Посилання на GitHub:

https://github.com/Oxanka/ProjectPracticalWork.git

1. Функціональні вимоги
   1. Формування словників предметної області:

**Бизнес-требования** (business requirement) — высокоуровневая бизнес-цель организации, которая строит продукт, или клиента, который приобретает продукт.

**Организационная структура** (Organizational structure) — документ, схематически отражающий состав и иерархию подразделений предприятия. Организационная структура устанавливается исходя из целей деятельности и необходимых для достижения этих целей подразделений, выполняющих функции, составляющие бизнес-процессы организации

**Система автоматизации документооборота, система электронного документооборота (СЭД)** — автоматизированная многопользовательская система, сопровождающая процесс управления работой иерархической организации с целью обеспечения выполнения этой организацией своих функций. При этом предполагается, что процесс управления опирается на человеко-читаемые документы, содержащие инструкции для сотрудников организации, необходимые к исполнению.

**Предприя́тие** — самостоятельный, организационно-обособленный хозяйствующий субъект с правами юридического лица, который производит и сбывает товары, выполняет работы, оказывает услуги.

**Документооборот** – это движение документов в организации с момента их создания или получения до завершения исполнения или отправки. Следует отметить, что в этом определении упор делается на словах "движение документов", то есть их пути из одного подразделения или от одного сотрудника к другому.

**Производственная структура предприятия** – это состав и взаимосвязь производственных подразделений предприятия.

* 1. Формування багаторівневої класифікації функціональних вимог:

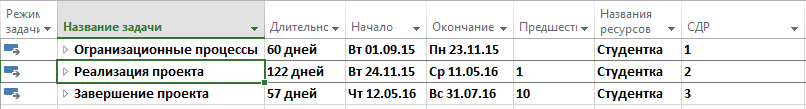
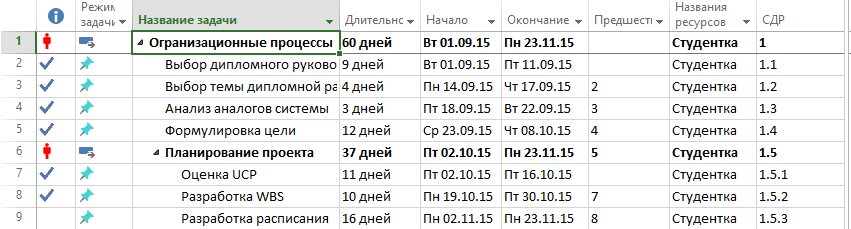
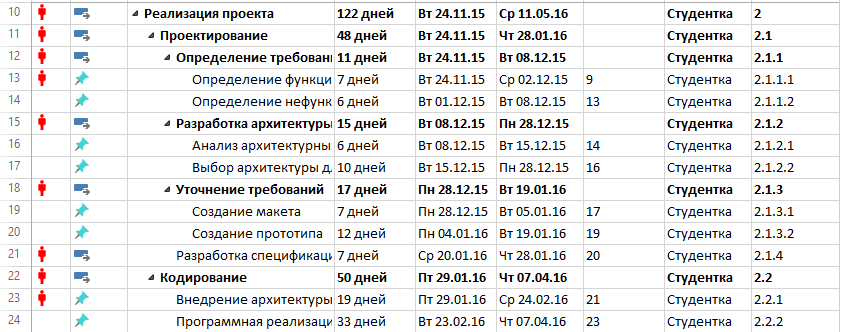
Зобразимо структуру декомпозиції робіт в згорнутому вигляді (рис 2.1):

Рисунок 2.1 – Структура декомпозиції робіт в згорнутому вигляді

Зобразимо структуру декомпозиції робіт в розгорнутому вигляді (рис. 2.2):



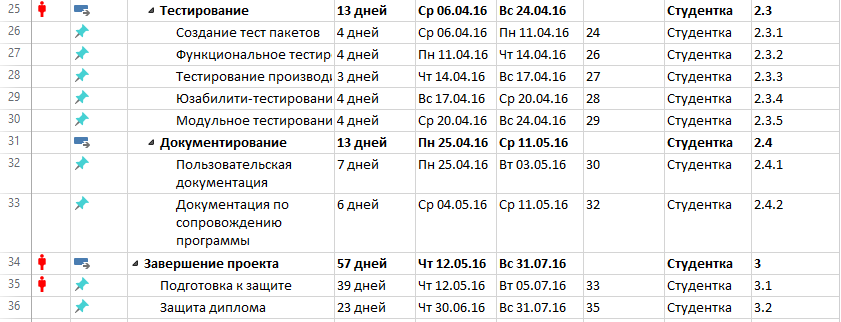


Рисунок 2.2 – Структура декомпозиції робіт в розгорнутому вигляді

Розглянемо час виконання кожної роботи.

Організаційні процеси займають 60 днів, тому що для правильного написання дипломної роботи необхідно проаналізувати систему, що розробляється, проаналізувати аналоги, коректність реалізацій функцій в аналогах, побудувати план виконання проекту. Бо якщо організаційні роботи будуть виконані з помилкою, то доведеться дуже часто повертатися до цього етапу і змінювати всю систему.

Після організаційних процесів настає реалізація проекту.

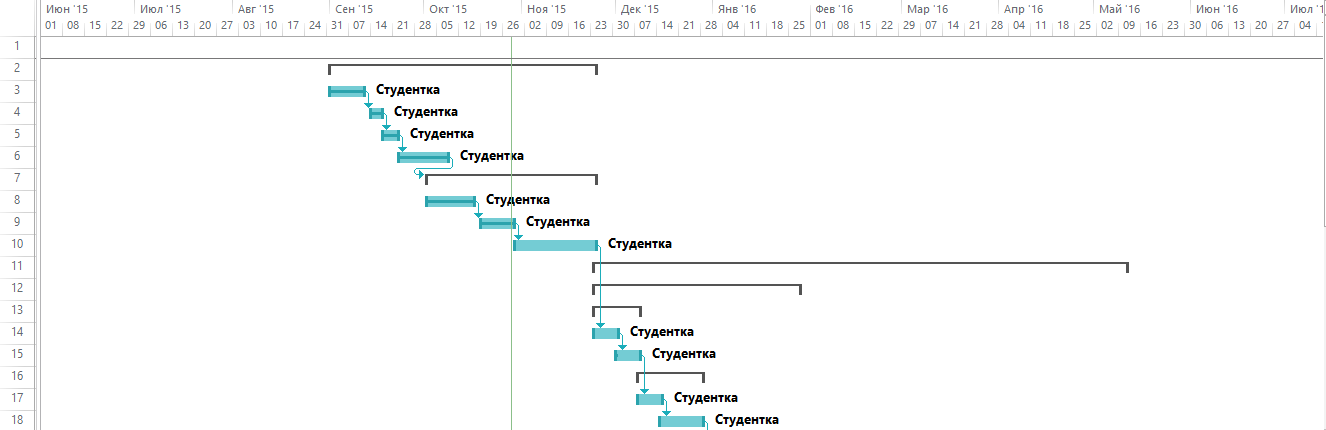
У реалізацію проекту виходить проектування, кодування, тестування і документування. Проектування займає 48 днів, так як необхідно розробити архітектуру проекту, створити макет і прототип для коректної розробки програмного продукту.

На кодування вистачить і 50 днів, так як був уже створений прототип програмного продукту, який показує весь функціонал, а так само описана вся архітектура.

Після того, як проект закодований можна переходити на етап тестування, для цього вистачить 13 днів, так як процес тестування проходить вручну і вимагає опису, а так же самого тесту програмного продукту.

Після того, як продукт був налагоджений можна робити документування проекту, на це вистачить 13 днів, в документації потрібно включити результати тестування, створити документацію до програми.

На діаграмі Ганта (рис 2.3) показано тимчасову шкалу виконання проекту.



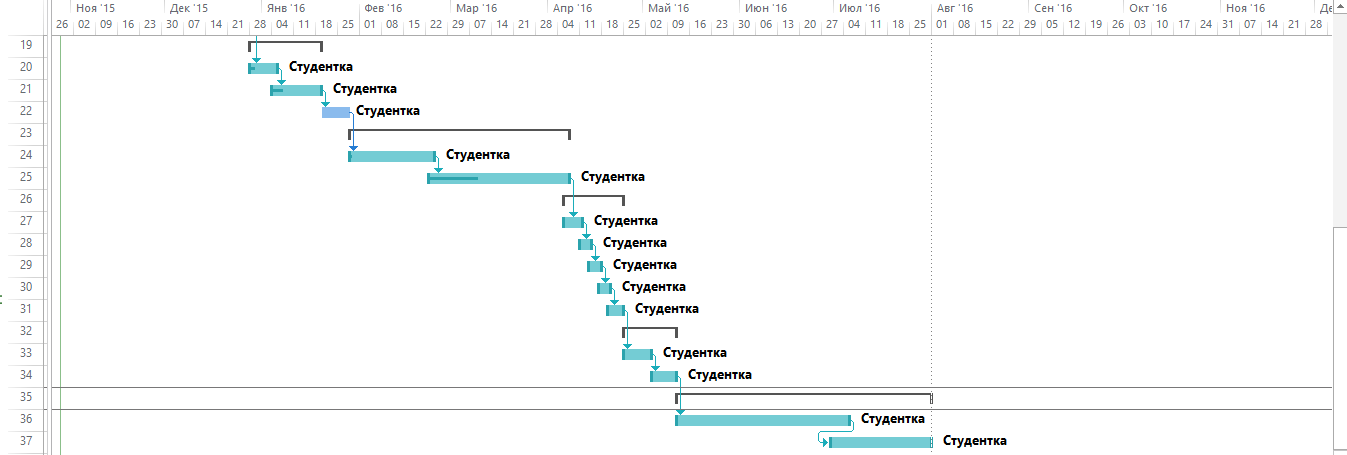
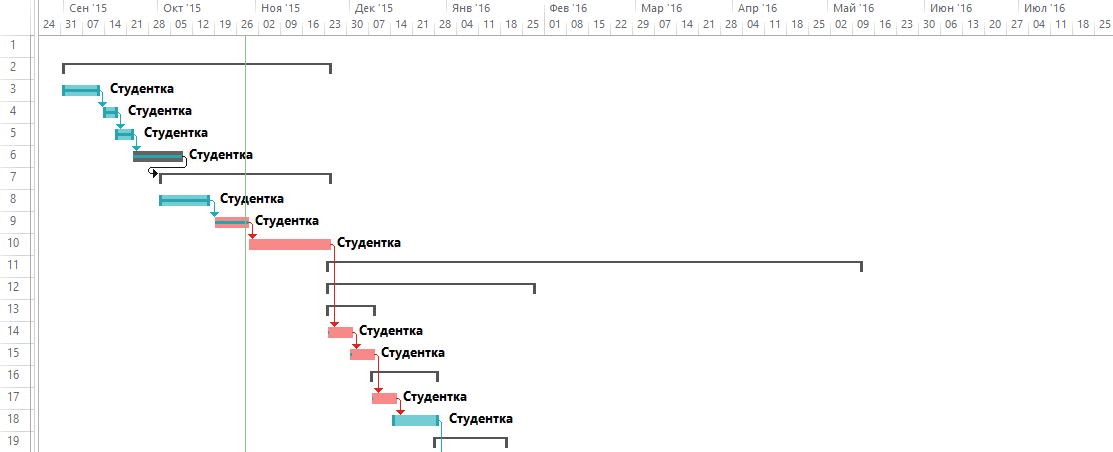


Рисунок 2.3 – Діаграма Ганта

Побудує структуру декомпозиції робіт, можна зробити висновок, що при аналізі UDP, були враховані не всі фактори, які впливають на тривалість робіт. Були втрачені частина організаційних робіт, оформлення документацій і завершальна стадія. А це трохи збільшує загальний час, необхідний на розробку. Але так, приблизно правильно було підраховано час на етапі підрахунку UDP.



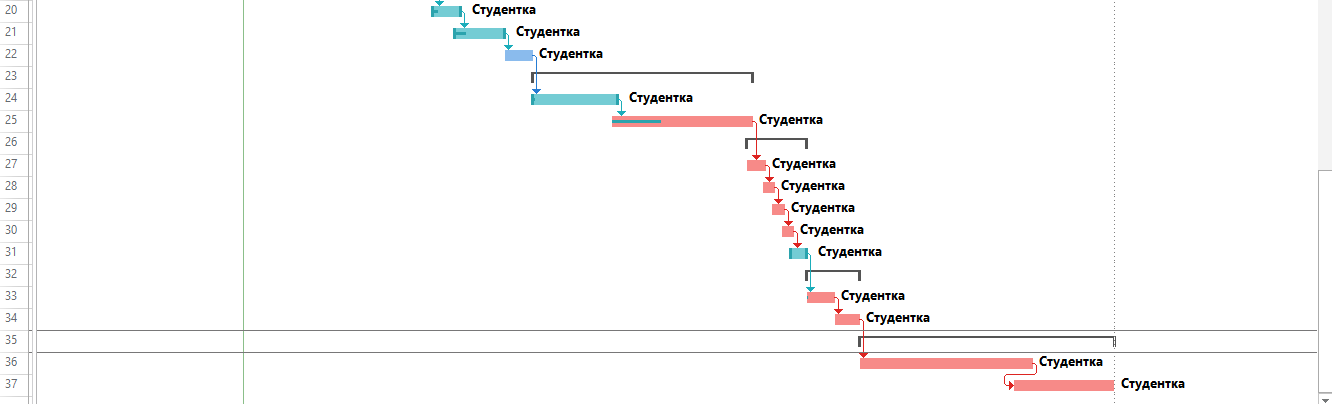


Рисунок 2.4 – Критичний шлях

* 1. Аналіз ризиків

Побудуємо діаграму Ішикава для всього проекту (рис 2.5). Дана діаграма показує чинники, що впливають на низьку оцінку за проектом.

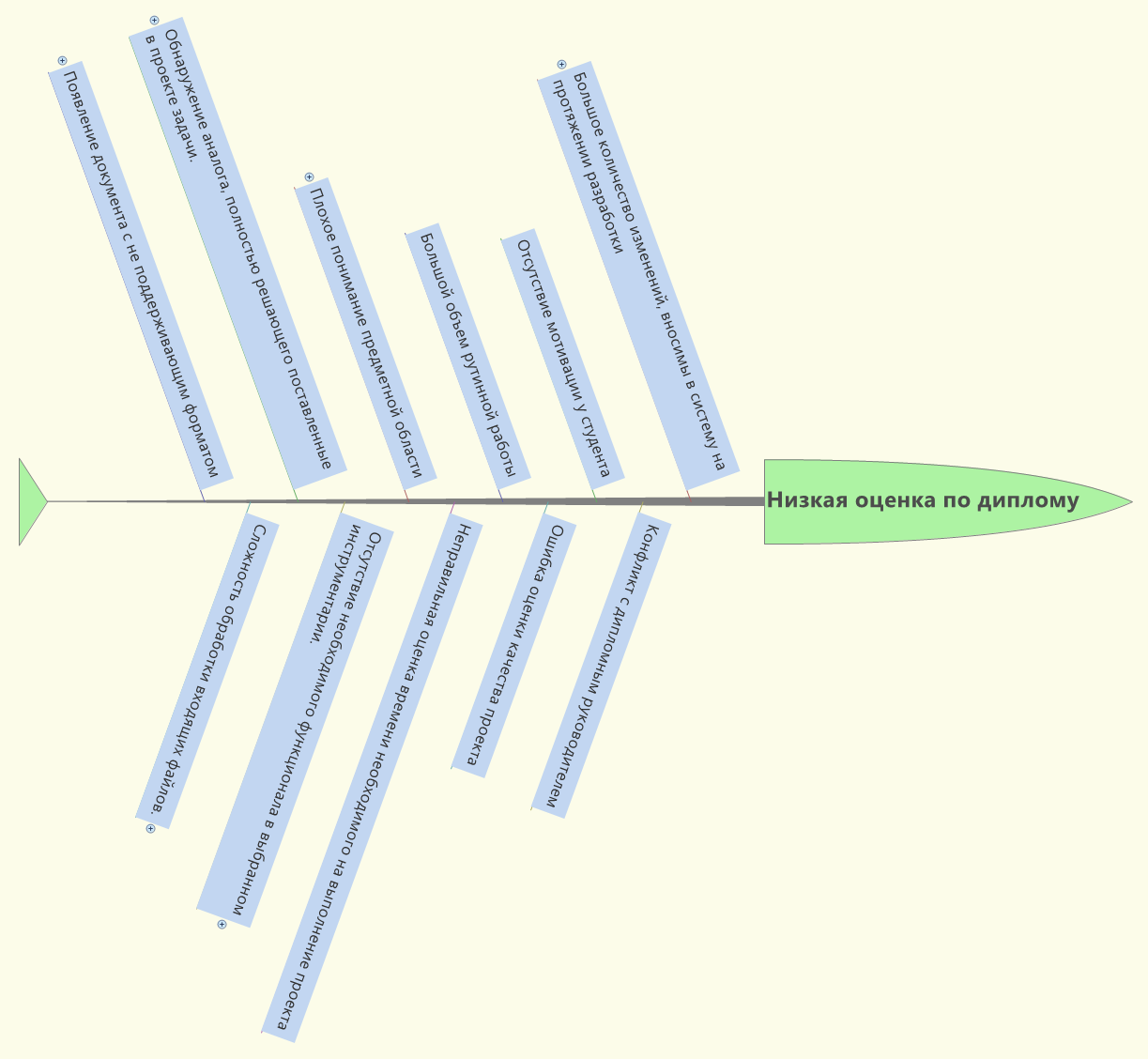


Рисунок 2.5 – Діаграма Ішикава

Розглянемо детально всі фактори, що впливають на проект, а також їх під причини.

Складність обробки вхідних файлів (2.6).

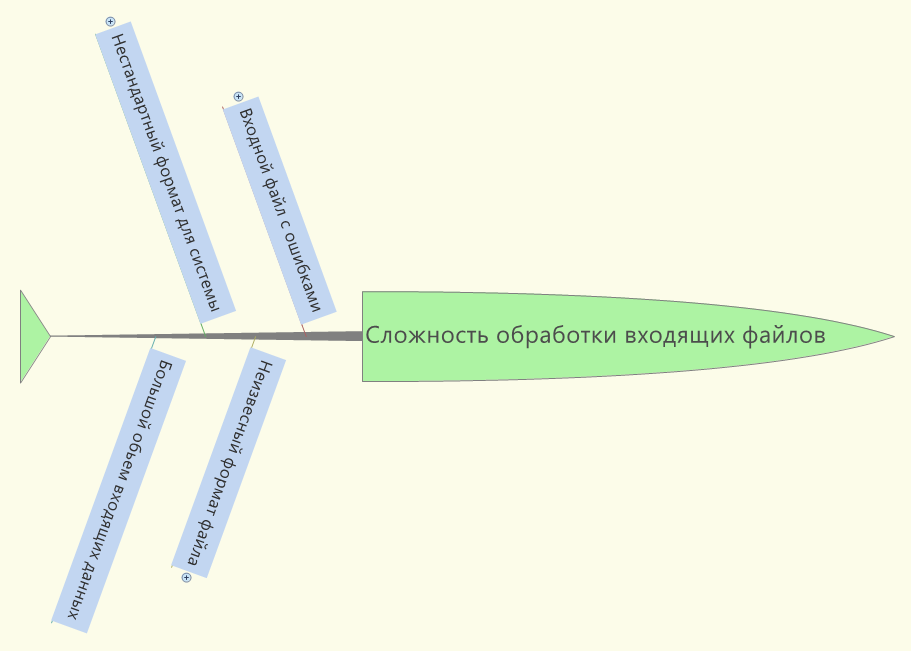


Рисунок 2.6 – Складність обробки вхідних файлів

Поява документа з не підтримуючим форматом (рис. 2.7)

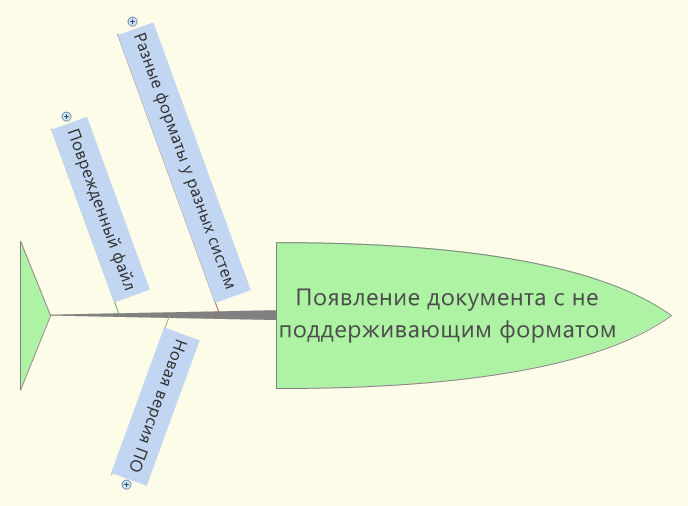


Рисунок 2.7 – Поява документа з не підтримуючим форматом

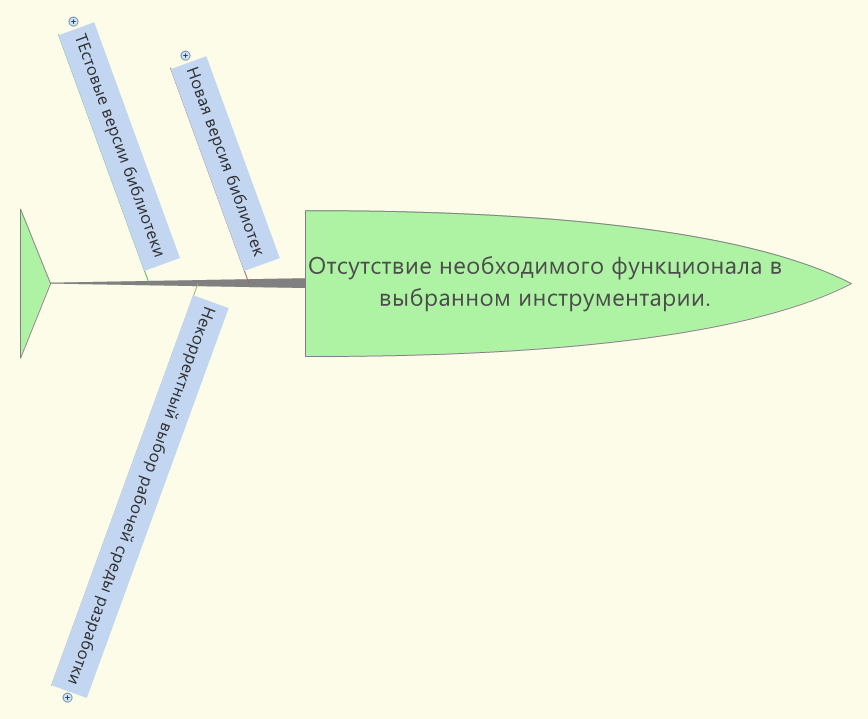
Відсутність необхідного функціоналу в обраному інструментарії (рис. 2.8)

Рисунок 2.8 – Відсутність необхідного функціоналу в обраному інструментарії

Виявлення аналога, повністю вирішального поставлені в проекті завдання (рис. 2.9)

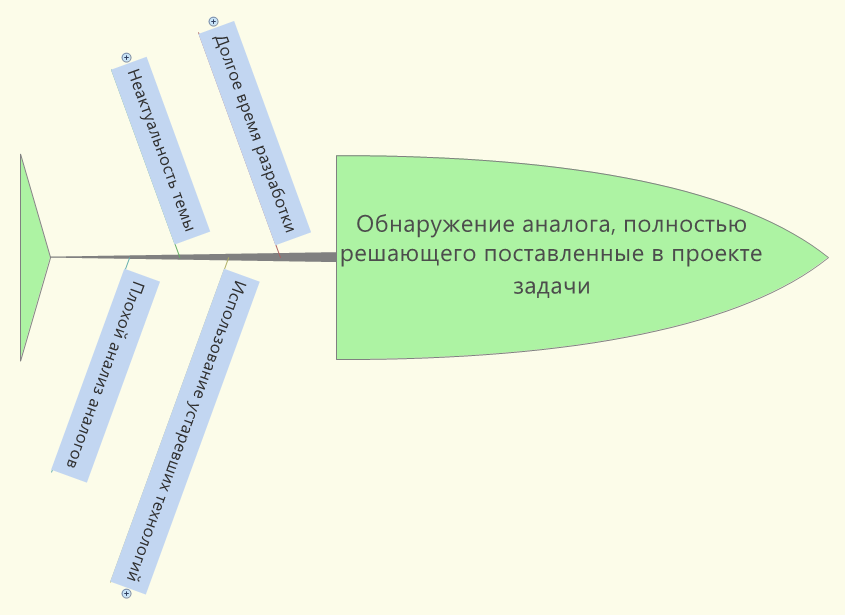


Рисунок 2.9 – Виявлення аналога, повністю вирішального поставлені в проекті завдання

Велика кількість змін, що вносяться до системи протягом розробки (рис.2.10)

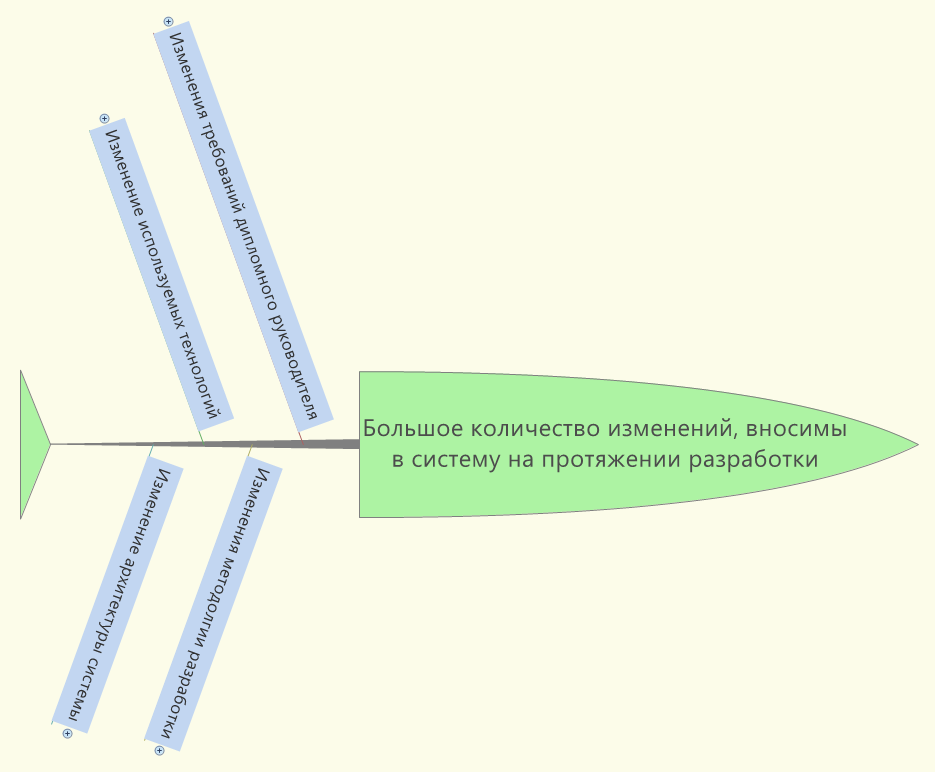


Рисунок 2.10 – Велика кількість змін, що вносяться до системи протягом розробки

Погане розуміння предметної області (рис. 2.11)

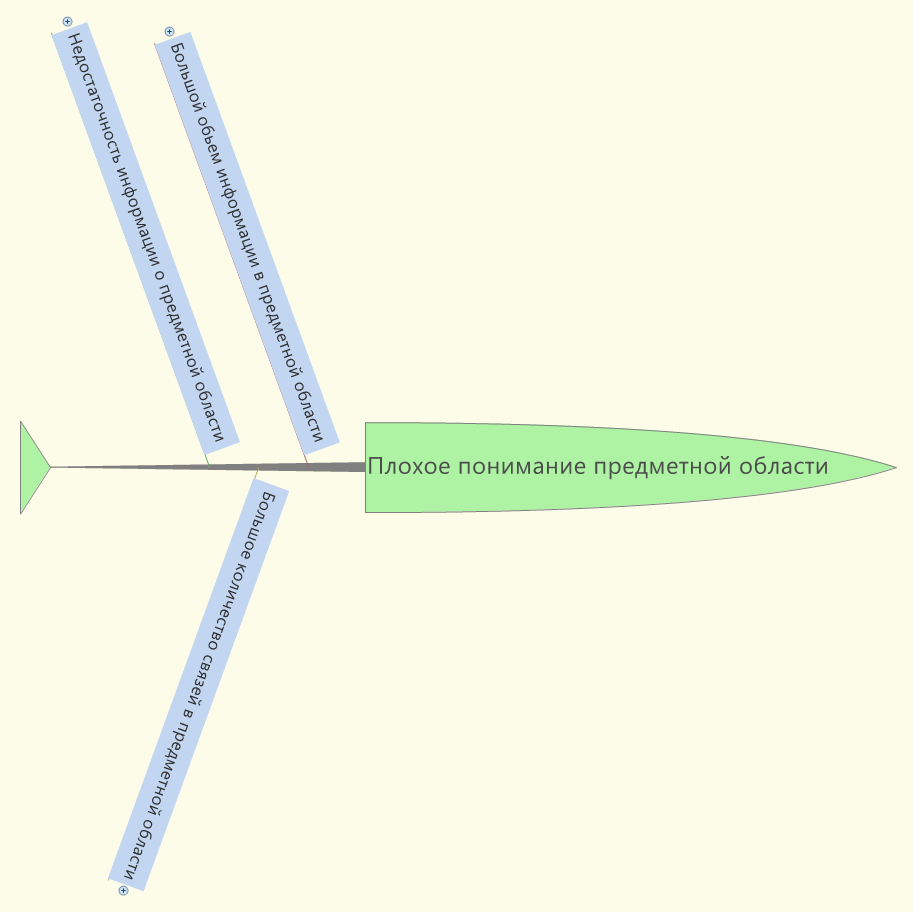


Рисунок 2.11 – Погане розуміння предметної області

Великий обсяг рутинної роботи (рис. 2.12)

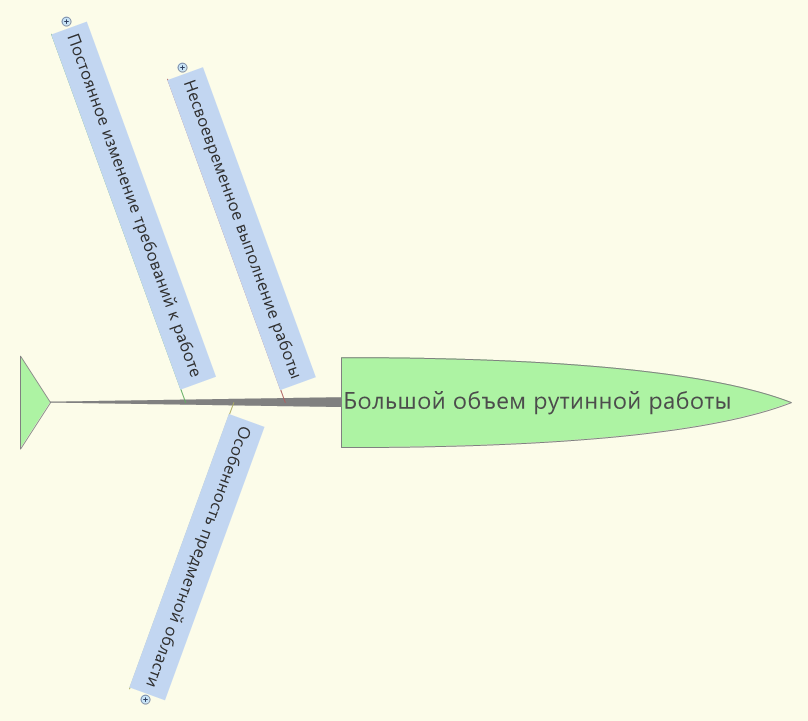


Рисунок 2.12 – Великий обсяг рутинної роботи

Неправильна оцінка часу необхідного на виконання проекту (рис. 2.13)

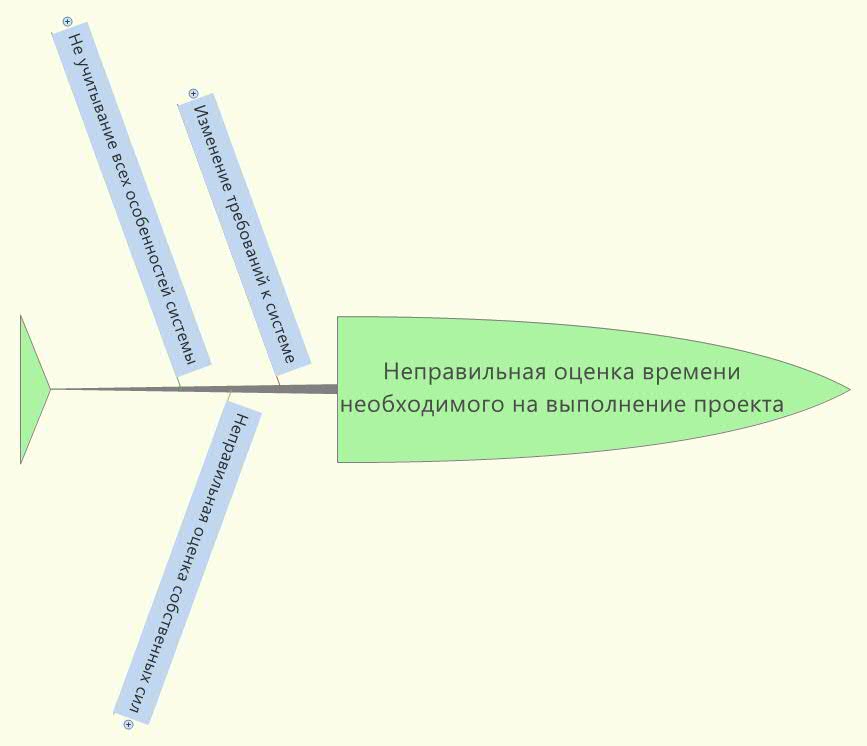


Рисунок 2.13 – Неправильна оцінка часу необхідного на виконання проекту

Відсутність мотивації у студента (рис. 2.14)

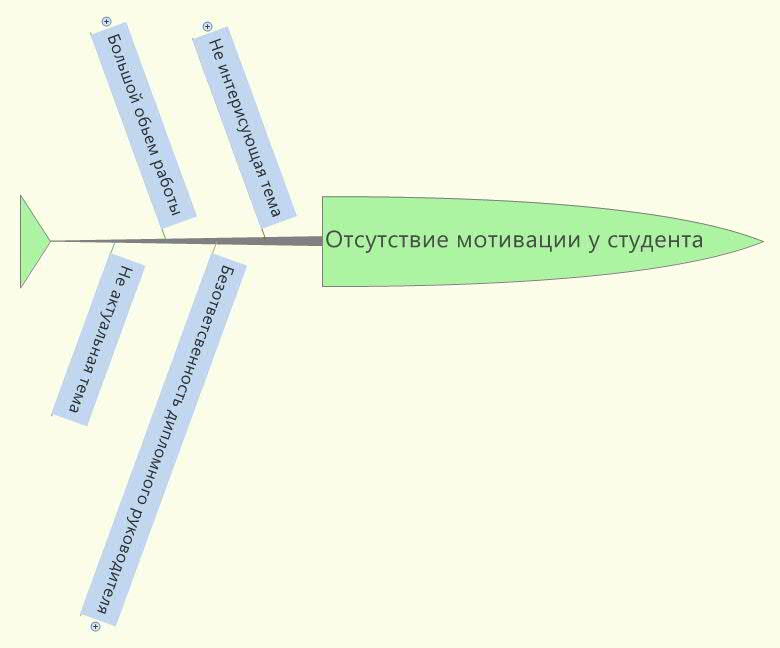


Рисунок 2.14 – Відсутність мотивації у студента

Помилка оцінки якості проекту (рис. 2.14)

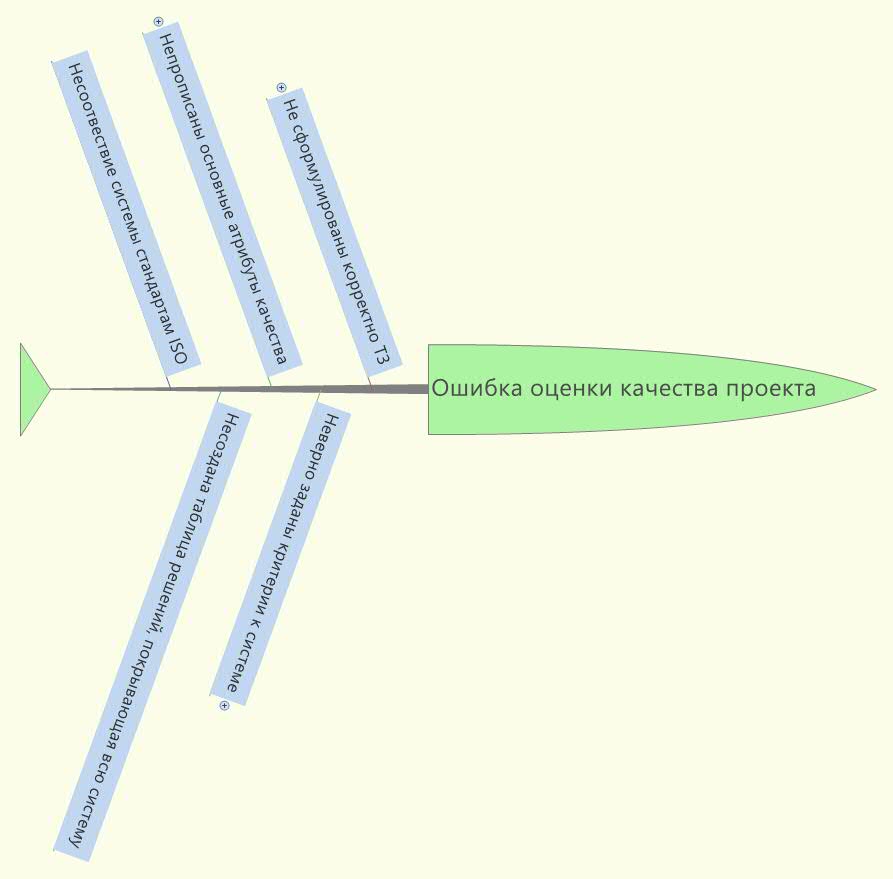


Рисунок 2.14 – Помилка оцінки якості проекту

Конфлікт з дипломним керівником (рис. 2.15)

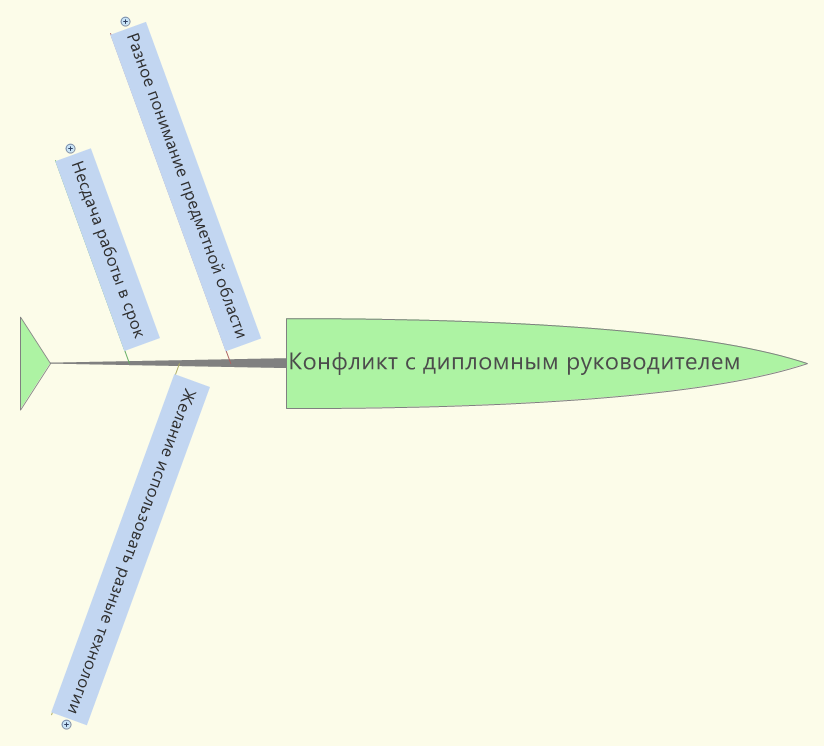


Рисунок 2.15 – Конфлікт з дипломним керівником

На основі побудованих діаграм, складемо таблицю ризиків нашого проекту (а саме дипломної роботи) (табл. 2.8)

Таблиця 2.8 – Ризики проекту

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Риск | Ймовірність | Вплив | Ранг |
| 1 | Складність обробки вхідних файлів. | 3 | 3 | 9 |
| 2 | Поява документа з не підтримують форматом | 2 | 3 | 6 |
| 3 | Відсутність необхідного функціоналу в обраному інструментарії. | 1 | 3 | 3 |
| 4 | Виявлення аналога, повністю вирішального поставлені в проекті завдання. | 1 | 3 | 3 |
| 5 | Велика кількість змін, що вносяться до системи протягом розробки | 2 | 1 | 2 |
| 6 | Погане розуміння предметної області | 2 | 3 | 6 |
| 7 | Великий обсяг рутинної роботи | 2 | 2 | 4 |
| 8 | Неправильна оцінка часу необхідного на виконання проекту | 2 | 3 | 6 |
| 9 | Відсутність мотивації у студента | 1 | 2 | 2 |
| 10 | Помилка оцінки якості проекту | 2 | 2 | 4 |
| 11 | Конфлікт з дипломним керівником | 1 | 3 | 3 |

Найвпливовіші ризики: складність обробки вхідних файлів, поява документа з не підтримують форматом, погане розуміння предметної області, неправильна оцінка часу необхідного на виконання проекту, помилка оцінки якості проекту.

* 1. Функціональний аналіз існуючих продуктів:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Отображение документооборота | Построение иерархии отделов | Отображение связей между отделами | Большое количество вложений | Возможность добавления комментариев к связям | Сохранение схемы |
| C4ITS:редактор орг структуры | + | + | + |  |  | + |
| ARIS |  | + | + | + | + | + |
| BPwin |  | + |  | + | + | + |
| Rational Rose |  | + | + |  |  | + |
| Fox Manager |  | + |  | + |  | + |
| Business Studio |  | + | + | + | + | + |

* 1. MoSCoW принцип розташування пріорітетів:

M: Збереження файлу у файл формату XML

S: Конвертація організаційної структури у граф специфікацій вимог

C: Можливість запуску на різних платформах

W: Платний продукт

1. Нефункціональні вимоги:
   1. Інтерфейс користувача:

Після запуску програмного продукту(рис. 2.1), користувач меню, за допомогою якого може створювати організаційну структуру підприємства, а також панель для візуалізації організаційної структури підприємства.

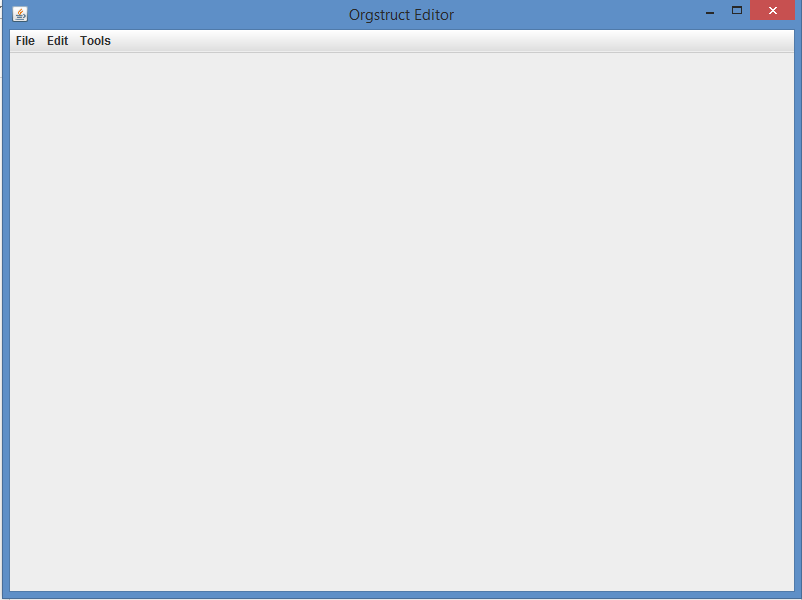


Рисунок 2.1 – Інтерфейс програмного продукту

Вибравши пункт меню File, та обравши підпункт меню Open користувач може завантажити збережену схему організаційної структури підприємства. На рис. 2.2 зображено даний процес.

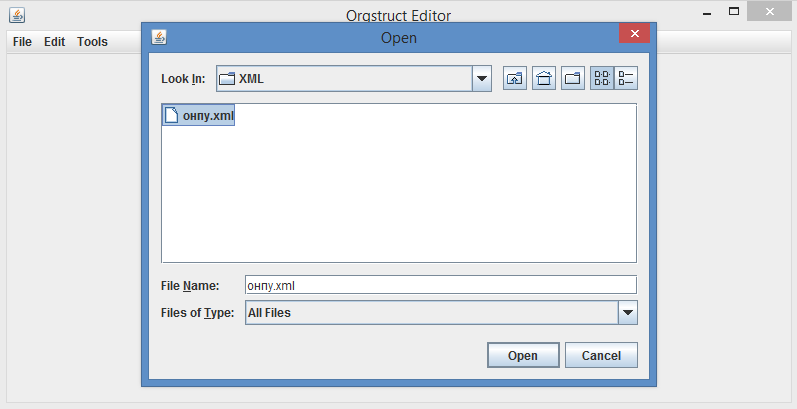


Рисунок 2.2– Вікно відкриття xml-схеми

Після завантаження xml-схеми користувач бачить організаційну структурі підприємства, я ку йому передали, або він зберіг заздалегідь.

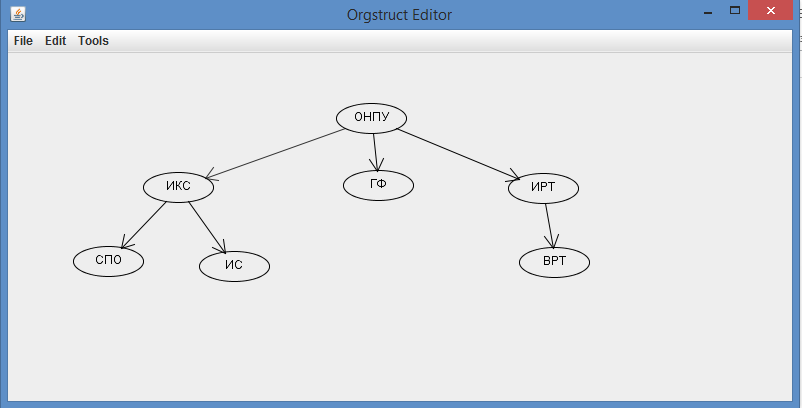


Рисунок 2.3 – Візуалізація організаційної структури підприємства

2.2 Атрибути якості:

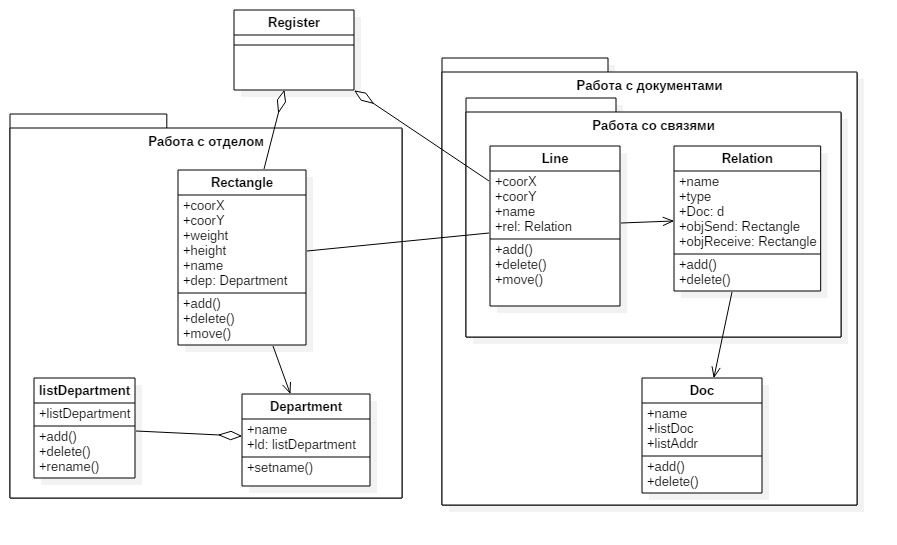
Таблиця 2.1 – Атрибути якості

Таблиця 1.2. Нефункціональні вимоги

| Набір характеристик (за ISO 9126) | Властивості програмного забезпечення |
| --- | --- |
| Функціональність | * Здатність до взаємодії (interoperability): Програмний продукт повинен взаємодіяти з Сховищем даних, для отримання інформації в побудові організаційної структури підприємства * Захищеність (security): Дані у БД захищені розділенням прав користувачів (тільки замовник може вносити зміни) |
| Ефективність | * Часові характеристики (time behaviour): Збереження схеми, повинно проводиться < чим через 15 секунд; Доданий новий елемент на схему відобразиться < чим через 5 секунд * Використання ресурсів (resource utilisation): Програма не повинна використати більше 500 МБ. |
| Надійність | * Здатність до відновлення працездатності при відмовах (recoverability): Програма повинна автоматично зберігати дані кожні 10 хвилин роботи і у разі збою програми запустити останній збережений варіант. |
| Переносимість (мобільність) | * Адаптованість (adaptability): Програма буде написана на java, підтримуватиме кросплатформовість, її можна буде запускати на різних платформах. |

* 1. Планування розробки:
  2. Архітектура програмної системи:

Проаналізував предметну область, діаграму пакетів, сценарії та вимоги до системи можна побудувати діаграму класів(рис. 3.13), яка являється основою програмного коду, та допоможе відобразити усі вимоги до системи.

Рисунок 3.13 – Діаграма класів

На діаграму класів є присутнім клас Register для взаємодії з усіма елементами систем.

Клас Rectangle містить в собі екземпляр класу Department, координати, розміри (висоту і ширину), назва. Екземпляр класу можна додати, перейменувати і видалити.

Клас Relation містить в собі зв'язки між відділами (класом Rectangle) і перелік документів, що відносяться до цього зв'язку. Містить в собі ім'я об'єкту посилача, ім'я об'єкту приймача, тип зв'язку і назву зв'язку. Зв'язок можна додати і видалити.

Клас Department містить в собі перелік відділів, які відображатимуться у блоці Rectangle, а так само перелік все існуючих відділів. Назву відділу можна перейменувати.

Клас Doc містить в собі список документів а так само шлях до місця розташування, які відносяться до якого-небудь зв'язку.

Клас listDepartment містить в собі список можливих підпорядкованих класів. Екземпляр класу можна додати, перейменувати і видалити.

* 1. Технології розробки:
* Операційні системи: Windows, Linux, Mac OS( але обов’язковою умовою являється наявність встановленої Java версії не нижче Java 1.7).
* Структури даних: файли формату XML для діаграм, бінарні файли для збереження діаграм та для переведення діаграм у файл формату XML.
* Інструменти для розробки: IntelliJ IDEA Community Edition 15.0.2 для самого програмного продукту, Sublime Text 3 для створення DTD опису файлу XML, git та github для системи контролю версій.
  1. План розробки:

Оцінювання тривалості розробки пропонується виконувати з використанням методики оцінки проектів на основі варіантів використання UCP (Use Case Points). В основі методики UCP лежить FPA (Functional Point Analysis).

*Етап 1* Оцінка акторів, для оцінки складності інтерфейсів системи

По діаграмі ВВ видно, що в системі присутні 3 актори: Сховище даних, Замовник і Проектувальник. Проаналізуємо їх типи з допомогою наданої таблиці(табл. 2.1).

Таблиця 2.1 – Визначення типів акторів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип актора | Вага | Кількість акторів | Опис |
| 1 | Простий | 1 | 1 | Інша система з певним API (REST, SOAP, dll) |
| 2 | Середній | 2 | 0 | Інша система, взіємодія з якою виконується за певним протоколом (наприклад, TCP/IP) |
| 3 | Складний | 3 | 2 | Переважно взаємодія користувача з допомогою GUI або веб-сторінки |
| Нескоректована оцінка акторів, AW | | | 7 |  |

Замовник та проектувальник являються складними типами акторів, так як вони взаємодіють з GUI, а сховище даних простим, так як система взаємодіє з допомогою вже існуючих API (табл. 2.2).

Таблиця 2.2 – Актори системи

|  |  |  |  |
| --- | --- | --- | --- |
| № | Актори | Вага | Назва Актора |
| 1 | Простий | 1 | Сховище даних |
| 2 | Складний | 3 | Замовник |
| 3 | Складний | 3 | Проектувальник |

UAW(AW) = 2\*3+1\*1=7

*Етап 2* Оцінка ВВ, що дає оцінку масштабу системи

Оцінимо ВВ, за кількістю класів, необхідних для реалізації

Розглянемо всі прецеденти нашої системи і на основ таблиці запишемо скільки приблизно класів буде використовуватися для реалізації того чи іншого прецеденту (табл. 1.3).

Таблиця 2.3 – Визначення типів ВІ кількості класів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип актора | Вага | Кількість класів | Опис |
| 1 | Простий | 5 | 15 | До 5 класів |
| 2 | Середній | 10 | 1 | От 5 до 10 класів |
| 3 | Складний | 15 | 1 | Більше 10 класів |
| Сумарний UUCP | |  | 100 |  |

Розглянемо детально типи варіантів використання (табл. 2.4)

Таблиця 2.4 – Вага ВВ

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тип ВИ | Вага | Назва ВВ |
| 1 | Простий | 5 | Збереження схеми |
| 2 | Середній | 10 | Створити схему |
| 3 | Простий | 5 | Робота з відділом |
| Продовження таблиці 2.4 | | | |
| 4 | Простий | 5 | Додати відділ |
| 5 | Простий | 5 | Видалити відділ |
| 6 | Простий | 5 | Вибір відділу |
| 7 | Простий | 5 | Робота з документами |
| 8 | Простий | 5 | Додати документ |
| 9 | Простий | 5 | видалити документ |
| 10 | Простий | 5 | Робота зі зв'язками |
| 11 | Простий | 5 | Додати зв'язок |
| 12 | Простий | 5 | видалити зв'язок |
| 13 | Простий | 5 | змінити зв'язок |
| 14 | Важкий | 15 | Передача даних |
| 15 | Простий | 5 | переглянути відділи |
| 16 | Простий | 5 | Переглянути зв'язок між відділами |
| 17 | Простий | 5 | Подивитися документи |

UUCW(UUCP) = 5\*15+10\*1+15\*1 = 100

*Етап 3.* Оцінка технічних факторів для обчислення коефіцієнта складності архітектури додатку

Оцінимо перераховані в таблиці фактори в залежності від їх впливу на складність побудови архітектури програмного продукту. 0 - відсутня, 3 - середнє вплив, 5 - значний вплив на архітектуру.

У процесі оцінки архітектури було виявлено для нашої системи є більш впливовими є: Простота використання, Час відгуку, Портативність (табл. 2.5).

TFactor = 4\*2+1\*5+1\*4+1\*2+1\*3+0.5\*4+0.5\*5+2\*4+1\*3+1\*2+1\*3+1\*3+1\*2 = 47,5

TCF = 0.6 + (0.01\*TFactor) = 0,6 + (0,01\*47,5) = 1,075

Таблиця 2.5 – Технічні фактори системи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Технічний фактор | Вага | Оцінка  (0-5) | Опис |
| 1 | Розподіленість систем | 2 | 2 | У програмі буде використовуватися багато модулів, які пов'язані між собою |
| 2 | Час відгуку | 1 | 5 | Так як у нас програма побудови організаційної структури підприємства орієнтована на роботу користувача з графічним компонентами, то час відгуку і створення елементів повинно проводиться дуже швидко. |
| 3 | Ефективність кінцевого користувача | 1 | 4 | Програма повинна бути інтуїтивна у використанні. |
| 4 | Складність обробки | 1 | 2 | При обробці даних немає потреби використовувати складні алгоритми, досить буде звичайних, які допоможуть проаналізувати інформацію, що надходить. |
| 5 | Фокус на повторному використанні коду | 1 | 3 | Повторне використання коду можливо, але не обов'язково, тільки для схожих властивостей елементів системи |
| 6 | Простота інсталяції | 0,5 | 4 | Процес інсталяції повинен бути стандартним і зрозумілим користувачеві |
|  | | | | |
| Продовження таблиці 2.5 | | | | |
| 7 | Простота використання | 0,5 | 5 | Так як у нас програма побудови організаційної структури підприємства орієнтована на роботу користувача з графічним компонентами |
| 8 | Портативність | 2 | 4 | Програма передбачає наявність платформ, тому буде писатися на мові програмувати Java |
| 9 | Простота змін | 1 | 3 | Програма повинна передбачати подальші зміни, так як вимоги замовника можуть зміниться, а внесення змін до системи не повинні значно вплинути на існуючу на той момент систему |
| 10 | Паралельні обчислення | 1 | 2 | Паралельні обчислення можна використовувати при роботі з інтерфейсом і завантаженням інформації зі сховища даних і відображенням їх на екран. |
| 11 | Способи захисту | 1 | 3 | Система повинна захищати отримані від замовника дані від зовнішніх програм і користувачів. |
| 12 | Доступ до третій стороні | 1 | 3 | Система повинна передбачати використання її іншими зацікавленими особами. |
| 13 | Потреба в спеціальному навчанні | 1 | 2 | Програма буде інтуїтивно зрозуміла користувачеві, тому немає особливої ​​необхідності навчання користувачів |
| Сумарна TCF | |  | 1,075 |  |

*Етап 4*. Оцінка зовнішніх факторів для обчислення коефіцієнта організаційних ризиків

Оцінимо вплив зовнішніх факторів на розробку програмного продукту. 0 - відсутня, 3 - середнє вплив, 5 - значний вплив на розробку (табл. 2.6).

Таблиця 2.6 – Зовнішні фактори до системи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Фактор | Вага | Оцінка  (0-5) | Опис |
| 1 | Знайомство з процесом розробки | 1,5 | 3 | Так як предметна область не зовсім знайома розробникам, то це вплине на якість розробки. Якщо розробник погано знає UML, що сповільнить час, яке він витратить на розуміння вимог до програмного продукту |
| 2 | Досвід в подібних проектах | 0,5 | 4 | Досвід подібних проект може зіграти значну роль в розробці, так як прискорить процес проектування і реалізації програмного продукту |
| 3 | Досвід ООП | 1 | 4 | Знання та досвід роботи в ООП програмування необхідний, так як програма буде розроблятися на високорівневій мовою програмування з графічним інтерфейсом |
| 4 | Досвідченість головного аналітика | 0,5 | 5 | Уміння і досвідченість аналітика напевно найголовніше в проекті, так як якщо він неправильно зрозуміє замовника, некоректно розробить вимоги до системи, то це може привести до непотрібної роботи з боку програміста, перероблення проекту кілька разів і велика кількість помилок |
| 5 | Мотивація | 1 | 3 | Мотивація не сильно впливає на розробку, так як користувач може переключиться з одного заняття на інше, потім повернуться до цього завдання |
| 6 | Стабільність вимог | 2 | 3 | Бажано що б вимоги були стабільні, тоді буде простіше розробнику, то таке в реальності не існує, так як замовник в міру розробки програмного продукту розуміє, що він не все тоді у вимогах з формулював. |
| Продовження таблиці 2.6 | | | | |
| 7 | Часткова зайнятість працівників | -1 | 3 | Часткова зайнятість працівників можливо, так як проект може бути розбитий на підзадачі і розрахувати так, що навіть незважаючи на часткову зайнятість працівників, вся необхідна робота виконається в термін |
| 8 | Складність мови програмування | -1 | 2 | Програмний продукт буде писатися на Java, а що б зрозуміти основи мови програмування і почати виконувати невеликі завдання системи вистачить приблизно місяць, з урахуванням, що в подальшому буде відбуватися поглиблення в мову, вивчення нюансів мови програмування. |
| Сумарне EF | |  | 0,89 |  | |

В ході аналізу при виконанні даної таблиці, можна зробити висновок, що на розробку найбільше впливають досвідченість розробників, знання і досвід роботи в ООП.

EFactor = 1.5\*3+0.5\*4+1\*4+0.5\*5+1\*3+2\*3+(-1)\*3+(-1)\*2 = 17

EF = 1.4+(-0.03\*EFactor) = 1.4+(-0.03\*17) = 0.89

*Етап 5.* Результуюча оцінка

На основі попередніх етапів можна визначити приблизну тривалість розробки в робочих годинах (табл. 2.7)

Таблиця 2.7 – Результуючі оцінки

|  |  |  |
| --- | --- | --- |
| Сумарна таблиця | | |
| TCF | Технічний фактор | 1,075 |
| EF | Зовнішній фактор | 0,89 |
| UUCP | Нескоректована оцінка | 100 |
| AW | Вага акторів | 7 |
| Сумарна оцінка ВВ | | |
| UCP | Оцінка ВВ | 102,4 |
| Підрахунок часу виконання проекту | | |
| Ratio | Час на кожен ВВ | 28 |
|  | | |
| Час виконання проекту | | 2 866 |

AUCP = UCP\*TCF\*EF = 28\* ( AW+UUCP)\*TCF\*EF = =28\*(7+100)\*1.075\*0.89=2866