Практичне завдання № 2.

Визначення функціональних / нефункціональних вимог і планування

розробки програмного продукту

2.1 Функціональні вимоги

2.1) Формування тезаурусів

1) База знань (knowledge base) - сукупність знань, що відносяться до деякої предметної області і формально представлених таким чином, щоб на їх основі можна було здійснювати міркування. Інформація, що міститься у базі знань, може являти собою не лише факти, а й правила, за допомогою яких може здійснюватися осмислена та обробка нових фактів.

2) Експертна система - це комплекс комп'ютерного програмного забезпечення, що допомагає людині приймати обґрунтовані рішення. Експертні системи використовують інформацію, отриману заздалегідь від експертів - людей, які в будь-якій області є кращими фахівцями.

2.2) Формування багаторівневої класифікації функціональних вимог

Рисунок 2.1 – Багаторівнева класифікація вимог

2.3) Вилучення функціональних вимог-дублів для обліку такої рекомендації IEEE-стандарту як модифікованості

Виходячи із аналізу функціональних вимог системи, було виявлено що вимоги дублі відсутні.

2.4) Вилучення суперечливих один одному вимог, щоб вимоги не суперечили вимогам свого рівня ієрархії і вимогам «батьківського» рівня (вимоги користувачів не повинні суперечити бізнес-вимогам, а функціональні вимоги - призначеним для користувача вимогам) для обліку таких рекомендацій IEEE-стандарту, як ясність і коректність.

В процесі розробки програмного продукту вже заздалегідь були виділені всі необхідні вимоги, тому немає сенсу додавати інші.

2.5) Визначення функціональних пріоритетів для обліку таких рекомендацій

IEEE-стандарту, як необхідність, корисність при експлуатації, здійсненність, наприклад по MoSCoW-принципом розстановки пріоритетів (M - повинні це зробити, S - повинні це зробити, якщо це взагалі можливо, C - могли б це зробити, якщо це не вплине негативно на щось інше, W - не буде достатньо часу на це, але в майбутньому хотілося б.

Усі стандарти IEEE відіграють важливу роль у розробці програми, тому за принципом MoSCoW можна виставити пріоритет S, якщо щось можливо реалізутати це потрібно зробити.

2.6) Функціональний аналіз існуючих програмних продуктів

Для аналізу функціоналу існуючих програмних продуктів було виділено наступні програми:

**MIXER**

Система надає допомогу програмістам в написанні мікропрограм для розробленої Texas Instruments НВІС TI990. По заданому опису прошивки система отримує оптимізовані прошивки для TI990. MIXER містить знання з мікропрограмування для TI990, взяті з керівництва і з аналізу прошивки керуючого ПЗУ TI990. Сюди відносяться знання про те, як перетворювати введені опису в набори проміжних операцій, як виділити відповідні регістри під змінні і як перетворити проміжні операції в набори мікрооперацій.

MIXER використовує ці знання, щоб визначити, які микрооперации є кращими для реалізації вбудованого. Система являє знання у вигляді правил і даних, володіє уніфікацією, керованої механізмом виведення, і динамічним поверненням. MIXER реалізована на мові Пролог. Вона була розроблена в Токійському університеті і доведена до рівня демонстраційного прототипу.

**ExpSystem PC**

Ця система допоможе людям не настільки добре розбираються в комплектуючих для персонального комп'ютера підібрати собі прийнятну конфігурацію майбутнього комп'ютера. У програму вводяться дані про вимогу до системи і максимальну суму грошей, передбачуваної для покупки.

Так само дана експертна система стане в нагоді для менеджерів комп'ютерних салонів що б автоматизувати свою роботу і зробити автоматизоване робоче місце для своїх покупців. Для підбору комплектуючих використовується два алгоритми розрахунку: для "багатого" і "бідного" покупателя. Данная ЕС була розроблена в грудні 2006 року.

**ACE**

Система ACE визначає несправності в телефонної мережі і дає рекомендації щодо необхідного ремонту і відновлювальним заходам. Система працює без людського втручання, аналізуючи зведення-звіти про стан, одержувані щодня за допомогою програми, яка стежить за ходом ремонтних робіт в кабельній мережі.

ACE виявляє несправні телефонні кабелі і потім вирішує, чи потребують вони в планово-попереджувальному ремонті і вибирає, який тип ремонтних робіт найімовірніше буде ефективним. Потім ACE запам'ятовує свої рекомендації в спеціальній базі даних, до якої користувач має доступ. Система приймає рішення, застосовуючи знання щодо телефонних станцій, повідомлення системи та стратегії аналізу мереж. Подання знань в системі засновано на правилах; використовується схема управління за допомогою прямої ланцюжка міркувань. Вона розроблена в Bell Laboratories. ACE пройшла дослідну експлуатацію і доведена до рівня комерційної експертної системи.

У таблиці 2.1 представлено порівняльний аналіз готових систем з системою що розробляється.

Таблиця 2.1 – Порівняльний аналіз систем

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функції | Програмні продукти | | | |
| ZEVS | MIXER | ExpSystem PC | ACE |
| Пошук інформації | + | + | + | - |
| Редагування | + | - | - | - |
| Консультація користувача | + | + | + | + |
| Статистика | + | - | - | + |

В процесі пошуку програмних продуктів було виявлено, що немає систем-аналогів які б повністю відтворювали функціонал розроблюваної системи. Та чи інша програма яка була знайдена частково відтворює функціонал але не повністю, і це є великим плюсом розроблюваної системи.

2.2 Нефункціональні вимоги

2.2.1) зовнішні інтерфейси:

* інтерфейс користувача

В процесі розробки потрібно дотримуватися принципу KISS. Принцип KISS. Від англ. «Keep it short and simple». Інтерфейс повинен бути простий і зрозумілий, завдання повинні вирішуватися мінімальним числом дій, все повинно бути зрозуміло і очевидно.

Не варто чіплятися за моду і робити щось тільки тому, що так роблять інші, краще віддати перевагу стандартним перевіреним елементам інтерфейсу.

На рисунку 1 і 2 зображено стандартне оформлення інтерфейсу користувача.

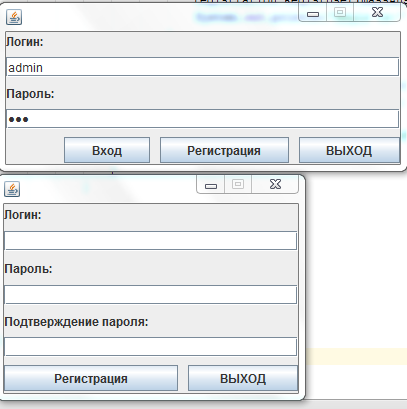


Рисунок 1 – Вікно авторизації користувача.

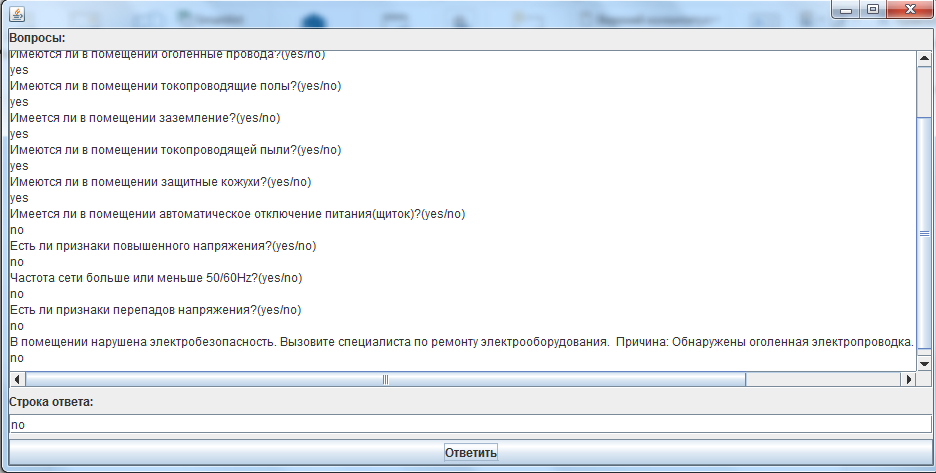


Рисунок 2 – Вікно консультації користувача

* інтерфейси із зовнішніми пристроями (обладнанням);

В розроблюваній програмі відсутні будь які інтерфейси із зовнішніми пристроями.

* програмні інтерфейси

Програма розроблюється платформонезалежною мовою JAVA, тому для корректної роботи необхідно встановити JVM (Java Virtual Machine).

* інтерфейси передачі інформації

Так як для підключення користувача до сервера використовуються сокети, інтерфейси передачі даних можуть бути як провідні так и без провідні. Сервер можна розвернути як в глобальній мережі інтернет так і в локальній мережі комп’ютерів.

* атрибути якості

Сформулюємо вимоги до програмної системи за ISO 9126 з такими атрибутами якості:

* **Функціональність** - набір атрибутів, що відносяться до суті набору функцій та їх конкретних властивостям. Функціями є ті, які реалізують встановлені або передбачувані потреби.
* **Ефективність** - набір атрибутів, що відносяться до співвідношення між рівнем якості функціонування програмного забезпечення і обсягом використовуваних ресурсів при встановлених умовах.
* **Надійність** - набір атрибутів, що відносяться до здатності програмного забезпечення зберігати свій рівень якості функціонування при встановлених умовах за встановлений період часу.
* **Переносимість** (мобільність) - набір атрибутів, що відносяться до здатності програмного забезпечення бути перенесеним з одного оточення в інше.

В Таблиці 1 представлені атрибути якості програмної системи за ISO 9126.

Таблиця 1 - Нефункціональні вимоги програмної системи

| **Набір характеристик (за ISO 9126)** | **Властивості програмного забезпечення** |
| --- | --- |
| **Функціональність** | **Здатність до взаємодії (interoperability):** Система активно взаємодіє з **Базою даних** яка містить у собі **Базу знань,** які необхідні при прийнятті рішень.  **Захищеність (security):** У зв'язку з тим що база даних є найважливішою складовою системи доступ строго обмежений, за допомогою розділення прав. |
| **Ефективність** | **Часові характеристики (time behaviour):** Додавання даних в базу має здійснюється з максимальною швидкістю, і воно повинно бути < 5 секунд. Отримання даних з бази, мабуть найголовніше, так як на цьому зав'язана робота системи час отримання даних повинне бути < 3 секунд  **Використання ресурсів (resource utilisation):** Іза того що мова java досить вимоглива до пам'яті, програмна система не повинна споживати < 1 гб пам'яті |
| **Надійність** | **Здатність до відновлення працездатності при відмовах (recoverability):** Програмна система повинна періодично зберігати дамп бази даних, щоб у разі збою бази, дані можна було відновити з допомогою дампера. |
| **Переносимість (мобільність)** | **Адаптованість (adaptability):** Програмна система буде написана мовою java, оскільки мова java платформонезалежна дана програма зможе запускатися на платформах з різними операційними системами. |

**2.3 Планування розробки**

**1) Завдання проектування програмної системи як узагальнення ієрархії**

**функціональних вимог**

Головна мета програмної системи є поліпшення безпеки роботи з електрообладнанням.

Дана система, аналізує вимоги з електробезпеки і приймає рішення про допустимість проведення робіт. Рішення програма буде приймати виходячи з правил, які будуть описані на основі експертної системи.

Користувач даної системи може виконувати наступні дії:

1) Додавати, видалення та змінювання правил експертної системи.

Дані дії користувач може виконати в разі виявлення недоліків знань, отриманих від експерта (наприклад протиріччя, неповноти або помилковості), і усунути їх шляхом використання даних функцій.

2) Додавати, видалення та змінювання інформаційної бази та користувачів системи.

Ця функція дозволяє користувачеві модифікувати систему під інші потреби або допоможе усунути недоліки системи.

3) Консультація користувача.

У разі виникнення питань щодо електробезпеки, користувач може скористатися цією функцією, в ході інтерв'ю відповівши на питання, система допоможе користувачеві вирішити якусь проблему.

4) Пошук інформації

Якщо користувачу необхідно знайти інформації з електробезпеки, ця функція допоможе знайти її в найкоротші терміни.

1. **Архітектура (концептуальний опис структури) програмної системи**

Патерн Model-View-Controller (MVC) основна мета застосування цієї концепції полягає у відділенні бізнес-логіки (моделі) від її візуалізації (уявлення, виду).

За рахунок такого поділу підвищується можливість повторного використання. Найбільш корисно застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах та / або з різних точок зору.

Для наочності розглянемо даний патерн по всім трьом пунктам.

1. Model (Модель) - надає дані та методи роботи з даними, для реалізації цього буде використано JDBC драйвер для доступу к даним, які будуть знаходитись в MySQL БД.
2. View (Вид) - реалізує графічний інтерфейс користувача, для реалізації графічного інтерфейсу буде використовуватися стандартна бібліотека Java, Swing.
3. Controller (Контролер) - забезпечує зв'язок між користувачем і системою: реалізуватися це буде за допомогою класів Java та серіалізації.

**3) Технології розробки**

* операційні системи:

Віртуальна машина Java є платформонезалежною тому вона буде нормально виконуватися на самих розповсюджених ОС.

* СУБД:

Для збереження необхідних даних буде використовуватися MySQL БД.

* програмна технологія розробки:

Програмна система повністю спрямована на Desktop.

* структури даних:

Система буде використовувати тільки реляційну БД.

1. **Інструменти розробки**

* IDE: Eclipse Kepler, MySQL Workbench;
* збирачі програмних модулів: вистроєні збирачі IDE;
* система контролю версій: GitHub;
* бібліотека модульного тестування: JUnit;
* автоматизація тестування: JUnit, JBehave, Selenium WebDriver;

1. **План розробки**

5.1) Структура декомпозиції робіт (Work Breakdown Structure – WBS)

Планування робіт проекту слід починати з визначення процесів, які повинні бути реалізовані в проекті. Відповідно до стандарту ISO / IEC 12207 всі процеси життєвого циклу програмного забезпечення розділені на групи (рис. 3).

Життєвий цикл

Організаційні процеси

Основні процеси

Допоміжні процеси

Управління проектом

Створення інфраструктури

Оцінка і поліпшення життєвого циклу

Навчання

Придбання

Поставка

Розробка

Експлуатація

Супровід

Документування

Управління конфігурацією

Забезпечення якості

Верифікація і атестація

Оцінка

Вирішення проблем

Рисунок 3 - Процеси життєвого циклу програмного забезпечення

Для планування робіт, пов'язаних з виконанням дипломної роботи, обов'язковим безліччю процесів є процеси розробки або супроводу, навчання, управління проектом, документування та забезпечення якості. Решта процеси вводиться в розгляд при необхідності.

Процес розробки визначає дії і завдання, які виконуються розробником в процесі створення програмного забезпечення та його компонентів відповідно до заданих вимог. Декомпозиції робіт процесу розробки пропонується виконувати на основі варіантів використання.

       Декомпозиції робіт організаційних і допоміжних процесів пропонується виконувати з групами процесів.

Декомпозицію (WBS) будемо виконувати в програмі Microsoft Project.

       Для почав створимо новий проект MS Project і встановимо тимчасові рамки виконання дипломної роботи. Відомості про проект представлені на рис. 4

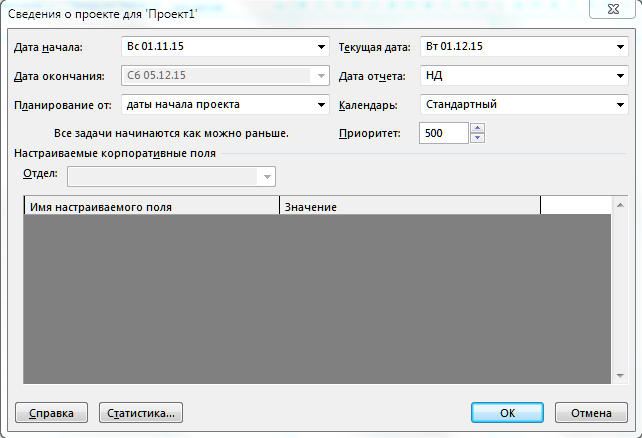


Рисунок 5 - Дані про дипломний проект

Для проекту встановлюється відлік від дати початку, тому дата закінчення проекту неактивна. Часто проект планується таким чином, що він повинен бути обов'язково завершений до певної дати.

Також для проекту встановлюється стандартний календар, в якому вже встановлено час і початок роботи, а також час перерви. Календарем не включені вихідні і укорочені дні.

Такий варіант роботи мене не влаштовує, тому потрібно змінити наявні налаштування або створити новий календар. Змінені настройки календаря представлені на рис. 6

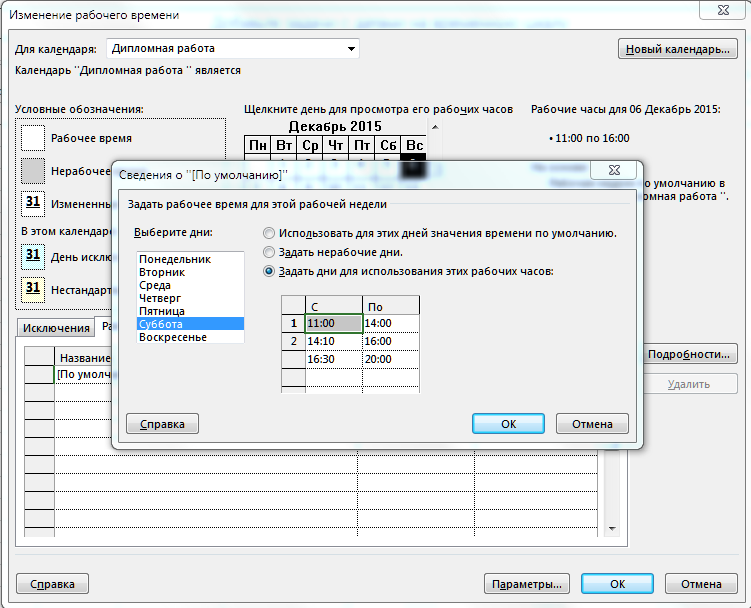


Рисунок 6 - Змінені тимчасові рамки розробки

У зв'язку з тим, що дипломний проект розробляється на протязі всього навчального процесу, і пари в середньому закінчуються о 13:15, було вибрано час по буднях з 17:30 грубо по 21:30 та у вихідні можна побачити на рис. 6

Після зміни часу виконання можна приступати до створення ресурсів проекту. У зв'язку з тим що я єдиний ресурс проекту, буде створено всього лише один ресурс. Результати створення і призначення ресурсу представлені на рис. 7

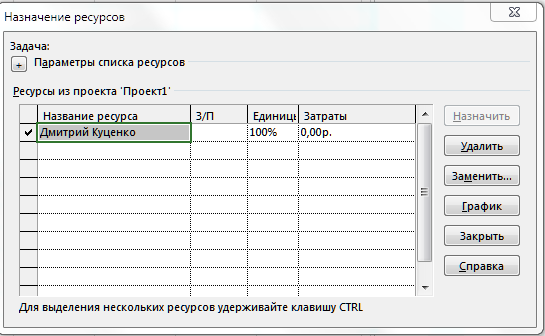


Рисунок 7 - Призначення ресурсу

Зробимо розподіл роботи. Розподіл представлено на рис. 8

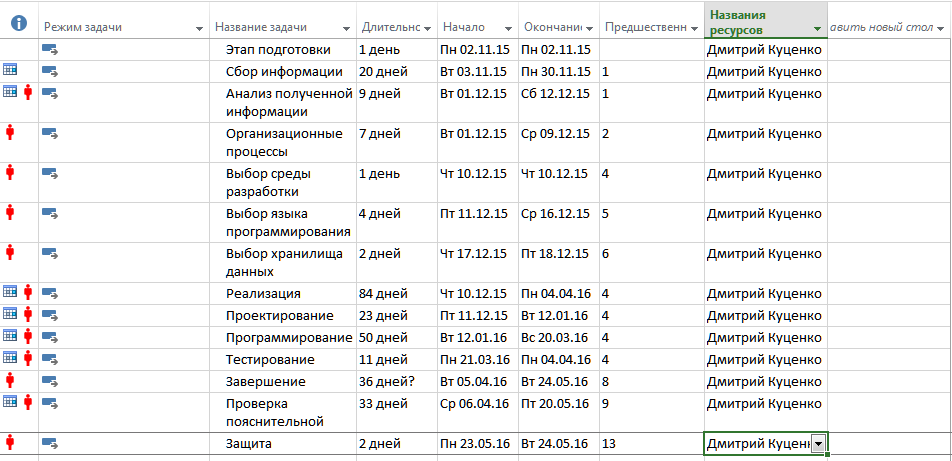


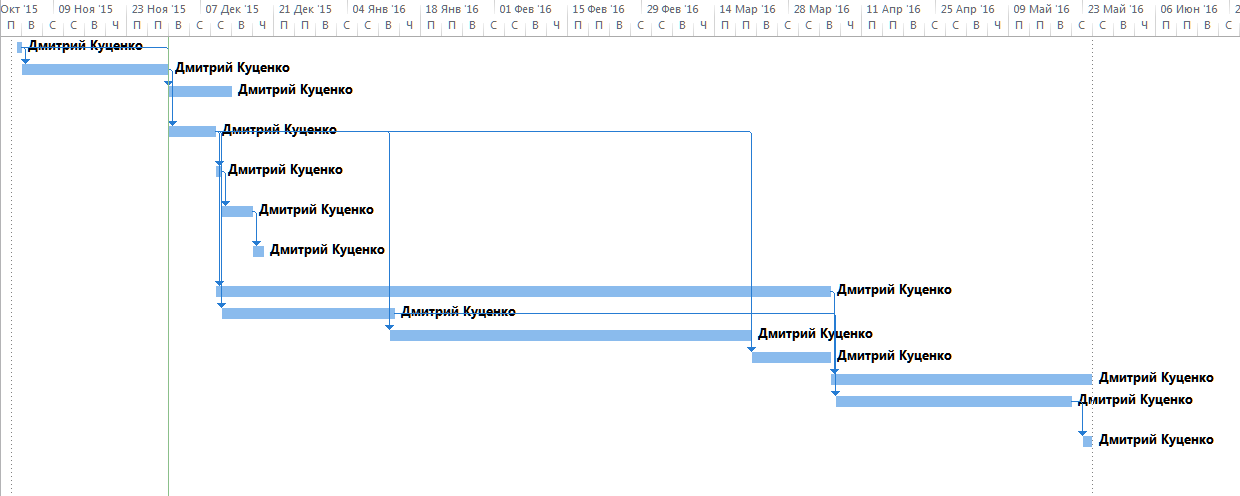
Рисунок 8 - Структура розподілу роботи

У сумі на розробку необхідно витратити 157 днів що приблизно дорівнює 5 місяців. В результаті отримуємо такий розподіл часу. Розподіл часу представлено в табл. 2

|  |  |
| --- | --- |
| Елемент WBS першого рівня | Стандартний розподіл |
| Етап підготовки | 15% |
| Організаційні процеси | 5% |
| Реалізація | 60% |
| Завершення | 20% |

Таблиця 2 - Розподіл часу

Для наочності побудуємо діаграму Ганта. Діаграма Ганта є графіком, на якому по горизонталі розміщена шкала часу, а по вертикалі розташований список завдань (рис. 9). Довжина відрізків, що позначають завдання, пропорційна тривалості завдань. Поруч з відрізками може відображатися додаткова інформація, наприклад, назви задіяних в них ресурсів. Склад діаграми визначається її налаштуванням.

Рисунок 9 – Діаграма Ганта.

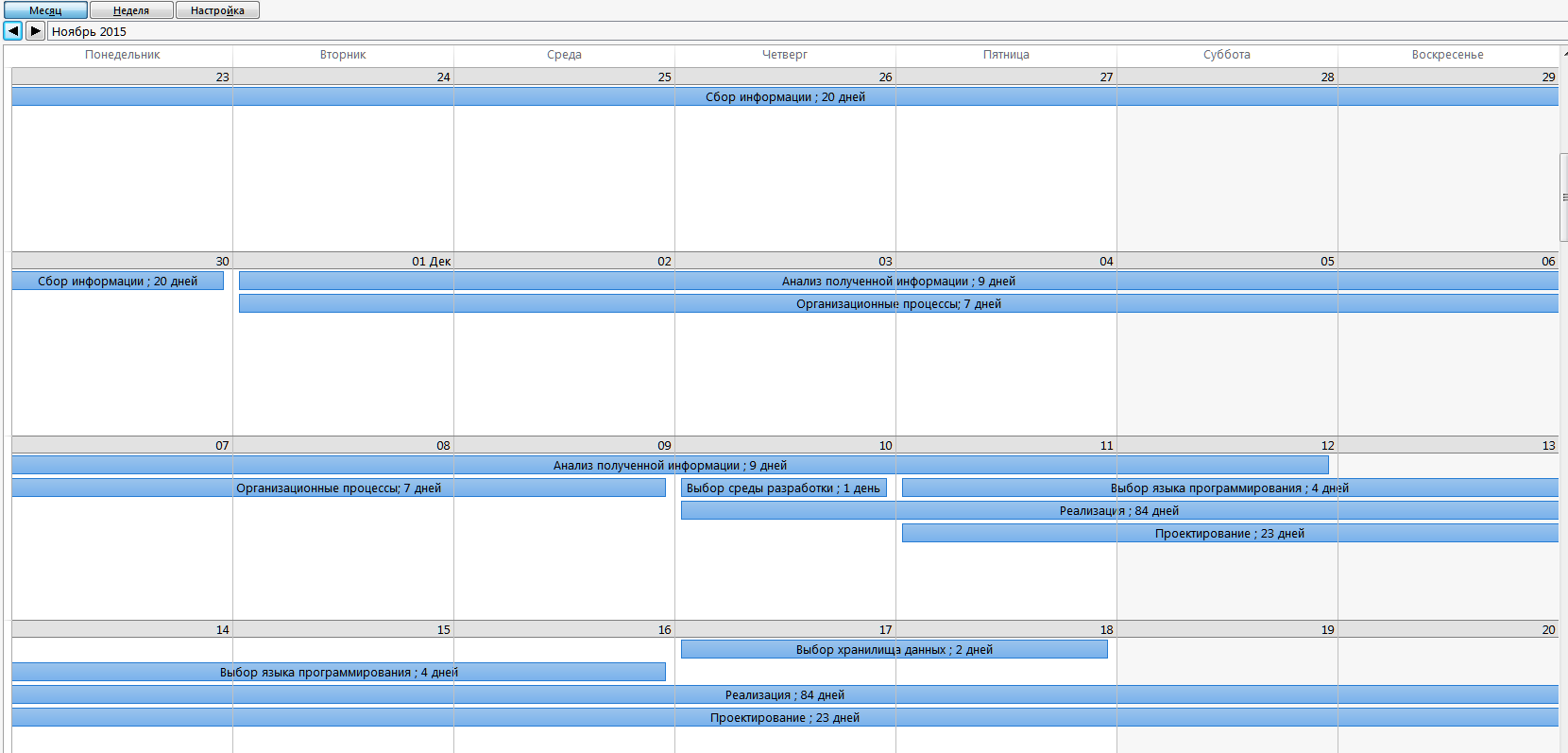
За допомогою діаграми Календар можна представити план робіт в звичному для більшості співробітників вигляді. Ця діаграма відображає інформацію про план проекту у вигляді таблиці з сімома або п'ятьма колонками, відповідні днях тижня, і безліччю рядів, відповідні тижнях. Завдання на календарі позначено відрізками, які починаються в день початку робіт над завданням і закінчуються в день їх закінчення (рис.10). Робота з завданнями (зміна термінів, створення, видалення і т.п.) здійснюється так само, як і на діаграмі Ганта.

Рисунок 10 - Фрагмент діаграмы Календар

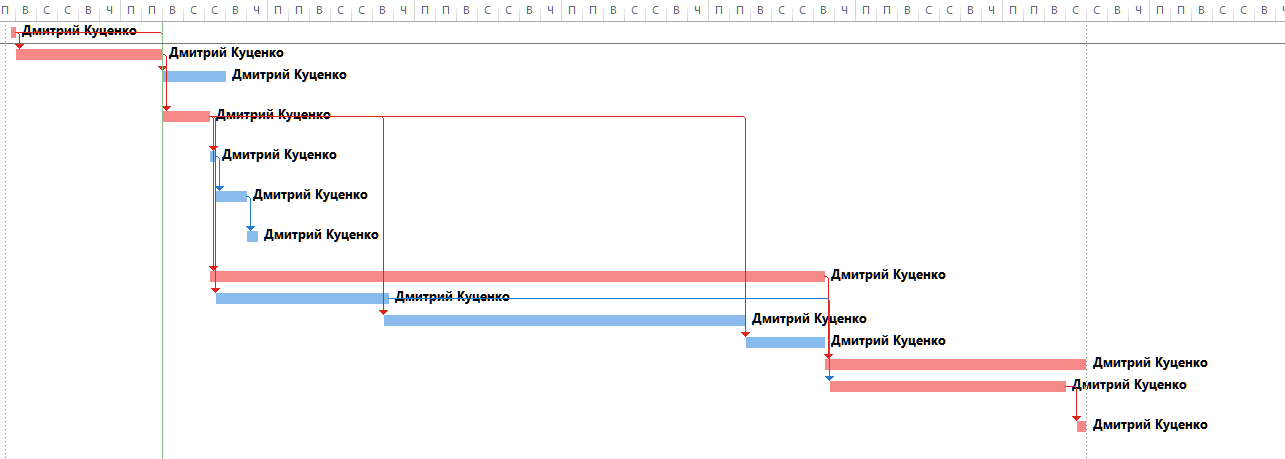
Критический путь проекта представлен на рис. 11 и изображен красным цветом.

Рисунок 11 - Критичний шлях проекту

Виходячи з графіка критичного шляху, можна без сумніву сказати що червоним кольором позначено сам процес розробки а саме проектування, програмування, тестування і т. д.

Мережевий графік є ще одним популярним засобом візуалізації плану проекту. На ньому завдання представлені у вигляді блоків, з'єднаних стрілками в блок-схему у відповідності з взаємозв'язків завдань в плані проекту.

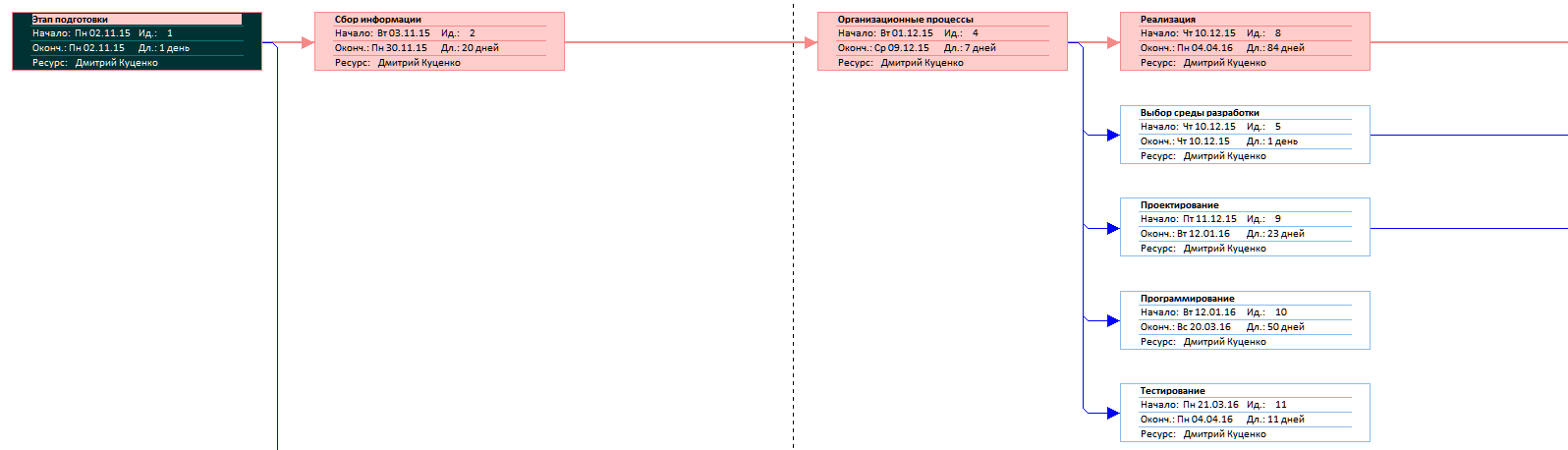
Фрагмент мережевого графіка представлений в рис. 12.

Рисунок 12 - Фрагмент мережевого графіка

5.2) Визначення обсягів робіт за методом UCP

Методика UCP дозволяє врахувати нефункціональні вимоги, організаційні ризики, компетенцію при оцінці і інші критерії.

Метод складається з 5 етапів.

• Етап 1. Оцінка акторів.

• Етап 2. Оцінка варіантів використання.

• Етап 3. Оцінка технічних факторів.

• Етап 4. Оцінка зовнішніх факторів.

• Етап 5. Результуючі оцінки.

**Етап 1. Оцінка акторів**

У методі UCP розрізняють три типи акторів, відповідно до складності проектування та реалізації інтерфейсу взаємодії системи з ними (табл. 3).

Таблиця 3 - Визначення типів акторів

|  |  |  |
| --- | --- | --- |
| Тип актора | Вага | Примітка |
| Простий | 1 | Інша система з певним API (REST, SOAP, dll) |
| Середній | 2 | Інша система, взаємодія з якою виконується за певним протоколу (наприклад, TCP / IP) |
| Складний | 3 | У більшості випадків взаємодія з користувачем за допомогою GUI або веб-сторінок |

У системі присутній два найважливіших актора це Користувач і База даних.

Виходячи з даних представлених в табл. 3 можна визначити, що: Користувач і Адміністратор це складний тип актора, так як саме він взаємодіє з системою за допомогою графічного інтерфейсу (GUI).

База даних це середній тип актора, тому що система має клієнт / серверну архітектуру і доступ до бази буде здійснюватися по мережевим протоколам передачі даних. У табл. 4 представлений короткий висновок.

|  |  |  |
| --- | --- | --- |
| Найменування актора | Тип актора | Вага |
| Користувач | Складний | 3 |
| Адміністратор | Складний | 3 |
| БД | Середній | 2 |

Таблиця 4 - Короткий висновок

Тепер зробимо скориговану оцінку наявних акторів.

UAW = 3\*1 + 3\*1 + 2\*1 = 3+ 3 + 2 = 8.

**Етап 2. Оцінка варіантів використання**

Тип варіанти використання найпростіше визначити відповідно до кількості транзакцій (неподільних операцій) в ньому, але наша система часто взаємодіє з базою даних тому краще використовувати інший підхід, а саме підхід який базується на кількості об'єктів в базі даних (БД), які змінюються в його рамках (табл. 5).

|  |  |  |
| --- | --- | --- |
| Тип варіанти використання | Вага | Кількість об'єктів в БД |
| Простий | 5 | 1 |
| Середній | 10 | 2 |
| Складний | 15 | 3 и більше |

Таблиця 5 - Визначення типів варіантів використання за кількістю змінених об'єктів в БД.

Визначаємо тип варіантів використання, опис представлено в табл.6

Таблиця 6 - Визначення варіантів використання

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування варіанта використання | Тип варіанти використання | Вага | Опис |
| Авторизація | Простий | 5 | Простий варіант використання, складається всього лише з перевірки даних, без будь-яких змін. |
| Реєстрація | Простий | 5 | Також як і авторизація простий варіант використання, але все ж відбувається зміна даних в базі за рахунок додавання нового користувача. |
| Консультація | Простий | 5 | Простий варіант використання, з БД не відбувається ніяких змін крім отримання даних. |
| Статистика | Простий | 5 | Як і у випадку з консультацією це простий варіант без змін БД. |
| Пошук інформації | Середній | 10 | Змін в БД не передбачає але реалізація функції на середньому рівні. |
| Редагування | Складний | 15 | Найскладніший варіант використання так як відбувається безпосередня робота з БД, в результаті чого дані в БД будуть активно зміняться. |

І так, зробимо обчислення нескорректированной оцінки варіантів використання. Для обчислення нескорректированной оцінки варіантів використання (UUCW) слід підрахувати кількості варіантів використання кожного типу, помножити ці кількості на відповідні вагові коефіцієнти і знайти суму отриманих творів.

Показник UCP обчислюють за формулою:

UCP = UAW + UUCW

Загальна кількість варіантів використання 6, з них 4 простих ,1 середній і 1 складний.

UUCW = 4\*5 + 1\*10+ 1\*15 = 30 + 15 = 45.

UAW = 8

UUCW = 45

UCP = UAW + UUCW.

UCP = 8 + 45 = 53.

**Етап 3. Оцінка технічних факторів**

Оцінка технічних факторів дає коефіцієнт для оцінки складності архітектури застосування (табл. 7). Оцінка проводиться за шкалою від 0 до 5, де 0 означає відсутність впливу, 3 - середній вплив, 5 - сильний вплив на розробку.

Таблиця 7 - Технічні фактори

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактор | Опис | Вага | Пояснення | Оцінка |
| Т1 | Розподіленість системи | 2 | Інформує про необхідність системи в розподілених обчисленнях | 0 |
| Т2 | Час відгуку | 1 | Визначає ефективність системи з точки зору часу відгуку, потоку робіт і т.п. | 5 |
| Т3 | Ефективність кінцевого користувача | 1 | Визначає ефективність користувача з точки зору його (її) сприйняття | 3 |
| Т4 | Складність обробки | 1 | Визначає, чи будуть застосовуватися складні алгоритми для обробки даних | 0 |
| Т5 | Фокус на повторному використанні коду | 1 | Визначає, чи будуть розміщені елементи коду системи використовуватися знову | 0 |
| Т6 | Простота інсталяції | 0,5 | Визначає спосіб призначення та простоту інсталяції для кінцевого користувача, або необхідності в фахівця для установки системи | 3 |
| Т7 | Простота використання | 0,5 | Зазначає узгодженість інтерфейсу з його потребами | 5 |
| Т8 | Портативність | 2 | Визначає, чи має застосування працювати в різних середовищах | 3 |
| Т9 | Простота зміни | 1 | Визначає, буде будуватися система таким чином, щоб спростити її модифікації в майбутньому | 3 |
| Т10 | Паралельні обчислення | 1 | Інформує, матимуть в системі місце паралельні обчислення | 0 |
| Т11 | Способи захисту | 1 | Визначає вимагає система спеціальні засоби захисту даних і системи | 3 |
| Т12 | Доступ до третьої сторони | 1 | Визначає ступінь використання системи зовнішніми системами або акторами | 3 |
| Т13 | Потреба в спеціальному навчанні | 1 | Визначає, чи потрібно організувати тренінги для користувачів | 0 |

Тепер на основі даних представлених в табл. 7 зробимо обчислення технічного чинника UCP.

Сума добутків вагових коефіцієнтів і оцінок для кожного з технічних факторів визначає показник TFactor. Оцінка технічного фактора обчислюється за формулою:

TCF = 0,6 + (0,01 \* TFactor)

TFactor = 28

TCF = 0,6 + (0,01 \* 28) = 0,6 + 0,28 = 0,88

**Етап 4. Оцінка зовнішніх факторів**

Оцінка зовнішніх факторів дає коефіцієнт для організаційних ризиків при розробці (табл. 8). Оцінка проводиться за шкалою від 0 до 5, де 0 означає відсутність впливу, 3 - середній вплив, 5 - сильний вплив на розробку.

Таблиця 8 - Зовнішні чинників

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фактор | Опис | Вага | Пояснення | Оцінка |
| F1 | Знайомство з процесом розробки | 1,5 | Визначає, знайома чи команда з предметною областю і технічними аспектами вирішення проблеми клієнта. Особливу увагу слід приділити знання методології, в якій виконується проект, а також знання мов моделювання системи | 3 |
| F2 | Досвід подібних проектів | 0,5 | Загальне уявлення про досвід команди в розробці програмного забезпечення | 0 |
| F3 | Досвід об'єктно-орієнтованої розробки | 1 | Досвід в проектуванні об'єктно-орієнтованих додатків, а також в підтримку засобів для розробки інформаційних систем | 5 |
| F4 | Досвідченість провідного аналітика | 0,5 | Здібності аналітика отримати вимоги від клієнта і знання по задач, які буде вирішувати система | 3 |
| F5 | Мотивація | 1 | Здатність команди займатися призначеної завданням | 5 |
| F6 | Стабільність вимог | 2 | Визначає, чи не будуть вимоги часто змінюватися | 5 |
| F7 | Часткова зайнятість працівників | -1 | Визначає, наскільки велика частка працівників часткової зайнятості | 0 |
| F8 | Складність мови програмування | -1 | Визначає, наскільки складно вивчити мову програмування:    0 - легко, за тиждень можна опанувати;    1 - не менше 2 тижні потрібно для вивчення мови;    2 - мінімум місяць потрібно для вивчення мови;    3 - потрібен спеціальний тренінг з мови;    4 - потрібен спеціальний тренінг і допомогу під час виконання проекту;    5 - складно, потрібні тільки досвідчені люди. | 1 |

Произведем вычисление внешних факторов UCP.

Сумма произведений весовых коэффициентов и оценок для каждого из внешних факторов определяет показатель EFactor. Оценка внешнего фактора вычисляется по формуле:

EF = 1,4 + (-0,03 \* EFactor)

EFactor = 22

EF = 1,4 + (-0,03 \* 22) = 1,4 -0,66 = 0,74

**Етап 5. Результуючі оцінки**

Скориговані UCP обчислюються за формулою

UCP = UCP \* TCF \* EF.

AUCP = UCP \* TCF \* EF = 53 \* 0,88 \* 0,74 = 35

Для визначення тривалості розробки потрібно знати, якою кількістю робочих годин відповідає один UCP. Для цього підраховується кількість чинників з безлічі F1 - F8, оцінки яких за абсолютним значенням перевищують 3. Якщо результат 2 або менше, то для розрахунку тривалості розробки приймається, що одному UCP відповідає 20 робочих годин. Якщо результат 3 або 4, то одному UCP відповідає 28 робочих годин. Якщо результат перевищує 4, то рекомендується переглянути умови виконання проекту. У разі неможливості це зробити приймається, що одному UCP відповідає 36 робочих годин.

Результат з безлічі F1 - F8 дорівнює 5, в слідстві чого одному UCP буде відповідає 36 робочих годин. В результати отримуємо 35 \* 36 = 1260 годин.

Для розробки даної системи необхідно 1260 год що приблизно дорівнює 53 дням.