2.1 Функциональные требования

**2.1) формирование тезаурусов (словарей слов предметной области, обнаруженных в сценариях)**

Предметная область: медицина, диагностика заболеваний дыхательных путей.

**Термины:**

**КВВВ-данные** (Конденсат влаги выдыхаемого воздуха) - на основе этих данных ставится диагноз.

**Медкарточка** – медицинская карта пациента, содержит такую информацию: ФИО, пол, год рождения, история болезней, аллергия, курящий/некурящий, спортсмен/нет, контактные данные и прочее.

**Спектрометр** – прибор, с помощью которого проводится сбор КВВВ-данных.

**Классификатор** – программа, которая группирует имеющиеся КВВВ-данные и определяет, к каким болезням что относится. Сначала проходит обучение, затем сопоставление с новыми данными и выдача результата.

**Объект автоматизации:** процесс диагностики

2.2 Нефункціональні вимоги

**Функціональність**

Правильність

Точність - А = 186, Б = 200 => 0.93, де А - кількість елементів даних, реалізованих із заданними рівнями точності, Б - кількість елементів даних, для яких в специфікації задані рівні точності.

Придатність

Повнота функціональної реалізації - система відповідає заявленим специфікаціям. 1 - А/Б = 1 - 2/20 = 0,9, де А - число нереалізованих функцій, Б - число функцій, описаних в специфікації.

**Мобільність**

Налагодження - система має бути встановлена в зазначеному середовищі.

Об’єм робот по установці - А/Б = 7/8 =  0.88, де А - кількість автоматичних кроків інсталяції, Б - необхідна кількість кроків.

Взаємозамінність - система може бути використана замість існуючого програмнго забезпечення.

Спадкоємність даних - А/Б = 15/20 = 0.75, де А - кількість елементів даних ПС, що продовжують використовуватися після заміщення, Б - кількість елементів старих даних, які повинні використовуватися зі старого ПС.

**Надійність**

Завершеність

Кількість тестових комбінацій, виконаних при перевірці = 6, кількість тестових комбінацій за вимогами = 7 => 6/7 = 0.85

Відновлення

Кількість реалізованих вимог до відновлення = 5, загальна кількість вимог до відновлення = 12 => 0.42

Стійкість до помилки

Кількість безпечних функцій =  45, кількість небезпечних дій = 67 => 45 / 67 = 0.67

**Функціональність**

1. Придатність - 0,9 (відсутні 2 функції, описані 20 => 1 - 2/20 = 0.9)
2. Правильність - (Кількість результатів з незадовільним рівнем точності - 3, час роботи - 25 годин => 3/25 = 0.12)
3. Здатність до взаємодії (Кількість випадків невдалих підключень до інших ПС - 1, всього підключень - 129 => 1 - 1/129 = 0.92)

**Мобільність**

1. Адаптованість (кількість працездатних даних при інсталяції - 32, очікуване число працездатних даних - 40 => 32/40 = 0.8)
2. Налагодження - число успішних випадків інсталяції = 7, загальне число = 11 => 7/11 = 0.63
3. Сумісність - кількість обмежень = 5, тривалість роботи = 10 годин => 5/10 = 0.5

**Надійність**

1. Стійкість до помилки - кількість запобіглих відмов = 10, тестових випадків 13 => 10/13 = 0.77
2. Відновлення - успішне відновлення = 8, згідно вимог 10 випадків потрібно розглянути => 8/10 =  0.8
3. **Продуктивність**

Коефіцієнт продуктивності

Продуктивний час = 20, тривалість задачі =15, тривалість допомоги = 7, тривалість обробки помилок = 4, тривалість пошуку = 2 => 20/15-(7+4+2) = 10

Система має опрацьовувати дані швидше за інші існуючі способи.

**Результативність**

Завершені задачі = 127, всого задач = 129 => 127/129 = 0.98

**Задоволеність**

Кількість випадків використання програми згідно специфікації = 120, кількість випадків запланованого використання = 129 => 0.93

Иерархия функциональных требований отображена в WBS (пункт 5).

2.3 Планирование разработки

1, 2) архитектура (концептуальное описание структуры) программной системы:

- перечисление компонент, ответственных за три уровня программной системы (представление, бизнес-логика, данные), например, используя шаблонную архитектуру MVC;

- связи передачи данных между компонентами;

- связи передачи управления между компонентами.

## **Діаграма класів**



Рисунок 1 - Діаграма класів (кінцева версія)

Перероблена версія діаграми класів дещо нагадує шаблон проектування MVC і є більш доречним варіантом для створення програмного коду для даної системи.

Інтерфейс програми відображається у класі View, база даних усієї системи представляється у вигляді двох класів MedCard – клас, який предоставляє інформацію про пацієнтів, DataStorage – дані про медичний персонал та діагностичні групи. Усі обчислення щодо класифікації відбуваються у класі Classificator.

**Діаграма послідовностей (нова версія)**

Розглянемо на рисунках 2.7 – 2.9 виконання деяких варіантів використання з плином часу, що дозволяє продемонструвати діаграма взаємодії (послідовностей).

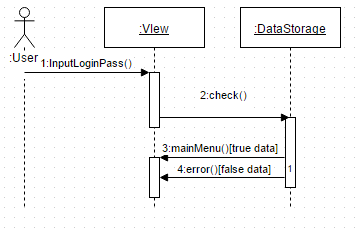


Рисунок 2 - Діаграма послідовностей ВВ «Authorization»

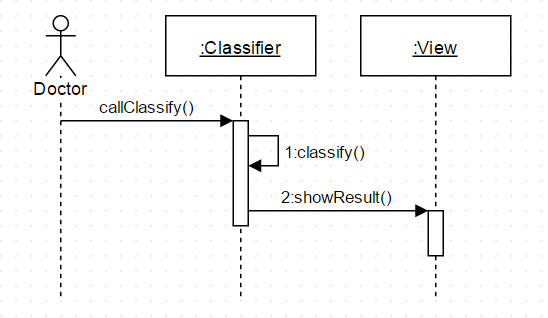


Рисунок 3 - Діаграма послідовностей ВВ «Identify patient’s condition»

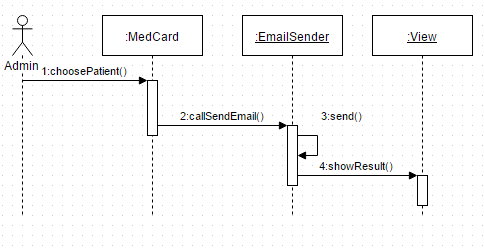


Рисунок 4 - Діаграма послідовностей ВВ «Send report via email»

**3) технологии разработки:**

**- операционные системы;**

**- СУБД;**

**- программные технологии разработки (Desktop, Web);**

**- структуры данных (файлы, реляционные БД, XML);**

Кроссплатформенное. СУБД MySQL. Структуры данных – РБД.

Представление данных в классификаторе – модель признаков.

**4) инструменты разработки:**

**- IDE;**

**- сборщики программных модулей;**

**- системы контроля версий;**

**- библиотеки модульного тестирования;**

**- автоматизация тестирования.**

IDE NetBeans (Java). Система контроля версий – Git.

Среда разработки классификатора – Weka

Система тестирования – FannTool

**5.1) структура декомпозиции работ (Work Breakdown Structure - WBS), который базируется на предпосылках: - верхний уровень – результат проекта (готовый программный продукт); - подуровни содержат рабочие пакеты, среднее количество подуровней – 3, сложность выполнения рабочего пакета не должна превышать 10 дней (две рабочих недели); - рабочие пакеты являются уникальными и не должны дублироваться**.

Перш за все, потрібно визначити терміни дипломного проектування. Роботи було розпочато в момент отримання відомостей про необхідність проектування у 7-8 семестрах навчання. Отже, початком робіт буде **1 вересня 2015** року. Великий захист дипломної роботи буде проведено **1 липня 2016 року**, отже, до цієї дати проектування має бути повністю виконане.

Припустимо, що, оскільки це дипломний проект, студент працює кожен день. Таким чином, в наявності є **304** дні.

Також в деякі етапи проектування залучений дипломний керівник, який працює лише по будніх днях. Будемо вважати, що керівник реагує моментально і дає відповіді без затримки, а отже, не впливає на критичний шлях. У першому етапі будо визначено, що для реалізації проекту буде потрібно 1714 годин, або 75 днів, якщо працювати 24 години на добу. Можна вивести оцінку кількості годин на добу, враховуючи оцінку кількості днів на реалізацію.

Nгод = 1714 / 304 = 5.63

Малоймовірно, що студент буде працювати 6 годин на добу кожен день, тому для зниження навантаження деякі етапи можна виконувати паралельно.

Можна побудувати структурну декомпозицію робіт (WBS). WBS у середовищі MS Project показана на рис. 2. Основні етапи робіт пов’язані також з датами виконання курсових проектів, адже вони перетинаються з дипломом.

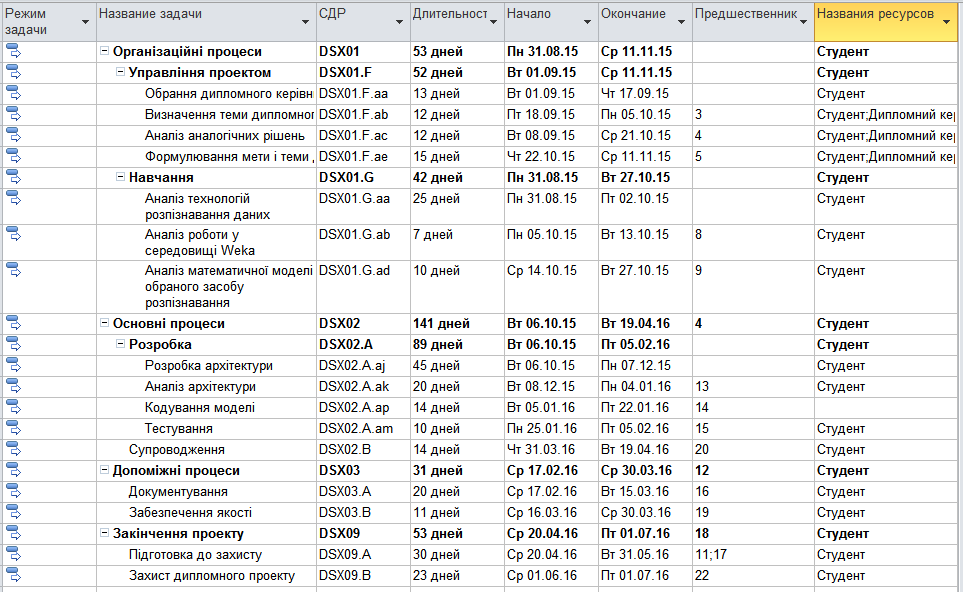


Рисунок 2.1 - Структурна декомпозиція робіт

На основі отриманих даних побудуємо діаграму Ґантта (рис. 2.2) Критичний шлях зображено червоним (сіреньким) кольором. Критичними операціями є операції лежать на критичному шляху З використанням паралельних процесів можна скоротити час роботи на добу приблизно втричі. Така можливість також додає буфер часу, на випадок недодержання плану.

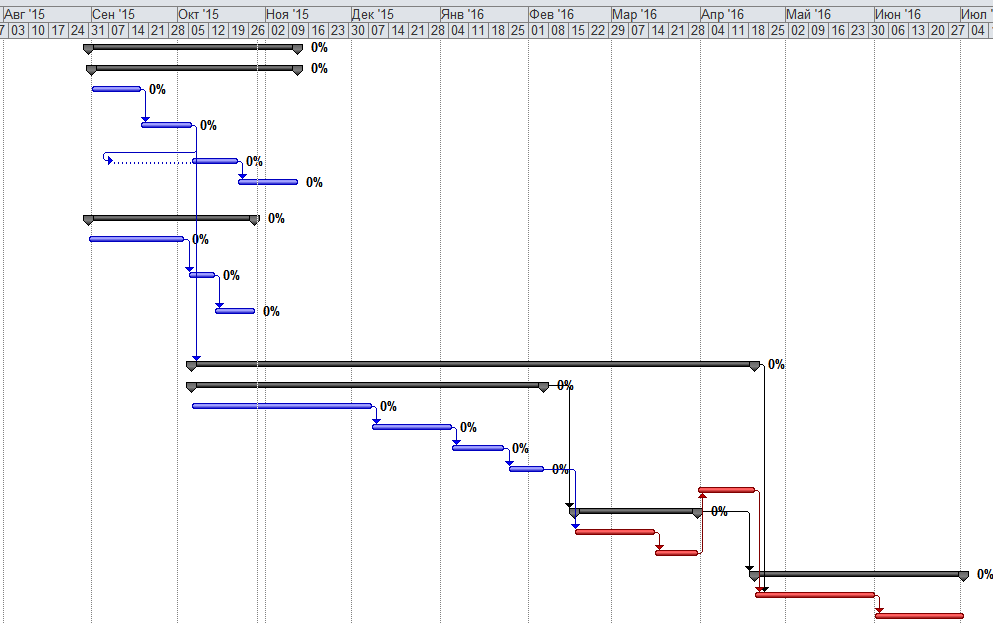


Рисунок 2.2 – Діаграма Ґантта

**5.2) Определение объемов работ по одному из методов оценки -**

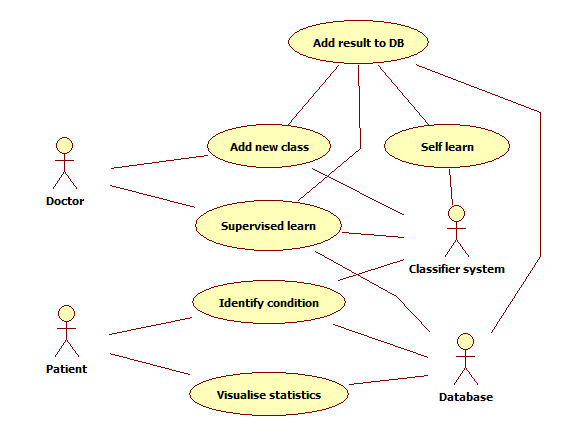


Рисунок 1.1 - Діаграма варіантів використання

## Етап 1. Оцінка акторів.

Таблиця 1.1 – Оцінка акторів

|  |  |  |  |
| --- | --- | --- | --- |
| Індивідуальні актори | | Множник | Назва |
| 1 | Простий | 1 | Database |
| 2 | Середньої складності | 2 | Classifier system |
| 3 | Складний | 3 | Doctor |
| 4 | Складний | 3 | Paient |

UAW = 2 \* 3 + 2 \* 1 + 1\*1 = 9

Складні актори – взаємодія з користувачем, Прості – інша система з визначеним API (у даному випадку взаємодія з СУБД).

## Етап 2. Оцінка варіантів використання.

Для зручності оцінювання буду використовувати підхід до визначення типу варіанту використання, що базується на кількості кроків, що будуть задіяні при реалізації варіантів використання.

Простий – менше за 3 кроки

Середньої складності – від 4 до 7 кроків

Складний – більше за 7 кроків

Таблиця 1.2 – Оцінка варіантів використання

|  |  |  |  |
| --- | --- | --- | --- |
| Складність | | Множник | Варіант використання |
| 1 | Простий | 5 | Visualize statistics |
| 2 | Середньої складності | 10 | Identify condition |
| 3 | Складний | 15 | Self learn |
| 4 | Складний | 15 | Add new class |
| 5 | Складний | 15 | Supervised learn |
| 6 | Простий | 5 | Add result to databse |

UUCW = 5\*2+10\*1+15\*3 = 65

UCP = UAW + UUCW

UCP = 9 + 65 = 74

Проект використовує нові технології, для реалізації яких немає готових компонентів.

Припустимо, що %REUSE = 10%

UCP = (100 – 10)/100 \* 74 = 66.6

## Етап 3. Оцінка технічних факторів

Таблиця 1.3 – Оцінка технічних факторів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Технічний фактор | | Множник | Відносний вплив | Опис |
| 1 | Розподілена система | 2 | **3** | Система не потребує розподілених обчислень, але їх можна застосувати |
| 2 | Час відклику важливий | 1 | **3** | Час відгуку сильно впливає на розробку системи |
| Продовження таблиці 1.3 | | | | |
| 3 | Ефективність кінцевого користувача | 1 | **1** | Ефективність користування має середній вплив на розробку |
| 4 | Складні внутрішні обчислення | 1 | **5** | Застосовуються алгоритми класифікації |
| 5 | Фокус на повторному використанні коду | 1 | **2** | При оновленні або додаванні даних до групи повторно використовується код |
| 6 | Простота інсталяції | 0,5 | **2** | При встановленні програми знадобиться допомога спеціаліста |
| 7 | Зручність використання | 0,5 | **5** | Система зосереджена на взаємодії з користувачем, тому простота використання має значний вплив на розробку |
| 8 | Кросплатформенність | 2 | **4** | Система є платформо-незалежною, але це не має важливий вплив на розробку |
| 9 | Легке внесення змін | 1 | **5** | При розробці системи береться до уваги той факт, що система повинна бути здатною до легкого внесення змін |
| 10 | Паралельні обчислення | 1 | **1** | Паралельні обчислення можуть мати місце, але не використовуються на даний момент |
| 11 | Виділений захист | 1 | **2** | Система потребує захист персональних даних пацієнтів |
| 12 | Залежність від чужого коду | 1 | **3** | Система може використовуватися зовнішніми системами/акторами |
| 13 | Навчання користувача | 1 | **2** | Потрібно провести тренінг користувача |
| Обчислений TCF | |  | **1,015** |  |

## Етап 4. Оцінка зовнішніх факторів

Таблиця 1.4 – Оцінка зовнішніх факторів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Зовнішній фактор | | Множник | Відносний вплив | Опис |
| 1 | Ознайомленість з проектом | 1,5 | **3** | Знання команди предметної області має значний вплив на розробку |
| 2 | Досвід розробки | 0,5 | **2** | Досвід команди має середній вплив |
| 3 | Знання ООП | 1 | **4** | Бажаними є знання ООП |
| 4 | Здібності аналітика | 0,5 | **4** | Важливо правильно отримати вимоги |
| 5 | Мотивація | 1 | **3** | Мотивація має середній вплив |
| 6 | Стабільність вимог | 2 | **4** | Вимоги не часто будуть змінюватись, тому це не дуже впливає на розробку |
| 7 | Участь частково зайнятих працівників | -1 | **3** | Всі розробники – студенти, тому частково зайняті |
| 8 | Складність мови програмування | -1 | **3** | Мова програмування – Java, також потрібні знання SQL. Для проекту вистачить місяця на опанування мов |
| Обчислений EF | |  | **0,905** |  |

## Етап 5. Результуючі оцінки.

Скоректовані UCP обчислюються за формулою AUCP = UCP \* TCF \* EF.

AUCP = 66.6 \* 1.015 \* 0.905 = 61.18.

У даному випадку 1 UCP відповідає 28 робочим годинам. Отже, загальний час на розробку дорівнює: 28 \* 61.18 = 1714 (год)