

Методичні вказівки до виконання курсової роботи з дисципліни «Технології створення програмних продуктів». Частина № 1

Вступ

В процесі розробки програмного продукту, в основному на етапі конструювання, проектна команда активно використовує репозиторії програмного коду. Якщо розбити структуру пояснювальної записки до курсової роботи з дисципліни ТСПП на окремі файли будь якого формату електронного документу, наприклад docx, тоді буде створено репозиторій проекту для сумісної розробки такої пояснювальної записки. Приклад такого проекту представлено за адресою - https://github.com/oleksandrblazhko/Project_Example

Завдання

Для виконання завдання учасники проектної команди повинні мати github-облікові записи. Подробиці роботи з github-репозиторієм - <https://git-scm.com/book/uk/v2>

Необхідно виконати наступні завдання.

Завдання 1. Створити github-репозиторій з назвою програмного продукту латиницею

Завдання 2. В репозиторії створити файл README.md, який містить наступне:

- назва програмного продукту
- мета програмного продукту
- список учасників команди (ПІБ та група)

Завдання 3. Підключити до репозиторію github-облікові записи всіх учасників проектної команди (розділ *Settings->Manage access*).

Завдання 4. Всім учасникам створити свої локальні git-репозиторії та клонувати копію проекту.

Завдання 5. Розподілити між учасниками проектної команди завдання зі створення каталогів та файлів, які описано в таблиці 1.

Завдання 6. Кожен учасник проектної команди повинен створити каталоги та файли у відповідності визначеними завданнями. Зміст файлів відповідає рішенням лабораторно-практичних завдань №1-5. Подробиці у Додатку А цих методичних вказівок.

Завдання 7. Кожен учасник проектної команди після завершення створення своїх каталогів та файлів оновлює github-репозиторій.

Завдання 8. Додати до кореня github-репозиторія файл з таблицею зі структурою таблиці 2.

Таблиця 1 – Опис підрозділів пояснювальної записки та відповідних каталогів, файлів у Github-репозиторію

Назва розділів, та пунктів пояснювальної записки	Назва каталогу або файлу у відповідності зі структурою пояснювальної записки	Тип файлу Github-репозиторію
1 Вимоги до програмного продукту	1_Software_product_requirements	Каталог
1.1 Визначення потреб споживача	1.1_Determining_consumer_needs	Каталог
1.1.1 Ієрархія потреб споживача	1.1.1_Hierarchy_of_consumer_needs	Файл
1.1.2 Деталізація матеріальної потреби	1.1.2_Material_needs_details	Файл
1.2 Бізнес-вимоги до програмного продукту	1.2_Business_requirements_for_software_Product	Каталог
1.2.1 Опис проблеми споживача	1.2.1_Consumer_problem_description	Каталог
1.2.1.1 Концептуальний опис проблеми споживача	1.2.1.1_Concept_description_of_consumer_problem	Файл

Таблиця - Продовження таблиці

Назва розділів, та пунктів пояснювальної записки	Назва каталогу або файлу у відповідності зі структурою пояснювальної записки	Тип файлу Github-репозиторію
1.2.1.2 Метричний опис проблеми споживача	1.2.1.2_Metric_description_of_consumer_problem	Файл
1.2.2 Мета створення програмного продукту	1.2.2_The_purpose_of_software_product	Каталог
1.2.2.1 Проблемний аналіз існуючих програмних продуктів	1.2.2.1_Problem_analysis_of_existing_software	Файл
1.2.2.2 Мета створення програмного продукту	1.2.2.2_Software_goal	Файл
1.2.3 Назва програмного продукту	1.2.3_Software_product_name	Каталог
1.2.3.1 Гасло програмного продукту	1.2.3.1_Software_product_slogan	Файл
1.2.3.2 Логотип програмного продукту	1.2.3.2_Software_product_logo	Файл
1.3 Вимоги користувача до програмного продукту	1.3_User_req_for_software	Каталог
1.3.1 Історія користувача програмного продукту	1.3.1_User_Story	Файл
1.3.2 Діаграма прецедентів програмного продукту	1.3.2_Use_Case_diagram	Файл
1.3.3 Сценаріїв використання прецедентів програмного продукту	1.3.3_Precedent_scenarios	Файл
1.4 Функціональні вимоги до програмного продукту	1.4_Func_req_for_software	Каталог
1.4.1. Багаторівнева класифікація функціональних вимог	1.4.1_Multilevel_classification_of_func_req	Файл
1.4.2 Функціональний аналіз існуючих програмних продуктів	1.4.2_Functional_analysis_of_existing_software	Файл
1.5 Нефункціональні вимоги до програмного продукту	1.5_Non-functional_requirements_for_the_software	Каталог
1.5.1 Опис зовнішніх інтерфейсів	1.5.1_Description_of_external_interfaces	Каталог
1.5.1.1 Опис інтерфейса користувача	1.5.1.1_User_interface_description	Каталог

Таблиця - Продовження таблиці

Назва розділів, та пунктів пояснювальної записки	Назва каталогу або файлу у відповідності зі структурою пояснювальної записки	Тип файлу Github-репозиторію
1.5.1.1.1 Опис INPUT-інтерфейса користувача	1.5.1.1.1_Description_of_INPUT_user_interface	Файл
1.5.1.1.2 Опис OUTPUT-інтерфейса користувача	1.5.1.1.2_Description_of_OUTPUT_user_interface	Файл
1.5.1.2 Опис інтерфейсу із зовнішніми пристроями	1.5.1.2_Description_of_the_interface_with_ext_dev	Файл
1.5.1.3 Опис програмних інтерфейсів	1.5.1.3_Description_of_software_interfaces	Файл
1.5.1.4 Опис інтерфейсів передачі інформації	1.5.1.4_Description_of_network_interfaces	Файл
1.5.1.5 Опис атрибутів продуктивності	1.5.1.5_Description_of_performance_attributes	Файл
2 Планування процесу розробки програмного продукту	2_Planning_the_software_development_process	Каталог
2.1 Планування ітерацій розробки програмного продукту	2.1_Planning_iterations_of_software_development	Файл
2.2 Концептуальний опис архітектури програмного продукту	2.2_Conceptual_description_of_software_architect	Файл
2.3 План розробки програмного продукту	2.3_Software_development_plan	Каталог
2.3.1 Оцінка трудомісткості розробки програмного продукту	2.3.1_Complexity_assessment_of_software_dev	Файл
2.3.2 Визначення дерева робіт з розробки програмного продукту	2.3.2_Defining_a_tree_of_work_on_software_dev	Файл
2.3.3 Графік робіт з розробки програмного продукту	2.3.3_Schedule_of_software_development	Каталог
2.3.3.1 Таблиця з графіком робіт	2.3.3.1_Table_with_work_schedule	Файл
2.3.3.2 Діаграма Ганта	2.3.3.2_Gantt_chart	Файл
3 Проектування програмного продукту	3_Software_design	Каталог
3.1 Концептуальне та логічне проектування структур даних програмного продукту	3.1_Conceptual_and_logical_design_of_data_structures	Каталог

Таблиця - Продовження таблиці

Назва розділів, та пунктів пояснювальної записки	Назва каталогу або файлу у відповідності зі структурою пояснювальної записки	Тип файлу Github-репозиторію
3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів	3.1.1_Conceptual_design_based_on_UML-diagram_of_conceptual_classes	Файл
3.1.2 Логічне проектування структур даних	3.1.2_Logical_design_of_data_structures	Файл

Таблиця 2 – Статистика участі проектної команди у створенні репозиторію

Назва розділів, та пунктів пояснювальної записки	Назва каталогу або файлу у відповідності зі структурою пояснювальної записки	Тип файлу Github-репозиторію	ПІБ учасника проектної команди
--	--	------------------------------	--------------------------------

Додаток А Структура розділів, підрозділів, пунктів і підпунктів пояснювальної записки до курсової роботи

1 Вимоги до програмного продукту

1.1 Визначення потреб споживача

1.1.1 Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

- фізіологія (вода, їжа, житло, сон);
- безпека (особиста, здоров'я, стабільність),
- приналежність (спілкування, дружба, любов),
- визнання (повага оточуючих, самооцінка),
- самовираження (вдосконалення, персональний розвиток).

Опишіть в цьому пункті одну ієрархію потреби вашого споживача, яку хотілося б задовольнити, використовуючи майбутній програмний продукт.

1.1.2 Деталізація матеріальної потреби

Для деталізації матеріальної потреби можна скористатися ментальними картами (MindMap). При створенні ментальних карт матеріальна потреба розташовується в центрі карти. Асоціативні гілки можна швидко створити, припускаючи, що в загальному вигляді з об'єктом пов'язані три потоки даних / інформації: вхідний, внутрішній, вихідний. Кожен потік - це асоціативна група, що включає можливі п'ять гілок, що відповідають на п'ять питань: Хто? Що? Де? Коли? Як? Відповідно до рекомендацій по створенню ментальних карт кожна гілка-асоціація може бути розділена на додаткові асоціативні гілки, які деталізують відповіді на поставлені питання.

Створіть в цьому пункті ментальну карту деталізації ієрархії потреби вашого споживача.

1.2 Бізнес-вимоги до програмного продукту

1.2.1 Опис проблеми споживача

1.2.1.1 Концептуальний опис проблеми споживача

Для скорочення часу і коштів при задоволенні реальних потреб людині потрібна інформація, що призводить до появи інформаційної потреби.

Відомо, що інформація визначається вимогами до даних для їх можливого перетворення в інформацію:

- 1) доступність, коли необхідно відповідати на питання «де щось-то?» і «чи є воно?»;
- 2) уявлення мовою, зрозумілою споживачеві, коли необхідно відповідати на питання «а що це?»;
- 3) цінність (корисність) для споживача, коли необхідно відповідати на питання «як використовувати?» і «з якою метою використовувати?»
- 4) актуальність, коли необхідно відповідати на питання «коли?».

Якщо зазначені вимоги не задовольняються виникає віртуальна проблема споживача як незадоволена інформаційна потреба.

При описі проблеми необхідно використовувати прислівники, явно вказують на незадоволеність процесом. Прикладами таких прислівників можуть бути слова: довго, повільно, дорого, важко, незрозуміло, неможливо.

Опишіть в цьому пункті вимога до інформації, яка не задовольняється для вашого споживача.

1.2.1.2 Метричний опис проблеми споживача

При описі проблеми споживача повинні бути використані метрики, якісно або кількісно в числовій формі описують проблему.

Програмне забезпечення може стати програмним продуктом, якщо воно забезпечить споживачеві задоволення умов переходу даних в інформацію.

Опишіть в цьому пункті метрику інформаційної потреби вашого споживача, яку можна виміряти.

1.2.2 Мета створення програмного продукту

1.2.2.1 Проблемний аналіз існуючих програмних продуктів

Проблемний аналіз існуючих програмних продуктів включає кроки:

- 1) формування переліку товарів через пошук в інтернеті;
- 2) формування таблиці рішення проблем (рядки - назви продуктів, стовпці - проблеми, осередки - позначки про рішення проблем продуктом).

Намалюйте в цьому пункті таблицю з проблемним аналізом існуючих програмних продуктів

1.2.2.2 Мета створення програмного продукту

Визначення цілей, які повинен досягати програмний продукт з урахуванням кількісних критеріїв, які демонструють економічний або інший відчутний ефект від впровадження продукту для зацікавлених осіб, з урахуванням раніше визначених проблем як їх антонімів, наприклад, «підвищення зручності (ергономіки) ...», «скорочення часу ... », «скорочення витрат ... » і інші.

Мета - це антонім по відношенню до проблеми. Мета також повинна бути представлена у вигляді метричних слів, які будуть підтверджені після впровадження програмного продукту.

Опишіть в цьому пункті метричну мета вашого програмного продукту.

1.2.3 Назва програмного продукту

1.2.3.1 Гасло програмного продукту

Рекомендується запропонувати унікальну назву програмного продукту, що включає:

- потреба матеріального світу;
- досягається мета задоволення інформаційної потреби.

Придумайте назву програмного продукту, яке стане гаслом, що включає проблему споживача і мета продукту, хоч і частково, але вирішальну його проблему.

1.2.3.2 Логотип програмного продукту

Відмінним способом представлення назви є його логотип, що поєднують зорові образи і короткі фрази-гасла.

Намалюйте картинку, яка візуально описує гасло продукту.

1.3 Вимоги користувача до програмного продукту

1.3.1 Історія користувача програмного продукту

Опишіть у довільній формі історію можливої взаємодії користувач з майбутнім ПП. Рекомендується кожне речення розміщувати окремим абзацем.

1.3.2 Діаграма прецедентів програмного продукту

Діаграма прецедентів (Use Case UML-діаграма) включає:

- актори (зацікавлені особи і зовнішні системи зі своїм API);
- прецеденти як основні функції ПП;
- зв'язки між прецедентами і акторами як множиною зацікавлених осіб;
- можливі зв'язки-узагальнення між акторами.

1.3.3 Сценарії використання прецедентів програмного продукту

Для кожного прецедента описати сценарій використання з урахуванням пунктів:

- назва прецеденту;

- передумови початку виконання прецеденту;
- актори як зацікавлені особи у виконанні прецеденту;
- актор-основна зацікавлена особа як ініціатор початку прецеденту;
- гарантії успіху (що отримають актори у разі успішного завершення прецеденту);
- основний успішний сценарій (формат опису: <номер кроку> <опис дії>);
- альтернативні сценарії, прив'язані до кроків основного успішного сценарію (формат опису: <номер кроку.номер розширення> <умова>: <дію або посилання на підлеглий варіант використання>).

1.4 Функціональні вимоги до програмного продукту

1.4.1. Багаторівнева класифікація функціональних вимог

З метою упорядкування функцій ПП створіть багаторівневу класифікацію функціональних вимог (*Functional Requirements - FR*), виявлених в сценаріях використання прецедентів. Рекомендується використати формат структур, схожих на структури декомпозиції робіт (*Work Breakdown Structure - WBS*), але без вказівки на складність і тривалість.

Кожну функцію в ієрархії позначте унікальним ієрархічним ідентифікатором FRi, наприклад, FR1, FR1.1

При визначенні функцій рекомендується функції першого рівня ієрархії визначати на основі назв прецедентів, а функції другого рівня ієрархії визначати на основі опису кроків основного успішного та альтернативного сценаріїв роботи прецедентів.

Повні назви функцій надайте у формі таблиці, приклад структури якої наведено в таблиці 1.

Таблиця 1 – приклад опису функцій з наданням унікальних ієрархічних ідентифікаторів

Ідентифікатор функції (назва)	Назва функції
FR1 (назва1)	Авторизація користувача
FR1.1 (назва1.1)	Створення запиту у користувача на отримання його параметрів ідентифікації та аутентифікації
FR1.2 (назва1.2)	Передача від користувача його параметрів ідентифікації та аутентифікації
...	
FR2(назва2)	Отримання меню
FR3(назва3)	Створення замовлення
...	

Приклад опису ієрархічної WBS-структури багаторівневої класифікації функціональних вимог представлено на рисунку 1.

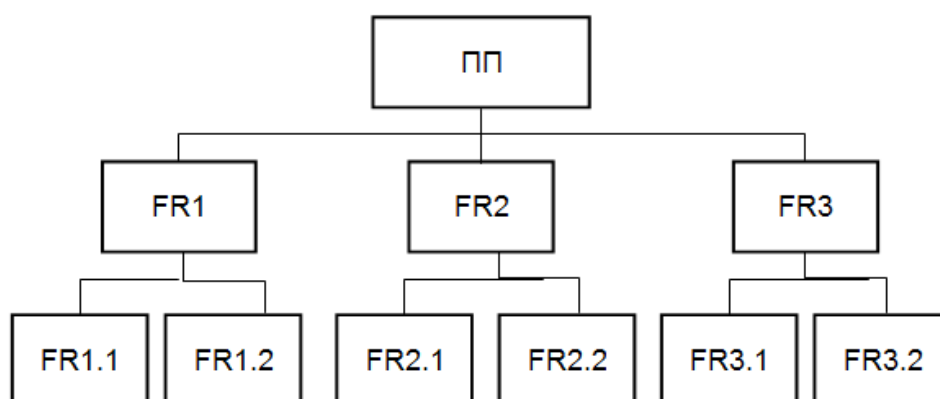


Рис. 1 – Приклад WBS-структури багаторівневої класифікації функціональних вимог

1.4.2 Функціональний аналіз існуючих програмних продуктів

З метою швидкого прототипування роботи ПП проведіть функціональний аналіз існуючих ПП у вигляді таблиці функцій, де:

- рядки – ідентифікатори функцій нижнього рівня ієрархії (листя в дереві ієрархії);
- осередки - позначки про реалізацію функції в ПП, наприклад, позначка «+» – функція реалізована в ПП, позначка «-» – функція не реалізована.

Приклад структури таблиці представлено в таблиці 2.

Таблиця 2 – приклад результатів функціонального аналізу існуючих ПП

Ідентифікатор функції (назва)	ПП1	...	ППn
FR1.1 (назва1)	+		-
FR1.2 (назва2)	+		
...			

1.5 Нефункціональні вимоги до програмного продукту

1.5.1 Опис зовнішніх інтерфейсів

1.5.1.1 Опис інтерфейса користувача

Опишіть інтерфейс користувача з урахуванням двох інформаційних потоків між користувачем та ПП: INPUT, OUTPUT.

Для опису проаналізуйте кожен функцію ПП нижнього (2-го) рівня ієрархії з метою вибору інтерфейсу, який пропонує користувачу передати дані до ПП (INPUT-засіб) або отримати від ПП (OUTPUT-засіб).

Важливо проводити вибір таких засобів інтерфейсу, які будуть наближати користувача до досягнення мети ПП, а не навпаки – ускладнювати досягнення мета!

Для однієї функції можливе використання декількох засобів.

1.5.1.1.1 Опис INPUT-інтерфейса користувача

Необхідно врахувати наступні засоби реалізації INPUT-потоків:

- контактні INPUT-засоби:
 - окремі кнопки, перемикачі;
 - стандартна комп'ютерна клавіатура;
 - 2/3-кнопочний маніпулятор типу "миша";
 - рукоятка керування «Joystick» (<https://en.wikipedia.org/wiki/Joystick>);
 - кульковий маніпулятор «Trackball» (<https://en.wikipedia.org/wiki/Trackball>);
 - багатофункціональний ігровий маніпулятор Gamepad (<https://en.wikipedia.org/wiki/Gamepad>);
 - сенсорний екран (Touchscreen, Touchpad, Multi-touch);
- безконтактні INPUT-засоби:
 - захоплення руху людини в - Motion capture (https://en.wikipedia.org/wiki/Motion_capture) в технології доповненої віртуальності (https://en.wikipedia.org/wiki/Mixed_reality#Augmented_virtuality);
 - голосовий інтерфейс (https://en.wikipedia.org/wiki/Voice_user_interface);
 - нейро-комп'ютерний інтерфейс (https://en.wikipedia.org/wiki/Brain%E2%80%93computer_interface).

Результат аналізу засобів INPUT-потоків необхідно представити у вигляді таблиці 3. В стовпчику «Особливості використання» надайте можливі деталі використання інтерфейсу, якщо вони є.

Таблиця 3 – приклад оформлення результатів аналізу засобів INPUT-потоків

Ідентифікатор функції (назва)	Засіб INPUT-потоків	Особливості використання
FR1.2 (назва1.2)	стандартна комп'ютерна клавіатура	
FR1.3 (назва1.3)	2/3-кнопочний маніпулятор типу "миша"	Використання колеса миші для завершення процесу вводу даних

1.5.1.1.2 Опис OUTPUT-інтерфейса користувача

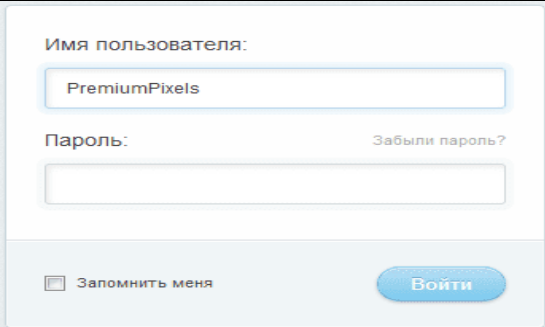
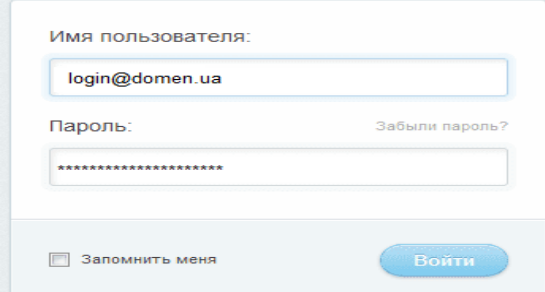
Необхідно врахувати наступні засоби реалізації OUTPUT-потоків:

- інтерфейс командного рядку;
- псевдографічний інтерфейс (https://en.wikipedia.org/wiki/Box-drawing_character);
- графічний інтерфейс;
- голосовий інтерфейс (https://en.wikipedia.org/wiki/Voice_user_interface);
- доповнена реальність (https://en.wikipedia.org/wiki/Augmented_reality)

Для графічного інтерфейсу рекомендується показати Mockup (макет) розміщення елементів інтерфейсу на екрані. Для створення макетів можна використати безкоштовний режим роботи програми <https://moqups.com/>

Результат аналізу засобів OUTPUT-потоків необхідно представити у вигляді таблиці 4.

Таблиця 4 – приклад оформлення результатів аналізу засобів OUTPUT-потоків

Ідентифікатор функції (назва)	Засіб OUTPUT - потоків	Особливості використання
FR1.1 (назва1.1)	графічний інтерфейс	
FR1.2 (назва1.2)	графічний інтерфейс	

1.5.1.2 Опис інтерфейсу із зовнішніми пристроями (обладнанням)

Опишіть інтерфейси із зовнішніми пристроями (обладнанням).

Для опису проаналізуйте кожну функцію ПП нижнього (2-го) рівня ієрархії з метою вибору зовнішніх пристроїв (обладнання)

Важливо проводити вибір таких приладів, які будуть наближати користувача до досягнення мети ПП, а не навпаки – ускладнювати досягнення мета!

Рекомендується використати такі зовнішні пристрої:

- Desktop-персональний комп'ютер;
- Notebook;
- смартфон;
- мобільний телефон;
- пристрої друку;
- прилади доповненої та віртуальної реальності;
- різноманітні сенсори мікроконтролерів з інтернету речей;
- спеціалізовані пристрої отримання даних та відображення даних.

Результат аналізу необхідно представити у вигляді таблиці 5.

Таблиця 5 – приклад оформлення результатів аналізу зовнішніх пристроїв

Ідентифікатор функції (назва)	Зовнішній пристрій
FR1.1 (назва1.1)	смартфон

1.5.1.3 Опис програмних інтерфейсів

Опишіть:

- версії операційних систем, які будуть необхідні користувачу при роботі з програмним продуктом;
- зовнішні програмні продукти, з якими буде взаємодіяти програмний продукт
- зовнішні програмні бібліотеки, або API-сервіси, які знадобляться для роботи нового програмного продукту.

1.5.1.4 Опис інтерфейсів передачі інформації

Опишіть інтерфейси передачі інформації, які знадобляться при реалізації більшості функцій ПП.

Рекомендується використати такі зовнішні пристрої:

- провідні інтерфейси:
 - Ethernet
 - GigabitEthernet
 - інше
- безпроводні інтерфейси:
 - Bluetooth;
 - Wi-Fi;
 - інше
- інші варіанти інтерфейсів передачі інформації

1.5.1.5 Опис атрибутів продуктивності

Найчастіше використовувати такі характеристики продуктивності:

- максимальний час реакції ПП на дії користувачів;
- максимальна кількість одночасно обслуговуваних користувачів.

В роботі необхідно визначити лише максимальний час реакції ПП на дії користувачів.

Для опису проаналізуйте кожну функцію ПП нижнього (2-го) рівня ієрархії та заповніть таблицю 6.

Таблиця 6 – приклад оформлення результатів аналізу характеристик продуктивності

Ідентифікатор функції (назва)	Максимальний час реакції ПП на дії користувачів, секунди
FR1.1 (назва 1.1)	3
FR1.2 (назва1.2)	3

2 Планування процесу розробки програмного продукту

2.1 Планування ітерацій розробки програмного продукту

З метою забезпечення для вимог таких рекомендацій IEEE-стандарту, як необхідність, корисність при експлуатації, здійсненість функціональних вимог до ПП, визначте функціональні пріоритети, які будуть використані при плануванні ітерацій розробки ПП.

При створенні пріоритетів необхідно врахувати:

- сценарні залежності між прецедентами, до яких належать функції, на основі аналізу пунктів передумов початку роботи прецедентів, вказаних в описі сценаріїв роботи прецедентів;

- вплив роботи прецеденту, до якого належить функція, на досягнення мети ПП, наприклад у відсотках, на основі аналізу пунктів гарантій успіху, вказаних в описі сценаріїв роботи прецедентів.

Сценарні залежності будуть перетворені у відповідні функціональні залежності.

Вплив роботи прецеденту буде поширено на всі підлеглі функції ієрархії.

При визначенні пріоритетів рекомендується використовувати наступні позначки:

- *M (Must)* – функція повинна бути реалізованою у перших ітераціях за будь-яких обставин;

- *S (Should)* – функція повинна бути реалізованою у перших ітераціях, якщо це взагалі можливо;

- *C (Could)* – функція може бути реалізованою, якщо це не вплине негативно на строки розробки;

- *W (Want)* – функція може бути реалізованою у наступних ітераціях.

Приклад опису представлено в таблиці 2.

Таблиця 2 – приклад опису функціональних пріоритетів

Ідентифікатор функції	Функціональні залежності	Вплив на досягнення мети, %	Пріоритет функції
F1.1	-	0	M
F1.1.1	-	0	M
F1.1.2	-	0	M
...			
F1.2	F1.1	15%	M
F1.3	F1.1	5%	S
...			

Проранжуйте всі функції за пріоритетами. В першу ітерацію розробки можуть потрапити функції Must, додатково Should. У другу ітерацію можуть потрапити інші функції.

2.2 Концептуальний опис архітектури програмного продукту

Опишіть компоненти ПП, відповідальні за три рівня роботи, які історично називають:

- рівень уявлення – Presentation Level (PL), який містить компоненти зовнішнього інтерфейсу (програмний інтерфейс користувача), наприклад, Client Operation System, Input, Output,

- рівень бізнес-логіки – Business Level (BL), який містить апаратно-програмні компоненти обробки даних, наприклад, Server Operation System, Application Server;

- рівень зберігання даних – Access Level (AL), який містить апаратно-програмні компоненти керування даними, наприклад, Server Operation System, DataBase Server.

В сучасній термінології програмні компоненти PL-рівня називають FrontEnd, а програмні компоненти BL-рівня та AL-рівня називають Backend (https://en.wikipedia.org/wiki/Front_end_and_back_end).

Необхідно представити компоненти графічно, вказуючи зв'язки двох типів:

- передачі даних між компонентами;
- передачі управління між компонентами.

Для кожного компонента необхідно вказувати особливості реалізації, наприклад: типи апаратних компонент, операційні системи, програмні технології розробки, системи керування базами даних, типи структур даних (реляційні, XML, JSON).

Рекомендується показувати ієрархію залежності компонентів від апаратних компонент до програмних компонент.

Для опису рекомендується використовувати UML-діаграму розгортання (https://en.wikipedia.org/wiki/Deployment_diagram). Прикладом FreeWare-інструменту створення UML-діаграм може бути draw.io

Приклад UML-діаграм розгортання представлено на рисунку 2.

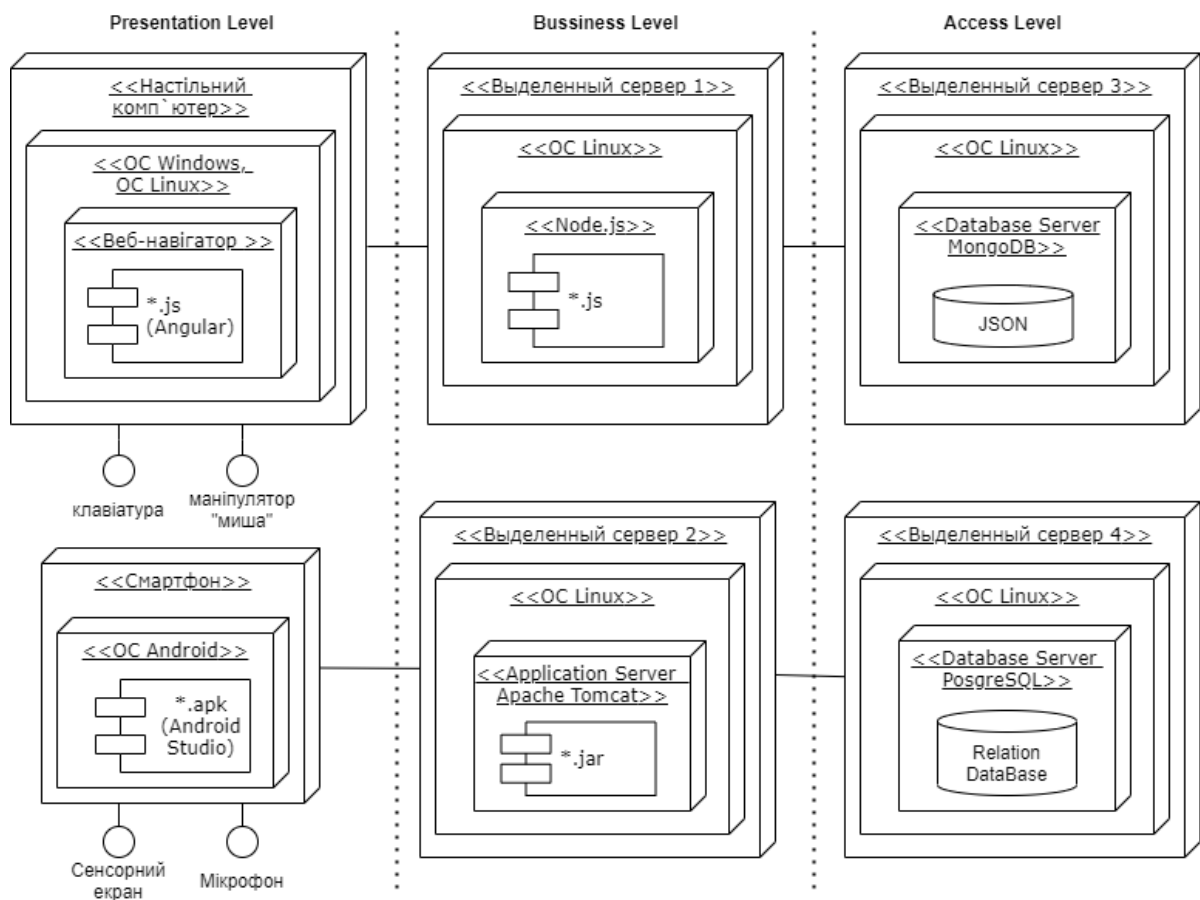


Рис. 2 – Приклад UML-діаграми розгортання ПП

2.3 План розробки програмного продукту

2.3.1 Оцінка трудомісткості розробки програмного продукту

Визначення обсягів робіт по одному з методів оцінки - <https://www.ibm.com/developerworks/ru/library/d-estimation-agile-or-conventional-trs/>

Рекомендується використовувати методику Use Case Point, яка має наступні кроки.

- Крок 1. Визначення вагових показників акторів А
- Крок 2. Визначення вагових показників прецедентів UC
- Крок 3. Визначення UUCP
- Крок 4. Визначення технічної складності проекту
- Крок 5. Визначення рівня кваліфікації розробників
- Крок 6. Визначення UCP
- Крок 7. Оцінка трудомісткості проекту

2.3.2 Визначення дерева робіт з розробки програмного продукту

При створенні дерева робіт (Work BreakDown Structure- WBS) використовується дерево функцій, яке було створено раніше.

Кожна функція 1-го рівня ієрархії перетворюється в Work Package (WP)

Кожна функція 2-го рівня ієрархії перетворюється в Work Task (WT).

Для кожної задачі визначаються підзадачі - Work SubTask (WST) з урахуванням базових процесів розробки програмних модулів: проектування, конструювання, модульне тестування, збірка та системне тестування. Приклад WBS представлено на рисунку 3.

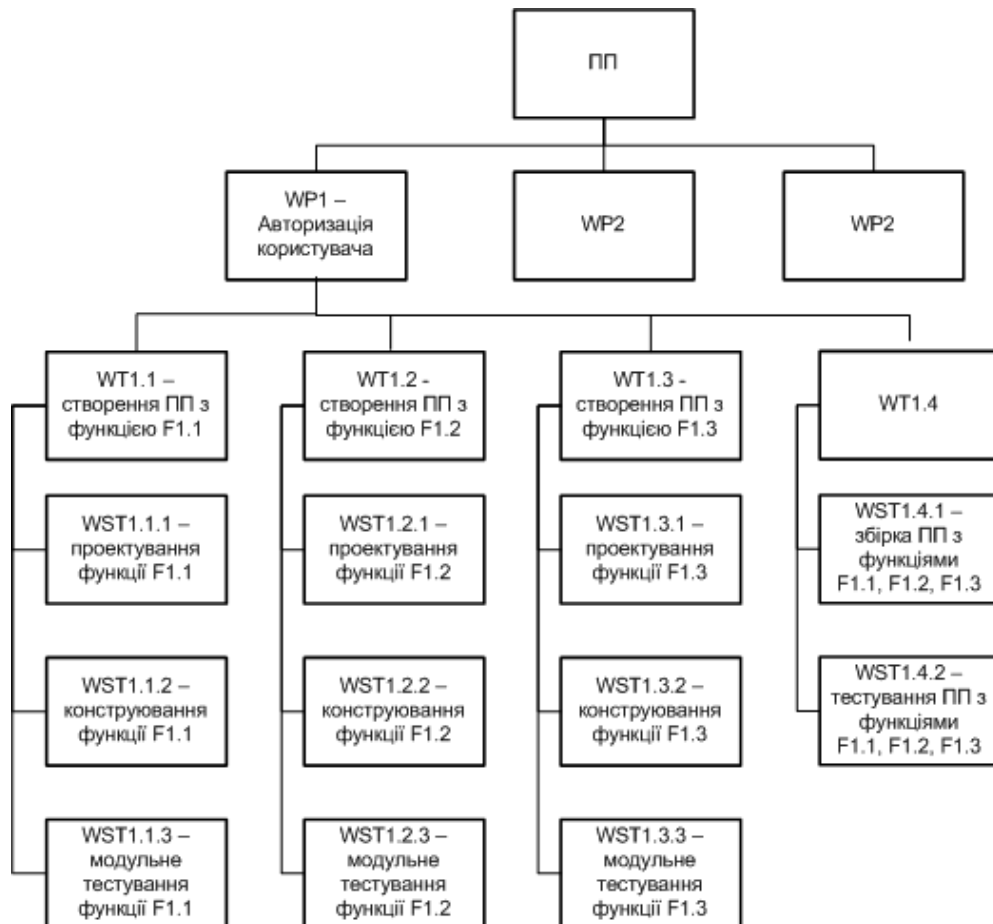


Рис. 3 – Приклад WBS

Для кожної підзадачі визначається виконавець, що фіксується у вигляді таблиці, приклад якої представлено в таблиці 5.

Таблиця 5 приклад опису підзадач із закріпленням виконавців

Підзадача	Виконавець
WST1.1.1	Петренко О.А.
WST1.1.2	Петренко О.А.
WST1.1.3	Петренко О.А.
WST1.2.1	Іваненко А.С.
WST1.2.2	Іваненко А.С.
WST1.2.3	Іваненко А.С.
WST1.3.1	Федоренко С.С.
WST1.3.2	Федоренко С.С.
WST1.3.3	Федоренко С.С.
WST1.4.1	Петренко О.А.

WST1.4.2	Петренко О.А.
----------	---------------

2.3.3 Графік робіт з розробки програмного продукту

2.3.3.1 Таблиця з графіком робіт

Необхідно підготувати таблицю з графіком робіт, приклад якої представлено на рисунку 4.

WST	Дата початку	Дні	Дата завершення	Виконавець
1.1.1	01.10.2020	1	01.10.2020	Петренко О.А.
1.1.2	02.10.2020	2	03.10.2020	Петренко О.А.
1.1.3	04.10.2020	1	04.10.2020	Петренко О.А.
1.2.1	01.10.2020	1	01.10.2020	Іваненко А.С.
1.2.2	02.10.2020	2	03.10.2020	Іваненко А.С.
1.2.3	04.10.2020	1	04.10.2020	Іваненко А.С.
1.3.1	01.10.2020	1	01.10.2020	Федоренко С.С.
1.3.2	02.10.2020	2	03.10.2020	Федоренко С.С.
1.3.3	04.10.2020	1	04.10.2020	Федоренко С.С.
1.4.1	05.10.2020	1	05.10.2020	Петренко О.А.
1.4.2	06.10.2020	1	06.10.2020	Петренко О.А.

Рис. 4 – Приклад таблиці графіка робіт

2.3.3.2 Діаграма Ганта

Діаграма Ганта (складається із смуг (вісь Y), орієнтованих уздовж осі часу (вісь X).

Кожна смуга – окрема підзадача в проекті, її кінці - моменти початку і завершення роботи, її протяжність - тривалість роботи. Мета діаграми - візуально показати послідовність процесів та можливість паралельного виконання робіт.

Для створення діаграми можна використати різні інструменти, наприклад, Excel або Google Sheet.

Приклад створення за посиланням - <https://intasker.com/blog/gantt-chart-in-excel/>

Приклад таблиці та діаграми представлено за посиланням - https://docs.google.com/spreadsheets/d/1DYfMZzMqfKFoqpye8rAynuPXk2_2xLOyggK3Tgf4m9U

За рекомендаціями вказаного посилання була створена діаграма Ганта, представлена на рисунку 5.

	A	B	C	D	E	F	G	H	I	J	K
1	WST	Дата початку	Дні	Дата завершення	Виконавець	01.10.2020	02.10.2020	03.10.2020	04.10.2020	05.10.2020	06.10.2020
2	1.1.1	01.10.2020	1	01.10.2020	Петренко О.А.						
3	1.1.2	02.10.2020	2	03.10.2020	Петренко О.А.						
4	1.1.3	04.10.2020	1	04.10.2020	Петренко О.А.						
5	1.2.1	01.10.2020	1	01.10.2020	Іваненко А.С.						
6	1.2.2	02.10.2020	2	03.10.2020	Іваненко А.С.						
7	1.2.3	04.10.2020	1	04.10.2020	Іваненко А.С.						
8	1.3.1	01.10.2020	1	01.10.2020	Федоренко С.С.						
9	1.3.2	02.10.2020	2	03.10.2020	Федоренко С.С.						
10	1.3.3	04.10.2020	1	04.10.2020	Федоренко С.С.						
11	1.4.1	05.10.2020	1	05.10.2020	Петренко О.А.						
12	1.4.2	06.10.2020	1	06.10.2020	Петренко О.А.						

Рис. 5 – Приклад діаграми Ганта

3 Проектування програмного продукту

3.1 Концептуальне та логічне проектування структур даних програмного продукту

3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів

Використовуючи кроки основного успішного та альтернативного сценаріїв роботи прецедентів ПП, спроектуйте UML-діаграми концептуальних класів.

Моделювання провести з урахуванням наступній послідовності кроків.

1. Визначити імена класів.
2. Визначити імена атрибутів класів.
3. Визначити зв'язку між класами (узагальнення, іменовані асоціації, агреговані асоціації).
4. Для зв'язків-асоціацій визначити назви, кратність і можливість агрегації.
5. Створити UML-діаграму класів в будь-якому графічному редакторі.

3.1.2 Логічне проектування структур даних

Перетворіть UML-діаграму концептуальних класів в опис структур даних з використанням моделі, яка була обрана в концептуальному описі архітектури ПП, наприклад, реляційної моделі даних. Створити графічну модель структур даних в будь-якому графічному редакторі.