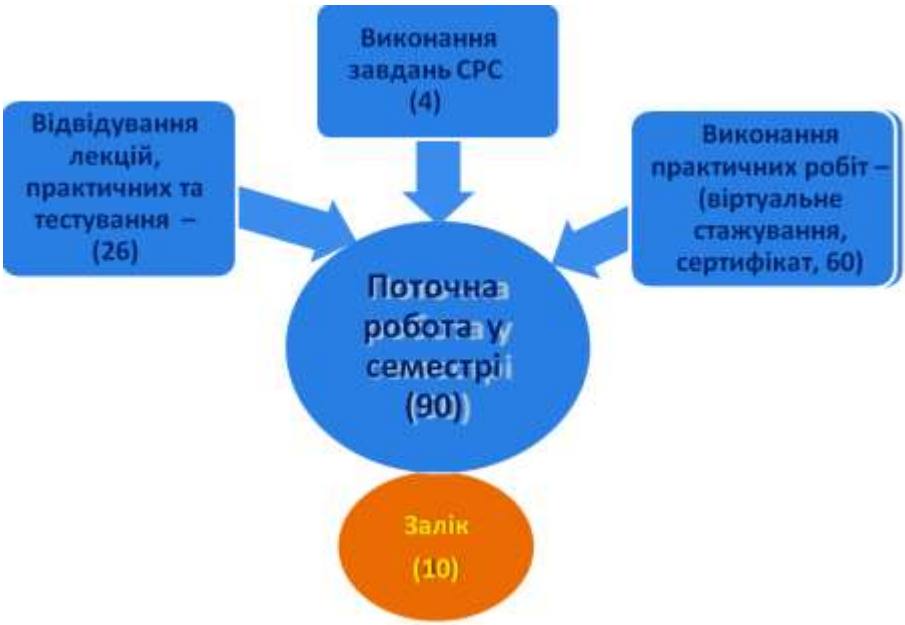

Рис. 6. Оцінювання поточної роботи студента у семестрі

Тестування засвоєння лекційного матеріалу відбувається в автоматизованому режимі на платформі Distance Learning - International European University (рис. 7). Кожен тест містить питання, пов'язані з ключовими дисциплінарними результатами навчання за даною лекцією. Кожен тест максимадбно оцінюється у 2 бали.

Рис. 7. Навчальні матеріали з дисципліни на платформі Distance Learning - International European University

Підсумковий контроль знань студентів проводиться у формі заліку.

Оцінювання відповіді під час заліку здійснюється в межах 0 – 10 балів та відбувається з використанням відповідних Критеріїв оцінювання (табл. 6). Залікова оцінка додається до оцінки поточної роботи, що представляє загальну підсумкову оцінку в балах за національною шкалою та за шкалою ECTS.

Таблиця 6

Оцінювання відповіді студента під час заліку на окреме питання з навчальної дисципліни

№ з/п	Оцінка	Кіль-кість балів	Критерії оцінювання відповіді
1	2	3	4
1	«відмінно»	8 - 10	розгорнутий, вичерпний виклад змісту вирішення заданої у питанні проблеми; повний перелік необхідних для розкриття змісту питання термінів та понять; порівняльний аналіз різних теорій, концепцій, підходів та самостійно зроблені логічні висновки й узагальнення; демонстрація здатності висловлення та аргументування власного ставлення до альтернативних поглядів на дане питання
2	«добре»	4 - 7	відносно відповіді на найвищий бал не зроблено розкриття хоча б одного з пунктів, вказаних вище (якщо він явно потрібний для вичерпного розкриття питання) або, якщо: при розкритті змісту питання в цілому правильно за зазначеними вимогами зроблені окремі помилки
3	«задовільно»	1-3	відносно відповіді на найвищий бал не зроблено розкриття пунктів, зазначених у вимогах до нього; висновки, зроблені під час відповіді, не відповідають правильним чи загально визначеним при відсутності доказів супротивного аргументами, зазначеними у відповіді; характер відповіді дає підставу стверджувати, що студент не зовсім правильно розуміє зміст питання чи не знає правильної відповіді і тому не відповів на нього по суті, допустивши грубі помилки у змісті відповіді

6. Глосарій

Аналіз вимог (Analysis of requirements) - відображення функцій системи і її обмежень в моделі вимог домену.

Артефакт (Artifacts) - будь-який продукт діяльності фахівців при розробці ПЗ.

Архітектура системи (Systems arhitecture) - визначення системи в термінах обчислюваних складових (підсистем) і інтерфейсів між ними, яке відображає правила декомпозиції проблеми на складові.

Аспектно-орієнтоване програмування (Aspect-oriented programming) - різновид складального програмування, заснована на збірці повнофункціональних додатків з багатоаспектних компонентів, інкапсулюючих різні варіанти реалізації.

Безвідмовність (Faultless) - атрибут надійності, який визначає відсутність або частоту відмов через помилки в програмному забезпеченні.

Валідація (Validation) - забезпечення відповідності розробки продукту вимогам її замовників.

Варіабельність (Variability) - здатність продукту (системи) до розширення, зміни, пристосування або конфігурації з метою використання в певному контексті і забезпечення його еволюції.

Верифікація (Verification) - перевірка правильності проекту в ході розробки.

Водоспадна (або каскадний) модель (Cascade model) - схема робіт, в якій кожна з робіт виконується один раз і в тому порядку, в якому роботи перераховані в моделі ЖЦ.

Відновлюваність (Restored) – атрибут надійності, який вказує на можливість до перезапуску програми для повторного виконання і відновлення даних після відмов.

Дефект (Fault) - результат помилок розробника у вхідних або проектних специфікаціях, текстах кодів програм, експлуатаційной документації і т.п

Динамічний метод тестування (Dynamic method of testing) - виконання програм з метою встановлення причин помилок з очікуваними реакціями на ці помилки.

Екстремальне програмування (Extreme programming, XP) - технологія програмування, яка багаторазово проходить цикл створення системи від аналізу потреб замовника до комплексного тестування та виготовлення готової системи. У функціонуючу систему послідовно додаються різні протестовані «можливості» (facilities), узгоджені з замовником, і цикл повторюється.

Імперативне програмування (Imperative programming) засноване на ідеї абстрактного обчислювача, що виконує послідовність команд програми, утворюючи деякий стан, яке змінюється при переході до іншого послідовності команд.

Інженерія якості (Quality Engineering) - процес надання програмному продукту заданих характеристик якості.

Інженерія вимог (Requirements Engineering) - збір, аналіз вимог замовника та подання їх в нотації, яка є зрозумілою і узгодженої як замовником, так і виконавцем

Компонент (Component) - тип, клас, проектне рішення, документація або інший продукт програмної інженерії, спеціально пристосований для повторного використання.

Компонент повторного використовуваний (Reuses) - КПВ - фрагмент знань про минулий досвіді розробки програмних систем, або артефакт, який можна адаптувати під час створення нових систем

Компонентна програма (Component program) - сукупність компонентів, які необхідні для забезпечення функціональних і не функціональних вимог, і яка побудована і функціонує відповідно до правил створення компонентних конфігурацій і взаємодії.

Компонентне програмування (Component programming) - підхід до створення компонентних програм з готових компонент шляхом складання компонентної програми на всіх етапах ЖЦ.

Комунікативний процес – забезпечення отримання, сприйняття і розуміння інформації, що є предметом обміну

Контейнер (Container) - оболонка, всередині якої реалізовані функції у вигляді примірників компонентів і яка забезпечує взаємодію з сервером через стандартні інтерфейси.. Примірники звертаються один до одного через системні сервіси даного або іншого контейнера.

Конфігурація (Configuration) - структура програмної системи з компонентів і набору варіантів модулів.

Концептуальне моделювання (Conceptual design) - процес побудови моделі проблеми, орієнтованої на розуміння її людиною.

Модель життєвого циклу (Life cycle model) - типова схема послідовності робіт на етапах розробки програмного продукту.

Модуль (Module) - незалежна функціональна частина програми, до якої можна звертатися як до самостійної одиниці через зовнішній інтерфейс.

Модульне програмування (Modular programming) - метод розробки програм, заснований на розбитті проблеми на незалежні частини - модулі, які реалізують окремі функції і їх інтерфейси (вхідні та вихідні дані) для взаємодії з іншими модулями, зібраними в велику програму.

Надійність (Reliability) - набір атрибутів, які вказують на можливість програми перетворювати вхідні дані в вихідні результати за умов, які залежать від періоду часу (знос і старіння не враховуються) і наявності відмов в результаті внутрішніх дефектів або порушення умов його застосування

Нефункціональні вимоги (Unfunctional requirements) - вимоги, які характеризують організаційні, виконавчі, операційні та безпечні аспекти роботи програмної системи.

Об'єктно орієнтована модель (Object-oriented model) - уявлення програм системи як сукупності об'єктів, які взаємодіють між собою і мають певні властивості і поведінку.

Оцінка якості (Quality estimation) - дії, спрямовані на визначення ступеня задоволення програмного забезпечення потреб відповідно до їх призначення.

Помилка (Mistake) - недоліки в операторах програми або в технологічному процесі розробки, що призводять до неправильної інтерпретації вхідної інформації, а отже, і до неправильного вирішення.

Переносимість (Bearable) - група властивостей, яка забезпечує пристосованість до переносу з одного середовища функціонування в інші, а також зусилля для перенесення програм в нове середовище.

Сервісно орієнтоване програмування (Service oriented programming) - сукупність взаємодіючих сервісів, веб-сервісу і їх інтерфейсів.

Сертифікація продукту (Certification of product) - процес, який підтверджує відповідність ідентифікованої програмної продукції (процесу або послуг) конкретному стандарту або технічним умовам з оцінкою спеціальним знаком або свідоцтвом.

Складальне програмування (Assmbling programming) - метод збирання різнорідних готових програмних ресурсів (модулів, об'єктів, компонентів, reuses, asets, services і ін.) Багаторазового застосування через їх інтерфейсні дані, якими обмінюються модулі між собою

Спадне (top down) програмування (програмування «зверху вниз») - методика розробки програм, при якій розробка починається з визначення цілей проблеми, після чого йде їх деталізація і опис в мовах програмування.

Специфікація (Specification) - уявлення правил, стандартів, критеріїв якості, обмежень користування об'єктами.

Структурне програмування (Structured programming) - методологія і технологія розробки за методом «зверху вниз» окремих

блоків системи, структура яких задається графічними елементами: послідовність, розгалуження і цикл.

Супровід (Accompaniment) - будь-які роботи по внесенню змін до ПС після того, як вона була передана користувачеві для експлуатації.

Тест (Test) - деяка програма, призначена для перевірки працездатності іншої програми і виявлення в ній помилкових ситуацій.

Тестування (Testing) - засіб семантичної налагодження (перевірки) програми, яка полягає в опрацюванні програмою послідовності різних контрольних наборів тестів, для яких відомий результат.

Техніка функціонального .моделювання SADT (Structured analysis and design technique) заснована на специфікації функцій системи у вигляді діаграм, які задають фрагменти опису програм і їх інтерфейсів (вхідні та вихідні дані) з іншими функціональними програмами.

Функціональні вимоги (Functional requirements) - вимоги, які визначають цілі і функції системи.

Функціональність (Functionality) - сукупність властивостей, які визначають можливість ПО виконувати перелік функцій в заданому середовищі і відповідають вимогам.

Якість програмного забезпечення (Software quality) - сукупність властивостей, які визначають можливість програмного забезпечення задовольнити замовника, який сформулював ці властивості як вимоги до розробки.

Література

1. Салов В.О., Письменкова Т.О. Рекомендації до створення комплексу навчально-методичного забезпечення дисциплін: метод. посіб. для наук.-пед. прац. Нац. гірн. ун-т, наук. метод. центр. Д. : НГУ, 2017. 48 с.
2. Будько М.М., Нестеренко О.В., Нетесін І.Є. Вільне програмне забезпечення: український вибір. К.: Альтерпрес, 2011. 400 с
3. Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 4.0. IEEE Computer Society, 2024.
4. Software Engineering 2004. Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. The Joint Task Force on Computing Curricula IEEE Computer Society Association for Computing Machinery. 2004. 135 p.
5. Computing Curricula 2020 (CC2020). Paradigms for Global Computing Education. Association for Computing Machinery (ACM) IEEE Computer Society (IEEE-CS). 2020. 205 с.
6. Pressman Roger S. Software engineering: a practitioner’s approach. 7th ed. 2010. 928 p.
7. Tom DeMarco, Timothy Lister. Peopleware: Productive Projects and Teams. 2013. 272 p.
8. Нестеренко О., Проскура С. Порівняльний аналіз освітніх програм в галузі інженерії програмного забезпечення. Інформаційні технології та суспільство, 2022, (2 (4)), 70-77.
9. Nesterenko O.V. Technological trends & software engineering education: a systematic review study. Проблеми програмування. 2022. № 3-4. С. 107-116.
10. Лавріщева К.М. Програмна інженерія. Київ, 2008. 319 с.
11. Бородкіна І.Л., Бородкін Г.О. Інженерія програмного забезпечення. Навч. посібник. НУБіП. Київ: Центр учбов. літератури. 2020. 204 с.

12. Pfleeger S.L. Software Engineering. Theory and practice. Printice Hall: Upper Saddenle River, New Jersey, 1998. 576 p.

13. Нестеренко О.В. Інформаційні системи управління підприємствами / Навч. посіб. Київ: УкрНЦ РІТ, 2019. 135 с.

14. Малиновський Б. М. Відоме і невідоме в історії інформаційних технологій в Україні». Київ, "Інтерлінк", 2004, 216 с.

15. Мартін Роберт. Чистий AGILE. Фабула. 2021. 224 с.

16. Martin Robert C. Agile Software Development, Principles, Patterns, and Practices. Pearson Education Limited 2014..

17. Kent Beck with Cynthia Andres. Extreme programming explained: embrace change. 2nd ed. 2005. 176 p.

18. Nesterenko, O. Trofymchuk, O. Decision Support for Requirements Prioritization in Software Engineering. CEUR Workshop Proceedings, 2024, 3806, 50–61.

19. Jacobson I. Object-Oriented Software Engineering. A use Case Driven Approach, Revised Printing. New York: Addison-Wesley Publ. Co., 1994. 529 p.

20. Рябокінь Ю. М. Оцінка вартості програмного забезпечення. Вісник КНУТД, Серія «Технічні науки». 2015, №1 (82). С. 117-124.

21. Erich Gamma ... [et al.]. Design Patterns: elements of reusable object-oriented software. Addison-Wesley professional computing series. 1996. 417 p.

22. Jez Humble, David Farley. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. 2010, 512 p.

23. Evans David C. Bottlenecks: Aligning UX Design with User Psychology. 2017, 281 p.

24. Steve McConnell. Code Complete: A Practical Handbook of Software Construction, Second Edition, 2016. 916 p.

25. Yablonski Jon. Laws of UX: Using Psychology to Design Better Products & Services. 2024, 164 p.

26. Nir Eyal. Hooked. How to Build Habit-Forming Products. 2014, 256 p.

27. Скібіцька Л.І. Організація праці менеджера. Навч. посібник. К.: Центр учбової літератури, 2010. 360 с.

28. Верес О. М. Технології підтримання прийняття рішень. Навч. посіб. Друге видання. Львів: Видавництво Львівської політехніки, 2013. 252 с.

29. Білас О. Є. Якість програмного забезпечення та тестування. Навч. посіб. Львів: Видавництво Львівської політехніки, 2011. 216 с.

30. Д’яченко Л.І., Ілін А.В., Шумиляк Л.М. та ін. Огляд програмних засобів та методологій для реалізації систем автоматизованого тестування. Вчені записки ТНУ імені В.І. Вернадського. Серія: Т ехнічні науки. 2021, Том 32 (71) Ч. 1 № 2. С. 122-129.

31. Яковина В., Мацелюх В. Огляд і аналіз моделей надійності програмного забезпечення. Вісник Національного університету "Львівська політехніка". Комп’ютерні науки та інформаційні технології. 2017. № 864. С. 130-140.

32. People Capability Maturity Model (PCMM). Version 2.0. CMU/SEI-2001-MM-01 / Bill Curtis, William E. Hefley, Sally A. Miller. Carnegie Mellon University (Software Engineering Institute). Pittsburgh, 2001. 735 p.

33. Brooks Fred. The Mythical Man-Month: Essays on Software Engineering. 1995.

34. Дж. Ханк Рейнвотер. Як пасти котів. Фабула, 2020. 320 с.

35. Тарасюк Г.М. Управління проектами: Навч. посібник. 3-є вид. Київ: Каравела, 2009. 320 с.

36. Колпаков В. М. Самоменеджмент: навч. посібник. К.: Персонал, 2008. 528 с.

37. Гринченко М. А., Колісник М. Е. Управління проектом з використанням Microsoft Project:. Навч.-метод. посіб.. Харків: НТУ «ХПІ», 2012. 76 с.

38. Добровська Л. М. Аверьянова О. В. Управління ІТ-проектами в Microsoft Project. Комп’ютерний практикум [Електронний ресурс] : навчальний посібник. КПІ ім. Ігоря Сікорського. Київ : КПІ ім. Ігоря Сікорського, 2020. 152 с.

39. Калашнікова С. Імплементація парадигми вдосконалення викладання в університетах України: інституційний та індивідуальний виміри. ITeachers Meet-Up #6. Інститут вищої освіти НАПН України, 2022. 27 с.

40. Нестеренко О. Інфологічне моделювання простору освіти з інженерії програмного забезпечення. Проблеми освіти, 2022, 2(97), 147-168.

Навчально-методичне видання

Нестеренко Олександр Васильович

ОСНОВИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Методичні рекомендації
до опанування лекційних занять
з навчальної дисципліни
для здобувачів вищої освіти спеціальності
121 «Інженерія програмного забезпечення»

Міжнародний європейський університет

Навчально-науковий інститут
«Європейська школа бізнесу»

Кафедра інформаційних технологій

Відповідальний за випуск
Ліщенко А.В.