Робоча програма дисципліни «Інженерія програмного забезпечення»

СПЕЦІАЛЬНІСТЬ 123 · КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

БАКАЛАВР

120

Годин

Загальний обсяг дисципліни

4

Кредити ЄКТС

Відповідно до стандарту

4

Модулі

Змістовних блоки курсу

Загальна інформація про дисципліну

Ідентифікація

Назва: Інженерія програмного забезпечення

Спеціальність: 123 Комп'ютерна інженерія

Ступінь: Бакалавр

Обсяг: 120 годин / 4 кредити ЄКТС

Модулів: 4 змістовних блоки

Лекції: 45 год. / **Лабораторні:** 15 год. / **СРС:** 60 год.

Призначення дисципліни

Дисципліна «Інженерія програмного забезпечення» формує у студентів системне розуміння принципів, методів та інструментів промислового розроблення програмних систем — від аналізу вимог до розгортання та супроводу продукту. Курс охоплює ключові процеси, моделі, методології та стандарти галузі, розвиває практичні навички проєктування, кодування, тестування та управління програмними проєктами.

Дисципліна є обов'язковою для підготовки бакалаврів зі спеціальності «Комп'ютерна інженерія» та забезпечує формування фахових компетентностей, передбачених освітньо-професійною програмою.

Загальні компетентності

Аналітичне мислення

Здатність до аналізу, синтезу та критичного оцінювання технічної інформації: виявлення причинно-наслідкових зв'язків у програмних системах, порівняння альтернативних архітектурних рішень, формування обґрунтованих інженерних висновків.

Комунікація

Здатність до усної та письмової технічної комунікації: чітко пояснювати архітектурні рішення, складати технічну документацію, специфікації вимог, звіти та аналітичні записки відповідно до стандартів галузі.

Самоорганізація

Уміння самостійно навчатися, планувати розробку програмного забезпечення та організовувати виконання завдань у визначені строки, зокрема під час лабораторних робіт і командних проєктів.

Командна робота

Відповідальність, ініціативність та готовність працювати в команді під час групових проєктів з розробки ПЗ, розподілу ролей (аналітик, розробник, тестувальник, менеджер) та колективного прийняття технічних рішень.

Студент повинен вміти: аналізувати технічну та проєктну інформацію; виділяти головне та узагальнювати результати досліджень; планувати власну розробницьку діяльність і дотримуватися дедлайнів; коректно спілкуватися з замовниками, колегами та викладачами; брати участь у командній роботі та розподіляти обов'язки в проєкті.

Фахові компетентності

Розуміння ІПЗ як дисципліни

Розуміння сутності та закономірностей розвитку інженерії програмного забезпечення як інженерної дисципліни. Знання ключових моделей, методологій і процесів ЖЦ ПЗ. Уміння застосовувати системний підхід до проєктування програмних систем різного масштабу.

Аналіз вимог та проєктування

Здатність збирати, аналізувати та документувати вимоги до програмного забезпечення; розробляти архітектурні та детальні проєкти з використанням мови UML, патернів проєктування та принципів SOLID, GRASP.

Розробка та тестування ПЗ

Знання методів та інструментів розробки якісного програмного коду, технік тестування (модульне, інтеграційне, системне), рефакторингу та забезпечення якості ПЗ. Розуміння зв'язку між якістю коду і надійністю системи.

Управління проєктами та стандарти

Орієнтування в стандартах ІПЗ (ISO/IEC, IEEE); застосування agile та класичних методологій управління проєктами; оцінювання ризиків, трудомісткості та термінів виконання проєктних завдань.

Результати навчання

01

Пояснення концепцій ІПЗ

Пояснювати сутність ключових понять і явищ: життєвий цикл ПЗ, вимоги, архітектура, патерн проєктування, рефакторинг, тестування, конфігураційне управління, якість, DevOps, agile.

02

Характеристика моделей та процесів

Характеризувати основні моделі ЖЦ та процеси розробки ПЗ, застосовувати методи порівняльного аналізу методологій, визначати відповідну методологію для конкретного типу проєкту.

03

Проєктування та реалізація

Розробляти специфікації вимог (SRS), UML-діаграми, архітектурні схеми програмних систем; реалізовувати окремі модулі ПЗ з дотриманням принципів якісного коду та патернів проєктування.

04

Тестування та оцінювання якості

Розробляти та виконувати тест-кейси для модульного й інтеграційного тестування; оцінювати якість програмних систем за метриками; застосовувати інструменти статичного аналізу та CI/CD.

МОДУЛЬ 1

Основи інженерії програмного забезпечення та процеси розробки

1

Тема 1.1

Введення в ІПЗ. Визначення, предмет, завдання. Програмне забезпечення як продукт. Стандарти SWEBOK.

2

Тема 1.2

Моделі та стандарти життєвого циклу ПЗ. Каскадна, ітеративна, спіральна моделі. ISO/IEC 12207.

3

Тема 1.3

Agile-методології розробки ПЗ. Scrum, Kanban, XP, SAFe. Маніфест Agile.

4

Тема 1.4

Управління програмними проєктами. Планування, оцінювання, ризики. Інструменти: Jira, Trello, MS Project.

МОДУЛЬ 1 · ТЕМА 1.1

Введення в інженерію програмного забезпечення

Лекційний матеріал (3 год.)

Лекція 1: Поняття інженерії програмного забезпечення (ІПЗ). Відмінність між програмуванням і програмною інженерією. Властивості програмного забезпечення як особливого продукту: нематеріальність, складність, змінність. Кризи програмного забезпечення та причини їх виникнення.

Лекція 2: SWEBOK — зведення знань з інженерії програмного забезпечення. Огляд галузей знань: вимоги, проєктування, конструювання, тестування, супровід, конфігурація, управління, процеси, якість. Огляд міжнародних стандартів: ISO/IEC 12207, ISO/IEC 25010, IEEE Std 830.

Ключові питання лекції

- Визначення ІПЗ та її місце серед інших дисциплін
- Особливості ПЗ як продукту інженерної діяльності
- Причини та прояви кризи програмного забезпечення
- Структура SWEBOK та галузі знань ІПЗ
- Роль міжнародних стандартів у практиці розробки

Лабораторна робота (1 год.)

Завдання 1. Ознайомлення з середовищем розробки та інструментами: встановлення та налаштування IDE (VS Code / IntelliJ IDEA), системи контролю версій Git, платформи GitHub/GitLab. Створення першого репозиторію проєкту.

Завдання 2. Аналіз прикладу реального програмного продукту: визначення його характеристик якості за ISO/IEC 25010 (функціональність, надійність, зручність використання тощо).

Самостійна робота (4 год.)

- Опрацювання: Цибульник С. О., Барандич К. С. «Технології розроблення програмного забезпечення. Ч. 1: Життєвий цикл ПЗ» — Київ: КПІ ім. Ігоря Сікорського, 2022. — розділ «Основи ІПЗ»
- Підготовка конспекту з ключових визначень SWEBOK (не менше 15 термінів)
- Огляд стандарту ISO/IEC 12207: структура та основні процеси ЖЦ ПЗ

Очікуваний результат: студент вміє пояснювати сутність ІПЗ, описувати галузі знань SWEBOK та характеристики якості ПЗ за міжнародними стандартами.

МОДУЛЬ 1 · ТЕМА 1.2

Моделі та стандарти життєвого циклу ПЗ

1

Каскадна модель

Послідовне виконання фаз: вимоги → проєктування → реалізація → тестування → супровід. Чіткі точки контролю.

2

Ітеративна модель

Поступове нарощування функціоналу через ітерації. RUP — Rational Unified Process як приклад.

3

Спіральна модель

Боема: кожен цикл — аналіз ризиків + прототипування. Орієнтована на управління ризиками.

4

ISO/IEC 12207

Міжнародний стандарт процесів ЖЦ ПЗ: основні, допоміжні та організаційні процеси розробки.

Лекційний матеріал (3 год.)

Поняття життєвого циклу програмного забезпечення. Основні фази: концепція, вимоги, проєктування, реалізація, тестування, впровадження, супровід. Каскадна (Waterfall) модель: переваги, недоліки, сфера застосування. Ітеративна і еволюційна моделі. Спіральна модель Боема: аналіз ризиків як ключовий елемент. Інкрементна модель. Порівняльна характеристика моделей. Стандарт ISO/IEC 12207: групи процесів і їх взаємодія.

Лабораторна робота (1 год.) та СРС (4 год.)

Лабораторна: Побудова діаграми Ганта для навчального проєкту з розробки ПЗ з використанням інструменту MS Project або Trello. Визначення фаз, контрольних точок і відповідальних осіб у команді.

СРС: Опрацювання: Трофименко О. Г., Манаков С. Ю., Ларін Д. Г. «Основи програмної інженерії» — Одеса: Фенікс, 2022. — розділ «Моделі ЖЦ ПЗ»; підготовка порівняльної таблиці моделей ЖЦ (критерії: вартість змін, участь замовника, підхід до ризиків, тривалість).

МОДУЛЬ 1 · ТЕМА 1.3

Agile-методології розробки програмного забезпечення

Лекційний матеріал (3 год.)

Передумови виникнення Agile: обмеження важких методологій. Маніфест Agile (2001): 4 цінності та 12 принципів. Scrum: ролі (Product Owner, Scrum Master, команда), артефакти (Product Backlog, Sprint Backlog, Increment), події (Sprint, Daily, Review, Retrospective). Kanban: принципи, WIP-ліміти, Kanban-дошка. Extreme Programming (XP): TDD, парне програмування, рефакторинг, безперервна інтеграція. Scaled Agile Framework (SAFe) для великих організацій. Порівняння Scrum vs Kanban vs XP.

Ключові питання

- Цінності та принципи Маніфесту Agile
- Ролі, артефакти та події Scrum
- Відмінності Kanban від Scrum
- Практики XP та їх переваги
- Вибір методології для конкретного типу проєкту

Лабораторна робота (1 год.)

Завдання 1. Організація Scrum-спринту для навчального проєкту: створення Product Backlog, оцінювання User Story методом Planning Poker, планування першого Sprint Backlog.

Завдання 2. Налаштування Kanban-дошки в Trello для командного проєкту: визначення стовпців, WIP-лімітів, пріоритетів завдань.

Самостійна робота (4 год.)

- Опрацювання: Марченко О. І. «Компоненти програмної інженерії. Ч. 1: Вступ до програмної інженерії» — Київ: КПІ ім. Ігоря Сікорського, 2022. — розділ «Гнучкі методології»
- Підготовка реферату: «Порівняльний аналіз Scrum та Kanban: критерії вибору методології»
- Складання схеми Scrum-процесу із зазначенням артефактів і подій

Очікуваний результат: студент вміє характеризувати agile-методології, порівнювати Scrum, Kanban і XP та обирати відповідну методологію для заданих умов проєкту.

МОДУЛЬ 1 · ТЕМА 1.4

Управління програмними проєктами

Лекційний матеріал (3 год.)

Поняття програмного проєкту та його характеристики. Планування проєкту: WBS (Work Breakdown Structure), мережеві графіки (CPM, PERT), діаграма Ганта. Оцінювання обсягу та трудомісткості: метод функційних точок, COCOMO II. Управління ризиками: ідентифікація, аналіз, реагування, моніторинг. Управління командою: ролі, комунікація, мотивація. Управління конфігурацією та змінами. Інструменти: Jira, Confluence, MS Project, GitLab CI.

Ключові питання

- Структура WBS та принципи декомпозиції
- Методи оцінювання трудомісткості проєкту
- Ідентифікація та управління ризиками
- Управління конфігурацією та змінами
- Інструментарій управління програмними проєктами

Лабораторна робота (1 год.)

Завдання 1. Розробка WBS для навчального проєкту: декомпозиція завдань, визначення залежностей, побудова мережевого графіка в Jira або MS Project.

Завдання 2. Ідентифікація та аналіз ризиків проєкту: заповнення реєстру ризиків (опис, імовірність, вплив, стратегія реагування).

Самостійна робота (4 год.)

- Опрацювання: Трофименко О. Г., Манаков С. Ю., Ларін Д. Г. «Основи програмної інженерії» — Одеса: Фенікс, 2022. — розділ «Управління проєктами»
- Підготовка аналітичної записки «Управління ризиками в IT-проєктах: методи та інструменти»
- Складання плану навчального проєкту з визначенням контрольних точок і критичного шляху

Очікуваний результат: студент вміє розробляти WBS, оцінювати трудомісткість, ідентифікувати ризики та застосовувати інструменти управління програмними проєктами.

МОДУЛЬ 2

Аналіз вимог та проєктування програмних систем

Тема 2.1

Інженерія вимог. Збір, аналіз, специфікація та управління вимогами. Стандарт IEEE 830. Документ SRS.

Тема 2.2

Моделювання програмних систем. UML: діаграми класів, послідовності, варіантів використання, компонентів, діяльності.

Тема 2.3

Архітектурне проєктування ПЗ. Архітектурні стилі: MVC, мікросервіси, REST, клієнт-сервер. Принципи SOLID.

Тема 2.4

Патерни проєктування. Creational, Structural, Behavioral патерни (GoF). Принципи GRASP. Антипатерни.

МОДУЛЬ 2 · ТЕМА 2.1

Інженерія вимог до програмного забезпечення

Лекційний матеріал (3 год.)

Поняття вимог до ПЗ: функціональні та нефункціональні вимоги. Рівні вимог: бізнес-вимоги, вимоги стейкхолдерів, системні вимоги. Методи збору вимог: інтерв'ю, анкетування, мозковий штурм, аналіз документів, спостереження, прототипування. User Stories та Use Cases як формати специфікації вимог. Стандарт IEEE 830 та структура документа SRS (Software Requirements Specification). Управління вимогами: трасування, пріоритизація (MoSCoW), управління змінами.

Ключові питання

- Класифікація вимог: функціональні, нефункціональні, обмеження
- Техніки виявлення та збору вимог
- Структура документа SRS за IEEE 830
- User Stories: формат, критерії прийнятності (INVEST)
- Трасування та управління змінами вимог

Лабораторна робота (1 год.)

Завдання 1. Збір та специфікація вимог для навчального проєкту: проведення інтерв'ю з «замовником» (викладачем), складання переліку User Stories з критеріями прийнятності.

Завдання 2. Складання структури документа SRS для навчального проєкту: розділи, функціональні та нефункціональні вимоги, обмеження, глосарій.

Завдання 3. Пріоритизація вимог за методом MoSCoW.

Самостійна робота (4 год.)

- Опрацювання: Марченко О. І. «Компоненти програмної інженерії. Ч. 1» — КПІ ім. Ігоря Сікорського, 2022. — розділ «Інженерія вимог»
- Підготовка повного документа SRS для навчального проєкту
- Складання матриці трасування вимог (Requirements Traceability Matrix)

Очікуваний результат: студент вміє збирати, аналізувати та специфікувати вимоги до ПЗ у форматі User Stories та SRS-документа.

МОДУЛЬ 2 · ТЕМА 2.2

Моделювання програмних систем засобами UML

Use Case

Діаграма варіантів використання: актори, прецеденти, відносини include/extend.

Class Diagram

Діаграма класів: атрибути, методи, відносини (асоціація, агрегація, композиція, успадкування).

Sequence

Діаграма послідовності: взаємодія об'єктів у часі, повідомлення, lifecycle.

Activity

Діаграма діяльності: бізнес-процеси, паралельні потоки, умови.

Component / Deploy

Компонентні та розгортання: фізична архітектура системи, вузли та залежності.

Лекційний матеріал (3 год.)

Уніфікована мова моделювання (UML 2.x): призначення, класифікація діаграм. Структурні діаграми: класів, об'єктів, компонентів, розгортання. Поведінкові діаграми: варіантів використання, діяльності, станів, послідовності, комунікації. Засоби моделювання: draw.io, Visual Paradigm, Enterprise Architect, PlantUML. Зв'язок UML-моделей із вимогами та кодом.

Лабораторна робота (1 год.) та СРС (4 год.)

Лабораторна: Розробка комплекту UML-діаграм для навчального проєкту в draw.io або PlantUML: діаграма варіантів використання, діаграма класів (не менше 5 класів), діаграма послідовності для одного сценарію.

СРС: Опрацювання: Цибульник С. О., Барандич К. С. «Технології розроблення ПЗ. Ч. 1» — КПІ ім. Ігоря Сікорського, 2022. — розділ «UML-моделювання»; розробка діаграм діяльності та компонентів для навчального проєкту.

МОДУЛЬ 2 · ТЕМИ 2.3–2.4

Архітектурне проєктування та патерни

Тема 2.3 — Архітектурне проєктування (Лекція, 3 год.)

Поняття архітектури ПЗ. Архітектурні стилі та зразки: клієнт-сервер, шарова (layered), мікросервісна, подієво-орієнтована (event-driven), REST. Принципи SOLID: Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion. Принципи GRASP: Creator, Information Expert, Low Coupling, High Cohesion. Якісні атрибути архітектури: масштабованість, відмовостійкість, безпека, продуктивність.

Лабораторна (1 год.): Проєктування архітектури навчального застосунку: вибір архітектурного стилю, обґрунтування, розробка компонентної діаграми та опис інтерфейсів між компонентами.

СРС (4 год.): Опрацювання: Трофименко О. Г. та ін. «Основи програмної інженерії» — Одеса: Фенікс, 2022. — розділ «Архітектура ПЗ»; підготовка реферату «Мікросервісна архітектура: переваги, виклики та кейси застосування».

Тема 2.4 — Патерни проєктування (Лекція, 3 год.)

Поняття патерну проєктування. Класифікація GoF (Gang of Four): породжувальні (Singleton, Factory, Abstract Factory, Builder, Prototype), структурні (Adapter, Decorator, Facade, Proxy, Composite), поведінкові (Observer, Strategy, Command, Iterator, Template Method). Антипатерни: God Object, Spaghetti Code, Golden Hammer. GRASP-патерни та їх застосування в об'єктно-орієнтованому проєктуванні. Вибір патерну відповідно до контексту задачі.

Лабораторна (1 год.): Реалізація 3–4 патернів GoF у навчальному проєкті (мовою Java або Python): написання коду з коментарями та демонстрація результату.

СРС (4 год.): Підготовка каталогу патернів проєктування: для кожного з 10 обраних патернів — опис, UML-діаграма, приклад застосування на псевдокоді.

МОДУЛЬ 3

Конструювання, тестування та забезпечення якості ПЗ

Тема 3.1

Конструювання ПЗ. Принципи написання якісного коду. Clean Code, Code Review, рефакторинг.

Тема 3.2

Тестування програмного забезпечення. Рівні, типи та техніки тестування. TDD. Інструменти автоматизації.

Тема 3.3

Забезпечення якості ПЗ (SQA). Метрики якості. Статичний аналіз. Стандарт ISO/IEC 25010.

Тема 3.4

Безперервна інтеграція та доставка (CI/CD). DevOps-практики. Інструменти: Jenkins, GitHub Actions, Docker.

МОДУЛЬ 3 · ТЕМА 3.1

Конструювання програмного забезпечення

Лекційний матеріал (3 год.)

Поняття конструювання ПЗ як фази ЖЦ. Принципи написання якісного коду (Clean Code): змістовні імена, малі функції, коментарі, форматування, обробка помилок. Принципи DRY (Don't Repeat Yourself), KISS (Keep It Simple, Stupid), YAGNI (You Aren't Gonna Need It). Рефакторинг: визначення, цілі, техніки (Extract Method, Rename Variable, Replace Magic Numbers). Code Review: цілі, процес, інструменти (GitHub PR, Gerrit). Технічний борг: причини виникнення, оцінювання, управління.

Ключові питання

- Принципи Clean Code та їх практичне застосування
- DRY, KISS, YAGNI — як ці принципи впливають на якість коду
- Техніки рефакторингу: коли і як застосовувати
- Організація ефективного Code Review
- Технічний борг: природа та управління

Лабораторна робота (1 год.)

Завдання 1. Аналіз «брудного» коду: ідентифікація порушень принципів Clean Code, виявлення запахів коду (code smells). Проведення рефакторингу з фіксацією змін у Git.

Завдання 2. Проведення Code Review для коду одногрупника: підготовка рецензії у форматі GitHub Pull Request із конкретними зауваженнями та пропозиціями.

Самостійна робота (4 год.)

- Опрацювання: Марченко О. І. «Компоненти програмної інженерії. Ч. 1» — КПІ ім. Ігоря Сікорського, 2022. — розділ «Конструювання ПЗ»
- Підготовка реферату «Рефакторинг як безперервний процес підтримки якості коду»
- Виконання рефакторингу власного навчального коду з документуванням 5+ застосованих технік

Очікуваний результат: студент вміє застосовувати принципи Clean Code, виконувати рефакторинг та проводити Code Review відповідно до стандартів галузі.

МОДУЛЬ 3 · ТЕМА 3.2

Тестування програмного забезпечення

Модульне тестування (Unit)

Тестування окремих модулів/функцій. Фреймворки: JUnit (Java), pytest (Python), NUnit (.NET). Техніка TDD.

Інтеграційне тестування

Тестування взаємодії компонентів: стратегії Big Bang, Top-Down, Bottom-Up. Моки та стаби.

Системне та приймальне

Тестування системи загалом: функціональне, навантажувальне, безпеки. UAT — приймальне тестування замовником.

Лекційний матеріал (3 год.)

Поняття тестування, відлагодження, верифікації та валідації ПЗ. Рівні тестування: модульне, інтеграційне, системне, приймальне. Типи тестування: функціональне, нефункціональне (навантажувальне, безпеки, зручності використання). Техніки тест-дизайну: еквівалентне розбиття, граничні значення, таблиці рішень, попарне тестування. Test-Driven Development (TDD): цикл Red-Green-Refactor. Автоматизація тестування: Selenium, pytest, JUnit. Тест-план та тест-кейси.

Лабораторна робота (1 год.) та СРС (4 год.)

Лабораторна: Розробка та запуск модульних тестів для навчального проєкту (pytest або JUnit): написання не менше 10 тест-кейсів із перевіркою граничних значень; досягнення покриття коду ≥ 70%.

СРС: Опрацювання: Трофименко О. Г. та ін. «Основи програмної інженерії» — Одеса: Фенікс, 2022. — розділ «Тестування ПЗ»; підготовка тест-плану для навчального проєкту; реалізація TDD-циклу для однієї з функцій системи.

МОДУЛЬ 3 · ТЕМИ 3.3–3.4

Якість ПЗ та DevOps-практики

Тема 3.3 — Забезпечення якості ПЗ (Лекція, 3 год.)

Software Quality Assurance (SQA): поняття, цілі, процеси. Характеристики якості за ISO/IEC 25010: функціональна придатність, надійність, захищеність, ефективність виконання, зручність обслуговування, переносність. Метрики якості коду: цикломатична складність, щільність дефектів, покриття тестами, зв'язність (coupling), згуртованість (cohesion). Статичний аналіз коду: інструменти SonarQube, ESLint, Pylint, Checkstyle. Процес SQA: планування якості, верифікація, аудит.

Лабораторна (1 год.): Налаштування статичного аналізу коду в навчальному проєкті за допомогою SonarQube або Pylint: аналіз результатів, ідентифікація проблем, усунення критичних помилок та code smells.

СРС (4 год.): Опрацювання: Добровольський Ю. Г. «Стандартизація в інженерії програмного забезпечення» — Чернівці: ЧНУ ім. Ю. Федьковича, 2022. — розділи зі стандартом ISO/IEC 25010; підготовка аналітичної записки «Метрики якості ПЗ: підходи до вимірювання та інтерпретації».

Тема 3.4 — CI/CD та DevOps (Лекція, 3 год.)

DevOps: культура, принципи, практики. Безперервна інтеграція (CI): автоматизовані збірка та тестування при кожному commit. Безперервна доставка (CD): автоматизоване розгортання на тестові та продуктові середовища. Конвеєр CI/CD: стадії build, test, analyze, deploy. Інструменти: Jenkins, GitHub Actions, GitLab CI/CD, Travis CI. Контейнеризація: Docker — образи, контейнери, Dockerfile, Docker Compose. Основи Kubernetes для оркестрації. Моніторинг та логування: Prometheus, Grafana, ELK Stack.

Лабораторна (1 год.): Налаштування GitHub Actions для навчального проєкту: автоматичний запуск тестів при кожному push, перевірка статичного аналізу, збірка Docker-образу.

СРС (4 год.): Підготовка аналітичної записки «DevOps як культура та практика: вплив на швидкість і якість доставки ПЗ»; написання Dockerfile для навчального проєкту та документування процесу розгортання.

МОДУЛЬ 4

Супровід, безпека та сучасні технології розробки ПЗ

Тема 4.1

Супровід та еволюція програмного забезпечення. Типи супроводу. Технічний борг. Реінжиніринг.

Тема 4.2

Безпека програмного забезпечення. Захищене кодування. OWASP Top 10. Вразливості та методи захисту.

Тема 4.3

Сучасні технології та тренди ІПЗ. Хмарні обчислення, мікросервіси, AI-assisted development, Low-code/No-code.

Тема 4.4

Комплексний програмний проєкт. Підсумковий захист. Презентація розробленого програмного продукту.

МОДУЛЬ 4 · ТЕМА 4.1

Супровід та еволюція програмного забезпечення

Лекційний матеріал (3 год.)

Поняття супроводу ПЗ (software maintenance). Типи супроводу за ISO/IEC 14764: коригувальний (виправлення помилок), адаптивний (адаптація до змін середовища), вдосконалювальний (нові функції), превентивний (рефакторинг для майбутнього). Моделі супроводу. Технічний борг: природа, класифікація (Cunningham), накопичення та управління. Реінжиніринг: зворотна інженерія, редокументація, реструктуризація, реархітектурування. Управління конфігурацією: Git, GitFlow, семантичне версіонування (SemVer). Планування кінця ЖЦ продукту.

Ключові питання

- Класифікація та типи супроводу ПЗ
- Технічний борг: виникнення та стратегії зменшення
- Реінжиніринг застарілих систем: підходи та інструменти
- GitFlow та стратегії управління гілками
- Семантичне версіонування програмних продуктів

Лабораторна робота (1 год.)

Завдання 1. Реалізація стратегії GitFlow у навчальному проєкті: main, develop, feature, release, hotfix гілки; виконання злиття через Pull Request з перевіркою CI.

Завдання 2. Оцінювання технічного боргу в навчальному проєкті: аналіз звіту SonarQube (Debt), визначення пріоритетів для усунення, складання плану рефакторингу.

Самостійна робота (4 год.)

- Опрацювання: Цибульник С. О., Барандич К. С. «Технології розроблення ПЗ. Ч. 1» — КПІ ім. Ігоря Сікорського, 2022. — розділ «Супровід ПЗ»
- Підготовка реферату «Управління технічним боргом у великих IT-компаніях: кейси та методи»
- Складання хронологічної таблиці еволюції підходів до супроводу ПЗ (1960–2024)

Очікуваний результат: студент вміє характеризувати типи супроводу ПЗ, оцінювати технічний борг та застосовувати GitFlow у командній розробці.

МОДУЛЬ 4 · ТЕМА 4.2

Безпека програмного забезпечення

OWASP Top 10

Найпоширеніші вразливості вебзастосунків: Injection, Broken Auth, XSS, CSRF, Insecure Deserialization та ін.

Моделювання загроз

STRIDE-модель: Spoofing, Tampering, Repudiation, Info Disclosure, DoS, Elevation. Threat modeling як частина SDLC.

1 2 3 4

Захищене кодування

Принципи Secure Coding: валідація вхідних даних, принцип найменших привілеїв, шифрування, безпечне зберігання паролів.

DevSecOps

Інтеграція безпеки в CI/CD pipeline: SAST, DAST, Dependency Scanning, Secret Detection.

Лекційний матеріал (3 год.)

Поняття безпеки ПЗ: конфіденційність, цілісність, доступність (CIA Triad). OWASP Top 10: детальний огляд найпоширеніших вразливостей та механізми захисту. Принципи захищеного кодування (CERT Secure Coding Standards). Моделювання загроз (STRIDE, DREAD). Статичне (SAST) та динамічне (DAST) аналізування безпеки. Інтеграція безпеки в SDLC: Security by Design, DevSecOps. Огляд інструментів: OWASP ZAP, Snyk, Bandit, SonarQube Security.

Лабораторна робота (1 год.) та СРС (4 год.)

Лабораторна: Тестування безпеки навчального вебзастосунку за допомогою OWASP ZAP: виявлення вразливостей, класифікація за критичністю, розробка рекомендацій щодо усунення знайдених вразливостей.

СРС: Опрацювання: Добровольський Ю. Г. «Стандартизація в інженерії програмного забезпечення» — ЧНУ ім. Ю. Федьковича, 2022. — розділ «Стандарти безпеки ПЗ»; підготовка аналітичної записки «OWASP Top 10 у реальних проєктах: аналіз відомих інцидентів безпеки».

Очікуваний результат: студент вміє ідентифікувати вразливості ПЗ, застосовувати принципи захищеного кодування та використовувати інструменти аналізу безпеки.

МОДУЛЬ 4 · ТЕМА 4.3

Сучасні технології та тренди ІПЗ

Лекційний матеріал (3 год.)

Хмарні обчислення у розробці ПЗ: моделі IaaS, PaaS, SaaS; AWS, Azure, Google Cloud Platform. Мікросервісна архітектура: принципи, переваги, виклики. API Gateway, Service Mesh (Istio). Serverless Computing: функції як сервіс (FaaS). Штучний інтелект у розробці ПЗ: AI-assisted coding (GitHub Copilot, ChatGPT), генерація тестів, автоматичний Code Review. Low-code та No-code платформи. Квантові обчислення та їх перспективи для ІПЗ. Edge Computing та IoT-розробка.

Ключові питання

- Хмарні платформи та моделі розгортання ПЗ
- Мікросервіси vs моноліт: критерії вибору
- Застосування ШІ у процесах розробки ПЗ
- Low-code/No-code: можливості та обмеження
- Перспективні напрямки розвитку ІПЗ до 2030 р.

Лабораторна робота (1 год.)

Завдання 1. Розгортання навчального застосунку у хмарі: публікація Docker-образу на Docker Hub; розгортання на безкоштовному рівні хмарної платформи (Render, Railway, Vercel або аналог).

Завдання 2. Практика AI-assisted розробки: використання GitHub Copilot або аналогу для генерації коду, тестів, документації; критична оцінка якості згенерованого матеріалу.

Самостійна робота (4 год.)

- Опрацювання: Тищенко К. В. та ін. «Алгоритмічні мови програмування в інформаційних системах» — Суми: СумДУ, 2024. — розділи «Сучасні технології»
- Підготовка аналітичної записки «Тренди розробки ПЗ 2024–2026: огляд State of DevOps та Stack Overflow Developer Survey»
- Дослідження однієї хмарної платформи: архітектура, сервіси, порівняння з конкурентами

Очікуваний результат: студент вміє характеризувати сучасні тенденції ІПЗ, розгортати застосунки у хмарі та застосовувати AI-інструменти у процесі розробки.

МОДУЛЬ 4 · ТЕМА 4.4

Комплексний програмний проєкт — підсумковий захист

Узагальнення курсу (3 год. лекції)

Системне повторення та інтеграція знань усіх модулів курсу. Огляд повного SDLC: від вимог до розгортання та супроводу на прикладі реального проєкту. Типові помилки та найкращі практики в промисловій розробці ПЗ. Огляд кар'єрних шляхів: розробник, архітектор, тестувальник QA, DevOps-інженер, менеджер проєкту, аналітик вимог. Актуальні вимоги ринку праці в IT-галузі України. Підготовка до підсумкового іспиту.

Ключові питання підсумку

- Інтеграція всіх артефактів SDLC у єдиний проєкт
- Критерії якості готового програмного продукту
- Типові помилки молодших розробників та їх уникнення
- Кар'єрні можливості у сфері ІПЗ
- Практики провідних IT-компаній України та світу

Лабораторна робота — Захист проєкту (2 год.)

Завдання. Захист комплексного програмного проєкту, розробленого протягом курсу. Кожна команда (2–3 особи) представляє повний комплект артефактів:

- Документ SRS (специфікація вимог)
- UML-діаграми (Use Case, Class, Sequence, Component)
- Реалізований програмний код у репозиторії Git
- Набір модульних тестів (покриття ≥ 70%)
- Налаштований CI/CD pipeline (GitHub Actions)
- Розгорнутий застосунок у хмарі або Docker-контейнері

Самостійна робота (4 год.)

- Фінальне доопрацювання та документування проєкту
- Підготовка презентації (10–12 слайдів) для захисту
- Написання README.md з описом архітектури та інструкцією запуску

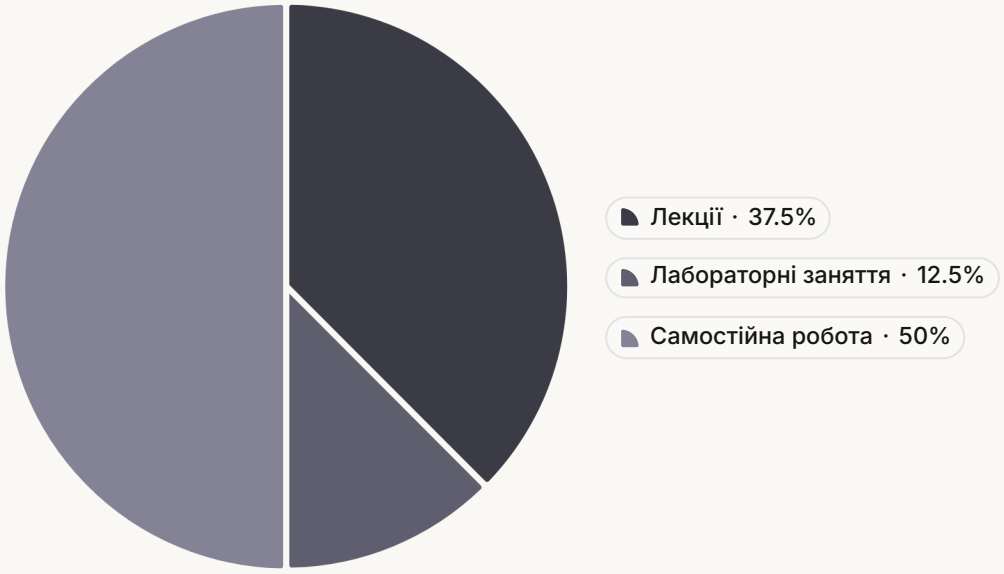
Очікуваний результат: студент демонструє здатність розробити повноцінний програмний продукт із застосуванням усіх методів та інструментів курсу.

Розподіл навчального часу — 120 годин

Модуль	Тема	Лекції (год.)	Лабораторні (год.)	СРС (год.)
Модуль 1	Тема 1.1 — Введення в ІПЗ	3	1	4
	Тема 1.2 — Моделі ЖЦ ПЗ	3	1	4
	Тема 1.3 — Agile-методології	3	1	4
	Тема 1.4 — Управління проєктами	3	1	4
Модуль 2	Тема 2.1 — Інженерія вимог	3	1	4
	Тема 2.2 — UML-моделювання	3	1	4
	Тема 2.3 — Архітектурне проєктування	3	1	4
	Тема 2.4 — Патерни проєктування	3	1	4
Модуль 3	Тема 3.1 — Конструювання ПЗ	3	1	4
	Тема 3.2 — Тестування ПЗ	3	1	4
	Тема 3.3 — Забезпечення якості SQA	3	1	4
	Тема 3.4 — CI/CD та DevOps	3	1	4
Модуль 4	Тема 4.1 — Супровід ПЗ	3	1	4
	Тема 4.2 — Безпека ПЗ	3	1	4
	Тема 4.3 — Сучасні технології ІПЗ	3	1	4
	Тема 4.4 — Комплексний проєкт / Захист	3	2	4
Разом	**16 тем**	**45**	**15**	**60**

Загальний обсяг: 120 годин = 4 кредити ЄКТС. Розподіл є орієнтовним і може коригуватися відповідно до специфіки навчального процесу.

Структура навчального часу



Коментар до розподілу

Загальний обсяг — **120 годин (4 кредити ЄКТС)** — розподілено між трьома формами роботи для забезпечення балансу між теорією та практикою.

Лекції — 45 год. (37,5%)

Системне подання теоретичного матеріалу з використанням наочних схем, UML-діаграм і прикладів із реальної IT-практики. Акцент на концептуальному розумінні процесів і методологій розробки ПЗ.

Лабораторні — 15 год. (12,5%)

Практичне відпрацювання інструментів і технологій: Git, UML-редактори, фреймворки тестування, CI/CD, Docker, аналізатори якості коду. Наскрізний навчальний проєкт.

Самостійна робота — 60 год. (50%)

Поглиблене вивчення, підготовка завдань, робота з технічною літературою та документацією. Самостійна робота становить **50% обсягу дисципліни**.

Самостійна робота студентів — Детальний опис

Опрацювання літератури

Поглиблене вивчення теоретичного матеріалу за підручниками та посібниками з ІПЗ (Цибульник, Трофименко, Марченко, Добровольський та ін.), стандартами SWEBOK, ISO/IEC, IEEE, науковими статтями у фахових виданнях. Студент конспектує ключові положення, виділяє головні поняття та порівнює підходи різних авторів.

Індивідуальні завдання

Виконання аналітичних завдань (порівняльний аналіз методологій, архітектурних рішень, інструментів розробки); творчих завдань — реферати, аналітичні записки, міні-дослідження; наскрізна розробка артефактів програмного проєкту (SRS, UML, тести, CI/CD).

Робота з інформаційними ресурсами

Опрацювання офіційних ресурсів: SWEBOK (**computer.org**), IEEE Standards (**standards.ieee.org**), OWASP (**owasp.org**), документація Docker, GitHub Actions, офіційні стандарти ISO/IEC. Аналіз відкритих репозиторіїв і кейсів реальних проєктів.

Підготовка до занять

Підготовка до лабораторних занять (налаштування інструментів, попереднє вивчення технологій); підготовка до тестувань (ключові терміни, концепції, стандарти); підготовка до захисту модульних та підсумкового проєкту (систематизація артефактів, формування презентації).

Форми контролю знань

1	2	3
Поточний контроль	**Проміжний контроль**	**Підсумковий контроль**
Завдання: аналітичні та тестові вправи за темами змістовного модуля. Оцінюються якість виконання лабораторних робіт, правильність відповідей на усні запитання, рівень засвоєння матеріалу. Поточний контроль формує накопичувальну оцінку протягом семестру. Проводиться після кожної теми у формі тестування (10–15 питань) або захисту лабораторної роботи.	Захист артефактів навчального проєкту — комплексне завдання, що охоплює теми змістовних модулів (специфікація вимог, UML-діаграми, код, тести). Оцінюються повнота, технічна якість, обґрунтованість рішень та здатність захистити прийняті рішення. Проводиться після завершення кожного змістовного модуля.	Іспит відповідно до навчального плану. Форма проведення — тестування та письмові відповіді на питання + захист комплексного проєкту. Підсумкова оцінка враховує результати поточного, проміжного та підсумкового контролю відповідно до встановленої шкали: **100-бальна система** з переведенням у національну шкалу та ЄКТС (A, B, C, D, E, FX, F).

Методи викладання

Викладання дисципліни «Інженерія програмного забезпечення» поєднує класичні академічні та практично-орієнтовані методи, що забезпечує формування не лише міцної теоретичної бази, а й практичних навичок розробки реальних програмних систем, необхідних для майбутньої інженерної діяльності у сфері комп'ютерної інженерії.

Лекції

Системне подання теоретичного матеріалу з використанням наочних схем, UML-діаграм і прикладів із реальної практики; студенти ведуть конспекти та задають уточнювальні запитання.

Лабораторні роботи

Практичне освоєння інструментів ІПЗ: Git, UML-редактори, IDE, фреймворки тестування, CI/CD, Docker; реалізація наскрізного навчального проєкту від вимог до розгортання.

Кейс-стаді

Аналіз реальних програмних проєктів (відомих систем, відкритих репозиторіїв); студенти виявляють архітектурні рішення, оцінюють якість та обґрунтовують альтернативи.

Дискусії

Обговорення дискусійних питань ІПЗ (вибір методології, архітектурний вибір, монолітна vs мікросервісна архітектура); розвиток технічного мислення та навичок аргументації.

Самостійне дослідження

Підготовка аналітичних записок, рефератів і презентацій з актуальних тем ІПЗ; розвиток навичок роботи з технічною документацією, стандартами та науковими публікаціями.

Командна розробка

Виконання наскрізного проєкту в мікрокомандах (2–3 особи) з розподілом ролей (розробник, QA, DevOps); імітація реального Scrum-процесу зі Sprint Review та Retrospective.

Рекомендована література — Підручники та посібники (2022–2024 рр.)

1. Цибульник С. О., Барандич К. С.

Технології розроблення програмного забезпечення. Ч. 1: Життєвий цикл програмного забезпечення : підручник. — Київ : КПІ ім. Ігоря Сікорського, 2022. — 270 с.

2. Трофименко О. Г., Манаков С. Ю., Ларін Д. Г.

Основи програмної інженерії : навч.-метод. посіб. — Одеса : Фенікс, 2022. — 220 с.

3. Марченко О. І.

Компоненти програмної інженерії. Ч. 1: Вступ до програмної інженерії : лаб. практикум. — Київ : КПІ ім. Ігоря Сікорського, 2022. — 98 с.

4. Добровольський Ю. Г.

Стандартизація в інженерії програмного забезпечення : навч. посіб. — Чернівці : Чернівецький нац. ун-т ім. Ю. Федьковича, 2022. — 140 с.

5. Кублій Л. І.

Алгоритми та структури даних. Основи алгоритмізації : підручник. — Київ : КПІ ім. Ігоря Сікорського, 2022. — 528 с.

6. Тищенко К. В., Логвинов А. М., Пилипенко О. В.

Алгоритмічні мови програмування в електронних інформаційних системах та комп'ютерних технологіях : навч. посіб. — Суми : СумДУ, 2024. — 95 с.

7. Висоцька В. А., Оборська О. В.

Python: алгоритмізація та програмування : навч. посіб. — Львів : Новий Світ-2000, 2023. — 514 с.

8. Корнієнко Б. Я.

Сучасні мобільні операційні системи : лаб. практикум. — Київ : КПІ ім. Ігоря Сікорського, 2023. — 71 с.

Додаткова література та наукові ресурси

9. Ковалюк Д. О.

Технології розроблення програмного забезпечення: практикум : підручник. — Київ : КПІ ім. Ігоря Сікорського, 2022. *Рекомендується для тем 2.1–2.4 (проєктування та патерни).*

10. Ткаліченко С. В.

Штучні нейронні мережі : навч. посіб. — Кривий Ріг : Держ. ун-т економіки і технологій, 2023. — 150 с. *Рекомендується для теми 4.3 (AI-assisted development).*

11. Єршова О. О., Гончаренко І. М.

Сучасні моделі управління розвитком бізнесу в умовах цифрової трансформації. Журнал стратегічних економічних досліджень. — 2022. — № 2(7). — С. 75–84. *Рекомендується для тем 1.4 та 4.3.*

12. SWEBOK Guide v4.0

IEEE Computer Society. Guide to the Software Engineering Body of Knowledge. — 2024. — **computer.org/swebok**. *Базовий довідник для всіх модулів курсу.*

Нормативно-правова база та стандарти

→ ISO/IEC 12207:2017 — Процеси ЖЦ систем та ПЗ

Міжнародний стандарт, що визначає рамкову модель для процесів ЖЦ програмного забезпечення. Базовий стандарт курсу.

→ ISO/IEC 25010:2023 — Якість систем та ПЗ

Стандарт моделі якості продукту та якості у використанні. Визначає характеристики якості ПЗ, що вивчаються у Темі 3.3.

→ IEEE Std 830 / IEEE Std 29148:2018 — Специфікація вимог

Стандарти специфікації вимог до ПЗ (SRS). Основа для вивчення Теми 2.1 (Інженерія вимог).

→ Закон України «Про вищу освіту» від 01.07.2014 № 1556-VII

Визначає засади вищої освіти в Україні, стандарти підготовки бакалаврів. **zakon.rada.gov.ua**

→ Стандарт вищої освіти України зі спеціальності 123 «Комп'ютерна інженерія»

Визначає компетентності та результати навчання для бакалаврів зі спеціальності 123. **mon.gov.ua**

Офіційні ресурси та фахові видання

IEEE Computer Society

computer.org — SWEBOK, стандарти ІПЗ, наукові публікації, конференції з програмної інженерії (ICSE, FSE).

OWASP Foundation

owasp.org — матеріали з безпеки ПЗ: OWASP Top 10, CheatSheet Series, інструменти тестування безпеки.

GitHub / GitLab Documentation

docs.github.com — документація з Git, GitHub Actions, CI/CD, управління проєктами; практична база для лабораторних робіт.

Національна бібліотека України ім. Вернадського

nbuv.gov.ua — електронні ресурси, наукові статті з комп'ютерних наук та ІПЗ, доступ до фахових видань.

Журнал «Проблеми інформатизації та управління»

НАУ (Національний авіаційний університет) — фахове видання з інформаційних технологій та програмної інженерії.

Stack Overflow Developer Survey

survey.stackoverflow.co — щорічні дані про технології, інструменти та тренди у розробці ПЗ. Актуальне джерело для тем 3.4, 4.3.

Docker Documentation

docs.docker.com — офіційна документація Docker та Docker Compose; основа для лабораторних тем 3.4 та 4.3.

Міністерство освіти і науки України

mon.gov.ua — освітні стандарти, навчальні програми зі спеціальності 123, методичні матеріали.

Зв'язок дисципліни зі спеціальністю 123 «Комп'ютерна інженерія»

Чому «Інженерія ПЗ» для комп'ютерних інженерів?

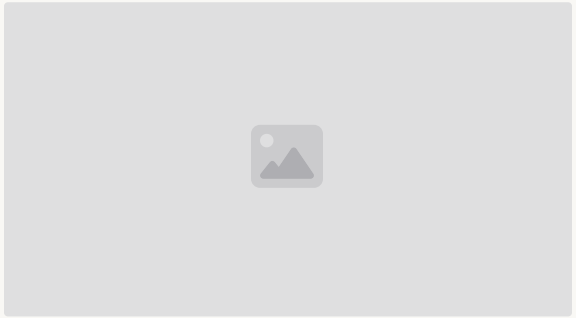
Дисципліна «Інженерія програмного забезпечення» є невід'ємною складовою підготовки бакалаврів зі спеціальністю 123 «Комп'ютерна інженерія». Розробка програмного забезпечення — це не просто написання коду, а системна інженерна діяльність, яка вимагає дотримання методологій, стандартів і процесів.

Сучасний фахівець у сфері комп'ютерної інженерії постійно взаємодіє з програмними системами: вбудовані системи, операційні системи, драйвери, мікропрограми (firmware). Знання ІПЗ дозволяє розробляти надійне, безпечне та якісне ПЗ для апаратних платформ.

Практичне значення для майбутньої кар'єри

- Проєктування надійного вбудованого ПЗ для мікроконтролерів і FPGA
- Розробка драйверів та системного ПЗ з дотриманням стандартів якості
- Застосування agile-методологій у командній апаратно-програмній розробці
- Тестування та верифікація критичного ПЗ (safety-critical systems)
- Управління конфігурацією та версіонуванням вбудованого ПЗ
- Застосування DevOps-практик у циклі розробки комп'ютерних систем

Ключові компетентності курсу — Підсумок



Курс охоплює повний спектр компетентностей сучасного інженера програмного забезпечення — від аналізу вимог до розгортання та супроводу продукту — і формує у студентів цілісне розуміння промислових процесів створення програмних систем.

Бажаємо успіхів у вивченні Інженерії програмного забезпечення!

120

Годин

Загальний обсяг дисципліни

4

Кредити ЄКТС

Відповідно до стандарту

16

Тем

У 4 змістовних модулях

12

Джерел

Рекомендованої літератури 2022–2024 рр.

> «Програмне забезпечення — це мистецтво, замасковане під інженерію.» — Фредерік Брукс

Спеціальність 123 · Комп'ютерна інженерія · Бакалавр · 4 кредити ЄКТС