НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра системного проектування

До захисту допущено:
Завідувач кафедри

_____ Вадим МУХІН

«___»_____2022 р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою
“Інтелектуальні сервіс-орієнтовані розподілені обчислювання”
зі спеціальності 122 "Комп'ютерні науки"
на тему: «Створення автоматизованної системи формування вимог для IT проекту»

Виконав:
студент IV курсу, групи ДА-82
Вороной Микита Юрійович _____

Керівник:
доцент, к.т.н., с.н.с.
Кисельов Геннадій Дмитрович _____

Консультант з економічного розділу:
доцент, к.е.н.
Рощина Надія Василівна _____
Рецензент:
Завідувач кафедри ММСА
НН «ІПСА» к.т.н, доцент
Тимощук О.Л.

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ 2022

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра системного проектування

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інтелектуальні сервіс-орієнтовані розподілені обчислювання»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Вадим МУХІН
«___»_____2022 р.

ЗАВДАННЯ
на дипломну роботу студенту
Вороному Микиті Юрійовичу

1. Тема роботи «Створення автоматизованної системи формування вимог для ІТ проекту», керівник роботи Кисельов Г.Д. к.т.н., доцент, затверджені наказом по університету від «06» червня 2022 р. №906-с

2. Термін подання студентом роботи 18.06.2022

3. Вихідні дані до роботи: онтологічні моделі знань дисциплін з комп’ютерних наук, згорткові нейронні мережі.

4. Зміст роботи
   1) Огляд сучасних концепцій управління ІТ проектами.
   2) Цілі та задачі, що потрібні для реалізації проекту.
   3) Вимоги до ІТ проекту згідно стандартів SWEBOK,PMBOK.
   4) Розробка концепції проекту: ідеї, можливі ризики, складність, терміни
   5) Існуючі інструментальні засоби автоматизації створення вимог (ТЗ)на проекту.

6) Створення і тестування вимог на програмний проект.
7) Пропозиції зі створення авторської системи автоматизованого створення вимог до програмного проекту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Н. В., к.е.н., доцент		

7. Дата видачі завдання 03.02.2022

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	03.02.2022	
2	Збір інформації	25.02.2022	
3	Ознайомлення з літературою і підготовка теоретичної частини роботи	15.03.2022	
4	Аналіз вимог завдання, вибір методів	15.04.2022	
5	Створення і тестування вимог на програмний проект	20.05.2022	
6	Розробка економічної частини дипломного проекту	26.05.2022	
7	Оформлення дипломної роботи	03.06.2022	
8	Отримання допуску до захисту та подача	12.06.2022	

Студент Вороной Микита

Керівник Геннадій КИСЕЛЬОВ

АНОТАЦІЯ

Процес управління вимогами традиційно вважається одним з ключових при створені автоматизованих систем. Найбільші ризики проектів пов’язані з високою мінливостью вимог та помилками у їх визначенні. Методика, яка заноснова у галузі управління життєвим циклом автоматизованих систем, спрямована на зниження таких ризиків. Проблеми при виконанні проектів створення автоматизованих систем можуть виникати через неформальний збір інфомармації, передебачуваної функціональності, помилкових або нефункціональних вимог до системи,а також нерегламентовану процедуру їх зміни.

Організація управління вимогами насамперед спрямована на спростування таких проблем за рахунок удосконалення способів збору, документування, погодження та модифікації вимог до системи, відстеження вимог від зацікавлених осіб та інших джерел, що їх породжують. Регламентація процедур управління вимогами має забезпечити високу якість робити з ними у проектах, пов’язаних з розробкою, супроводом, створення автоматизованої системи, розробкою їх організаційно – методичного забезпечення, а також впровадження готових систем за рахунок зменшення типових помилок при роботі з вимогами.

Дипломна робота: 81 с., 26 рис., 5 табл., 24 джерела.

Створення автоматизованої системи формування вимог для IT проекту

Метою дипломної роботи є дослідження сучасних моделей представлення знань та методів контролю знань, вибір оптимального методу та огляд систем дистанційного тестування знань.

Проведено огляд основних сучасних концепцій управління

Проведено огляд основних вимог згідно стандарту SWEBOK,PMBO

Проведена розробка програмного проекту для тестування.

Оцінено вартість та варіанти реалізації програмного продукту.

Ключові слова: сучасні концепції управління, стандарт PMBOK,SWEBOK , автоматизація створення вимог.

ABSTRACT

The process of requirements management is traditionally considered one of the key in creating automated systems. The greatest risks of projects are associated with high variability of requirements and errors in their definition. The methodology, which is based on the life cycle management of automated systems, aims to reduce such risks. Problems with the implementation of projects to create automated systems may arise due to informal collection of information, defaulted functionality, erroneous or non-functional requirements for the system, as well as unregulated procedure for their change.

The requirements management organization is primarily aimed at refuting such problems by improving the methods of collecting, documenting, agreeing and modifying system requirements, tracking requirements from stakeholders and other sources that generate them. Regulation of requirements management procedures should ensure high quality to do with them in projects related to the development, maintenance, creation of an automated system, development of their organizational and methodological support, as well as implementation of ready-made systems by reducing common errors in dealing with requirements

The total volume of work is 81 pages, 26 figures, 5 tables, 24 sources.

Keywords: modern management concepts, PMBOK standard, SWEBOK, automation of requirements creation.

ЗМІСТ

ВСТУП

Процес управління вимогами традиційно вважається одним з ключових при створені автоматизованих систем. Найбільші ризики проектів пов’язані з високою мінливостью вимог та помилками у їх визначенні. Методика, яка заноснова у галузі управління життєвим циклом автоматизованих систем, спрямована на зниження таких ризиків. Проблеми при виконанні проектів створення автоматизованих систем можуть виникати через неформальний збір інфомармації, передебачуваної функціональності, помилкових або нефункціональних вимог до системи,а також нерегламентовану процедуру їх зміни.

Організація управління вимогами насамперед спрямована на спростування таких проблем за рахунок удосконалення способів збору, документування, погодження та модифікації вимог до системи, відстеження вимог від зацікавлених осіб та інших джерел, що їх породжують. Регламентація процедур управління вимогами має забезпечити високу якість робити з ними у проектах, пов’язаних з розробкою, супроводом, створення автоматизованої системи, розробкою їх організаційно – методичного забезпечення, а також впровадження готових систем за рахунок зменшення типових помилок при роботі з вимогами.

1. Огляд сучасних концепцій управління проектами

Теоретично можна використовувати будь яку методологію, незалежно від того, яке програмне забезпечення для управління проектами ви використовуєте.

Насправді більшість систем управління завданнями і проектами підходять для декількох різних методологіх. Отже, необхідно зрозуміти, які існують види методологій управління проектами, їх переваги та недоліки, і для яких вони проектів вони найкраще підходять.

Отже розглянемо деякі з найпопулярніших методологій управління проектами.

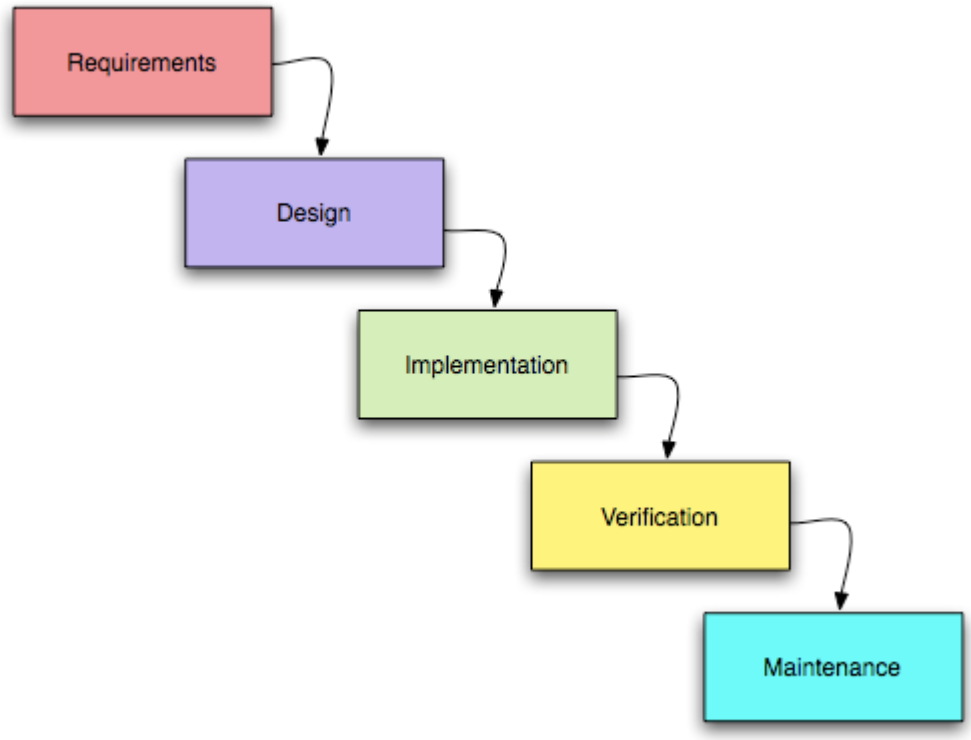
1.1 Каскадна модель WATERFALL(«водоспад)

Методологія Waterfall – найстаріша з усіх. Вперше вона була викладена американським вченим в галузі інформатики Уінстоном Уокером Ройсом у 1970 році у відповідь на потребу управління процесом розробки програмного забезпечення, що дедалі більше ускладнюється. З того часу вона набула широкого поширення, особливо у сфері програмного забезпечення.

Каскадна модель характеризується послідовністю. Крім того, вона значною мірою орієнтована на вимоги.

Коли проект вже буде у розробці, ви не зможете скоригувати його курс. Методологія Waterfall поділяється на три окремі етапи. Спочатку необхідно зібрати та проаналізувати вимоги, потім розробити рішення і підхід, впровадити рішення та виправити проблеми, якщо вони з’явились.

Кожен етап цього процесу є автономним. Щоб перейти до наступного, необхідно завершити попередній етап.



Малюнок 1.1 - Каскадна модель (Waterfall model)

Вищеописане стосується програмного забезпечення. Для швидкого початку планування необхідно скористатись готовим шаблоном діаграми Ганта для розробки програмного забезпечення.

В рамках інших проектів, наприклад, творчих, етапи будуть іншими, але підхід залишається таким самим.

ПЕРЕВАГИ WATERFALL

Витрачаючи час на ранніх стадіях розвитку проекту, створюються умови для своєчасного виконання вимог. Це дозволяє заощадити час та сили на

виправленні недоліків та вирішення проблем надалі. Таким чином є низка переваг:

- **Простота використання –** цю модель просто зрозуміти та використати
- **Структура** – жорсткість методології це і недолік, і явна перевага. Чіткий поділ на етапи дозволяє організувати та розподілити роботу.Оскільки назад повернутись не можна, необхідно ідеально справлятись з виконанням кожного етапу, що дозволяє домогтися кращих результатів
- **Документація**

Недоліки:

- **Підвищенний ризик –** жорсткість методології означає, що якщо ви виявите помилку або знадобиться внести зміни, доведеться починати проекту спочатку.
- **Складність першого етапу**

1.2 Agile

Agile це ще одна методологія управління з акцентом на розробці програмного забезпечення. З'явилася вона як наслідок незастосовності методології Waterfall у межах складних проектів. Хоча ідеї, властиві Agile, вже давно використовуються у сфері розробки програмного забезпечення, формально методологія з'явилася лише у 2001 році, коли кілька представників із IT випустили Agile-маніфест. Agile повністю протилежна методології Waterfall за

підходом та ідеологією. Сама назва з англійської перекладається як «Гнучкий», а це означає, що в управлінні використовується швидкий і гнучкий підхід. Методологія швидше характеризується невеликими циклічними змінами, які впроваджують у відповідь зміну вимог.

Малюнок 1.2 - Agile методологія

ПЕРЕВАГИ AGILE

- **Гнучкість і свобода –** не потрібно чітко позначати етапи і наголошувати на вимогах
- **Знижений ризик**

Недоліки:

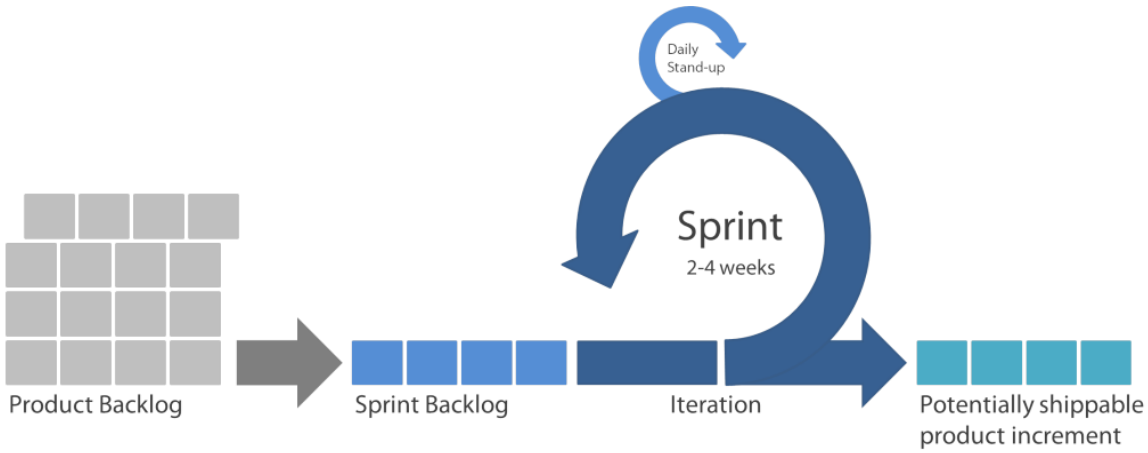
- **Відсутність чіткого плану**
- **Складність взаємодії**

1.3 Scrum

Scrum - це повнофункціональна методологія управління проектами. Це швидше підхід до методології Agile з акцентом на командах проекту, спринтах та щоденних зборах.

Незважаючи на те, що Scrum запозичує принципи та процеси з Agile, цьому підходу властиві свої методи та тактики управління проектами.

У рамках підходу Scrum у центрі проекту – команда. Найчастіше менеджера проекту немає. Тому передбачається, що команда характеризується самоорганізацією та самоврядуванням. Саме тому такий підхід ідеально підійде для досвідчених мотивованих команд, але навряд чи підійде решті.



Малюнок 1.3 - Scrum методологія

ПЕРЕВАГИ AGILE

- **Спрінти –** у підході Scrum акцент робиться на 30-денні спринти, або відрізки часу.
- **Динамічність**
- **Командна робота**

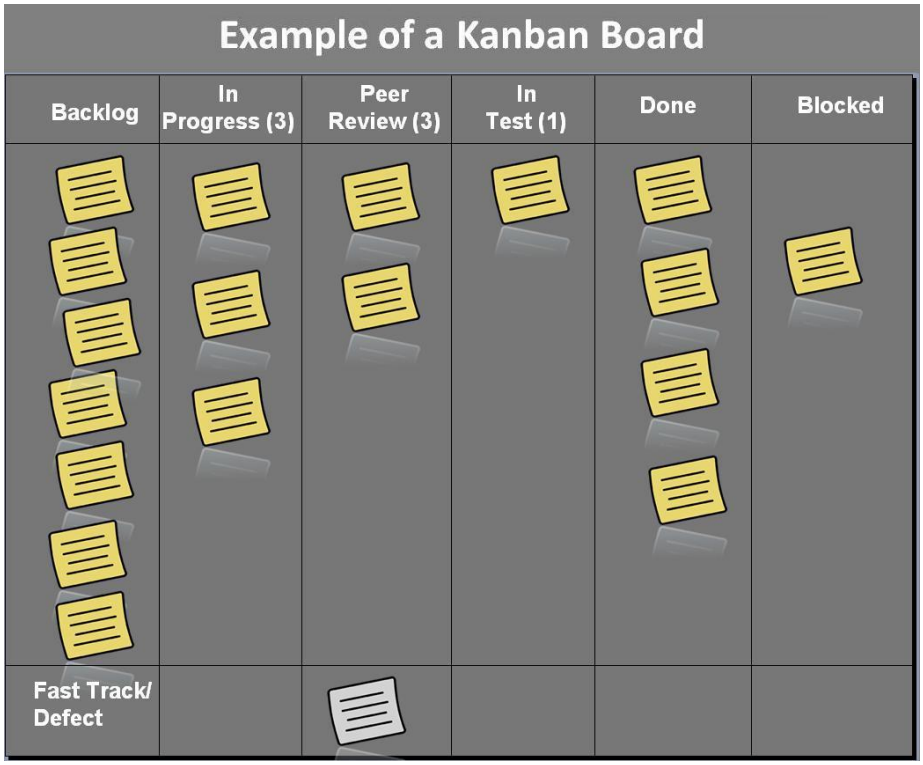
НЕДОЛІКИ:

- **Неконтрольоване розширення масштабів -** оскільки дата завершення проекту не встановлена і відсутня менеджер проекту, який займався б плануванням та бюджетом, Scrum може спричинити неконтрольоване розширення масштабів проекту.
- **Складність взаємодії**
- **Підвищенний ризик -** оскільки команда проекту займається самоорганізацією, збільшується ризик провалу, якщо команда недисциплінована та невмотивована. Якщо у команди недостатньо досвіду, робота в рамках Scrum з великою ймовірністю закінчиться невдачею.
- **Недостатня гнучкість -** акцент на команді проекту означає, що відхід будь-якого ресурсу вплине на результат.

1.4 Kanban

Kanban набагато менш строгий, ніж Scrum - він не обмежує час спринтів, немає ролей, за винятком власника продукту. Kanban навіть дозволяє члену команди вести кілька завдань одночасно, що не дозволяє Scrum. Також ніяк не регламентовані зустрічі щодо статусу проекту – можна робити це як Вам зручно, а можна не робити взагалі.

Для роботи з Kanban необхідно визначити етапи потоку операцій (workflow). У Kanban вони зображуються як стовпці, а завдання позначають спеціальні картки. Картка переміщається по етапах, подібно до деталі на заводі, що переходить від верстата до верстата, і на кожному етапі відсоток завершення стає вищим. На виході ми отримуємо готовий до постачання замовнику елемент продукту. Дошка зі стовпцями та картками може бути як справжньою, так і електронною – навіть тут Kanban не накладає жодних обмежень на користувачів.



Малюнок 1.4 - Канбан методологія

ПЕРЕВАГИ KANBAN

- **Немає встановлених жорстких дедлайнів**
- **При правильному налаштуванні та управлінні, Kanban може принести велику користь команді проекту. Точний розрахунок навантаження на команду, правильне розміщення обмежень і концентрація на постійному поліпшенні - все це дозволяє Kanban серйозно економити ресурси і вкладати в дедлайн і бюджет.**

НЕДОЛІКИ:

Kanban найкраще підходить для команд, навички членів яких перетинаються. Таким чином вони можуть допомагати один одному долати труднощі під час вирішення завдань. Без цього Kanban буде не таким ефективним, як міг би бути. Також, як уже було сказано, Kanban краще підходить у випадках, коли немає жорстких дедлайнів. Для жорстких дедлайн краще підходить Scrum

1.4 Висновки до розділу 1

Отже, зробимо висновки для яких проектів більше за все підійде WATERFALL, AGILE, SCRUM, KANBAN. Методологія управління Waterfall найчастіше використовується у сфері розробки програмного забезпечення. Вона найкраще підходить для невеликих не складних проектів. Проектів із чіткими вимогами. Та проектів, де змінюються ресурси, які залежні від докладної документації. Agile же дозволяє адаптувати проекти до проектів різного типа. Вона буде працювати коли немає впевненості яким потрібен бути фінальний результат. Коли проект потрібно швидко підлаштувати під зміни. Та коли є немає проблем з комунікацією та взаємодією але планування виступає слабкої стороною. Методології Scrum властиві всі переваги та недоліки Agile. Її можна застосовувати для роботи над великими проектами, але вона не підходить командам із багатьма учасниками. Вона більше підійде до досвідчених, дисциплінованих та мотивованих команд.

KANBAN набагато менш строгий,ніж SCRUM - він не обмежує час спринтів, немає ролей,за винятком власника продукту.KANBAN навіть дозволяє члену команди вести кілька завдань одночасно,що не дозволяє SCRUM. Також ніяк не

регламентовані зустрічі щодо статусу проекту – можно робити це як буде зручно,а можна не робити взагалі.

2. Цілі та задачі, результати що потрібні для реалізації проекту

Перший етап будь-якого проекту зводиться до відповіді запитання «що?» і навіщо?". У ключових зацікавлених осіб достатньо впливу та повноважень, щоб від них залежала успішність проекту, і тому їх цілі обов'язково потрібно досягти. Навіть якщо виконати проект вам доручив генеральний директор, вам все одно знадобиться його відношення.

Під час початкових переговорів необхідно порозумітися, сформулювати цілі та визначити цінність проекту. На першому етапі планування потрібно обговорити потреби та очікування та закласти основу для визначення обсягу робіт за проектом, складання кошторису та календарного плану. Всі ці дані стануть надійною платформою для плану проекту.

2.1 Список цілей, OKR та розробка плану

Відсутність чітко поставленої мети стає причиною зриву 37% проектів. Якщо у вас немає чіткої мети, не буде і зв'язку між вимогами, завданнями та термінами,

зазначеними у плані проекту. Але давайте припустимо що на даний момент у вас вже є список потреб основних учасників, ви заручилися їхньою підтримкою, і настав час призначати їм цілі та ключові результати (OKR).

OKR – це методика планування та постановки цілей, яка стала відомою завдяки таким компаніям, як Intel та Google. Ваш проект має бути узгоджений з корпоративними OKR та з OKR вашої команди.

Необхдіно записати цілі проекту на дошці планування проекту та зв'язати їх із вимогами ключових учасників, які обов'язково потрібно задовольнити. Від цієї вихідної точки потрібно починати вибудовувати структуру, віхи та завдання, необхідні для досягнення цілей. Віхи можуть виконувати роль контрольних точок протягом усього проекту та допомагати учасникам стежити за ходом виконання робіт, оцінювати прогрес та пам'ятати про очікування.

2.2 Розробка документації

Ми склали попередній план, погодили завдання та цілі та заручилися підтримкою команди. Тепер настав час скласти опис обсягу робіт за проектом та детально задокументувати всі елементи проекту, перелічені у пункті попередньому пункті.

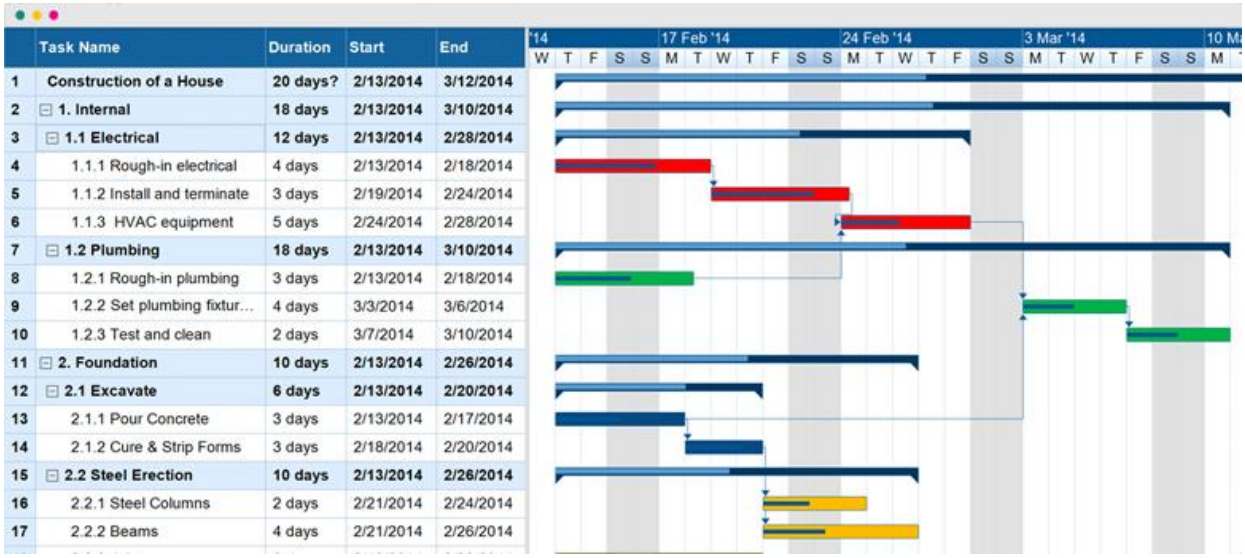
Потрібно поглянути на кожен із звітних матеріалів та визначити серію завдань, які мають бути виконані для створення даного матеріалу. Для кожного завдання вказується тривалість її виконання, необхідні ресурси та відповідального виконавця. У процесі потрібно допрацювати та записати усі відомості про проект, щоб співробітники змогли користуватися єдиним джерелом достовірної інформації. Також потрібно забезпечити вільний доступ

до цього документа, наприклад, розмістивши його в системі управління проектами, щоб запобігти можливим непорозумінням, які ведуть до зайвих витрат.

Хоча підготовка документації про обсяг робіт за проектом має бути стандартною процедурою, кожен четвертий менеджер проекту, який взяв участь в опитуванні управління проектами Wellingstone State, зізнався, що «ніколи» не займався підготовкою таких документів або становив їх «зрідка». Створення такої документації надасть перевагу та допоможе всім учасникам проекту бути в курсі подій.

2.3 Календарний план проекту

Коли цілі, завдання та віхи вже намічені, настав час приступити до складання календарного плану. Діаграма Ганта – зручний інструмент, що дозволяє уявити тимчасову шкалу проекту у наочному вигляді. Ця інтерактивна тимчасова шкала дає повне уявлення про хід виконання проекту, обсяг робіт та залежності.



Малюнок 2.1 – Шаблон діаграми Ганта

Залежності вказують, які завдання необхідно виконати до початку інших завдань. При плануванні термінів використовуються підзавдання, щоб розбивати завдання на більш зручні елементи. Це спрощує подальше складання звітів та управління завантаженістю команди. Давайте дамо визначення:

- Задачі — це окремі операції, які потрібно виконати для досягнення цілей.
- Підзадачі з тривалістю не більше кількох днів дозволяють розбити завдання на дрібніші складові етапи.
- Віхи — великі етапи або події в ході проекту, які допомагають розбити проект. Віхи можна використовувати як контрольні точки під час роботи над проектом.

2.4 Визначення ролей, обов'язків та ресурсів

Під ресурсами зазвичай маються на увазі персонал, обладнання або гроші, необхідні для виконання проекту. Вибравши інструменти і отримавши фінансування, непотрібно забувати про потрібних фахівців. Навіть люди, які вміють складати робочий план проекту та робили це дуже багато разів, можуть недооцінити свої потреби у робочій силі.

Схема розподілу відповідальності допоможе визначити, хто виконуватиме ту чи іншу роботу за проектом. Це таблиця з перерахуванням всіх завдань за проектом та зазначенням відповідальних виконавців (призначених для виконання роботи), підзвітних (з правом голосу та правом накласти вето), консультантів (що беруть участь у погодженні або обговоренні робіт) та співробітників, які ставляться до відома (мають знати про виконаній дії або прийнятому рішенні).

RACI Chart (Roles and Responsibilities Matrix)

For instructions / training material visit http://www.racichart.org

Process Name / Description:	*Plant maintenance project: Repair and resurface plant parking lot during plant shutdown in July*
Created On:	*1-Jan-16* Revision: *3/12/16*
Created by:	*Kelly Bradley (facilities mgr), Mike Cole (plant manager), Joe Pallino (HR), Brian Sullivan (security), Billy Owens (project manager)*

	Facilities	Plant Mgr	HR	Security	Project Mgr
Identify a minimum of three asphalt contractors from Angie's List	C	-	-	-	R
Arrange for contractor visits and quotes	I	-	-	-	R
Review quotes and references, make contractor selection	A	I	I	-	R
Review and finalize contract, lock in plant shutdown week	I	I	-	-	R
Communicate project to shutdown maintenance crew, make sure all vehicles are removed from the lot	I	I	R	I	I
Provide security gate access codes for asphalt crew by June 15	I	-	A	R	I
Oversee the project during the plant shutdown week, ensure it is completed on time	A	I	I	-	R

R = Responsible, A = Accountable, C = Consulted, I = Informed

Малюнок 2.2 - Матриця RACI

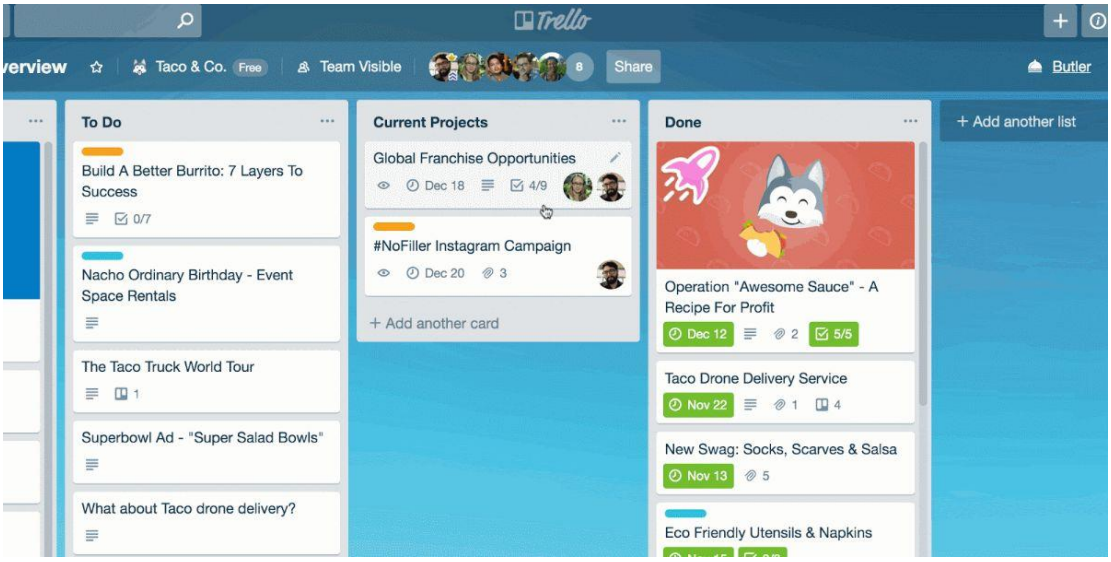
Коли почнуть призначати завдання, обов'язково потрібно зважати на можливості співробітників. Чітко визначити коло обов'язків та очікування, які покладаються на кожного з них. Також потрібно не забувати, що 95% працівників працюють із більш ніж однією командою або над кількома проектами. Якщо ці проекти не узгоджені, робоче завантаження стає надмірним. Стрес — саме та причина, через яку 50% працівників починають шукати іншу роботу, а 25% звільняються.

У ході планування проекту потрібно продумати, як будуть фільтрувати вхідні запити, що впливають на терміни та кошторис проекту. Такі інструменти, як Wrike Resource, допомагають менеджерам проектів представити у наочному вигляді завдання щодо проекту з погляду робочих процесів команди, дозволяючи наочно та гнучко збалансувати робоче завантаження.

2.5 Визначити процеси взаємодії та перевірки

За даними дослідження McKinsey, за тиждень співробітники витрачають близько 20% робочого часу на збирання та пошук потрібної інформації. До того ж, неефективне спілкування та співпраця — це дві головні причини стресу на робочому місці. Коли учасникам доводиться ритися в купі електронних повідомлень або постійно ставити запитання, щоб дізнатися про останні новини, від їх мотивації не залишається і сліду.

Можна знизити рівень стресу, якщо зберігатимуть всю інформацію щодо проекту — матеріали, обговорення, завдання, терміни, новини, звіти — централізовано, наприклад, у вирішенні спільної роботи. Так буде простіше стежити за перебігом робіт, обмінюватися новинами та вносити виправлення. Встановлення правил для спілкування під час роботи над проектом та зберігання всіх даних в одному програмному інструменті, щоб у всіх учасників був доступ до інформації.



Малюнок 2.3 – Приклад управління процесами у Trello

2.6 План випадок якщо щось піде не за планом

Навіть якщо ви кваліфікований спеціаліст і вмієте складати плани проектів, правда в тому, що кожен з проектів може зробити несподіваний сюрприз, і саме це робить їх цікавими. Допустимо, був передбаченний буферний час при складанні календарного плану, ви переконалися, що всі учасники знають свою роль і впровадили правила обміну інформацією.

Але перед запуском все ж таки варто пошукати можливі джерела проблем, такі як відпустка когось із співробітників, свята або залучення сторонніх фахівців. Встановлення чіткого ланцюжку команд та складання списку ключових контактних осіб. Заздалегідь потрібно обговорити із командою можливі ризики, щоб співробітники були готові до можливих форс-мажорів.

Висновки до розділу 2

У ході кожного успішного проекту обов'язково проводиться нарада. Організація зустріч із ключовими учасниками та розробка чіткого порядку денного. Головна мета — донести до всіх інформацію про цілі, ролі, процеси та терміни. До порядку денного має бути включена вся інформація, про яку було проговорено вище:

- Визначення цілі проекту та користь, які він принесе.
- Складання списку матеріалів, створених під час проекту.
- Складання схеми зв'язків між вимогами учасників та завданнями проекту.
- Продемонструвати тимчасову шкалу проекту (діаграму Ганта), щоб усі могли бачити залежні завдання та очікувані дати.

- Опис ролей та обов'язків всіх учасників.
- Пояснити, як здійснюватиметься взаємодія в ході проекту, де шукатиме потрібну інформацію, наприклад, документ про обсяг робіт, і до кого звертатися з питаннями.

Обговоріть ризики та переконайтеся, що команда готова до них

3. Вимоги до IT проекту згідно стандарту SWEBOK,PMBOK

У сфері управління IT-проектами існує низка міжнародних стандартів. Це такі, як PMBoK, SWEBoK та ряд інших.

Project Management Body of Knowledge (PMBOK) – Стандарт управління проектами PMBOK Guide 3-rd Edition (Project Management Body of Knowledge), Американського інституту управління проектами (PMI – Project Management Institute) визначає коло знань, необхідних для ефективного управління проектами. Стандарт PMBOK Guide 3-rd Edition включають процеси, що охоплюють всі стадії життєвого циклу проекту (ініціація, планування, виконання, контроль і завершення). Результати чи виходи одного процесу є входами іншого процесу.

3.1 PMBOK

Звід знань з управління проектами є сумою професійних знань з управління проектами. Керівництво PMBOK фіксує частини Зводу з управління проектами, що зазвичай вважається гарною практикою. PMI використовує цей документ як основний довідковий матеріал для своїх програм з професійного розвитку. Є американським національним стандартом.

У цьому стандарті описуються суть процесів управління проектами в термінах інтеграції між процесами та взаємодій між ними, а також цілі, яким вони служать. Ці процеси розділені на п'ять груп, званих "групи процесів управління проектом".

Стандарт PMBOK Guide 3-rd Edition включають процеси, що охоплюють всі стадії життєвого циклу проекту (ініціація, планування, виконання, контроль і завершення). Результати чи виходи одного процесу є входами іншого процесу.

3.2 Опис методологій PMBoK

Процеси областей знань представлені в PMBOK у вигляді дискретних елементів, які мають чітко визначені межі. На практиці ці процеси є ітеративними — можуть взаємодіяти між собою і накладатися один на одного. PMBOK взаємодії не описує

Малюнок 3.1 - 10 областей знань проектного управління

Отже,стандарт розглядає ось такі галузі знань управління проектам:

- **Управління інтеграцією проекту(Project Integration Management**
- **Управління Вмістом проекту (Project Scope Management)**
- **Управління термінами проекту (Project Time Management)**
- **Управління вартістю проекту (Project Cost Management)**
- **Управління якістю проекту (Project Quality Management)**
- **Тощо.**

3.3 Групи процесів PMBoK

1. **Група процесів ініціації**

Група процесів ініціації складається з процесів, що сприяють формальному ініціювання нового проекту.

- Розробка Статуту проекту (Develop Project Charter)
- Визначення зацікавлених сторін (Identity Stakeholders)

2. **Група процесів планування**

Уточнення і планування дій, необхідних для досягнення цілей і змісту, заради яких було розпочато проект.

3. **Група процесів виконання**

- Об’єднує людські та інші ресурси для виконання плану управління проектом даного проекту. До групи процесів виконання входять наступні процеси: Керівництво та управління виконанням проекту (Direct and Manage Project Execution),забезпечення якості (Perform Quality Assurance),набір команди проекту (Acquire Project Team)

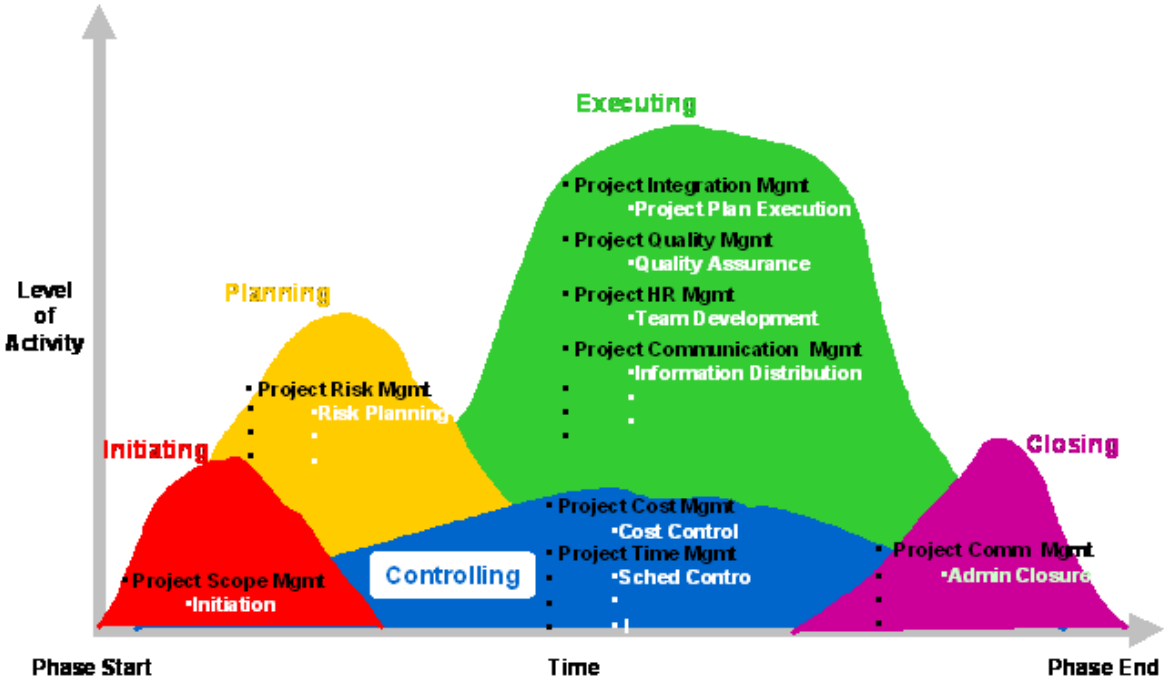
4. **Група процесів ініціації**

Охоплює оцінку і моніторинг прогресу проекту, щоб виявити можливі відхилення від плану управління проектом

5. **Група завершальних процесів**

Формалізує прийомку товару, послуги або результату і підводить проект чи фазу проекту до завершення

Уточнення і планування дій, необхідних для досягнення цілей і змісту, заради яких було розпочато проект.



Малюнок 3.2 – Групи процесів по PMBoK

3.4 SWEBOK

Важливо розуміти, що програмна інженерія є дисципліною, що розвивається. Більш того, дана дисципліна не стосується питань конкретизації застосування тих чи інших мов програмування, архітектурних рішень або, тим більше, рекомендацій, що стосуються більш-менш розповсюджених чи тих, що розвиваються, з тим чи іншим ступенем активності/помітності технологій (наприклад, web-служб).

Керівництво до зводу знань, а таким є SWEBOK, включає базове визначення й опис галузей знань (наприклад, конфігураційне управління – configuration management) і, безумовно, є недостатнім для охоплення всіх питань, що відносяться до питань створення програмного забезпечення, але, у той же час потрібним для їхнього розуміння.

Необхідно відзначити, що однією з найважливіших цілей SWEBOK є саме визначення тих аспектів діяльності, що складають суть професії інженера-програміста.

3.5 Структура і зміст SWEBOK

Опис галузей знань у SWEBOK побудовано за ієрархічним принципом, як результат структурної декомпозиції. Така ієрархічна побудова звичайно нараховує два-три рівня деталізації, прийнятих для ідентифікації тих чи інших загальновизнаних аспектів програмної інженерії. При цьому, структура декомпозиції галузей знань деталізована тільки до того рівня, що потрібен для розуміння природи відповідних тем і можливості пошуку джерел компетенції й інших довідкових даних і матеріалів. У принципі, вважається, що як такий “звід знань ” з програмної інженерії представлений не в обговорюваному керівництві (SWEBOK),а в першоджерелах (як зазначених у ньому, так і представлених за його рамками)

SWEBOK описує 10 галузей знань:

1. Software requirements – програмні вимоги

2. Software design – дизайн (архітектура)

3. Software construction – конструювання програмного забезпечення

4. Software testing - тестування

5. Software maintenance – експлуатація (підтримка)програмного забезпечення

6. Software configuration management – конфігураційне управління

7. Software engineering management – управління в програмній інженерії

8. Software engineering process – процеси програмної інженерії

9. Software engineering tools and methods – інструменти і методи

10. Software quality – якість програмного забезпечення.

На додаток до них SWEBOK також включає огляд суміжних дисциплін, зв'язок з якими представлений як фундаментальний, важливий та обгрунтований для програмної інженерії:

1. Computer engineering

2. Computer science

3. Management

4. Mathematics

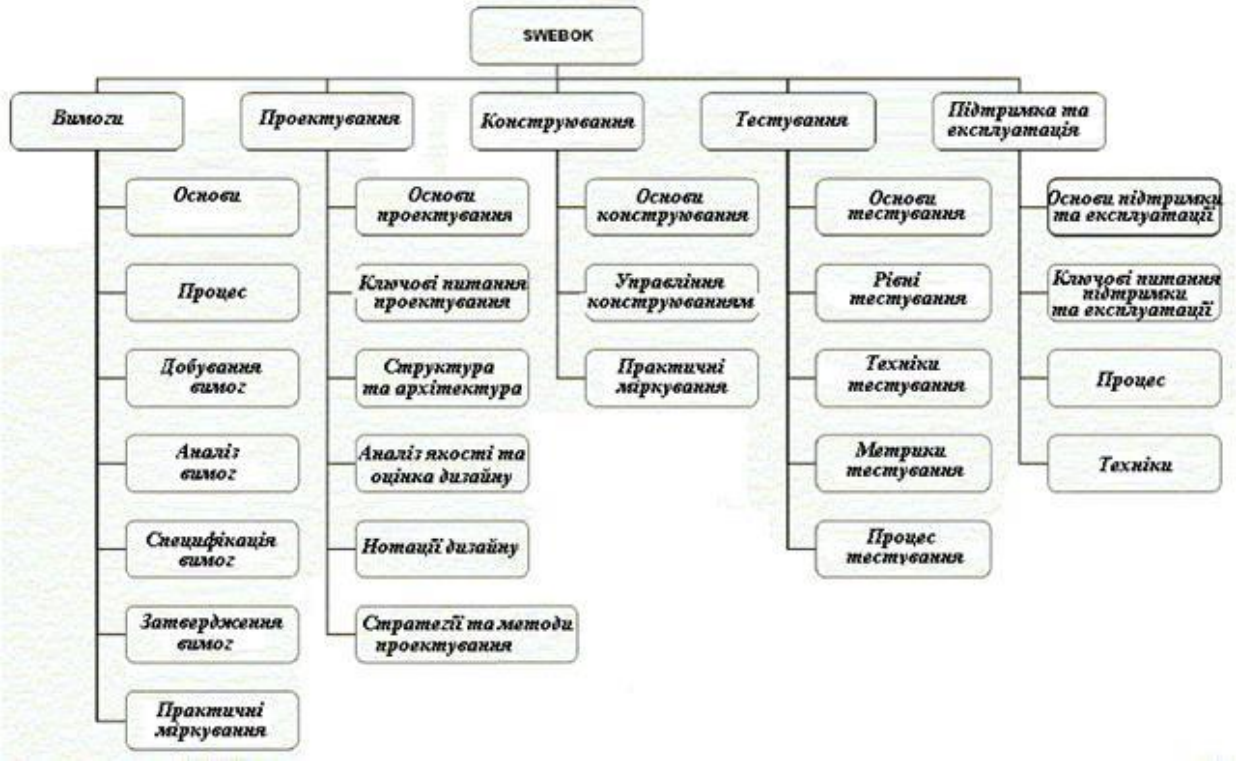
5. Project management

6. Quality management

7. Software ergonimics

8. Systems engineering

Варто відзначити, що прийняті розмежування між галузями знань, їх компонентами (subareas) і іншими елементами досить довільні. При цьому, на відміну від PMBOK, галузі знань SWEBOK не включають “входи ” і “виходи ”. Деякою мірою така декомпозиція пов'язані з тим, що SWEBOK не асоційовано з тією чи іншою моделлю (наприклад, життєвого циклу) чи методом. Хоча на перший погляд перші п'ять галузей знань у SWEBOK представлені в традиційній послідовній (каскадній -waterfall) моделі, це не більш ніж слідування прийнятій послідовності висвітлення відповідних тем. Інші галузі і структура декомпозиції галузей представлені за абеткою.



Малюнок 3.3 – Перші п’ять галузей знань

3.6 Інженерія вимог

Вимоги до ПЗ - сукупність властивостей, які повинно мати ПЗ. Призначені для адекватного визначення функцій, умов і обмежень виконання ПЗ, а також обсягів даних, технічного забезпечення і середовища його виконання.

Вимоги відбивають потреби людей (замовників, користувачів, розробників), зацікавлених у створенні ПЗ. Замовник і розробник спільно виявляють вимоги, аналізують, переглядають, визначають необхідні обмеження і умови, а також описують їх. Розрізняють вимоги до продукту і до процесу, а також функціональні, не функціональні і системнівимоги. Вимоги до продукту і до процесу визначають умови виконання і режими роботи ПЗ в операційному середовищі, обмеження на структуру і пам'ять комп'ютерів та принципи взаємодії програм.

Функціональні вимоги визначають призначення і функції системи, а не функціональні - умови стосовно виконання ПЗ, його переносності і доступу до даних. Системні вимоги описують вимоги до програмної системи, яка складається з взаємозалежних програмних і апаратних підсистем і різних застосувань. Вимоги можуть бути кількісні (наприклад, кількість оброблених запитів на секунду, середній показник помилок і т.п.). Значна частина вимог стосується атрибутів якості: безвідмовність, надійність і ін., а також захисту і безпеки як ПЗ, так і даних.

Область знань «Вимоги до ПЗ (Software Requirements))) складається *у* таких розділів:

- інженерія вимог (Requirement Engineering),

- виявлення вимог (Requirement Elicitation),

- аналіз вимог (Requirement Analysis), специфікація вимог (Requirement Specification).

- валідація вимог (Requirement validation),

- керування вимогами (Requirement Management).

Інженерія вимог до ПЗ- це дисципліна аналізу і документування вимог до ПЗ, що полягає в перетворенні запропонованих замовником вимог до системи на опис вимог до ПЗ і їх валідації. Інженерія базується на моделі процесу визначення вимог і діяльності осіб, що забезпечують керування і формування вимог, а також на методах досягнення показників якості.

Модель процесу визначення вимог - це схема процесів ЖЦ, що виконуються від початку проекту і доти, поки не будуть визначені і погоджені вимоги. Таким процесом може бути маркетинг і перевірка виконання вимог у даному проекті.

Керування вимогами до ПЗ полягає в контролі за виконанням вимог і плануванні використання ресурсів (людських, програмних, технічних, часових, вартісних) у процесі розроблення проміжних робочих продуктів на етапах ЖЦ і продукту в цілому.

Якість і процес поліпшення вимог - це процес формулювання характеристик і атрибутів якості (надійність, реактивність і ін.), які повинно

мати ПЗ, методи їх досягнення на етапах ЖЦ і оцінювання отриманих результатів.

Виявлення вимог- це процес витягування інформації з різних джерел (договорів, матеріалів аналітиків з декомпозиції задач і функцій системи й ін.), проведення технічних заходів (співбесід, збирання пропозицій і ін.) для формування окремих вимог до продукту і до процесу розроблення. Вимоги погоджуються з замовником.

Аналіз вимог- процес вивчення потреб і цілей користувачів, класифікація і перетворення їх на вимоги до системи, апаратури і ПЗ, встановлення і вирішення конфліктів між вимогами, визначення пріоритетів, меж системи і принципів взаємодії із середовищем функціонування.

Специфікація вимог до ПЗ- процес формалізованого опису функціональних і не функціональних вимог, вимог до характеристик якості відповідно до стандарту якості ISO/IEC 9126, які будуть відпрацьовуватися на етапах ЖЦ ПЗ. У специфікації вимог відбивається структура ПЗ, вимоги до функцій, якості і документа-функцій, якості і документації, а також задається архітектура системи і ПЗ, алгоритми, логіка керування і структура даних. Специфікуються також системні вимоги, не функціональні вимоги і вимоги до взаємодії з іншими компонентами і платформами (БД, СКБД, маршаллінг даних, мережа й ін.).

Валідація вимог- це перевірка викладених у специфікації вимог, що виконується для того, щоб шляхом відстеження джерел вимог переконатися, що вони визначають саме дану систему. Замовник і розробник ПЗ проводять експертизу сформованого варіанта вимог для того, щоб розробник міг далі продовжувати проектування ПЗ. Один з методів валідації - прототипування, тобто швидке відпрацьовування окремих вимог на конкретному інструменті і дослідження масштабів зміни вимог, вимірювання обсягу функціональності і вартості, а також створення моделей оцінки зрілості вимог.

Верифікація вимог- це процес перевірки правильності специфікацій вимог щодо їх відповідності потребам, несуперечності, повноти і можливості реалізації, а також узгодженості зі стандартами. Як результат перевірки вимог складається погоджений вихідний документ, що встановлює повноту і коректність вимог до ПЗ, а також можливість продовження проектування ПЗ.

Керування вимогами- це керування процесами формування вимог на всіх етапах ЖЦ, а також змінами й атрибутами вимог, проведення моніторингу - відновлення джерела вимог. Керування змінами виникає після того, ПЗ починає працювати в заданому середовищі і виявляє помилки щодо трактування вимог, невиконання деякої окремої вимоги тощо. Невід'ємною складовою процесу керування є ***трасування вимог*** для відстеження правильності встановлення і реалізації вимог до системи і ПЗ на етапах ЖЦ, а також зворотний процес відстеження в отриманому продукті реалізованих вимог. Для уточнення деяких вимог або додавання нової вимоги складається план зміни вимог, що погоджується з замовником. Внесені зміни спричиняють і зміни в створеному продукті або в окремих його компонентах.

Висновки до розділу 3

Project Management Body of Knowledge (PMBOK) – Стандарт управління проектами PMBOK Guide 3-rd Edition (Project Management Body of Knowledge), Американського інституту управління проектами (PMI – Project Management Institute) визначає коло знань, необхідних для ефективного управління проектами. Стандарт PMBOK Guide 3-rd Edition включають процеси, що охоплюють всі стадії життєвого циклу проекту (ініціація, планування, виконання, контроль і завершення). Результати чи виходи одного процесу є входами іншого процесу.

Варто зазначити, що прийняті розмежування між областями знань, їх компонентами (subareas) та іншими елементами є довільними. Певною мірою така декомпозиція пов'язана з тим, що SWEBOK не асоційований з тією чи іншою моделлю (наприклад, життєвим циклом) або методом. Хоча на перший погляд перші п'ять областей знань у SWEBOK представлені в традиційній послідовній (каскадній – waterfall) моделі, це не більше ніж слідування прийнятої послідовності висвітлення відповідних тем. Інші області та структура декомпозиції областей представлені в алфавітному порядку.

4. Розробка концепції проекту: ідеї, можливі ризики, складність, терміни

Однією з основних ознак системи, що відрізняється від розрізнених компонентів, є підпорядкованість всієї організації системи — певної мети. Проектна робота команди, є теж якоюсь системою і відповідно повинна «йти на поводу» в якійсь меті. Тому, встановивши комунікації між учасниками проекту, почнемо визначати цілі, яких ми хочемо досягти внаслідок створення нового продукту. Мета цієї групи робіт: визначити основні завдання, які ставлять собі групи зацікавлених осіб, що у проекті.

4.1 Формування ідеї

З погляду керівника проекту правильно сформульовані цілі мають дозволити:

1. Ідентифікувати продукт, який команда має виробити в ході реалізації проекту, та блага, які замовник має отримати у результаті його завершення;
2. Визначити критерії успішності виконання проекту загалом, з погляду основних зацікавлених осіб, зокрема як замовника, а й команди проекту.

Тому мета аналітика на даному етапі – забезпечити команду та замовника саме цією інформацією.

Визначаючи цілі, важливо враховувати два важливі моменти:

1. Замовник найчастіше неспроможна самостійно якісно сформулювати цілі;

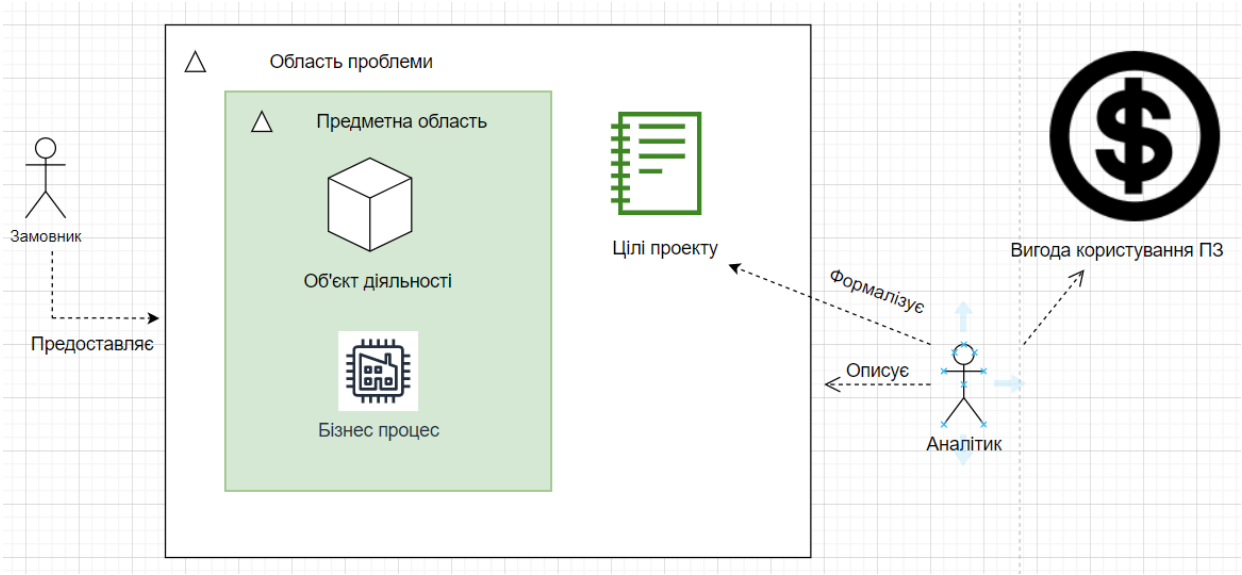
2. Цілі в проекті у різних представників замовника, підрядників та виконавців різні, а іноді й суперечать один одному.

Тому, по-перше: Ми повинні допомогти представникам замовника сформулювати цілі, яких вони хочуть досягти в результаті впровадження цільового продукту. По-друге: при формулюванні цілей замовника повинні враховувати і свої власні стратегічні цілі розвитку, які хочемо досягти в результаті виконання робіт. Наприклад, це може бути спроба закласти в функціональні можливості системи, що розробляється - характеристики, не суттєві для замовника, але що представляють інтерес для виведення на ринок, розробленого цільового продукту.

Окремо загострю увагу на тому, що при формулюванні цілей проекту, треба постаратися закласти в них свої власні (команди реалізації) стратегічні інтереси, які Ви хочете досягти в результаті своєї діяльності в проекті.

А ось при визначенні цілей замовника, спільно з його представниками, аналітик повинен насамперед оперувати поняттям вигоди, яку отримає власник від використання кожної цільової системи, що розглядається. Тому коли замовник просить реалізувати Вас якусь можливість, привчіть його відразу замислитися, а який профіт він з цього отримає. Так Ви можете ще на початковому етапі відсунути частину зайвих вимог і заощадити час на стадії визначення меж проекту, але трохи пізніше.

На малюнку 1 зображено процес формування цілей створення інформаційної системи.



Малюнок 4.1 – модель процесу формування цілей

Відкинувши загальні цілі процесу розробки вимог, спробуємо виділити головні завдання, які має вирішити навчальна система «Управління вимогами», що проектується нами, для задоволення існуючих потреб замовника. Оскільки в навчальному проекті я виконую і функцію аналітика, і імітую діяльність замовника, то наведу наші спільні роздуми щодо формулювання кожної мети.

Перш за все, слід звернутися до доступних джерел, що описують проблеми та вирішення предметної області, що автоматизується. Відповідно до SWEBOK (Software Engineering Body of Knowledge), аналіз вимог Requirement Process складається з таких основних складових:

- Requirements Elicitation (Вилучення вимог),
- Requirements Analysis (Аналіз вимог у вузькому значенні),
- Requirements Specification (Специфікація вимог),
- Requirements Validation (Перевірка вимог).

Таким чином, ми маємо охопити весь цей потік робіт. І виконувати їх бажано, як ми з'ясували у попередньому розділі, дотримуючись єдиного стилю, використовуючи єдине «довкілля» для всіх процесів та учасників проекту.

4.2 Розробка концепції

У кожної свідомої людини в світі розпорядок дня займає велику частину його життя. Потрібно виділяти час для праці, та деякий час для відпочинку. Як провести свій відпочинок вирішує кожен сам для себе, але велику частку у відпочинку займає перегляд фільмів. Але який фільм обрати? На це потрібен час. А ще ж хочеться щоб був свій список переглянутих або «watch later» фільмів. Тож, як приклад, візьмемо користувацький сервіс для вибору та перегляду кінофільмів та серіалів, а також з можливістю створення свого списку кінофільмів та серіалів.

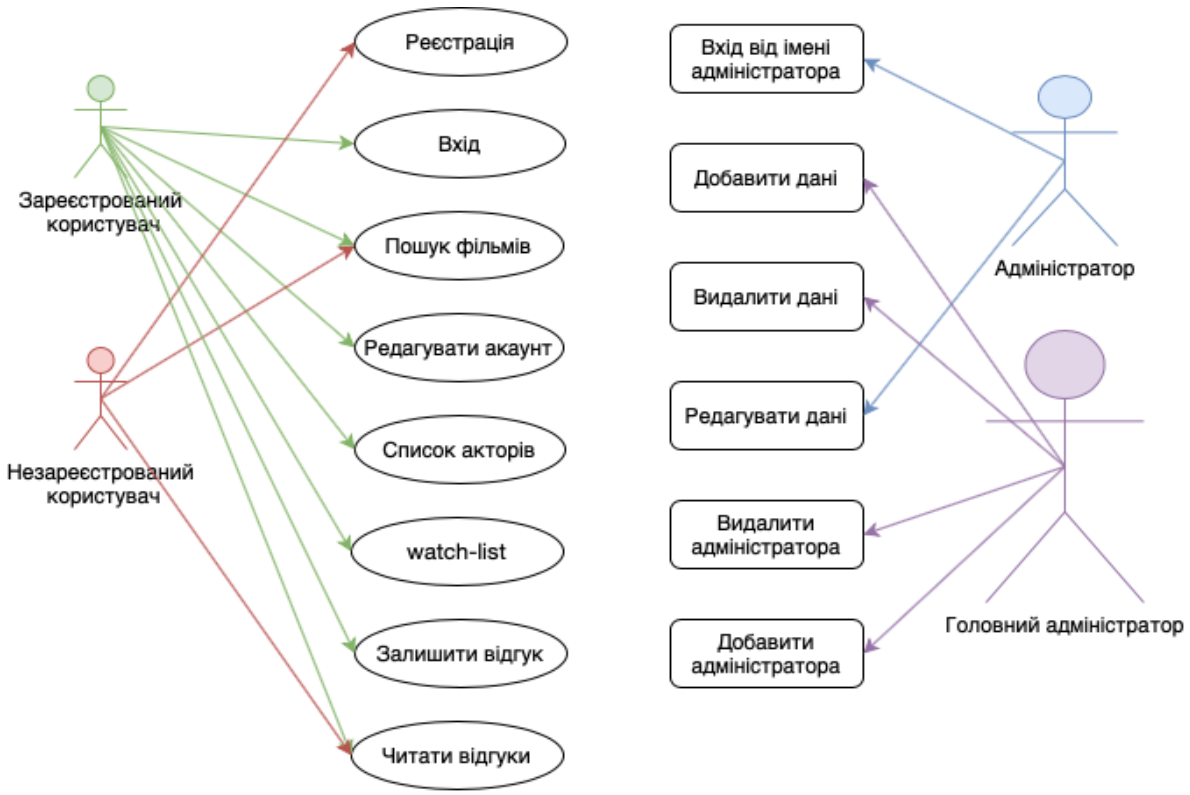
Опис проекту:

1. Користувач зможе обирати фільми з запропонованих йому тем та жанрів, також можна продивитись опис до фільму, його рейтинг, а також відгуки.
2. Користувач зможе зареєструватися та створити свій список фільмів, котрі хоче подивитися або вже подивився. Також буде можливість зробити свій список приватним або публічним
3. Після реєстрації буде доступна функція додавати рекомендації(будь-який користувач може побачити, що ви рекомендуєте до перегляду) та залишити свій відгук.

Актуальність проекту:

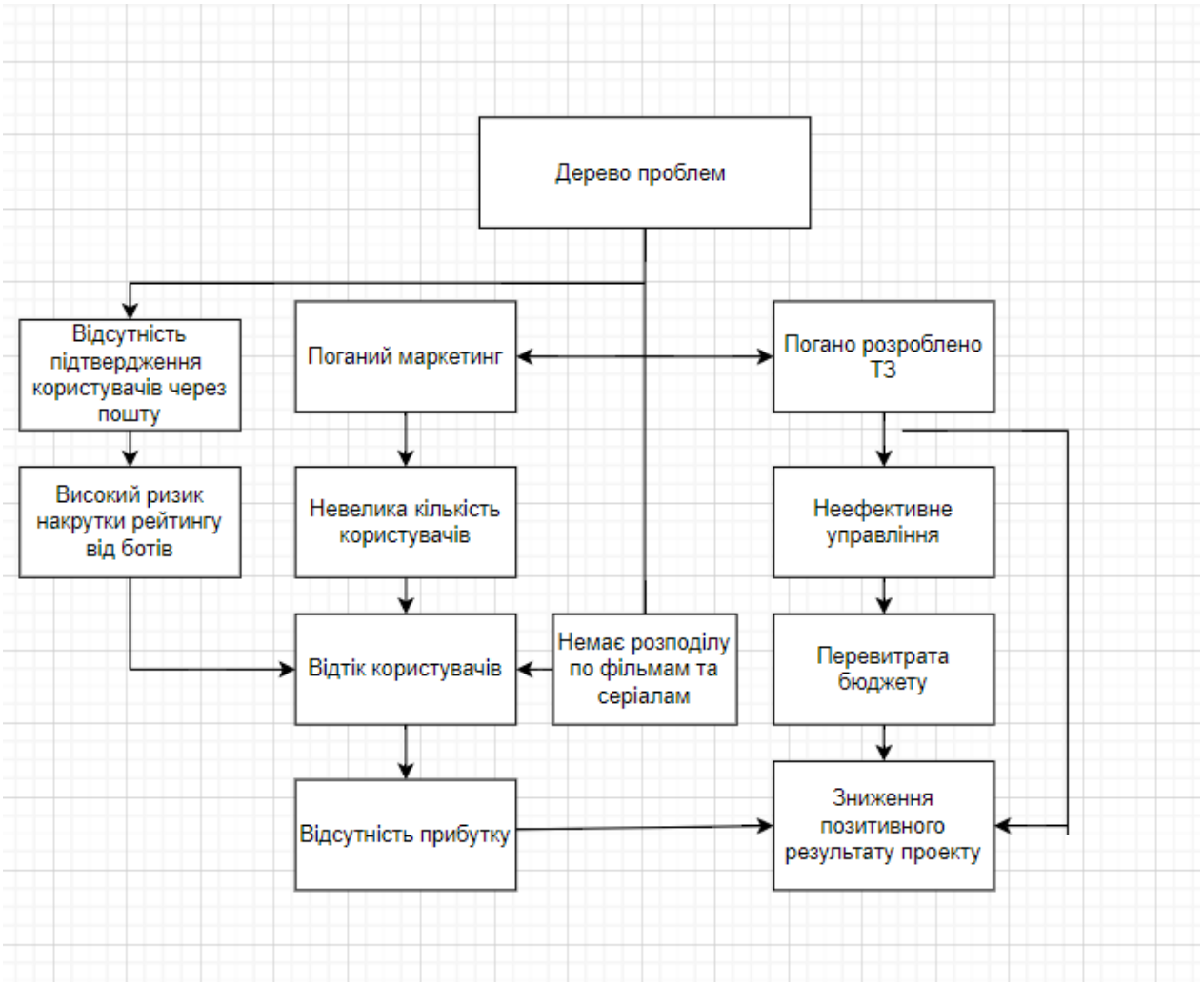
- Існують такі аналоги як IMDb та «Кинопоиск» (але у 2017 році за рішенням президента цей та багато інших сайтів стали недоступні для користувачів з України).Тому цей аналог буде актуальний для користувачів з України.
- Як було сказано раніше: у сучасному світі у людини є актуальна проблема економії часу, а пошук цікавого фільму буде займати певен час, тому цей сайт призначений виправити цю проблему.

Use Case diagram



Малюнок 4.2 – Use case diagram проекту.

Дерево проблем

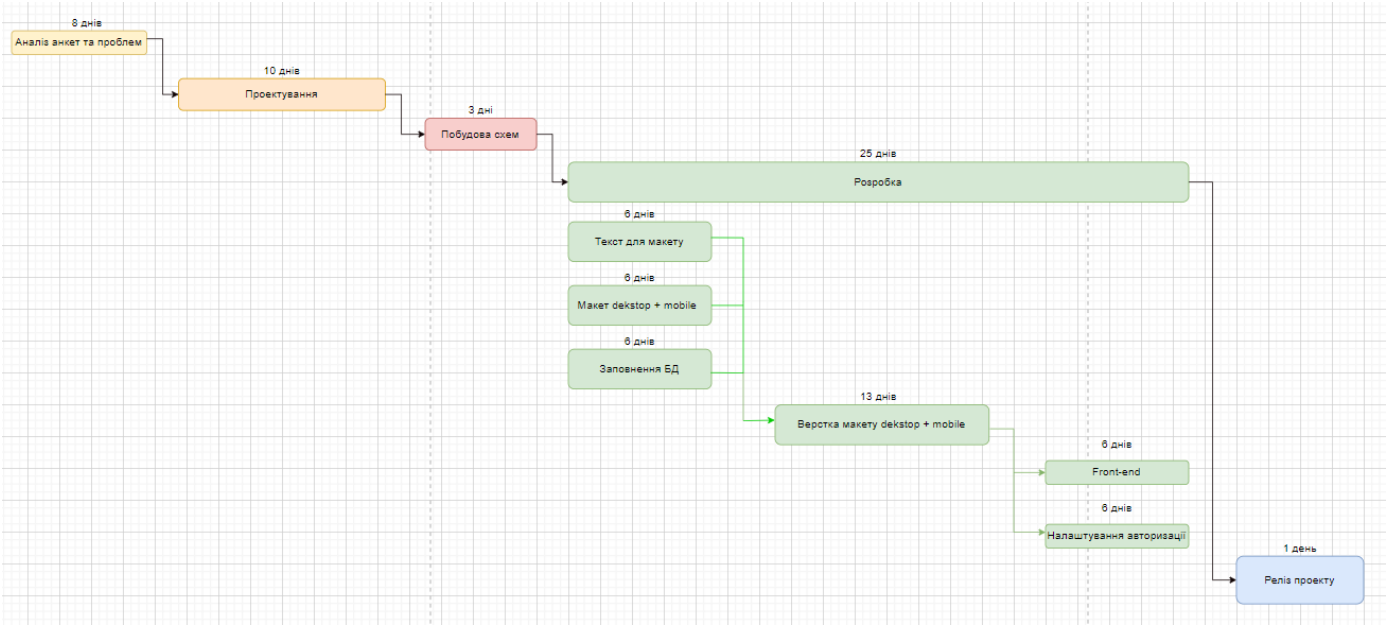


Малюнок 4.3 – Дерево проблем проекту

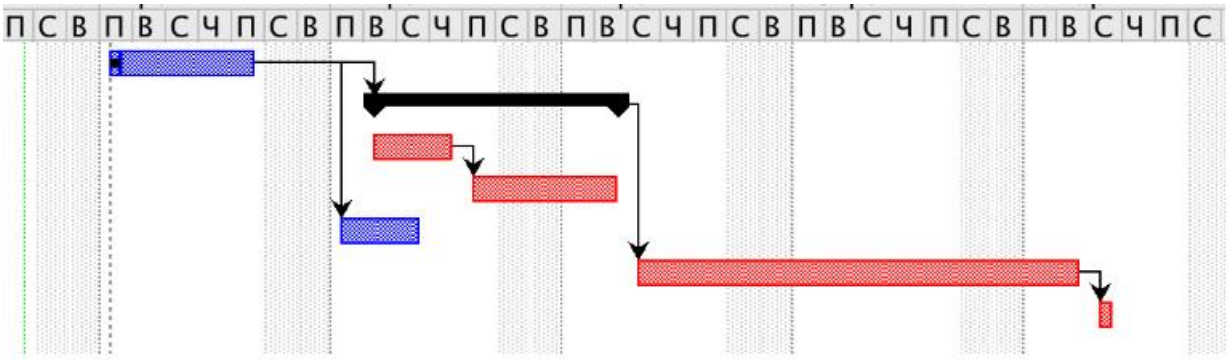
Терміни виконання проекту

Таблиця 4.1 – Таблиця дедлайнів проекту

Задача		***Час***
Аналіз анкет та проблем		02.03.2022-10.03.2022
Проектування		10.03.2022-20.03.2022
Побудова схем		20.03.2022-23.03.2022
Розробка	Текст для макету	23.03.2022-01.04.2022
	Макет desktop+mobile	
	Заповлення БД	
	Верстка макету desktop+mobile	02.04.2022-15.04.2022
	Front-end	15.04.2022-21.04.2022
	Налаштування авторизації	
Реліз проекту		22.04.2022



Малюнок 4.4 – Графічна ієрархічна структура роботи



Малюнок 4.5 - Графічна ієрархічна структура роботи в ProjectLibre.

Висновки до розділу 4

В результаті був придумана концепція веб-сайт для перегляду інформації про актуальні фільми.

Було створена ідея пошукового веб-сайт, що має мінімалістичний та простий дизайн, можливість реєстрації користувача задля створення власних фільм-листів та для автоматичних рекомендацій фільмів на основі фільм-листів користувача, а також можливість дізнатися більше інформації про фільм, акторів, режисерів, поставити рейтинг фільму і написати власний коментар щодо нього, що допоможе наступним користувачам легче та швидше обрати фільм для перегляду.

5. Створення і тестування вимог на програмний проект

Тестування є невід'ємною частиною життєвого циклу програмного забезпечення. Саме собою тестування – тривалий процес перевірок на відповідність очікуваного результату. Не можна виділити якийсь один етап як важливий, кожен із них має однакову вагу. Під час створення продукту тестувальник непросто грає значної ролі, а бере участь кожному етапі розробки від концепції до виходу товару світ.
Тестування документації - це початкова стадія процесу тестування, що постає як система раннього оповіщення про помилки. Процес тестування так чи інакше починається з документації та вимог. Тестування документації передбачає початок тестування ще до розробки товару. Тестувальник може вказати на логічні помилки у постановці завдання, невідповідності у вимогах, а також скласти чек-лист, список перевірок на надану вимогу.

У процес тестування документації важливо залучати різноманітних фахівців: тестувальники, проджект-менеджери, бізнес-аналітики, розробники.

Якщо помилка у вимогах буде знайдено на етапі тестування вимог, її вирішенням може бути лише виправлення кількох слів у тексті, тоді як знайдений дефект у вже реалізованому програмному продукті, може призвести до закриття проекту.

5.1 Документація

Документація – це ще одна складова програмного продукту будь-якої організації, що займається розробкою програмного забезпечення. Але не всі організації приділяють достатню кількість часу розробці документації, що стоїть дуже дорого. Дуже часто не щасливиться мати справу з тлумачним програмним продуктом і непоказним, незрозумілим і безпорадним документним супроводом.

Давайте спробуємо зібрати воєдино критерії тестування, що утворюють квінтесенцію якісної документації.

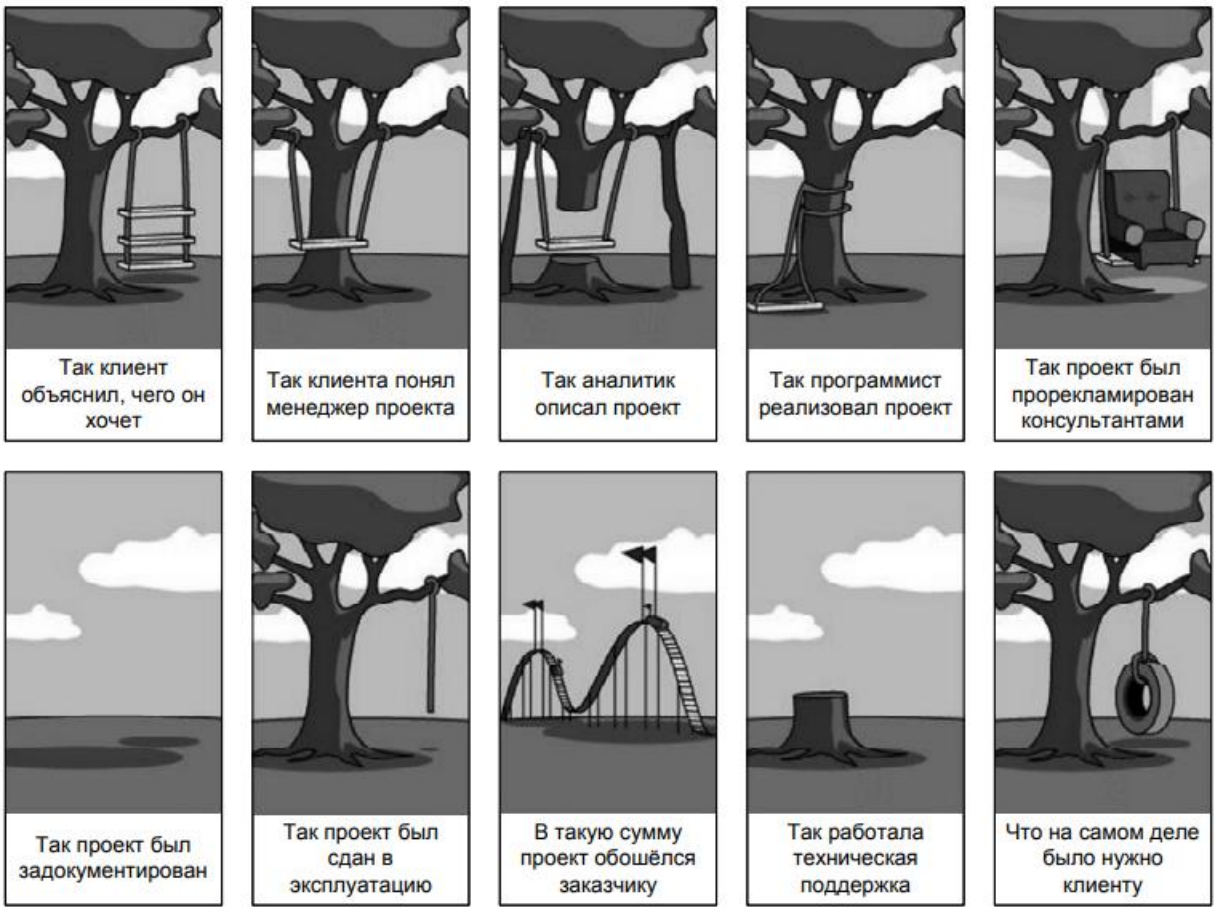
В цілому, документація створюється для можливості грамотно і без паніки знайти вихід або рішення з будь-якої проблемної ситуації людині з найнижчим знанням принципів розробки програмного забезпечення. Від цього принципу необхідно відштовхуватися, продумуючи зміст та структуру

Окремо відзначу, що при формулюванні цілей проекту, треба постаратися закласти в них свої власні (команди реалізації) стратегічні інтереси, які менеджер хочете досягти в результаті своєї діяльності в проекті.

А ось при визначенні цілей замовника, спільно з його представниками, аналітик повинен насамперед оперувати поняттям вигоди, яку отримає власник від використання кожної цільової системи, що розглядається. Тому коли замовник просить реалізувати якусь можливість, привчіть його відразу замислитися, а який профіт він з цього отримає. Так, можна ще на початковому етапі відсунути частину зайвих вимог і заощадити час на стадії визначення меж проекту.

Продуктна документація використовується проектною командою під час розробки та підтримки продукту. Вона включає:

- План проекту , і в тому числі тестовий план
- Вимоги до програмного продукту (та функціональні специфікації
- Архітектуру та дизайн.
- Тест-кейси та набори тест-кейсів.
- Технічні специфікації такі як схеми баз даних, описи алгоритмів, інтерфейсів тощо.



Малюнок 5.1 – Ілюстрація проекту з поганими вимогами

5.2 Функціональне тестування

ФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ — це тип тестування програмного забезпечення, який перевіряє програмну систему на відповідність функціональним вимогам/специфікаціям. Мета функціональних тестів полягає в тому, щоб перевірити кожну функцію програмної програми, надаючи відповідне введення, перевіряючи вихід відповідно до функціональних вимог.

Функціональне тестування в основному включає тестування чорної скриньки та не стосується вихідного коду програми. Це тестування перевіряє інтерфейс користувача, API, базу даних, безпеку, зв'язок клієнт / сервер та інші функціональні можливості тестованого додатка. Тестування може проводитися або вручну або з використанням автоматизації.

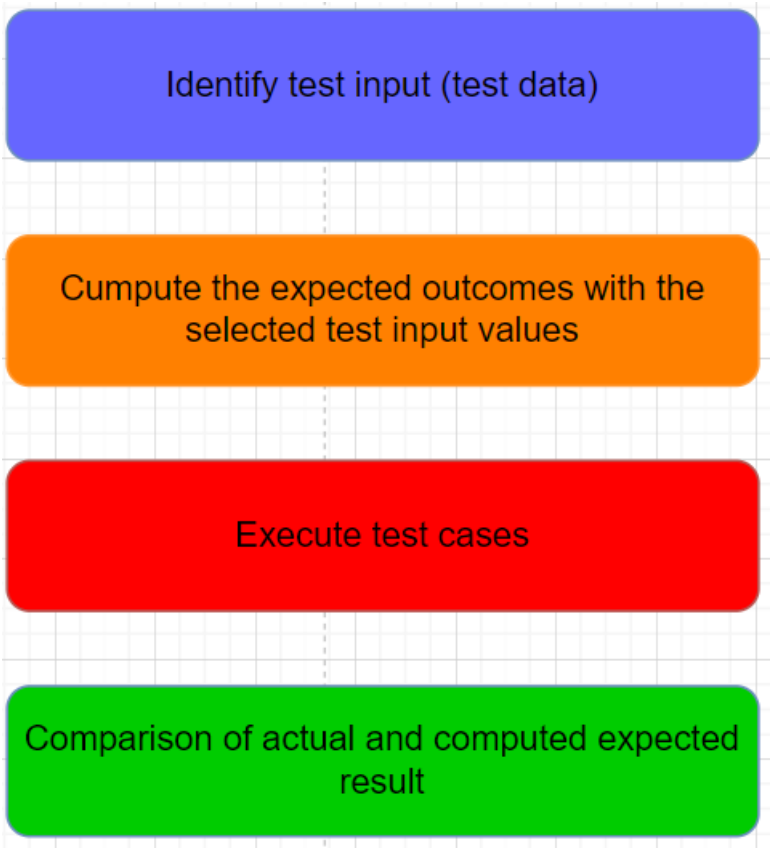
Мета функціонального тестування:

Основною метою функціонального тестування є перевірка функціональності системи програмного забезпечення. Він здебільшого концентрується на -

- Основні функції: тестування основних функцій програми
- Базова зручність використання: включає базове юзабіліті-тестування системи. Він перевіряє, чи може користувач вільно пересуватися по екранах без жодних труднощів.
- Доступність: перевіряє доступність системи для користувача.
- Умови помилки: Використання методів тестування для перевірки помилок. Він перевіряє, чи відображаються відповідні повідомлення про помилки.

Як виконувати функціонального тестування:

Щоб функціонально протестувати програму, необхідно дотримуватися наступних кроків.



Малюнок 5.2 – Послідовність кроків для функціонального тестування

- Розуміти вимоги до програмної інженерії
- Визначте вхід тесту (дані тесту)
- Обчислити очікувані результати з вибраними вхідними значеннями тесту
- Виконати контрольні приклади
- Порівняння фактичного та розрахункового очікуваного результату

5.3 Нефункціональне тестування

НЕФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ - визначається як тип тестування програмного забезпечення для перевірки дисфункцій (продуктивність, зручність використання, надійність і т. д.) Програмного додатка. Він призначений для перевірки готовності системи за нефункціональними параметрами, які ніколи не враховуються під час функціонального тестування..

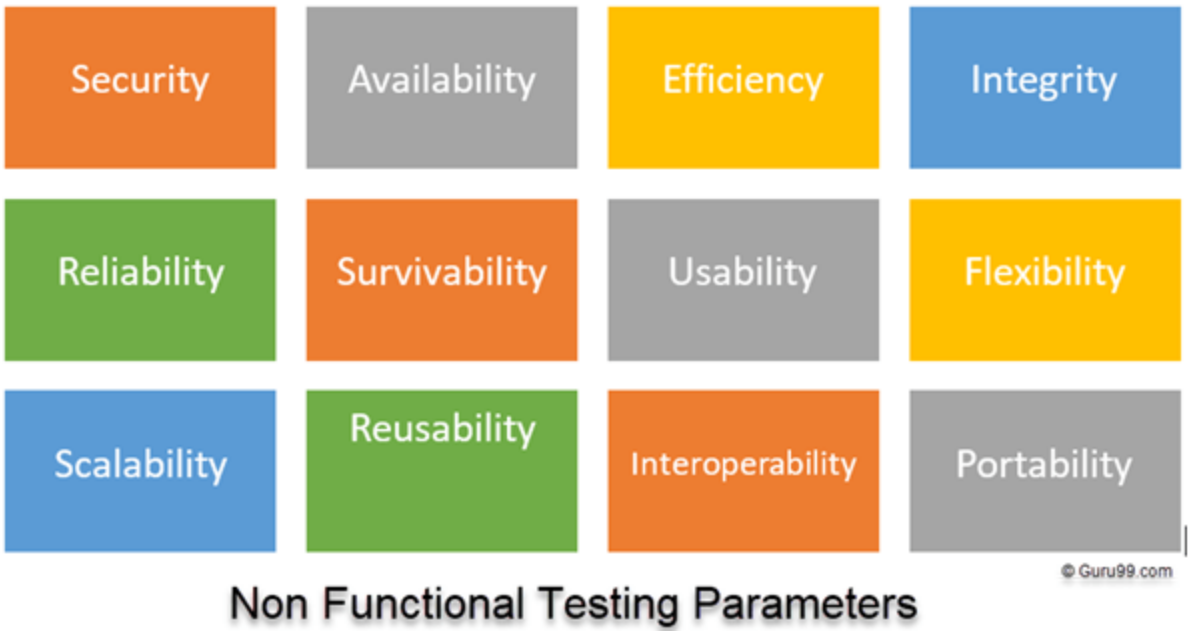
Відмінним прикладом дисфункції тесту може бути перевірка того, скільки людей можуть одночасно увійти в програмне забезпечення.

Нефункціональне тестування не менш важливе, ніж функціональне тестування, і впливає на задоволеність клієнтів.

Мета нефункціонального тестування:

- Нефункціональне тестування має підвищити зручність використання, ефективність, ремонтопридатність та переносимість продукту.
- Допомагає знизити виробничий ризик та витрати, пов'язані з нефункціональними аспектами продукту.
- Оптимізуйте спосіб встановлення, налаштування, виконання, керування та моніторингу продукту.
- Збирати та проводити вимірювання та метрики для внутрішніх досліджень та розробок.
- Поліпшення та розширення знань про поведінку продукту та використовувані технології.

Параметри нефункціонального тестування:



Малюнок 5.3 – Параметри нефункціонального тестування

1) **Безпека:** Параметр визначає, як система захищена від навмисних та раптових атак із внутрішніх та зовнішніх джерел.
2) **Надійність:** Ступінь, в якій кожна програмна система неперервно виконує вказані функції без збою.
3) **Живість**: Параметр перевіряє, що система програмного забезпечення продовжує функціонувати і відновлюється у разі збою системи.

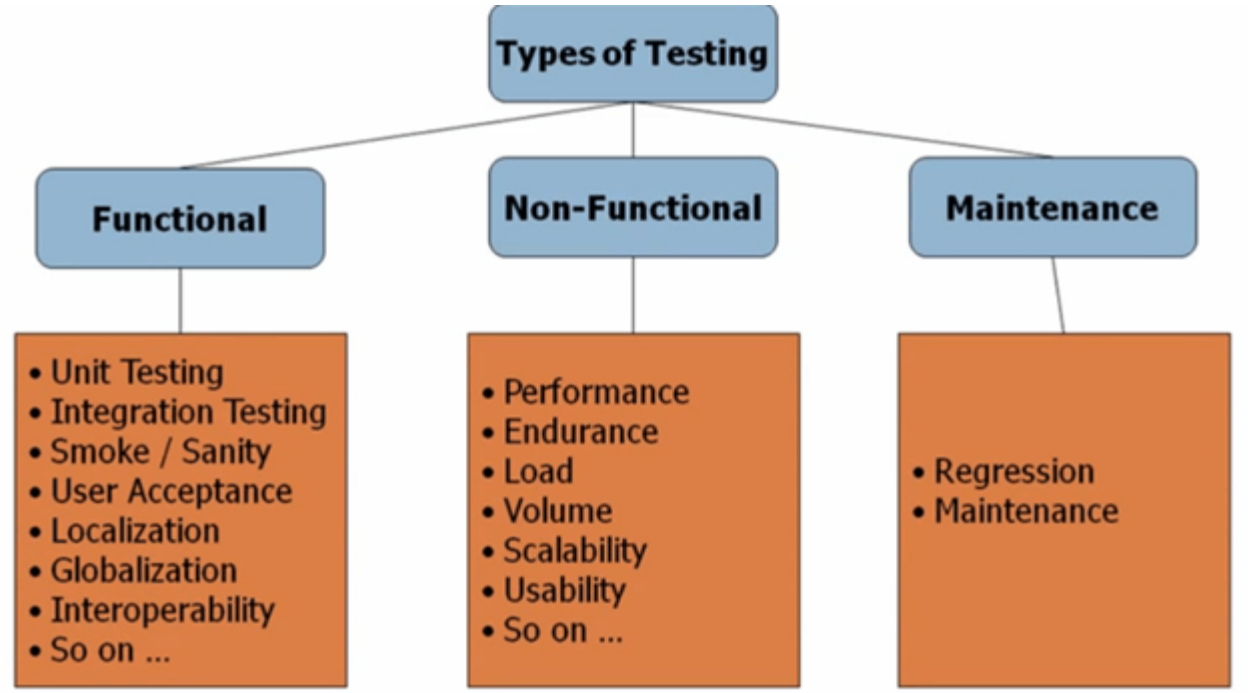
4) **Наявність:** Параметр визначає рівень, в якому користувач може залежати від системи під час її роботи.
5) **Зручність використання:** Легкість, з якою користувач може вчитися, працювати, готувати вхідні та вихідні дані за допомогою взаємодії із системою.
6) **Масштабованість:** Термін відноситься до ступеня, в якому будь-яка програмна програма може розширити свої обчислювальні потужності, щоб задовольнити збільшення попиту.
7) **Сумісність:** Цей нефункціональний параметр перевіряє взаємодію програмної системи з іншими програмними системами
8) **Ефективність:** Ступінь, в якій будь-яка програмна система може обробляти ємність, кількість та час відгуку.
9) **Гнучкість:** Термін відноситься до легкості, з якою програма може працювати в різних апаратних та програмних конфігураціях. Як мінімум оперативної пам'яті, вимоги до процесора.

Наведений вище список не є повним, оскільки існує понад 100 типів тестування та підрахунку.

5.4 Порівняння функціонального і нефункціонального тестування

Як ми вже визначили,є три типи тестування

- Функціональне
- Нефункціональне
- Технічне обслуговування



Малюнок 5.4 – типи тестування

Давайте зробимо таблицю наглядну таблицю для порівняння функціонального та нефункціонального тестування:

Таблиця 5.1 – Порівняння функціонального та нефукціонального тестування

Функціональне тестування	**Нефункціональне тестування**
Функціональне тестування виконується з використанням функціональної специфікації, наданої клієнтом, та перевіряє систему на відповідність функціональним вимогам.	Нефункціональне тестування перевіряє продуктивність, надійність, масштабованість та інші дисфункції системи програмного забезпечення.
Функціональне тестування виконується першим	Нефункціональне тестування має виконуватись після функціонального тестування.
Для функціонального тестування можуть використовуватись інструменти ручного тестування або автоматизації.	Використання інструментів буде ефективним для цього тестування
Бізнес-вимоги є вхідними даними для функціонального тестування	Параметри продуктивності, такі як швидкість, масштабованість є вхідними даними для нефункціонального тестування.
Функціональне тестування визначає, що робить продукт	Нефункціональне тестування визначає, наскільки добре працює продукт

Легко зробити ручне тестування	Складно зробити ручне тестування
Приклади функціонального тестування: • Модульне тестування • Тестування диму • Тестування в здоровому глузді • Інтеграційне тестування • Тестування білої скриньки • Тестування чорної скриньки • Тестування користувача • Регресійне тестування	Приклади нефункціонального тестування: • Тестування продуктивності • Тестування навантаження • Об'ємне тестування • тестування навантаження • Тестування безпеки • Тестування установки • Перевірка на проникність • Тестування сумісності • Міграційне тестування

- **<u>Ключові відмінності:</u>**
  Функціональне тестування перевіряє кожну функцію/функцію програмного забезпечення, тоді як нефункціональне тестування перевіряє нефункціональні аспекти, такі як продуктивність, зручність використання, надійність тощо.
- Функціональне тестування можна виконати вручну, тоді як нефункціональне важко виконати вручну.
- Функціональ не тестування базується на вимогах замовника, тоді як нефункціональне тестування базується на очікуваннях замовника.
- Функціональне тестування має на меті перевіряти дії програмного забезпечення, тоді як нефункціональне тестування має на меті перевірити ефективність програмного забезпечення.

- Приклад функціонального тестування — це перевірка функціональності входу, тоді як приклад нефункціонального тестування — перевірка, чи інформаційна панель завантажується за 2 секунди.
- Функціональний описує, що робить продукт, тоді як нефункціональний описує, як працює продукт.
- Функціональне тестування проводиться перед нефункціональним тестуванням.

5.5 Тестування вимог на програмний проект

Головна ціль	**Попередні вимоги**	**Шлях до виконання тесту**	**Очікуваний результат**	**Дії для повернення на початок**
Оцінювати фільм	Бути зареєстрованим	Обрати в описі фільму бажану оцінку; Натиснути кнопку “Save	Отримати повідомлення про успішне визначення оцінки фільму	Видалити свій рейтинг
Дивитись рейтинг фільму	Бути зареєстрованим необов’язково	Обрати вподобаний фільм та перейти на його сторінку	Можна побачити рейтинг будь - якого фільму	Натиснути кнопку «повернутись»
Шукати фільм по жанрам	Бути зареєстрованим необов’язково	Переходимо на пошукову сторінку та можемо побачити перелік жанрів фільму.Обираємо жанри(від 1го) та фільм підбирає нам фільми за переліченими жанрами	Підбір фільм за переліченими жанрами	Видалити обранні жанри

Переглядати інформацію по фільмам,акторам, режисирам	Бути зареєстрованим необов'язково	Є два маршрути: 1)переходимо до пошуку та вписуємо ключові букви та користувач зможе побачити та обрати приблизні результати 2) користувач на вже обраній сторінці,наприклад фільму, може перейти до сторінки актора/режисира фільму та побачити його інформацію	Шукана сторінка з можливостю перегляду інформації та переходом на інші сторінки	Нажати кнопку «повернутись» або натиснути кнопку «головна сторінка»
Додавати фільми до списку "Wish-list"	Бути зареєстрованим.	Обираємо бажаний фільм та натискаємо кнопку «додати до Wish List"	Фільм додається до аккаунту зареєстрованого користувача до його особистої бібліотеки	Заходим до «Wish list(-у)» користувача та видаляємо фільм

Таблиця 5.2 -Таблиця вимог на програмний проект

Тестовий сценарій:

Для всіх відвідувачів:

Юзер заходить на сайт, на **початкову сторінку**, на якій розташований пошук по базі фільмів, акторів, режисерів, нижче на початковій сторінці є категорії з назвами 9 жанрів для того, щоб юзер міг одразу перейти до списку фільмів, відфільтрованих за жанром. Вгорі завжди буде меню сайту, яке буде складатися з посилань на списки фільмів, акторів, режисерів, на особистий watch-list (для зареєстрованих користувачів), також кнопка "Вхід".
На сторінці списку фільмів початково буде розташовано список усіх фільмів, відсортованих за рейтингом (за рейтингом IMDB, у перспективі — за рейтингом сайту Проекта), також буде пошук, та різноманітні фільтри (країна, рік, жанр і т.д.). При кліку на окремий блок з фільмом, юзер буде переходити на сторінку з

інформацією цього фільму. На цій сторінці буде основна інформація по фільму, відгуки, також посилання на сторінки акторів, що знімалися у цьому фільмі, і посилання на сторінку режисера. Також буде можливість перейти “назад” до списку фільмів. Аналогічно буде відображено сторінки списку акторів і режисерів, з відповідними посиланнями.
Для зареєстрованих користувачів буде доступна можливість залишати відгуки та ставити бали рейтингу на сторінках фільму, актора(-ки) або режисера(-ки), а також можливість створити власний watch-list фільмів, які він планує подивитись. Також буде доступна сторінка особистого кабінету, де можна буде створювати різні watch-list-и, змінювати дані.

Опис варіанта використання

Можливості

- Оцінювати фільми.
- Дивитись рейтинг фільмів.
- Шукати фільм по жанрам.
- Переглядати інформацію по фільмам, акторам, режисерам.
- Добавляти фільми до списку «Watch-list»
- Мати список індивідуальних рекомендацій на основі його оцінювань інших фільмів.

Потік подій:

1. На сайті нажимає на «Вхід».

2. Заповнює особисті дані (логін, пароль).

3. Переходить до особистої сторінки.
    3.1 Переглядає особисті дані.
    3.2 Переглядає watch-list.
    3.3 Редагує особисту сторінку.
    3.3.1 Видаляє аккаунт.
    3.3.2 Добавляє/Видаляє фільми з watch-list.
    3.3.3 Редагує особисту сторінку.

4. Переходить до списку фільмів.
    4.1 Переходить на сторінку конкретного фільму.
    4.1.1 Читає інформацію про фільм.
    4.1.2 Переходить вниз сторінки до відгуків.
    4.1.3 Добавляє до watch-list.
    4.1.4 Читає/Залишає відгук.
    4.1.5 Оцінює фільм.
    4.2 Переходить на сторінку акторів або режисерів.

5. Переходить до списку акторів.
   5.1 Читає біографію актора.
   5.2 Список фільмів, де він знімався.
   5.3 Читає відгуки.
   5.4 Залишає відгук.
   5.5 Переходить до фільму, в якому цей актор знімався.

6. Переходить до списку режисерів.
   6.1 Читає біографію режисера.
   6.2 Дивиться список фільмів знятих цим режисером.
   6.3 Переходить до списку фільмів

Відвідувач сайту:

Можливості:

- Дивитись рейтинг фільмів.
- Шукати фільм по жанрам.
- Переглядати інформацію по фільмам, акторам, режисерам.
- Зареєструватися на сайті.

Потік подій:

1. На сайті здійснює пошук фільму.

2. Читає інформацію про фільм.

3. Дивиться рейтинг фільму.

4. Переходить до режисера, який зняв цей фільм.

5. Читає біографію, список фільмів режисера.

6. Переходить на головну сторінку.
   6.1 Реєструються на сайті.
   6.2 Переходить до списку фільмів/акторів/режисерів.

Адміністратор:

Короткий опис: Адміністратор слідкує для справністю сайту і добавляє інформацію про нові фільми, актори та режисери, та може видаляти коментарі.

Малюнок 5.5 – діаграма прецидентів

Висновки до розділу 5

Функціональне тестування - це перевірка працездатності, функціональності програмного забезпечення, сервісу, програми.
Нефункціональне тестування - це теж тестування програми, програми, але в цей вид тестування входить все, що не входить безпосередньо до функціонального тестування. Тобто. окрім перевірки функціональності можна перевіряти, наприклад, наскільки швидко працює програма, програма, в яких браузерах працює сайт, в яких оточеннях працює сайт. Все це відноситься до дисфункції тестування.
Таким чином, ми коротко описали, у чому різниця між функціональним та нефункціональним тестуванням програмного забезпечення.
Отжем, ми в цьому розділі ми продивились що таке функціональне та нефункціональне тестування, а також провели їх порівняльну характеристику.

Також було протестовані вимоги до проекту та написан тестовий сценарій для нашого програмного проекту

6. Існуючі інструментальні засоби автоматизації створення вимог (ТЗ) на проект

Кількість доступних на ринку автоматизованих інструментів управління проектами швидко зростає. Після значного розвитку цих інструментів багато керівників проектів почали використовувати різні програмні засоби управління проектами для управління та підтримки своєї проектної діяльності. Ці інструменти в основному використовуються при плануванні, моніторингу та контролю проектів. Функції, надані цими інструментами, розраховуються. Менеджери проектів повинні вибрати відповідний набір інструментів з необхідними функціями серед багатьох інструментів, знайдених на ринку

У складних програмних проектах успішного планування проектів у значній мірі використовує автоматизовані інструменти планування проектів. Отже, для власників або менеджерів проектів є важливим вибрати найбільш підходящий інструмент або набір інструментів для своїх потреб в управлінні проектами

У цьому розділі ми проаналізуємо набір програмних засобів управління проектами (PMST). В наступному розділі ми порівняємо ці інструменти, використовуючи набір критеріїв. Були вибрані найпопулярніші та доступні інструменти для управління проектами.

Список інструментів, що підлягають даному дослідженню, виглядає наступним чином:

1) **GanttProject**
2) **LiquidPlanner**
3) **OpenProj**

6.1 GanttProject

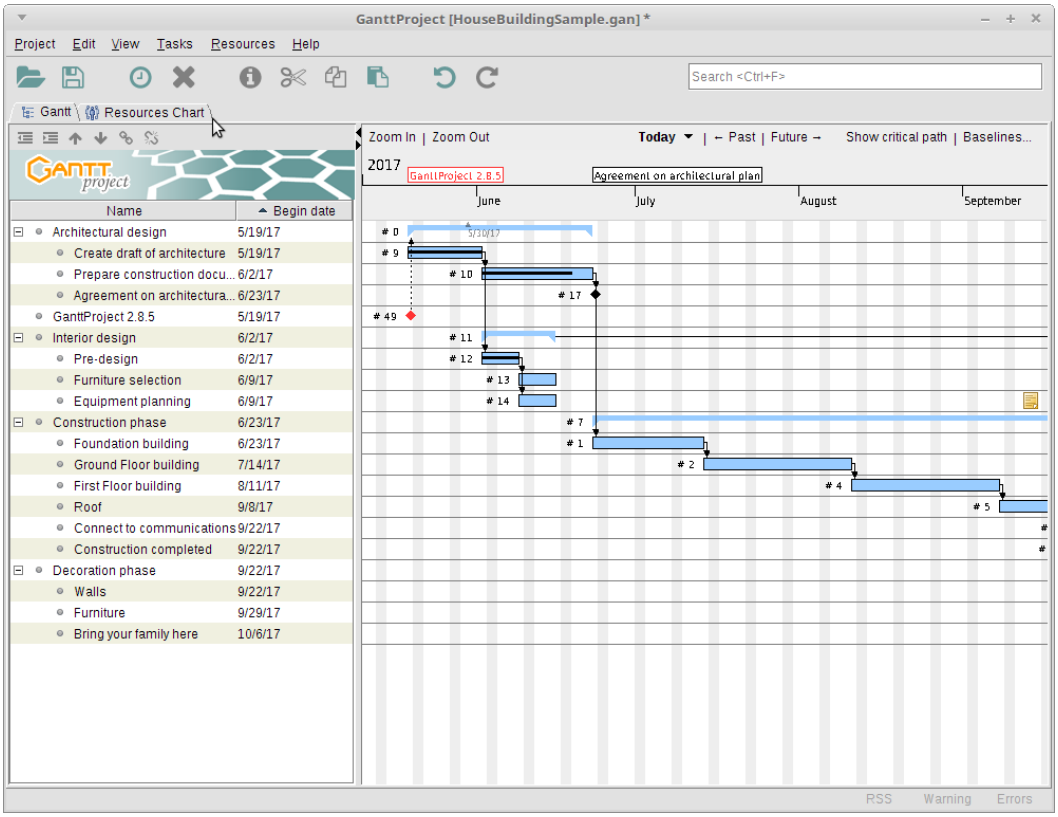
GanttProject - це програмне забезпечення для керування проектами з відкритим вихідним кодом на основі Java, яке працює під керуванням Windows, Операційні системи Linux та Mac OS X.
Члени команди та їх інформація, такі як облікові записи електронної пошти, номери телефонів можуть бути додані до інструменту, а завдання можуть бути призначені кожному з членів команди. Ключова особливість GanntProject полягає в тому, що він зберігає файли за допомогою an.формат xml, таким чином робить їх доступними через інтернет.
Серед його функції:

- Вартість зароблennої вартості
- Діаграма Ганта
- Діаграма PERT
- Структура розподілу ресурсів (RBS);
- Звіти про використання завдань;
- Структура розподілу робіт (WBS);
- Мережна діаграма;
- Проста звітність про завдання та ресурси;
- Імпорт / експорт файлів проектів з MS Project.
- Обмін даними з електронними таблицями
- Групова робота на основі WebDav

Використання:
GanttProject має інтуїтивно зрозумілий інтерфейс, який практично не відрізняється від аналогів, а також містить мінімальний набір найнеобхідніших функцій для вирішення поставлених завдань. Потрібні етапи проекту можуть бути виділені різними кольорами з метою швидшого орієнтування. Ця програма підтримує побудову таблиць, графіків, діаграм і календарних планів, а тому буде корисна як для бізнесменів-початківців, так і для великих підприємств. При необхідності, дана утиліта може бути відправлена в системний трей та продовжити свою роботу у фоновому режимі, що особливо актуально для регулювання та моніторингу завдань із мінімальними термінами на виконання. Програма написана мовою Java, за рахунок чого підтримується на різних операційних системах. Побудова діаграм здійснюється із застосуванням

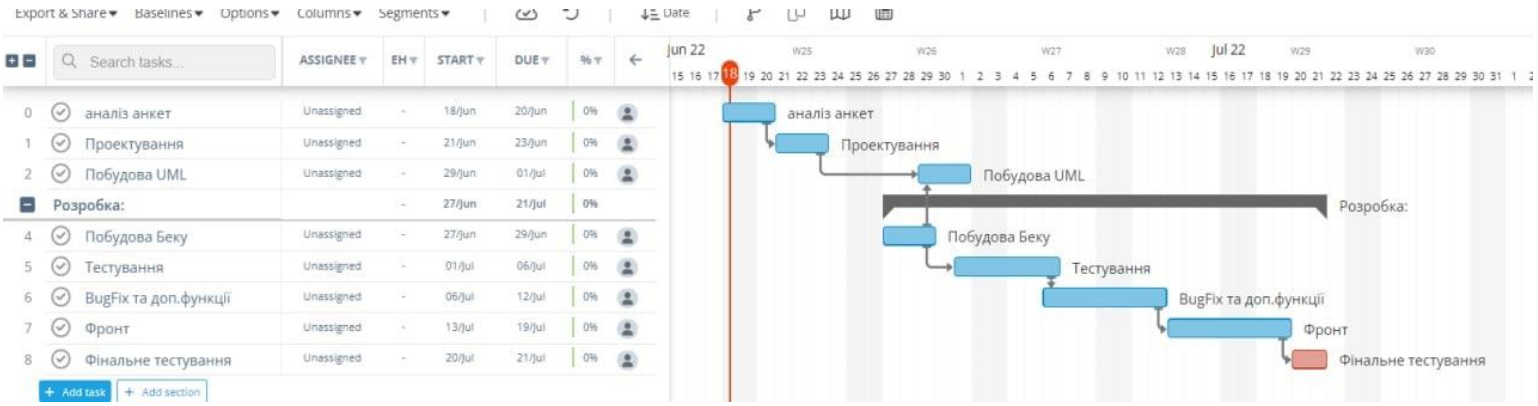
технології Гантта. Вся необхідна інформація може бути вивантажена за допомогою XSL-перетворень у документи HTML та PDF, що можливо завдяки XML. Крім того, є функція завантаження та збереження файлу проекту на FTP, тим самим цей документ зможуть переглядати одночасно кілька користувачів. Існує можливість експорту проектів у форматах txt та XML безпосередньо у цю утиліту.

Під час створення нового користувача для участі у проекті вказуються ПІБ, номер телефону, відпускні дні та електронна пошта. Кожному учаснику завдання може бути наданий статус координатора, невизначеного та інший, вказаний вами в налаштуваннях. Досягши необхідної мети, звіт можна зберегти у вигляді текстового або графічного зображення.



Малюнок 6.1 – Головна сторінка GanttProject версії 2.8.5 зразкового проекту

Також для демонстрації в GuttProject зробимо наше проектне завдання. Але зробимо поправку на те що на малюнку 10 було показана зразок версії GuttProject 2.8.5.



Малюнок 6.2 – Демонстрація роботи програмного проекту в GuttProject 3.2

Переваги GuttProject:

1) Можливість додавати підзавдання та змінювати кольорове кодування, щоб допомогти організувати роботу, яку потрібно виконати
2) Можливість пов'язувати різні завдання разом (залежності завдань), що допомагає уявити, що ще потрібно зробити поряд з поточним завданням
3) Перегляд тривалості, яку це може зайняти, та діаграми прогресу, щоб допомогти кількісно оцінити та візуалізувати свій прогрес

Недоліки:

1) GanttPRO не дозволяє додавати підзавдання до підзадачі, позбавляючи налаштування та гнучкості для різних випадків використання
2) GanttPRO розширює завдання до дати підзавдання, тому якщо завдання в цілому займає більше часу, ніж підзавдання, вам потрібно вказати це як таке

GanttPRO не дозволяє призначати матеріали для завдань

6.2 OpenProj

OpenProj – безкоштовний аналог Microsoft Project. Дане кроссплатформенное програмне забезпечення призначене для планування проектів і є дуже гарною заміною платного ПЗ. Програма включає всі необхідні функції: діаграма Ганта, мережевий трафік, розподіл ресурсів, звіти, а також підтримує імпорт/експорт документів Microsoft Project. Програма існує у двох варіаціях: платна, для

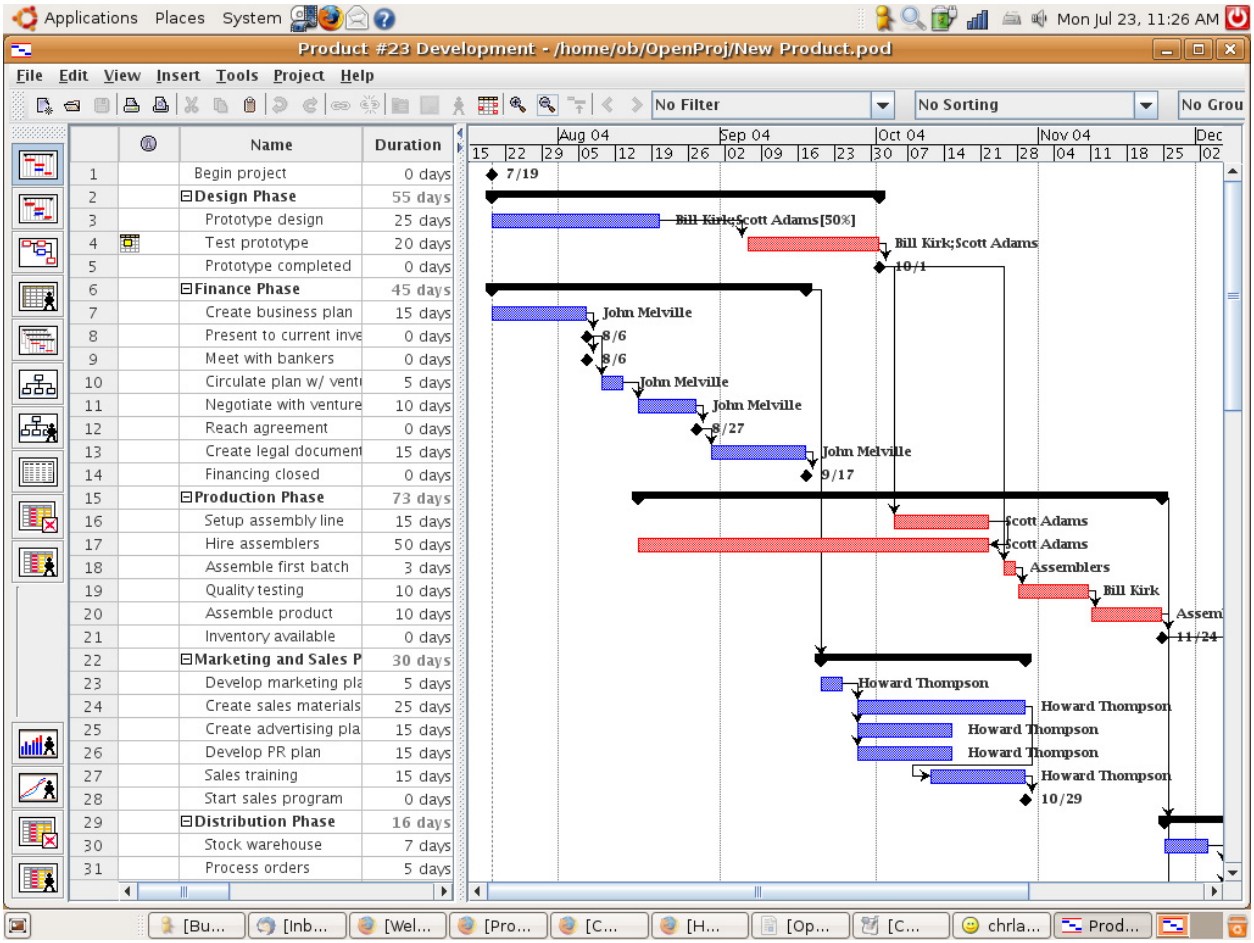
спільного користування та безкоштовна, а саме OpenProj, де доступне використання продукту тільки на своєму персональному комп'ютері.

Серед його функції:

- Вартість заробленної вартості
- Діаграма Ганта
- Діаграма PERT
- Структура розподілу ресурсів (RBS);
- Звіти про використання завдань;
- Структура розподілу робіт (WBS);
- Мережна діаграма;
- Проста звітність про завдання та ресурси;
- Імпорт / експорт файлів проектів з MS Project.

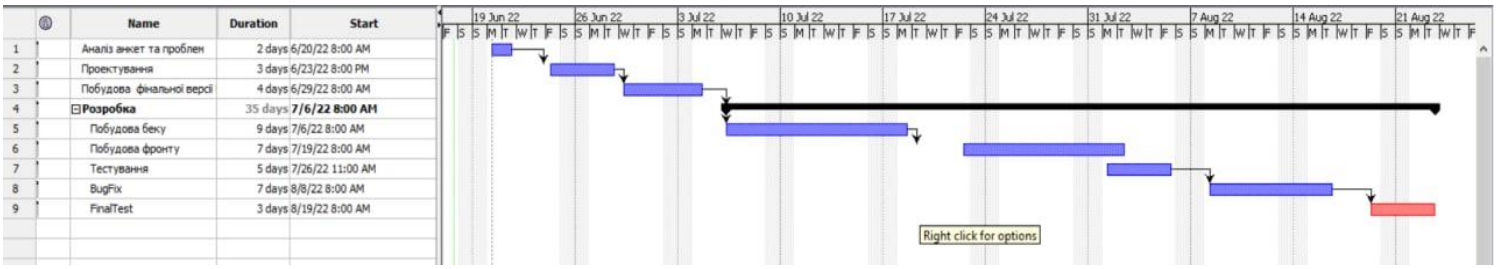
Використання:

Як відомо, програма призначена для створення та управління вашими проектами. Перше, що хочеться відзначити – її функціонал. Створення проекту не є складним, є безліч осередків для введення своїх критеріїв. Також в окремому вікні можна індивідуально налаштувати % завершення робіт, їх вартість, тривалість та багато іншого.
Що ж до інтерфейсу, він максимально схожий з Microsoft Project, є часткова підтримка імпорту файлів із цієї програми. І все-таки повернемося до інтерфейсу. Все виконано досить якісно та зручно. Багато вкладок з списком, що розгортається, допоможуть вам не заплутатися в пошуку потрібного інструменту або налаштування.
Після створення проекту ви можете при відкритті вкладки «Вид» вибрати зручний режим огляду: діаграма Ганта, мережевий графік, перегляд ресурсів та їх використання тощо.



Малюнок 6.3 – Головна сторінка OpenProj версії 1.4 зразкового проекту

Далі для демонстрації в OpenProj зробимо наше проектне завдання



Малюнок 6.4 – Головна сторінка OpenProj версії 1.4 програмного проекту

Переваги OpenProj:

1) Безкоштовно. Хоча є деякі платні функції, основний продукт виконує те, що нам потрібно.
2) Можливість пов'язувати різні завдання разом (залежності завдань), що допомагає уявити, що ще потрібно зробити поряд з поточним завданням

3) Перегляд тривалості, яку це може зайняти, та діаграми прогресу, щоб допомогти кількісно оцінити та візуалізувати свій прогрес

Недоліки:

1) Планування та управління ресурсами є дещо обтяжливим для OpenProject
2) Звіти про завантаження ресурсів потрібно створювати вручну за допомогою інших інструментів (електронних таблиць), щоб показати, чи є ресурси доступні чи заповнені.
3) Может бути не оптимізованним

Програма по своєму інтерфейсу дуже схожа на Microsoft Project, що істотно облегчить перехід на безкоштовне ПО користувача, який привик до комерційного продукту. Також велике значення має те, що вона дозволяє працювати напрямком з файлами MS Project і Primavera. Слід помітити, що сумісність файлів буде лише частинною. Також програма легко встановлюється і налаштовується. Програма не досить гарна, щоб отримати платний аналог, але чудово підходить для тех, кому потрібно ознайомитися з проектом, але немає можливості платити великі гроші тільки із-за можливості в навчанні. Дану програму хочеться більше порекомендувати для встановлення та вивчення в освітніх установах із-за її простоти та вільного поширення.

6.3 Microsoft Project

Цей інструмент управління проектами, розроблений корпорацією Майкрософт, закликаний запропонувати керівникам проектів розробку планів проектів і вимог, призначити завдання роботодавцям і контролювати стан проектів.Кроме того, в MS Project користувачі розділяються на групи в залежності від призначених ім задач або должностей. Таким чином, користувачі можуть мати різні рівні доступу до різних проектних документам. За допомогою надстройки MS Project Server MS Project можна використовувати для облегчения спільної роботи в інтернете. В якості резюме, як рекламується на офіційному сайті. Серед його функції:

1) сумісна работа;

2) комунікація;
3) управління задачами;
4) Управление проектами;
5) швидкий доступ до інформації;
6) покращений інтерфейс;
7) управління портфелем;
8) Управление документами.

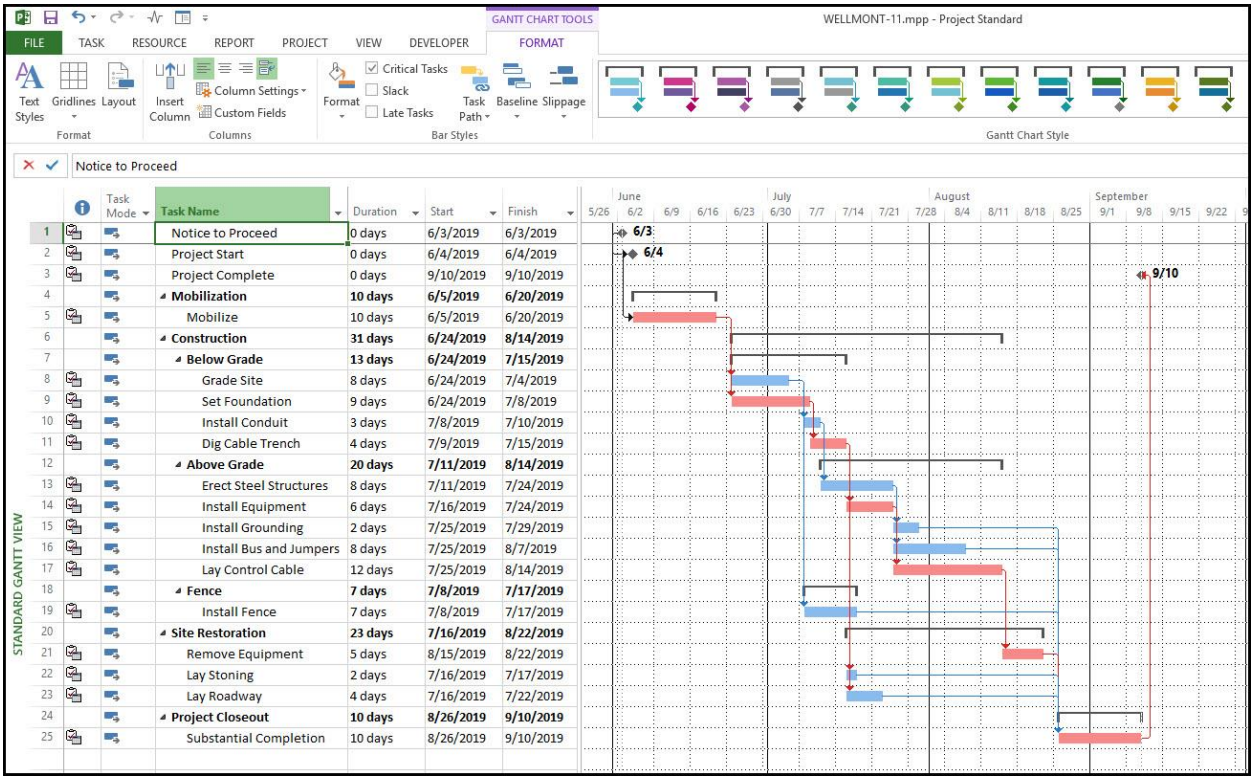
Використання:

Незважаючи на зовнішню складність, MS Project є дуже простим в ідейному плані. Він оперує трьома сутностями – завдання, ресурси, календар та зв'язки між ними.

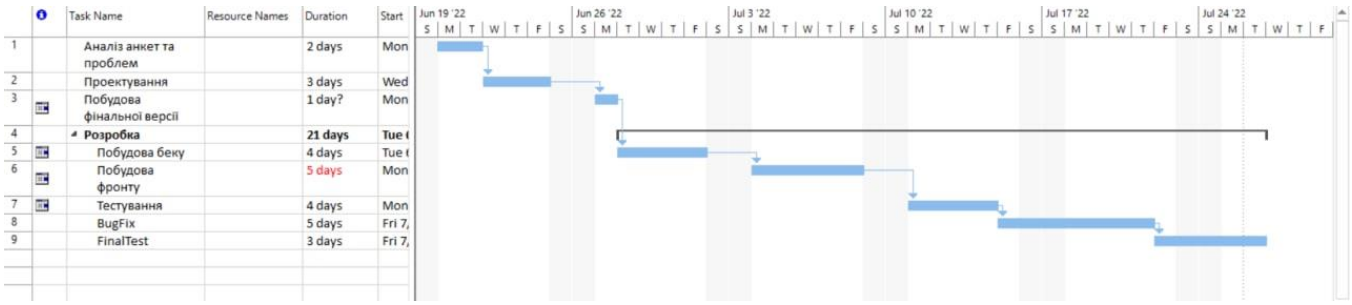
Завдання має тривалість, обсяг, призначений ресурс і ще велику різних якостей. Якщо вбудованих властивостей не вистачає, можна додати свої. Завдання можуть бути пов'язані між собою різними відносинами (попередники, послідовники тощо).

Ресурс має багато описових властивостей, але найголовніше – для нього можна встановити доступність у часі, для цього використовується календар. Ресурс може бути призначений на завдання.

На основі цих даних Project вміє робити різні уявлення з використанням фільтрів, угруповань, сортувань тощо. Крім цього він вміє за деяким алгоритмом обчислювати терміни початку та закінчення завдань з урахуванням доступності призначених ресурсів та зв'язків між завданнями.



Малюнок 6.5 – Головна сторінка Microsoft Project зразкового проекту



Малюнок 6.6 – Головна сторінка Microsoft Project програмного проекту

Переваги Microsoft Project:

1) Додаток пропонує широкий спектр інструментів для управління проектами. MS Project є гнучким і має ефективні інструменти.

2) Програма також допомагає контролювати прогрес на різних етапах розробки проекту та вивчати всі вимоги, необхідні для розробки модулів

проекту. MS Project отримав функції, які розробники проекту можуть використовувати для визначення та виправлення помилок, виявлених під час встановлення та оновлення будь-якого бізнес-проекту.

3) Забезпечує резервне копіювання файлів і документів

Недоліки:

1) Освоїти програму займає багато часу

2) Встановити проект також дорого, оскільки для його ефективної роботи потрібні складні ресурси.

3) Програма також може мати труднощі з сумісністю з останніми версіями операційних систем, а також з операційною системою UNIX

6.4 Висновок до розділу 6

У той час як кількість програмних засобів управління проектами (Pmst) зростає, менеджери повинні вибрати підходящий інструмент для своїх проектів. У цьому дослідженні ми вибрали набір PMST та досліджували їх. Далі ми визначили деякі переваги та недоліки , які можна використовувати для порівняння цих та інших інструментів на ринку. Зрештою, були наведені наші висновки. Завдяки цьому дослідженю керівникам проектів та членам команди представляється стислий опис інструментів щоб отримати уявлення про них, та оцінити їх сильні та слабкі сторони.

Знайти найбільш ефективне та дешеве програмне забезпечення для управління проектами може бути дуже важким завданням. Основні речі, які слід враховувати при пошуку програмного рішення для управління проектами: вартість продукту, сумісність з іншими продуктами, що працюють у цій системі,

простота використання та підтримки постачальників, безпека та досвід, необхідні для використання продукту.

7 Пропозиції зі створення авторської системи автоматизованого створення вимог до програмної системи

В минулому розділі ми з вами продивилися три інструмента для управління проектами та винесли їх переваги та недоліки.В цьому розділі зробимо більш конкретніше порівняння MS PROJECT, GunttProject, OpenProj.

Для кожного з інструментів, які ми досліджували, підтримується функціональність або функції цього інструменту, які перераховуються. Якщо інструмент підтримує певну функціональність або функцію, ми ставим під ним «+», в протилежному випадку ми ставим «ні». Порівняння для оцінки цих інструментів ми визначаємо основні функції та основні характеристики, які забезпечують основу процесу оцінки. У роботі було обрано та визначено дванадцять критеріїв (1–12) для порівняння та оцінки інструментів управління програмними проектами.

Таблиця 7.1 – порівняння систем формування вимог

Система	GanttProject	MicrosoftProject	OpenProject
Технічна документація	-	-	+
Ліцензія	Open Source	Комерційна Web based service (SaaS)	Open Source
Multi User	-	+(онлайн)	+

API	+	+	+
Властивості планування проекту			
Система ієрархії	+	+	+
Відслідковування часу	-	+	+
Оцінка	+	+	+
Оцінка ризику	-	+	-
Управління ресурсами	+	+	-
Критичний путь	+	+	-
Контроль якості	-	+	-
Планування задач	+	+	-

Судячи з нашої таблиці можно зробити висновок що при порівнянні MS Project, OpenProj та GanttProject більше переваг для саме у MS Project.

7.1 Висновок до розділу 7

Усі системи управління мають певні цілі та завдання і про них варто сказати кілька слів.

Ціль системи управління проектами:

- Підвищити ефективність працівників в проектній роботі
- Підвищити якість проектного менеджемнту керівника проектів
- Підвищити ефективність управліня загальним проектним портфелем організації

Серед можливих вигод від застосування систем управління проектами можна виділити в першу чергу те, що скорочується кількість проектів, що не відповідають стратегії компанії, а це означає, що й витрати по всьому.

Вимоги до функціональних можливостей систем управління проектами залежать від нюансів управління та самих проектів в кожній конкретній організації.

8. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ СИСТЕМИ ФОРМУВАННЯ ВИМОГ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для розробки автоматизованої системи формування вимог.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі

оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

- для кожної функції визначаються повні річні витрати й кількість робочих часів.
- для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.
- після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

8.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу, тестування та вибору рішення для побудови автоматизованної системи формування вимог. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, який призначений обробляти та аналізувати дані. Технічні вимоги до будуть наступні:

- застосування на персональних комп’ютерах із стандартною конфігурацією за допомогою веб-браузера;
- Висока швидкість обробки даних
- система має передбачати зручний спосіб встановлення, підтримки та оновлення
- зручний спосіб взаємодії з компонентами
- швидкий виклик та виконання функцій

Обґрунтування функцій програмного продукту

Головна функція F_0– вибір програмного продукту, що відповідає за розгортання мережі та виконує функції по керуванню нею. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір системи

F_2 – спосіб встановлення системи;

F_3 – наявність додаткових службових компонентів;

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) Dart;

б) Kotlin.

Функція F_2:

а) Native;

б) Crosplatform.

Функція F_3:

а) Bitrise;

б) Codemagic

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 8.1).

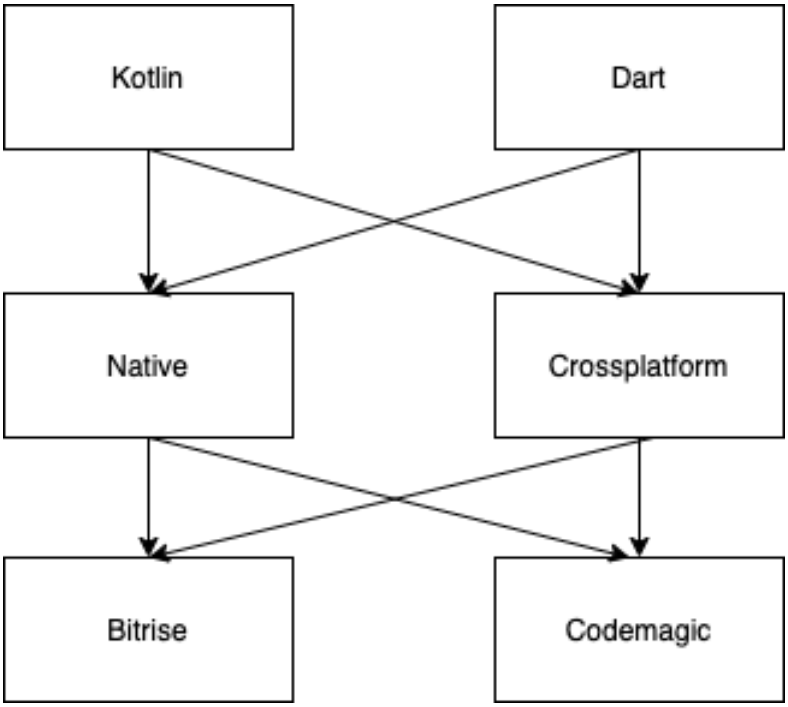


Рис. 8.1 - морфологічна карта

Морфологічна карта відображає множину всіх можливих варіанти основних функцій.

Таблиця 8.2 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	*А*	Кросплатформеність, легкість у використанні	Мала поширеність
	Б	Нативна підтримка	Більша складність у використанні, менша гнучкість
F_2	*А*	Швидкість роботи	Лише під одну платформу
	Б	Зменшення затрат часу на інші платформи	Відсутність платформо-орієнтованих компонентів

F_3	*А*	Необхідність лише документування готових API	Незручність використання кінцевими користувачами
	Б	Легкість у використанні кінцевими користувачами	Більше часу на розробку

На основі цієї карти будуємо позитивно-негативну матрицю варіантів основних функцій (Таб.8.2). Робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1:

Перевагу віддаємо швидкості вивчення, простоті використання та наявності стандартних бібліотек для обчислення. Для спрощення роботи по написанню коду варіант Б має бути відкинутий.

Функція F_2:

Аналогічно до F_1.

Функція F_3:

Можна розглянути обидва варінти.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а – F_2а – F_3а

F_1а – F_2а – F_3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – швидкодія системи;
- *X2* – кількість додаткових службових компонентів;
- *X3* – кількість доступних мов програмування;
- *X4* – потенційний об’єм додаткового функції.

X1: Відображає час, за який система обробить тестову функцію.

X2: Відображає кількість додаткових службових компонентів, потрібних для роботи системи.

X3: Відображає кількість мов програмування, які підтримує система.

X4: Відображає скільки службового коду потрібно написати для запуску функції у системі.

Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 8.1.

Таблиця 8.4 - Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Час обробки запиту	X1	мс	1000	500	100
Кількість сервісів	X2	К-ть	2	3	4
Об’єм даних, що передаються	X3	Кб	2000	800	500
Потенційний об’єм програмного коду	X4	кількість рядків коду	5000	3500	2000

За даними таблиці 8.1 будуються графічні характеристики параметрів (рис. 8.1 – рис. 8.4).

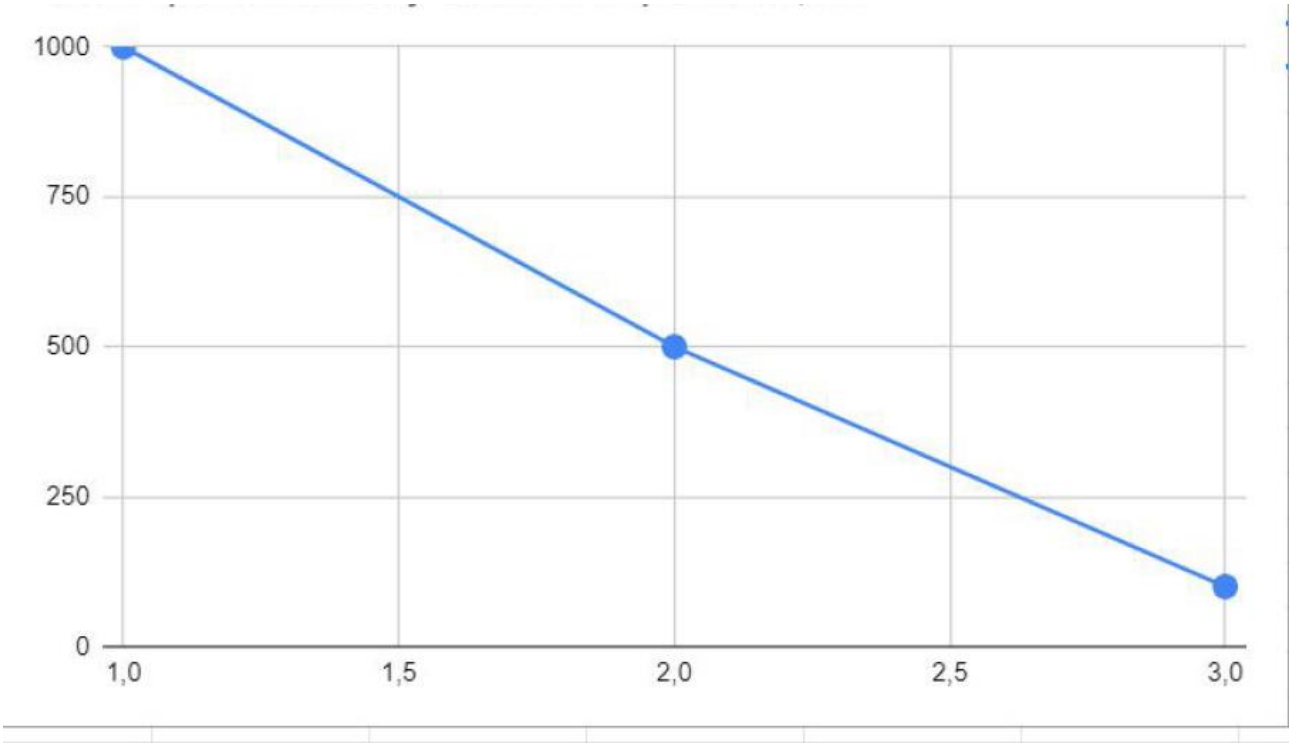


Рисунок 8.1 – X1, швидкодія системи

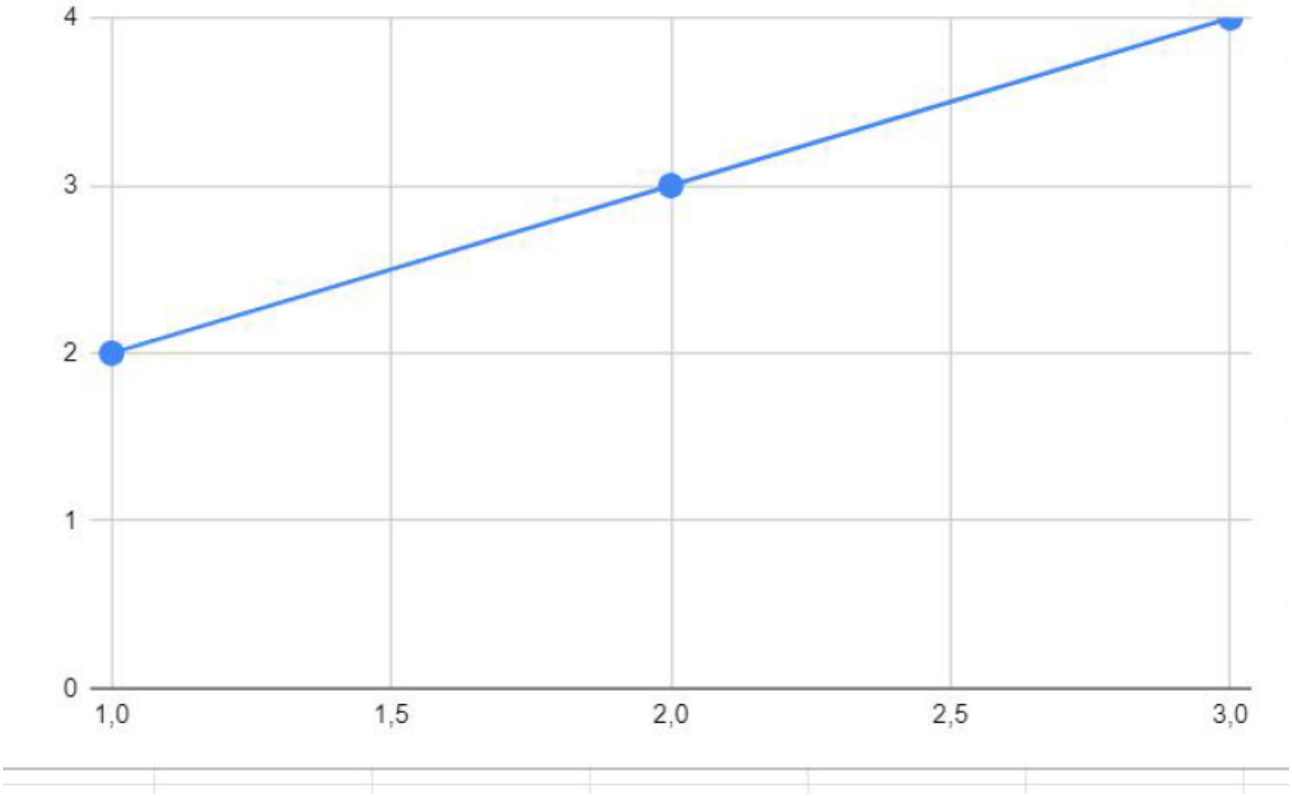


Рисунок 8.2 – X2, кількість додаткових службових компонентів

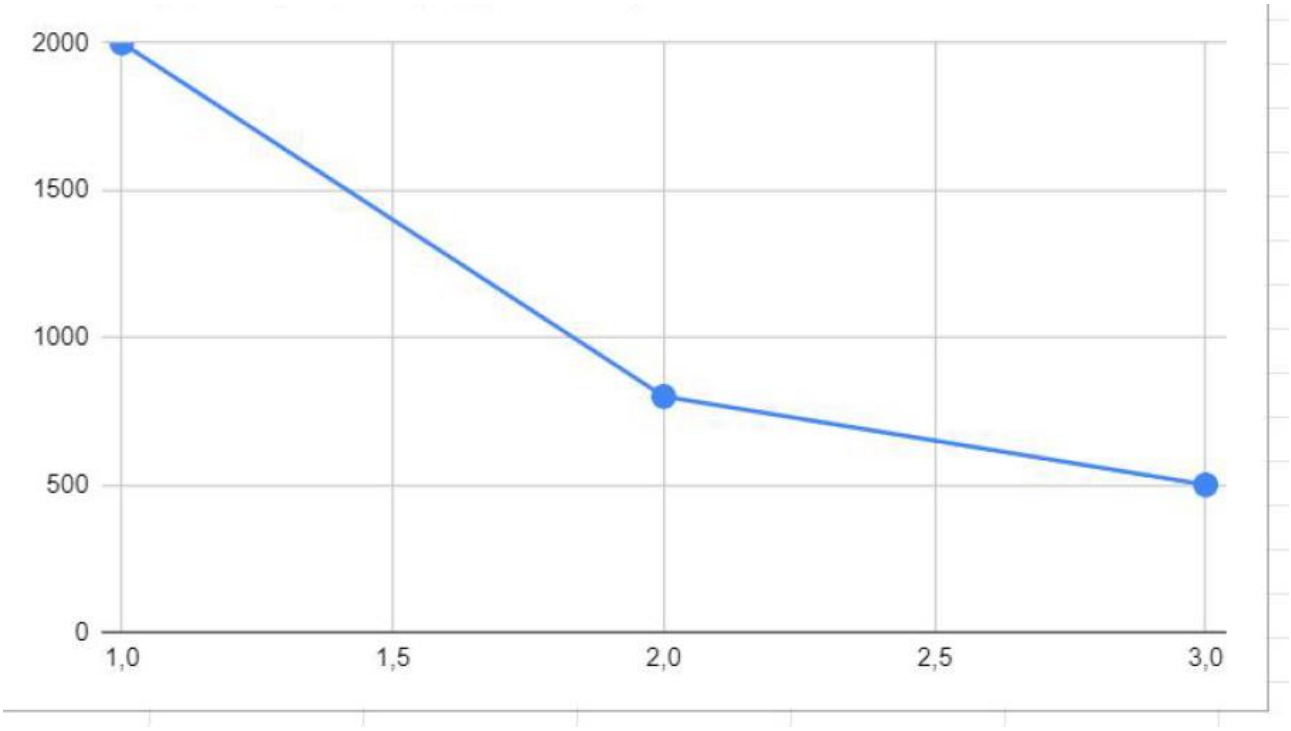


Рисунок 8.3 – X3, кількість доступних мов програмування

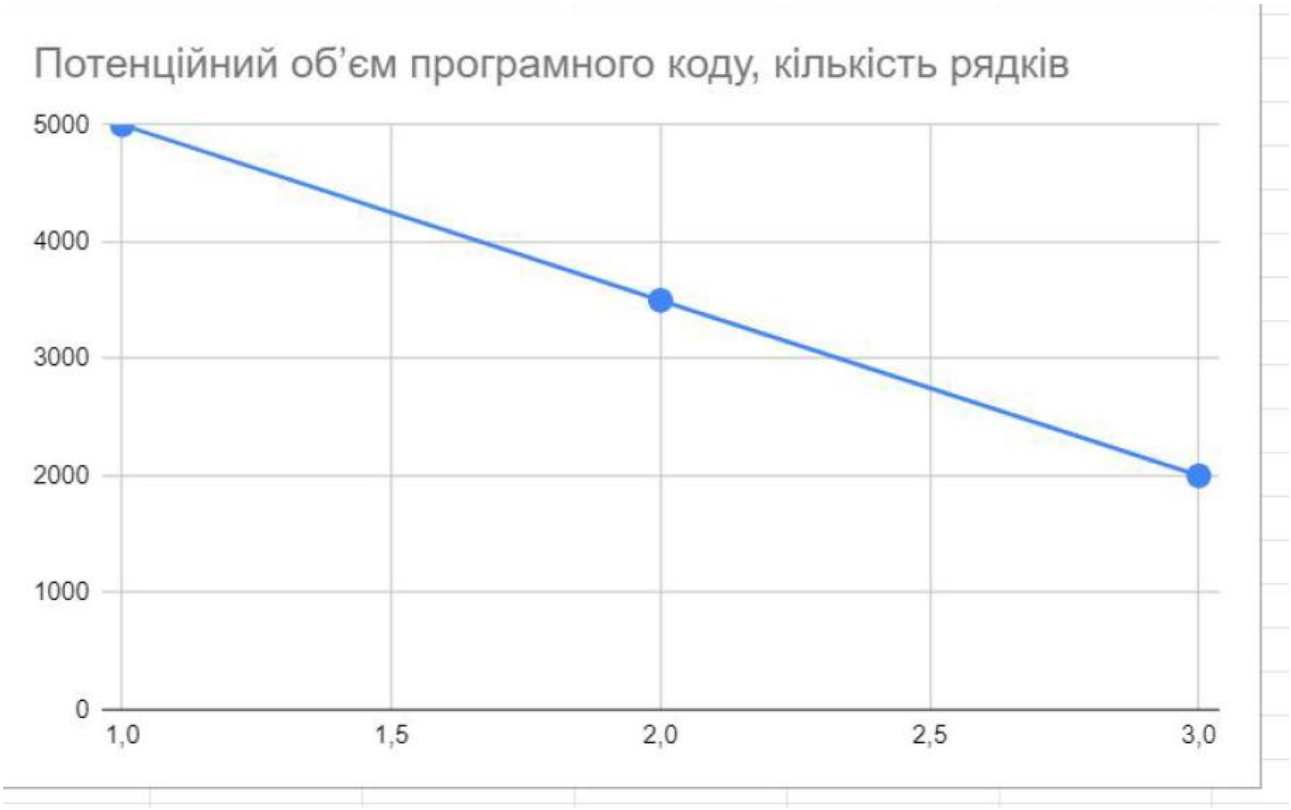


Рисунок 8.4 – X4, потенційний об’єм додаткового програмного коду

Таблиця 8.3 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Час обробки запиту	мс	4	4	4	4	4	4	4	28	10,5	110,25
X2	Кількість сервісів	К-ть	2	3	3	3	3	3	3	20	2,5	6,25
X3	Об’єм даних, що передаютьс я	Кб	1	2	1	1	1	1	1	8	-9,5	90,25

X4	Потенційний об'єм програмного коду	кількість рядків коду	3	1	2	2	2	2	2	14	-3,5	12,25
	Разом		10	10	10	10	10	10	19	70	0	219

Аналіз експертного оцінювання параметрів

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів:

$$R_i = \sum_{j=1}^{N} r_{ij} \ (8.1)$$

б) загальна сума рангів:

$$R_i = \frac{Nn(n+1)}{2} = 70 \ (8.2)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5 \ (8.3)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \ (8.4)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 219 \ (8.5)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 219}{7^2(4^3 - 4)} = 0{,}89 > W_k = 0{,}67 \ (8.6)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 8.5.

Таблиця 8.5 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1,5
X1 і X3	>	>	>	>	>	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	>	>	>	>	>	>	>	>	1,5
X2 і X4	<	>	>	>	>	>	>	>	1,5
X3 і X4	<	>	<	<	<	<	<	<	0,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \quad (8.7)$$

З отриманих числових оцінок переваги складемо матрицю $A=\|a_{ij}\|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$K_{\text{ві}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (8.8)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (8.9)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \quad (8.10)$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \quad (8.11)$$

.Як видно з таблиці 8.6, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 8.6 - Розрахунок вагомості параметрів

Параметриx_i	Параметриx_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1,0	1,5	1,5	1,5	5,5	0,344	21,25	0,36	77,875	0,36
X2	0,5	1,0	1,5	1,5	4,5	0,281	16,25	0,275	59,125	0,274
X3	0,5	0,5	1,0	0,5	2,5	0,156	9,25	0,157	34,125	0,158
X4	0,5	0,5	1,5	1,0	3,5	0,218	12,25	0,208	44,875	0,208
Всього:					16	1	59	1	216	1

8.2 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об'єм пам'яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 5.7):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} \,, \tag{8.11}$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 8.7 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X2	3	5	0,274	1,37
F2	А	X1	500	7	0,36	2,52
		X3	800	4	0,158	0,632
F3	А	X4	2000	6	0,208	1,248
	Б	X4			0,208	

За даними з таблиці 8.7 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \quad (8.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1{,}37 + 2{,}52 + 0{,}632 + 1{,}248 = 5{,}77,$$

$$K_{K2} = 1{,}37 + 2{,}52 + 0{,}632 + 1{,}04 = 5{,}562.$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

8.3 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Моделювання бізнес процесу;

2. Технічна реалізація бізнес процесу.

Завдання 1 за ступенем новизни відноситься до групи В, завдання 2 – до групи В. За складністю алгоритми, які використовуються в завданні 1 належать до групи 2; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як

$$T_О = T_Р \cdot К_П \cdot К_{СК} \cdot К_М \cdot К_{СТ} \cdot К_{СТ.М}, \quad (8.13)$$

де $T_Р$ – трудомісткість розробки ПП;

$К_П$ – поправочний коефіцієнт;

$К_{СК}$ – коефіцієнт на складність вхідної інформації;

$К_М$ – коефіцієнт рівня мови програмування;

$К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни В та групи складності алгоритму 2, трудомісткість дорівнює: $T_Р$ = 21 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт,

який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $К_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ} = 0.8$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$Т_1 = 21 \cdot 1.7 \cdot 0.8 = 28.56 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р = 12$ людино-днів, $К_П = 0.9$, $К_{СК} = 1$, $К_{СТ} = 0.8$:

$$Т_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$Т_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328{,}64 \text{ людино-годин.}$$

$$Т_{II} = (122.4 + 19.44 + 6.91 + 19.44) \cdot 8 = 1345.52 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 13500 грн., один аналітик в області даних з окладом 16500. Визначимо середню зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.}, \quad (8.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$С_Ч = \frac{13500 + 13500 + 16500}{3 \cdot 21 \cdot 8} = 86{,}30 \text{ грн.} \quad (8.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_Ч \cdot Т_i \cdot К_Д, \quad (8.16)$$

де $С_Ч$– величина погодинної оплати праці програміста;

$Т_i$ – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 86.30 \cdot 1328.64 \cdot 1.2 = 137593{,}95$ грн.

II. $С_{ЗП} = 86.30 \cdot 1345.52 \cdot 1.2 = 139342{,}51$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 137593{,}95 \cdot 0.22 = 30270{,}66$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 139342{,}51 \cdot 0.22 = 30655{,}25$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 10000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 13500 \cdot 0,2 = 32400 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 32400 \cdot (1 + 0.2) = 38800 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 38800 \cdot 0,22 = 8536 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 23000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.17 \cdot 0.25 \cdot 23000 = 6725 \text{ грн.},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.17 \cdot 23000 \cdot 0.05 = 1345 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} =(Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 12 – 12) \cdot 8 \cdot 0.93 =$$
$$= 1763.28 \text{ годин},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} =1763.28 \cdot 0,33 \cdot 3,51 \cdot 0,5 = 1021,81 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 20000 \cdot 0,67 =13400 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \qquad (8.17)$$

$C_{ЕКС}$ = 28800 + 6336 + 10589,76 + 5750 + 1150 + 2496,27 + 13400= 57932,27 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$C_{М\text{-}Г} = C_{ЕКС} / T_{ЕФ}$ = 57932,27 /1677.6 = 34,53 грн/год.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_М = C_{М-Г} \cdot T, \quad (8.18)$$

I. $C_М$ = 34,53 · 1328,64 = 45881,69 грн.

II. $C_М$ = 34,53 · 1345.52 = 46464,61 грн.

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0{,}67, \quad (8.19)$$

I. $C_Н$ = 102811,43 · 0,67 = 68883,66 грн.

II. $C_Н$ = 104117,62 · 0,67 = 69758,80 грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н, \quad (8.20)$$

I. $С_{ПП}$ = 102811,43 + 22618,51 + 45881,69 + 68883,66 = 240195,29 грн.

II. $С_{ПП}$ = 104117,62 + 22905,88 + 46464,61 + 69758,80 = 243246,91 грн.

8.4 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj} / С_{Фj}, \quad (8.21)$$

$К_{ТЕР1}$ = 4,25/ 240195,29 = 1,77 ·10^{-5},

$К_{ТЕР2}$ = 3,74/ 243246,91 = 1,53·10^{-5}.

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}$= 1,77 ·10^{-5}.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР}$ = 1,77 ·10^{-5}.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Dart;

– Використання моделей з великою ємністю

– Використання стандартного інтерфейсу візуалізації, швидкість розробки

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

Висновки до розділу 8

В даному розділі було проведено функціонально-вартісний аналіз програмного продукту, що буде створено. Процес аналізу складався з двохчастин.

У першій проведено дослідження програмного продукту з технічної точкизору: були поставлені основні параметри, що повинні бути головними при обранні кращої реалізації. На основі отриманих значень параметрів, оцінок експертів було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

У другій частині виконувалися обрахунки вартості розробки ПП з урахуванням витрат на заробітні плати, електроенергії, накладні витрати. Трудомісткість обох варіантів реалізації вважалась

однаковою, адже різниця полягала лише у виборі фреймворку.

Після виконання функціонально-вартісного аналізу програмного продуктущо розроблюється, можна зробити висновок, що перший варіант є найбільш оптимальним для реалізації (через більше значення коефіцієнту технічного рівня). Його показник техніко-економічного рівня якості $К_{ТЕР1}= 0,145 \cdot 10^{-3}$;

Цей варіант реалізації програмного продукту має такі параметри:

– UI-фреймворк Flutter;

– Підключення Firebase;

– Система Bitrise.

Даний варіант реалізації є найбільш продуктивним за рахунок використання більш оптимального для проекту фреймворку та системи CI/CD, а також надає користувачу зручний інтерфейс, гарну документацію, підтримку

ВИСНОВКИ

Управління проектом (УП) або (PM) – це наука і мистецтво керівництва та координації людських і матеріальних ресурсів на протязі життєвого циклу проекту шляхом застосування сучасних методів і техніки управління для досягнення певних результатів проекту за складом та обсягом робіт, коштами, годиною, якістю та задоволенням учасників проекту. Управління проектами дозволяє визначити цілі проекту та провести його обґрунтування, виявити структуру проекту, цілі, основні етапи роботи тощо, визначити необхідні джерела фінансування, підібрати виконавців через процедуру торгів та конкурсів, підготувати та закрити контракти, визначити терміни виконання проекту графік його реалізації та розрахувати ресурси, провести калькуляцію та аналіз витрат, спланувати та врахувати ризики, організувати реалізацію проекту у тому числі підібрати команду та забезпечити контроль за ходом виконання проекту.

Як можемо побачити наш аналіз показує, що жоден із цих інструментів не надає всієї функціональності чи функцій.

Деякі з цих функцій є майже у всіх інструментах, навіть у тих що не були представлені в роботі.
До них відносяться планування завдань, управління ресурсами, спільна робота та управління документами. Крім того, два інструменти мають веб-інтерфейс. Хотів б відзначити, що існує також досить багато програмних засобів управління проектами з відкритим вихідним кодом, доступних практикам. Два інструменти, які ми проаналізували, є відкритими кодами або не вимагають ліцензії. Деякі функціональні можливості вимагають подальшого вивчення. До них відносяться управління змінами, метод розробки процесів та управління

якістю. У цьому дослідженні ми просто визначили, чи ці інструменти підтримують перелічені функції або функціональні можливості певною мірою. Звичайно, якість функціональності, що забезпечується цими інструментами, варіюється, і вважаємо, що це питання потребує подальшого вивчення.

Саме тому це дослідження дозволяє керівникам проектів та членам команди швидко отримати уявлення про інструменти, що підлягають дослідженню, та оцінити їх сильні та слабкі сторони. Крім того, це прагнення допомогти керівникам проектів у виборі відповідних PMST, надаючи набір критеріїв. Це дослідження є одним із кроків при виборі бажаного PMST. Крім того, ми хотіd б визначити потреби у програмному засобі управління проектами. Виявлення цих потреб допоможе розробникам інструментів. Ще одним напрямком дослідження може стати розробка основи для PMST яка би брала в себе найкраще з представлених тестових PMST.

ПЕРЕЛІК ПОСИЛАНЬ

1. SWEBOK: Керівництво до зводу знань з прогамної інженерії [Електронний ресурс] - Режим доступу до ресурсу: https://infopedia.su/4x32bf.html

   Дата доступу - 27.04.2022
2. Програмна інженерія – К.М.Лавріщев, Київ 2008 [Електронний ресурс] – Режим доступу до ресурсу: http://www.immsp.kiev.ua/postgraduate/Biblioteka_trudy/Lavrishcheva_Progr.ingeneriia2008.pdf

   Дата доступу - 27.04.2022
3. Життєвий цикл програмного забезпечення[Електронний ресурс] - Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Життэвий_цикл_програмного_забезпечення

   Дата доступу - 27.04.2022
4. Конспект лекцій для студентів спеціальності 125 «Кібербезпека», Луцьк 2016 [Електронний ресурс] - Режим доступу до ресурсу: http://compi.com.ua/konspekt-lekcij-dlya-studentiv-specialenosti-125-kiberbezpeka-v2.html?page=2

   Дата доступу - 27.04.2022
5. Структура і зміст SWEBOK [Електронний ресурс] - Режим доступу до ресурсу: https://studopedia.com.ua/1_280644_struktura-i-zmist-SWEBOK

   Дата доступу - 30.04.2022
6. Вимоги до ПО (Softwate Requirements) [Електронний ресурс] - Режим доступу до ресурсу: http://um.co.ua/3/3-6/3-66867.html

   Дата доступу - 30.04.2022

7. Звід знань з управління PMBoK [Електронний ресурс] - Режим доступу до ресурсу: http://ni.biz.ua/8/8_3/8_33100_svod-znaniy-po-upravleniyu-proektami-PMBoK.html
   Дата доступу - 30.04.2022
8. Навчальне електронне видання ОСНОВИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ для студенту напряму «Програмна інженерія» [Електронний ресурс] - Режим доступу до ресурсу: http://irttri.com/OPI/OPI_konspekt.doc
   Дата доступу - 30.04.2022
9. Как составить план проекта за 8 простых шагов [Електронний ресурс] - Режим доступу до ресурсу: https://www.wrike.com/ru/blog/kak-sostavit-plan-proekta-za-vosem-prostyh-shagov/

   Дата доступу - 30.04.2022
10. 6 Simple Steps to start any project [Електронний ресурс] - Режим доступу до ресурсу: https://www.brightwork.com/blog/6-steps-to-start-any-project
    Дата доступу - 30.04.2022

11. Top 10 Project Management Methodologies: An Overview [Електронний ресурс] - Режим доступу до ресурсу: https://www.projectmanager.com/blog/project-management-methodology

    Дата доступу - 08.05.2022

12. 9 Of The Most Popular Project Managemen Methodologies Made Simpe [Електронний ресурс] - Режим доступу до ресурсу: https://thedigitalprojectmanager.com/project-management-methodologies-made-simple/

    Дата доступу - 08.05.2022
13. Kanban [Електронний ресурс] - Режим доступу до ресурсу: https://www.atlassian.com/ru/agile/kanban
    Дата доступу - 08.05.2022

14. Agile [Електронний ресурс] - Режим доступу до ресурсу: https://www.atlassian.com/ru/agile

Дата доступу - 08.05.2022

15. Scrum [Електронний ресурс] - Режим доступу до ресурсу: https://www.atlassian.com/ru/agile/scrum/

Дата доступу - 09.05.2022

16. Functional Testing[Електронний ресурс] - Режим доступу до ресурсу: https://www.javatpoint.com/functional-testing

Дата доступу - 09.05.2022

17. Non Fuctional Testing [Електронний ресурс] - Режим доступу до ресурсу: https://www.perfecto.io/blog/what-is-non-functional-testing

Дата доступу - 09.05.2022

18. Functional Testing VS Non Functional Testing [Електронний ресурс] - Режим доступу до ресурсу: https://www.softwaretestinghelp.com/functional-testing-vs-non-functional-testing/

Дата доступу - 09.05.2021

19. Microsoft Project [Електронний ресурс] - Режим доступу до ресурсу: https://www.microsoft.com/en-us/microsoft-365/project/project-management-software

Дата доступу -10.05.2022

20. OpenProject [Електронний ресурс] - Режим доступу до ресурсу: https://www.openproject.org/

Дата доступу -10.05.2022

21. Test documentation in SoftWare Testing [Електронний ресурс] - Режим доступу до ресурсу: https://www.guru99.com/testing-documentation.html

Дата доступу -10.05.2022

22. How to use Microsoft Project – With Linkedln Learning [Електронний ресурс] - Режим доступу до ресурсу: https://www.linkedin.com/learning/subscription/topics/microsoft-project?src=go-pa&trk=sem-ga_campid=15440322604_asid=130576405379_crid=565757955339_kw=use%20microsoft%20project_d=c_tid=kwd-300801887229_n=g_mt=b_geo=1012859_slid=&mcid=68737459009514946 67&cid=&gclid=CjwKCAjwtcCVBhA0EiwAT1fY77F1m0sUVgSgzY6CwWIZinIc-98fRGuer11Rc8Amhk_ZOO8Ma9NjuRoCfKcQAvD_BwE&gclsrc=aw.ds

Дата доступу - 18.05.2022

23. Gantt Project – Free Management Application [Електронний ресурс] – Режим доступу до ресурсу: https://www.ganttproject.biz

Дата доступу - 20.05.2022