Міністерство освіти і науки України
Міжнародний гуманітарний університет
Кафедра комп’ютерних наук

**Русу О.П.**

# ВСТУП ДО СПЕЦІАЛЬНОСТІ

Методичні вказівки до виконання практичних робіт
для здобувачів вищої освіти,
які навчаються за спеціальностями
галузі «Інформаційні технології»

Одеса 2025

Затверджено Вченою Радою Міжнародного гуманітарного університету.
Протокол № 5 від 20 січня 2025 р.

Русу О.П. Вступ до спеціальності. Методичні вказівки до виконання практичних робіт для здобувачів вищої освіти, які навчаються за спеціальностями галузі «Інформаційні технології» [Електронне видання]. Кафедра комп'ютерних наук Міжнародного гуманітарного університету. Одеса, 2025. 23 с.

Дисципліна «Вступ до спеціальності» спрямована на формування у здобувачів цілісного уявлення про сферу професійної діяльності, структуру ІТ-галузі, особливості функціонування ІТ-компаній, основні процеси життєвого циклу програмного забезпечення та ІТ-проєктів, а також правові, етичні та організаційні засади професійної діяльності. У межах дисципліни розглядаються професійні стандарти, нормативні документи, базові підходи до розробки програмних систем, принципи управління проєктами та основи професійного саморозвитку.

Результатом вивчення дисципліни є формування у здобувачів здатності орієнтуватися в структурі сучасної ІТ-галузі, розуміти основні процеси створення та супроводження програмного забезпечення, усвідомлювати роль стандартів і нормативних вимог у професійній діяльності, аналізувати організацію роботи команди розробки та базові принципи управління ІТ-проєктами, а також дотримуватися професійної етики й принципів доброчесності у сфері інформаційних технологій.

**ЗМІСТ**

## Практична робота 1. Аналіз структури ІТ-компанії та професійних ролей у команді розробки

### Ключові положення

Сучасні ІТ-компанії є складними організаційними системами, діяльність яких спрямована на створення, розвиток і супроводження програмного забезпечення. Ефективність роботи таких компаній значною мірою залежить від правильної організації внутрішньої структури, розподілу функцій між підрозділами та налагодженої взаємодії між спеціалістами різних професійних напрямів.

Організаційна структура ІТ-компанії визначає спосіб розподілу обов'язків, повноважень і відповідальності між працівниками та підрозділами організації. Структура компанії формується залежно від масштабу діяльності, типу програмних продуктів, бізнес-моделі компанії та особливостей ринку інформаційних технологій. У загальному випадку структура ІТ-компанії містить керівництво, технічні підрозділи, підрозділи підтримки, а також адміністративні та бізнесові служби.

До основних підрозділів ІТ-компанії зазвичай належать відділи розробки програмного забезпечення, тестування, системного аналізу, технічної підтримки, управління проєктами, маркетингу та продажів. У великих організаціях можуть також функціонувати підрозділи досліджень і розробок, інфраструктурні служби, підрозділи кібербезпеки та управління якістю.

Ключову роль у створенні програмного продукту відіграє команда розробки. Команда розробників програмного забезпечення є групою спеціалістів, які спільно виконують завдання зі створення програмної системи. У межах такої команди кожен учасник виконує певну професійну роль, що визначає його функції та відповідальність у процесі розроблення програмного продукту.

До основних професійних ролей у команді розробки належать програмісти, тестувальники, системні аналітики, архітектори програмного забезпечення, фахівці з забезпечення якості, дизайнери користувацьких інтерфейсів, менеджери проєктів та інші спеціалісти. Кожна з цих ролей має власні функції та сприяє реалізації різних етапів життєвого циклу програмного забезпечення.

Програмісти займаються безпосередньою реалізацією програмного коду, розробляють алгоритми оброблення даних та створюють програмні модулі. Системні аналітики аналізують вимоги до програмної системи та формують технічні специфікації. Архітектори програмного забезпечення визначають структуру програмної системи та принципи взаємодії її компонентів.

Тестувальники програмного забезпечення відповідають за перевірку правильності роботи програмного продукту та виявлення помилок. Фахівці з забезпечення якості контролюють дотримання стандартів розроблення програмного забезпечення та організовують процес тестування програмних систем. Менеджери проєктів координують роботу команди розробників, контролюють строки виконання завдань та забезпечують взаємодію із замовниками.

Аналіз структури ІТ-компанії та ролей у команді розробки є важливим для розуміння організації діяльності у сфері інженерії програмного забезпечення. Майбутні фахівці повинні розуміти, як організовується робота ІТ-компаній, які функції виконують різні підрозділи та

яким чином здійснюється взаємодія між учасниками процесу створення програмного продукту.

**Практичне завдання**

1. Обрати приклад IT-компанії, що займається розробленням програмного забезпечення. Це може бути відома міжнародна компанія або українська IT-компанія.
2. Дослідити організаційну структуру обраної компанії.
3. Визначити основні підрозділи компанії та описати їх функції.
4. Проаналізувати структуру команди розробки програмного забезпечення.
5. Визначити основні професійні ролі у команді розробки.
6. Описати функції кожної ролі у процесі створення програмного продукту.
7. Проаналізувати взаємодію між учасниками команди розробки під час реалізації програмного проєкту.
8. Сформулювати висновки щодо ефективності організаційної структури дослідженої компанії.

**Методичні вказівки до виконання практичного завдання**

Перед виконанням практичного завдання здобувач повинен ознайомитися з теоретичним матеріалом щодо організації діяльності IT-компаній, ролей у команді розробки та принципів управління IT-проєктами.

На першому етапі виконання роботи необхідно обрати IT-компанію для дослідження. Доцільно використовувати приклади відомих компаній, діяльність яких добре висвітлена у відкритих джерелах інформації. Джерелами інформації можуть бути офіційні вебсайти компаній, технічні блоги, публікації у професійних виданнях та інші інформаційні ресурси.

Після вибору компанії необхідно дослідити її організаційну структуру. Для цього слід визначити основні підрозділи компанії, їх функції та взаємозв’язки. Важливо звернути увагу на роль технічних підрозділів, що займаються розробленням програмного забезпечення.

На наступному етапі потрібно проаналізувати структуру команди розробки програмного забезпечення. Необхідно визначити, які спеціалісти входять до складу команди, які функції вони виконують та як взаємодіють між собою у процесі створення програмного продукту.

Опис професійних ролей повинен містити коротку характеристику діяльності кожного спеціаліста, його основні завдання та відповідальність у процесі розроблення програмної системи. Важливо також показати взаємозв’язок між різними ролями та пояснити, як здійснюється координація роботи команди.

Особливу увагу слід приділити аналізу взаємодії між учасниками команди розробки. Необхідно описати, яким чином організовується обмін інформацією між розробниками, тестувальниками, аналітиками та менеджерами проєкту. Також доцільно звернути увагу на використання сучасних інструментів командної роботи.

Після виконання аналізу необхідно сформулювати висновки щодо організаційної структури дослідженої компанії. У висновках слід оцінити ефективність розподілу ролей у команді, взаємодію між підрозділами та загальні принципи організації діяльності IT-компанії.

Під час виконання роботи необхідно використовувати професійні джерела інформації, дотримуватися принципів академічної доброчесності та правильно оформлювати використані джерела.

**Контрольні питання**

1. Що розуміють під організаційною структурою ІТ-компанії?
2. Які основні підрозділи можуть входити до складу ІТ-компанії?
3. Яку роль відіграє команда розробки програмного забезпечення?
4. Які основні професійні ролі існують у команді розробників?
5. Які функції виконує програміст у процесі створення програмного продукту?
6. Які завдання виконує системний аналітик?
7. У чому полягає роль тестувальника програмного забезпечення?
8. Які функції виконує архітектор програмного забезпечення?
9. Яку роль відіграє менеджер проєкту у команді розробників?
10. Чому важливою є ефективна взаємодія між членами команди розробки?

**Зміст звіту**

Звіт з практичної роботи повинен містити такі структурні елементи:
– назву практичної роботи;
– мету виконання практичної роботи;
– стислі відомості щодо структури ІТ-компаній та ролей у команді розробки;
– опис обраної ІТ-компанії;
– аналіз організаційної структури компанії;
– опис команди розробки програмного забезпечення;
– характеристику основних професійних ролей у команді;
– аналіз взаємодії між учасниками команди розробки;
– висновки за результатами виконання практичної роботи;
– перелік використаних джерел інформації.

## Практична робота 2. Дослідження професійних стандартів у сфері інженерії програмного забезпечення

### Ключові положення

Сфера інженерії програмного забезпечення характеризується високим рівнем складності процесів розроблення програмних систем, значною кількістю учасників проєктної діяльності та необхідністю забезпечення якості програмних продуктів. Для організації ефективної діяльності у цій галузі використовуються професійні стандарти, які визначають вимоги до процесів створення програмного забезпечення, професійних компетентностей спеціалістів та методів управління розробкою програмних систем.

Професійні стандарти у сфері програмної інженерії формують основу для організації діяльності IT-компаній, визначають правила виконання робіт та сприяють підвищенню якості програмного забезпечення. Вони встановлюють узагальнені вимоги до процесів життєвого циклу програмного забезпечення, методів управління проєктами, процедур тестування та забезпечення якості програмних продуктів.

Однією з важливих функцій професійних стандартів є уніфікація підходів до розроблення програмного забезпечення. Завдяки використанню стандартів різні організації можуть застосовувати узгоджені методи проєктування програмних систем, що сприяє підвищенню ефективності співпраці між компаніями, розробниками та замовниками програмних продуктів.

У сфері інженерії програмного забезпечення широко застосовуються міжнародні стандарти, які розробляються спеціалізованими організаціями. До таких організацій належать міжнародні стандартизаційні організації, а також професійні спільноти у сфері інформаційних технологій. Ці стандарти описують процеси розроблення програмного забезпечення, визначають основні етапи життєвого циклу програмного продукту та встановлюють вимоги до якості програмних систем.

Одним із найбільш відомих стандартів у сфері програмної інженерії є стандарт, що визначає структуру процесів життєвого циклу програмного забезпечення. У цьому стандарті описано комплекс процесів, які виконуються під час створення програмного продукту, зокрема процеси розроблення, тестування, впровадження та супроводження програмного забезпечення.

Крім стандартів, що регламентують процеси розроблення програмного забезпечення, важливу роль відіграють професійні рекомендації щодо формування знань і компетентностей фахівців у галузі програмної інженерії. Такі рекомендації містять узагальнений перелік знань, необхідних для діяльності інженера програмного забезпечення, а також описують основні напрями професійної підготовки у цій галузі.

Використання професійних стандартів має важливе значення для підготовки майбутніх фахівців у сфері інженерії програмного забезпечення. Освітні програми університетів формуються з урахуванням міжнародних рекомендацій та стандартів, що визначають перелік компетентностей, якими повинні володіти випускники відповідних спеціальностей.

Таким чином, дослідження професійних стандартів у сфері інженерії програмного забезпечення дозволяє зрозуміти принципи організації процесів розроблення програмних систем, визначити основні вимоги до професійної діяльності IT-фахівців та сформувати уявлення про сучасні підходи до забезпечення якості програмного забезпечення.

**Практичне завдання**

1. Ознайомитися з основними міжнародними стандартами та рекомендаціями у сфері інженерії програмного забезпечення.
2. Проаналізувати один або декілька стандартів, що регламентують процеси розроблення програмного забезпечення.
3. Визначити основні процеси життєвого циклу програмного забезпечення, описані у досліджених стандартах.
4. Дослідити професійні рекомендації щодо підготовки фахівців у галузі програмної інженерії.
5. Проаналізувати основні області знань, які визначають професійні компетентності інженера програмного забезпечення.

6. Сформулювати висновки щодо значення професійних стандартів у діяльності IT-компаній.

**Методичні вказівки до виконання практичного завдання**

Перед виконанням практичної роботи студент повинен ознайомитися з теоретичними відомостями щодо професійних стандартів у сфері програмної інженерії та їх ролі у процесі розроблення програмного забезпечення.

На першому етапі виконання роботи необхідно знайти інформацію про міжнародні стандарти, що використовуються у сфері інженерії програмного забезпечення. Для цього можна використовувати офіційні вебресурси міжнародних організацій стандартизації, технічну документацію, навчальні матеріали та професійні публікації.

Після ознайомлення з відповідними стандартами необхідно визначити їх основне призначення, структуру та сферу застосування. Особливу увагу слід приділити опису процесів життєвого циклу програмного забезпечення, які регламентуються цими стандартами.

На наступному етапі необхідно дослідити професійні рекомендації щодо підготовки фахівців у сфері інженерії програмного забезпечення. У межах цього дослідження слід визначити основні області знань, що формують професійну компетентність інженера програмного забезпечення.

Під час аналізу стандартів доцільно звернути увагу на такі аспекти: структуру процесів розроблення програмного забезпечення, вимоги до організації роботи команди розробників, методи забезпечення якості програмних систем та роль документації у процесі створення програмного продукту.

Результати дослідження необхідно систематизувати та представити у вигляді короткого аналітичного опису. У висновках слід оцінити значення професійних стандартів для організації діяльності IT-компаній та підвищення якості програмного забезпечення.

**Контрольні питання**

1. Що розуміють під професійними стандартами у сфері інженерії програмного забезпечення?

2. Яку роль відіграють стандарти у процесі розроблення програмного забезпечення?
3. Які міжнародні організації займаються розробленням стандартів у сфері інформаційних технологій?
4. Які основні процеси життєвого циклу програмного забезпечення визначаються у професійних стандартах?
5. Яке значення мають стандарти для забезпечення якості програмного забезпечення?
6. Які професійні рекомендації щодо підготовки фахівців у сфері інженерії програмного забезпечення існують?
7. Які основні області знань формують компетентність інженера програмного забезпечення?
8. Чому використання професійних стандартів є важливим для діяльності IT-компаній?
9. Яким чином стандарти сприяють уніфікації процесів розроблення програмного забезпечення?
10. Яке значення має використання стандартів для професійної підготовки майбутніх IT-фахівців?

**Зміст звіту**

Звіт з практичної роботи повинен містити такі структурні елементи:
– назву практичної роботи;
– мету виконання практичної роботи;
– стислі відомості щодо професійних стандартів у сфері інженерії програмного забезпечення;
– опис досліджених міжнародних стандартів або рекомендацій;
– аналіз процесів життєвого циклу програмного забезпечення відповідно до стандартів;
– характеристику основних професійних компетентностей інженера програмного забезпечення;
– узагальнення результатів дослідження професійних стандартів;
– висновки за результатами виконання практичної роботи;
– перелік використаних джерел інформації.

## Практична робота 3. Моделювання життєвого циклу програмного забезпечення

### Ключові положення

Створення сучасних програмних систем є складним інженерним процесом, що потребує системного підходу до організації робіт, чіткої послідовності виконання етапів розроблення та узгодженої взаємодії між учасниками проєкту. Для впорядкування процесу створення програмного продукту використовується поняття життєвого циклу програмного забезпечення. Життєвий цикл програмного забезпечення визначає послідовність процесів і робіт, що виконуються від моменту формування ідеї програмного продукту до завершення його використання.

Життєвий цикл програмного забезпечення включає етапи аналізу вимог, проєктування, реалізації програмного коду, тестування, впровадження та супроводження програмної системи. Кожен з цих етапів має власну мету, набір завдань та результати, що передаються до наступного етапу розроблення. Організація цих етапів визначається моделлю життєвого циклу програмного забезпечення.

Модель життєвого циклу описує спосіб організації процесу створення програмного продукту, визначає послідовність виконання робіт та взаємозв'язок між різними етапами розроблення. У практиці програмної інженерії використовується кілька моделей життєвого циклу, серед яких найбільш відомими є каскадна, ітераційна, інкрементна, спіральна та гнучкі моделі розроблення програмного забезпечення.

Каскадна модель передбачає послідовне виконання етапів життєвого циклу програмного забезпечення. Після завершення кожного етапу формується відповідна документація, і лише після цього розпочинається наступний етап. Такий підхід є простим для організації, проте має обмежену гнучкість у разі зміни вимог до програмної системи.

Ітераційна модель передбачає поступове вдосконалення програмної системи шляхом повторення циклів розроблення. У межах кожної ітерації виконуються роботи з аналізу вимог, проєктування, програмування та тестування. Це дозволяє поступово уточнювати функціональні можливості програмного продукту та враховувати нові вимоги користувачів.

Інкрементна модель передбачає створення програмної системи шляхом поступового додавання нових функціональних компонентів. На початковому етапі створюється базова версія програмного продукту, після чого її функціональні можливості розширюються за допомогою нових інкрементів.

Спіральна модель життєвого циклу поєднує елементи ітераційного підходу та аналізу ризиків. У межах кожного циклу розроблення здійснюється планування робіт, аналіз можливих ризиків, реалізація програмних компонентів та оцінювання результатів виконаної роботи.

У сучасній практиці розроблення програмного забезпечення широко застосовуються гнучкі підходи до організації життєвого циклу. Вони передбачають короткі цикли розроблення, активну взаємодію з користувачами та можливість швидкої адаптації програмного продукту до змін вимог.

Моделювання життєвого циклу програмного забезпечення є важливим інструментом планування процесу розроблення. Моделювання дозволяє візуально представити структуру процесу створення програмного продукту, визначити взаємозв'язки між етапами розроблення та оцінити послідовність виконання робіт.

У процесі моделювання можуть використовуватися різні графічні методи представлення процесів, зокрема діаграми етапів життєвого циклу, схеми взаємозв'язків між процесами або інші графічні моделі. Використання таких моделей сприяє кращому розумінню структури процесу розроблення програмного забезпечення та дозволяє ефективніше організувати роботу команди розробників.

Дослідження моделей життєвого циклу програмного забезпечення є важливим для майбутніх фахівців у сфері інженерії програмного забезпечення, оскільки дозволяє зрозуміти принципи організації процесу створення програмних систем та оцінити переваги різних підходів до розроблення програмного забезпечення.

**Практичне завдання**

1. Ознайомитися з основними моделями життєвого циклу програмного забезпечення.
2. Проаналізувати структуру етапів життєвого циклу програмного забезпечення.
3. Обрати одну з моделей життєвого циклу програмного забезпечення.
4. Побудувати модель життєвого циклу програмного забезпечення у графічному вигляді.
5. Визначити основні етапи розроблення програмної системи у межах обраної моделі.
6. Проаналізувати взаємозв'язки між етапами життєвого циклу програмного забезпечення.
7. Порівняти обрану модель з іншими моделями життєвого циклу програмного забезпечення.
8. Сформулювати висновки щодо особливостей застосування обраної моделі.

**Методичні вказівки до виконання практичного завдання**

Перед виконанням практичної роботи студент повинен ознайомитися з теоретичними положеннями щодо життєвого циклу програмного забезпечення та основних моделей його організації.

На першому етапі необхідно дослідити основні моделі життєвого циклу програмного забезпечення. Особливу увагу слід приділити аналізу послідовності етапів розроблення програмного продукту, а також принципам організації взаємодії між різними етапами життєвого циклу.

Після ознайомлення з різними моделями життєвого циклу необхідно обрати одну модель для подальшого аналізу. Доцільно обрати модель, яка найбільш повно відображає особливості сучасних процесів розроблення програмного забезпечення.

На наступному етапі необхідно побудувати графічну модель життєвого циклу програмного забезпечення. У моделі слід відобразити основні етапи розроблення програмної системи та показати взаємозв'язки між ними. Для побудови моделі можна використовувати графічні редактори, засоби створення діаграм або інші інструменти моделювання.

Під час побудови моделі слід звернути увагу на послідовність виконання етапів життєвого циклу програмного забезпечення, а також на можливі ітерації між різними етапами розроблення. Необхідно також визначити результати, які формуються на кожному етапі життєвого циклу.

Після побудови моделі необхідно виконати її аналіз. У межах аналізу слід оцінити переваги та обмеження обраної моделі життєвого циклу, а також визначити умови її ефективного застосування у процесі розроблення програмного забезпечення.

Результати виконаної роботи необхідно оформити у вигляді звіту, який повинен містити опис дослідженої моделі життєвого циклу, побудовану графічну модель та висновки щодо її застосування.

**Контрольні питання**

1. Що розуміють під життєвим циклом програмного забезпечення?
2. Які основні етапи життєвого циклу програмного забезпечення існують?
3. Що таке модель життєвого циклу програмного забезпечення?
4. У чому полягає сутність каскадної моделі розроблення програмного забезпечення?
5. Які особливості ітераційної моделі життєвого циклу програмного забезпечення?
6. У чому полягає відмінність інкрементної моделі від каскадної?
7. Які переваги має спіральна модель життєвого циклу програмного забезпечення?
8. Які особливості гнучких підходів до організації процесу розроблення програмного забезпечення?
9. Яку роль відіграє моделювання життєвого циклу програмного забезпечення?
10. Чому важливо правильно обирати модель життєвого циклу під час реалізації програмного проєкту?

**Зміст звіту**

Звіт з практичної роботи повинен містити такі структурні елементи:
– назву практичної роботи;
– мету виконання практичної роботи;
– стислі відомості щодо життєвого циклу програмного забезпечення;
– опис обраної моделі життєвого циклу програмного забезпечення;
– графічну модель життєвого циклу програмного забезпечення;
– аналіз етапів життєвого циклу програмної системи;
– порівняння обраної моделі з іншими моделями життєвого циклу;
– висновки за результатами виконання практичної роботи;
– перелік використаних джерел інформації.

## Практична робота 4. Формування вимог та проєктної структури програмної системи

### Ключові положення

Одним із найбільш важливих етапів створення програмного забезпечення є формування вимог до програмної системи та розроблення її проєктної структури. Якість програмного продукту значною мірою залежить від того, наскільки правильно визначено потреби користувачів, сформульовано функціональні можливості системи та розроблено її архітектурну структуру. Помилки на етапі формування вимог можуть призвести до значних ускладнень під час реалізації програмного продукту, збільшення витрат на розроблення та необхідності суттєвого доопрацювання системи.

Вимоги до програмного забезпечення визначають властивості, функціональні можливості та обмеження, яким повинна відповідати програмна система. Вимоги формуються на основі аналізу потреб користувачів, бізнес-процесів організації та технічних умов використання програмного продукту. Формування вимог є важливою складовою процесу аналізу програмної системи.

У процесі розроблення програмного забезпечення вимоги поділяють на кілька основних типів. Функціональні вимоги визначають, які саме функції повинна виконувати програмна система. Вони описують дії, які система повинна виконувати у відповідь на запити користувача або інші події. Наприклад, функціональною вимогою може бути можливість реєстрації користувачів у системі або виконання певних операцій з даними.

Нефункціональні вимоги описують властивості програмної системи, що визначають умови її використання. До таких вимог належать вимоги до продуктивності, надійності, безпеки, зручності використання та сумісності з іншими програмними системами. Нефункціональні вимоги визначають якісні характеристики програмного продукту.

Окремою групою є обмеження системи. Вони визначають технічні або організаційні умови, у межах яких повинна функціонувати програмна система. Наприклад, обмеження можуть стосуватися використання певної платформи, мови програмування або програмного середовища.

Після формування вимог до програмної системи виконується етап проєктування. Проєктування програмного забезпечення передбачає визначення структури програмної системи, її основних компонентів та принципів взаємодії між ними. На цьому етапі формуються архітектурні рішення, які визначають загальну організацію програмної системи.

Проєктна структура програмної системи описує склад програмних компонентів, їх функції та взаємозв’язки. Структура системи може включати програмні модулі, підсистеми, інтерфейси взаємодії між компонентами та механізми оброблення даних. Правильно сформована структура програмної системи забезпечує її модульність, розширюваність та зручність супроводження.

У процесі проєктування можуть використовуватися різні підходи до організації структури програмної системи. Одним із найбільш поширених є структурний підхід, який передбачає поділ програмної системи на окремі функціональні модулі. Іншим підходом є об’єктно-орієнтоване проєктування, у межах якого програмна система розглядається як сукупність взаємодіючих об’єктів.

Моделювання структури програмної системи часто виконується за допомогою графічних моделей. Такі моделі дозволяють наочно представити взаємозв’язки між компонентами системи та спростити аналіз її архітектури. Використання моделей сприяє

кращому розумінню структури програмного продукту та полегшує взаємодію між учасниками процесу розроблення.

Таким чином, формування вимог та проєктної структури програмної системи є важливим етапом створення програмного забезпечення. Якісне виконання цих етапів дозволяє забезпечити відповідність програмного продукту потребам користувачів, підвищити ефективність процесу розроблення та зменшити ризик виникнення помилок у програмній системі.

**Практичне завдання**

1. Обрати приклад програмної системи, що призначена для розв’язання певної прикладної задачі.
2. Визначити основних користувачів обраної програмної системи.
3. Сформулювати функціональні вимоги до програмної системи.
4. Сформулювати нефункціональні вимоги до програмної системи.
5. Визначити основні обмеження функціонування програмної системи.
6. Побудувати загальну структуру програмної системи.
7. Визначити основні компоненти програмної системи та їх функції.
8. Проаналізувати взаємодію між компонентами системи.
9. Сформулювати висновки щодо відповідності проєктної структури сформованим вимогам.

**Методичні вказівки до виконання практичного завдання**

Перед виконанням практичної роботи студент повинен ознайомитися з теоретичними положеннями щодо формування вимог до програмного забезпечення та принципів проєктування програмних систем.

На першому етапі необхідно обрати приклад програмної системи, для якої буде виконуватися формування вимог. Це може бути інформаційна система, вебзастосунок, мобільний застосунок або інший програмний продукт. Бажано обирати систему, що має зрозуміле призначення та чітко визначене коло користувачів.

Після вибору системи необхідно визначити основних користувачів програмного продукту. Доцільно описати їх роль у використанні системи та основні завдання, які вони виконують за допомогою програмного забезпечення.

На наступному етапі необхідно сформулювати функціональні вимоги до програмної системи. Формулювання вимог повинно бути чітким та однозначним. Кожна вимога повинна описувати конкретну функцію або можливість програмної системи.

Далі необхідно визначити нефункціональні вимоги до програмного продукту. Вони можуть стосуватися швидкодії системи, надійності, безпеки даних, зручності користування та інших характеристик програмного забезпечення.

Після формування вимог необхідно перейти до проєктування структури програмної системи. На цьому етапі слід визначити основні компоненти програмної системи, їх функції та взаємодію між ними. Для представлення структури системи доцільно використати схему або діаграму.

Під час аналізу структури системи слід оцінити, наскільки обрана архітектура забезпечує реалізацію сформованих вимог. Також необхідно звернути увагу на модульність системи, можливість її розширення та зручність подальшого супроводження.

Результати виконаної роботи необхідно систематизувати та представити у звіті. У висновках слід оцінити ефективність сформованої структури програмної системи та її відповідність визначеним вимогам.

**Контрольні питання**

1. Що розуміють під вимогами до програмного забезпечення?
2. Які основні типи вимог до програмної системи існують?
3. Що таке функціональні вимоги до програмного забезпечення?
4. Які характеристики описують нефункціональні вимоги?
5. Що розуміють під обмеженнями програмної системи?
6. У чому полягає процес проєктування програмного забезпечення?
7. Що таке проєктна структура програмної системи?
8. Які підходи використовуються під час проєктування програмних систем?
9. Яку роль відіграє моделювання структури програмної системи?
10. Чому важливо узгоджувати проєктні рішення з вимогами до програмного забезпечення?

**Зміст звіту**

Звіт з практичної роботи повинен містити такі структурні елементи:
– назву практичної роботи;
– мету виконання практичної роботи;
– стислі відомості щодо формування вимог до програмного забезпечення;
– опис обраної програмної системи;
– перелік основних користувачів системи;
– сформульовані функціональні вимоги до системи;
– сформульовані нефункціональні вимоги до системи;
– опис проєктної структури програмної системи;
– аналіз взаємодії компонентів системи;
– висновки за результатами виконання практичної роботи;
– перелік використаних джерел інформації.

## Практична робота 5. Базовий план ІТ-проєкту та аналіз ризиків його реалізації

### Ключові положення

Розроблення програмного забезпечення у сучасних умовах зазвичай здійснюється у формі ІТ-проєкту. ІТ-проєкт являє собою комплекс взаємопов'язаних робіт, спрямованих на створення або модернізацію інформаційної системи, програмного продукту чи програмно-апаратного комплексу. Основною особливістю проєктної діяльності є обмеженість у часі, наявність чітко визначеної мети, обмежені ресурси та необхідність координації роботи різних учасників проєкту.

Управління ІТ-проєктом передбачає планування, організацію та контроль виконання робіт, спрямованих на досягнення визначеного результату. Одним із ключових документів управління проєктом є базовий план проєкту. Базовий план ІТ-проєкту визначає основні параметри реалізації проєкту, зокрема перелік робіт, строки їх виконання, необхідні ресурси, відповідальних виконавців та очікувані результати.

Формування базового плану ІТ-проєкту є важливим етапом управління проєктною діяльністю. Планування дозволяє визначити послідовність виконання робіт, оцінити необхідні ресурси, узгодити строки реалізації проєкту та забезпечити координацію діяльності учасників команди. Чітко сформований план дозволяє зменшити невизначеність у процесі реалізації проєкту та підвищити ефективність управління роботою команди.

Базовий план ІТ-проєкту зазвичай включає кілька основних компонентів. Одним із них є опис цілей проєкту та очікуваних результатів. Цілі проєкту повинні бути чітко сформульованими та відображати призначення програмного продукту, що створюється. Очікувані результати повинні визначати функціональні можливості системи та критерії її успішного впровадження.

Іншим важливим компонентом базового плану є структура робіт проєкту. Для планування робіт використовується декомпозиція проєкту на окремі завдання або етапи. Така структура дозволяє визначити послідовність виконання робіт, встановити залежності між ними та оцінити тривалість виконання кожного етапу.

У процесі планування також визначаються ресурси, необхідні для реалізації проєкту. До ресурсів можуть належати людські ресурси, програмні інструменти, апаратне забезпечення, інформаційні ресурси та фінансові витрати. Визначення ресурсів дозволяє оцінити можливість виконання проєкту у встановлені строки.

Важливим аспектом управління ІТ-проєктами є аналіз ризиків. Ризик у проєктній діяльності визначається як можливість виникнення подій, що можуть негативно вплинути на реалізацію проєкту або призвести до відхилення від запланованих результатів. У сфері розроблення програмного забезпечення ризики можуть бути пов'язані з технічними труднощами, зміною вимог, нестачею ресурсів або організаційними проблемами.

Процес управління ризиками включає кілька основних етапів. На першому етапі здійснюється ідентифікація ризиків, тобто визначення можливих факторів, що можуть вплинути на реалізацію проєкту. Далі виконується аналіз ризиків, у межах якого оцінюється ймовірність виникнення ризикових подій та можливі наслідки їх реалізації.

Наступним етапом є оцінювання ризиків. На цьому етапі визначається рівень впливу ризику на реалізацію проєкту та встановлюється пріоритетність реагування на різні ризикові фактори. Після цього розробляються заходи щодо зменшення або усунення ризиків.

Управління ризиками дозволяє підвищити стійкість проєкту до непередбачуваних обставин та забезпечити більш ефективну організацію роботи команди. Аналіз ризиків є невід'ємною складовою планування ІТ-проєктів і сприяє підвищенню ймовірності успішної реалізації програмного продукту.

Таким чином, формування базового плану ІТ-проєкту та аналіз ризиків його реалізації є важливими елементами управління проєктною діяльністю у сфері програмної інженерії. Використання системного підходу до планування проєктів дозволяє ефективно організувати роботу команди, оптимально використовувати ресурси та забезпечити досягнення поставлених цілей.

**Практичне завдання**

1. Обрати приклад ІТ-проєкту, пов'язаного зі створенням програмної системи.
2. Сформулювати основну мету ІТ-проєкту та очікувані результати його реалізації.
3. Визначити основні етапи реалізації проєкту.
4. Сформувати структуру робіт ІТ-проєкту.
5. Визначити необхідні ресурси для виконання проєкту.
6. Побудувати базовий план реалізації ІТ-проєкту.
7. Визначити можливі ризики реалізації проєкту.
8. Виконати аналіз і оцінювання визначених ризиків.
9. Запропонувати заходи щодо зменшення або усунення ризиків.
10. Сформулювати висновки щодо ефективності сформованого плану проєкту.

**Методичні вказівки до виконання практичного завдання**

Перед виконанням практичної роботи студент повинен ознайомитися з теоретичними положеннями щодо управління ІТ-проєктами, принципів планування проєктної діяльності та методів аналізу ризиків.

На першому етапі необхідно обрати приклад ІТ-проєкту, який буде використано для виконання роботи. Це може бути розроблення вебзастосунку, мобільного застосунку, інформаційної системи або іншого програмного продукту. Важливо, щоб обраний проєкт мав чітко визначену мету та зрозумілий функціональний зміст.

Після визначення проєкту необхідно сформулювати його мету та очікувані результати. Формулювання мети повинно відображати основне призначення програмного продукту та задачі, які він повинен розв'язувати.

На наступному етапі необхідно визначити основні етапи реалізації проєкту. До таких етапів можуть належати аналіз вимог, проєктування програмної системи, реалізація програмного коду, тестування та впровадження програмного продукту.

Далі слід сформувати структуру робіт проєкту. Для цього необхідно визначити перелік завдань, що виконуються у межах кожного етапу проєкту. Завдання повинні бути чітко сформульованими та мати визначений результат виконання.

Після формування структури робіт необхідно визначити ресурси, необхідні для виконання проєкту. До ресурсів можуть належати учасники команди розробників, програмні інструменти, обчислювальні ресурси та інші необхідні засоби.

На наступному етапі формується базовий план ІТ-проєкту. План повинен містити перелік робіт, строки їх виконання, відповідальних виконавців та необхідні ресурси. Планування може виконуватися у вигляді таблиці або графічної схеми.

Після формування базового плану необхідно виконати аналіз ризиків. Для цього слід визначити можливі фактори, що можуть негативно вплинути на реалізацію проєкту. До таких факторів можуть належати технічні труднощі, зміна вимог до програмного продукту, недостатність ресурсів або організаційні проблеми.

Для кожного ризику необхідно оцінити ймовірність його виникнення та можливі наслідки для проєкту. Після цього необхідно запропонувати заходи щодо зменшення впливу ризиків або їх усунення.

Результати виконаної роботи необхідно узагальнити у звіті. У висновках слід оцінити ефективність сформованого плану ІТ-проєкту та обґрунтувати запропоновані заходи щодо управління ризиками.

**Контрольні питання**

1. Що розуміють під ІТ-проєктом?
2. Які основні характеристики ІТ-проєкту?
3. Що таке базовий план ІТ-проєкту?
4. Які основні компоненти містить базовий план проєкту?
5. Яку роль відіграє структура робіт у плануванні проєкту?
6. Які ресурси можуть використовуватися під час реалізації ІТ-проєкту?
7. Що розуміють під ризиком у проєктній діяльності?
8. Які етапи включає процес управління ризиками?
9. Яким чином виконується оцінювання ризиків?
10. Чому аналіз ризиків є важливим елементом управління ІТ-проєктом?

**Зміст звіту**

Звіт з практичної роботи повинен містити такі структурні елементи:
– назву практичної роботи;
– мету виконання практичної роботи;
– стислі відомості щодо планування ІТ-проєктів та управління ризиками;
– опис обраного ІТ-проєкту;
– сформульовану мету та результати проєкту;
– структуру робіт ІТ-проєкту;
– базовий план реалізації проєкту;
– перелік основних ризиків проєкту;
– результати аналізу та оцінювання ризиків;
– запропоновані заходи щодо зменшення ризиків;
– висновки за результатами виконання практичної роботи;
– перелік використаних джерел інформації.

## Практична робота 6. Аналіз етичних ситуацій у професійній діяльності інженера-програміста

### Ключові положення

Професійна діяльність інженера програмного забезпечення пов'язана не лише з технічними аспектами створення програмних систем, але й з необхідністю дотримання етичних норм, професійної відповідальності та принципів доброчесності. Програмні продукти використовуються у різних сферах суспільного життя, зокрема в економіці, освіті, медицині, транспорті, державному управлінні та інших галузях. Помилки або недобросовісні дії під час розроблення програмного забезпечення можуть мати значні соціальні, економічні та правові наслідки.

Етика професійної діяльності інженера програмного забезпечення визначає правила поведінки фахівця під час виконання професійних обов'язків, взаємодії з колегами, роботодавцями, замовниками та користувачами програмних продуктів. Дотримання етичних норм є важливою умовою формування довіри до інформаційних технологій та забезпечення відповідального використання програмних систем.

У сфері інженерії програмного забезпечення існують професійні кодекси етики, які визначають основні принципи відповідальної професійної діяльності. Такі кодекси формулюють загальні правила поведінки фахівців у процесі розроблення програмного забезпечення та взаємодії з іншими учасниками професійної діяльності. Вони визначають обов'язки інженера перед суспільством, роботодавцем, колегами та користувачами програмних систем.

Одним із важливих принципів професійної етики є відповідальність за результати розроблення програмного забезпечення. Інженер програмного забезпечення повинен забезпечувати належну якість програмних систем, дотримуватися технічних стандартів та використовувати перевірені технологічні рішення. Порушення цих принципів може призвести до появи помилок у програмних системах або виникнення ризиків для користувачів.

Іншим важливим аспектом професійної етики є дотримання принципів доброчесності. Доброчесність передбачає чесність у професійній діяльності, відповідальне використання інформації та повагу до інтелектуальної власності інших розробників. Недопустимими є прояви плагіату, неправомірне використання програмного коду або приховування недоліків програмного продукту.

У професійній діяльності інженерів програмного забезпечення можуть виникати різні етичні ситуації. Наприклад, розробник може зіткнутися з необхідністю прийняття рішення щодо використання сторонніх програмних компонентів без дотримання умов ліцензії. Іншою ситуацією може бути вимога з боку керівництва пришвидшити розроблення програмного продукту за рахунок зменшення обсягу тестування.

Етичні ситуації можуть виникати також під час роботи з конфіденційною інформацією користувачів. Інженер програмного забезпечення може мати доступ до персональних даних або комерційної інформації, що потребує дотримання правил інформаційної безпеки та конфіденційності.

Важливим етичним аспектом є також взаємодія з іншими учасниками команди розробників. У процесі роботи можуть виникати конфлікти інтересів, суперечки щодо технічних рішень або нерівномірний розподіл відповідальності між учасниками команди. У

таких ситуаціях важливо дотримуватися принципів професійної поваги, об’єктивності та конструктивного обговорення проблем.

Аналіз етичних ситуацій дозволяє сформувати у майбутніх фахівців навички відповідального прийняття рішень у професійній діяльності. Вміння оцінювати можливі наслідки професійних дій, враховувати інтереси користувачів та дотримуватися принципів професійної етики є важливою складовою підготовки інженерів програмного забезпечення.

Таким чином, етичні аспекти діяльності інженера програмного забезпечення є невід’ємною складовою професійної підготовки фахівців у сфері інформаційних технологій. Аналіз типових етичних ситуацій дозволяє сформувати відповідальне ставлення до професійної діяльності та підвищити рівень професійної культури майбутніх ІТ-фахівців.

**Практичне завдання**

1. Ознайомитися з основними принципами професійної етики у сфері інженерії програмного забезпечення.
2. Розглянути приклади типових етичних ситуацій, що можуть виникати у професійній діяльності інженера-програміста.
3. Обрати одну або декілька етичних ситуацій для детального аналізу.
4. Описати обрану ситуацію та визначити її основні особливості.
5. Визначити можливі варіанти дій інженера програмного забезпечення у цій ситуації.
6. Проаналізувати можливі наслідки кожного варіанту дій.
7. Обґрунтувати найбільш етично обґрунтоване рішення.
8. Сформулювати висновки щодо ролі етичних принципів у професійній діяльності інженера програмного забезпечення.

**Методичні вказівки до виконання практичного завдання**

Перед виконанням практичної роботи студент повинен ознайомитися з теоретичними положеннями щодо професійної етики у сфері інженерії програмного забезпечення та основними принципами відповідальної професійної діяльності.

На першому етапі необхідно дослідити основні принципи професійної етики, що застосовуються у сфері розроблення програмного забезпечення. Доцільно звернути увагу на питання відповідальності розробників перед суспільством, замовниками програмного продукту та користувачами програмних систем.

На наступному етапі необхідно розглянути приклади типових етичних ситуацій, що можуть виникати у професійній діяльності інженера програмного забезпечення. Приклади таких ситуацій можуть стосуватися використання стороннього програмного коду, порушення умов ліцензування програмного забезпечення, роботи з конфіденційною інформацією або прийняття рішень щодо якості програмного продукту.

Після вибору конкретної етичної ситуації необхідно виконати її детальний аналіз. У межах аналізу слід описати умови виникнення ситуації, визначити її основних учасників та сформулювати можливі варіанти дій інженера програмного забезпечення.

На наступному етапі необхідно проаналізувати наслідки кожного можливого варіанту дій. При цьому слід враховувати вплив прийнятого рішення на користувачів програмного продукту, роботодавця, інших членів команди та суспільство в цілому.

Після виконання аналізу необхідно визначити найбільш обґрунтований варіант дій з точки зору професійної етики. Вибір рішення повинен бути аргументованим та відповідати принципам відповідальної професійної діяльності.

Результати виконаної роботи необхідно систематизувати у звіті. У висновках слід оцінити значення етичних норм для професійної діяльності інженера програмного забезпечення та визначити роль етичних принципів у процесі розроблення програмних систем.

**Контрольні питання**

1. Що розуміють під професійною етикою інженера програмного забезпечення?
2. Які основні принципи професійної етики у сфері розроблення програмного забезпечення?
3. Чому важливо дотримуватися принципів доброчесності у професійній діяльності?
4. Які етичні ситуації можуть виникати у діяльності інженера програмного забезпечення?
5. У чому полягає відповідальність розробника програмного забезпечення перед користувачами?
6. Чому важливо дотримуватися конфіденційності інформації під час розроблення програмних систем?
7. Що таке конфлікт інтересів у професійній діяльності?
8. Яку роль відіграє професійний кодекс етики інженера програмного забезпечення?
9. Яким чином аналіз етичних ситуацій сприяє формуванню професійної культури ІТ-фахівців?
10. Чому професійна відповідальність є важливою складовою діяльності інженера програмного забезпечення?

**Зміст звіту**

Звіт з практичної роботи повинен містити такі структурні елементи:

– назву практичної роботи;

– мету виконання практичної роботи;

– стислі відомості щодо професійної етики у сфері інженерії програмного забезпечення;

– опис обраної етичної ситуації;

– аналіз можливих варіантів дій у межах цієї ситуації;

– оцінювання наслідків прийнятих рішень;

– обґрунтування найбільш етично обґрунтованого рішення;

– висновки за результатами виконання практичної роботи;

– перелік використаних джерел інформації.

## Рекомендована література

### Основна


1. Sommerville, Ian. Engineering Software Products: An Introduction to Modern Software Engineering. Pearson, 2020. 342 p. ISBN: 9780135210642.
2. Roger S. Pressman, Bruce R. Maxim. Software Engineering: A Practitioner's Approach. 2019. 1408 p. ISBN 9781260548006
3. Timinger, Holger. Modern Project Management: Successful Projects with Plan-Based, Agile and Hybrid Approaches. Германія: John Wiley & Sons, Incorporated, 2025. 592p. ISBN: 9783527850631
4. Winters, Titus., Manshreck, Tom., Wright, Hyrum. Software Engineering at Google: Lessons Learned from Programming Over Time. O'Reilly Media, 2020. 602p. ISBN: 9781492082767.
5. І.Л. Бородкіна, Г.О. Бородкін Інженерія програмного забезпечення: Посібник для студентів вищих навчальних закладів. К.: Центр учбової літератури, 2021. 204 с. ISBN 9786110112321.
6. Левус Є. В., Мельник Н. Б. Вступ до інженерії програмного забезпечення: навчальний посібник. Львів: Видавництво Львівської політехніки, 2018. 248 с. ISBN: 9789669411310.


### Допоміжна


7. ISO/IEC/IEEE 12207: Systems and Software Engineering – Software Life Cycle Processes. Швейцарія: IEEE. 2017. ISBN: 9781504442534.
8. Quinn, Michael Jay. Ethics for the Information Age. США: Pearson, 2024. 553. ISBN: 9780138238537.
9. Thomas, David., Hunt, Andrew. The Pragmatic Programmer: Your Journey to Mastery, 20th Anniversary Edition. Великобританія: Pearson Education. 2019. 352p. ISBN 9780135956915.
10. Forsgren, Nicole., Humble, Jez., Kim, Gene. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. США: IT Revolution, 2018. 254p. ISBN: 9781942788331.
11. Fairley, Richard E.. Systems Engineering of Software-Enabled Systems. CIF: Wiley, 2019. 432p. ISBN: 9781119535034.


### Інформаційні ресурси


12. IEEE Computer Society. Software Engineering Body of Knowledge (SWEBOK). URL: https://www.computer.org/education/bodies-of-knowledge/software-engineering
13. PMI – Project Management Institute. URL: https://www.pmi.org/standards/pmbok
14. Scrum Guides – The Scrum Guide. URL: https://scrumguides.org/download.html
15. ISO – ISO/IEC/IEEE 12207:2017. URL: https://www.iso.org/standard/63712.html
16. Git Documentation. URL: https://git-scm.com/docs

Навчальне видання

**Русу Олександр Петрович**

# ВСТУП ДО СПЕЦІАЛЬНОСТІ

Методичні вказівки до виконання практичних робіт
для здобувачів вищої освіти, які навчаються за спеціальностями
галузі «Інформаційні технології»

Українською мовою