

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему: **«Розроблення інформаційної підсистеми голосового управління
для розпізнавання мовних команд у медичній сфері»**

Виконав: студент групи 4141

_____ Трибрат Д. А.

(підпис)

Керівник роботи:

викладач

(посада, науковий ступінь вчене звання)

(підпис)

Меліксетов О.І.

м. Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. Гайда А.Ю.

" ____ " ____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти "бакалавр"

Студенту Трибрату Денису Андрійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Розроблення інформаційної підсистеми голосового управління для розпізнавання мовних команд у медичній сфері

Керівник роботи викладач Меліксетов Олег Ігорович

Затверджені наказом ректора № 295-уч від "27" березня 2025 року.

2. Термін подання роботи: 20 червня 2025 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів)

Аналіз предметної області та постановка задачі. Проектування прогнозової інформаційної системи. Розробка прогнозової інформаційної системи. Охорона праці.

5. Перелік презентаційних матеріалів Актуальність, концепція системи, Загальна схема роботи, Порівняльний аналіз систем, Діаграма Use Case аналітика підприємства, Схема роботи підприємства після впровадження нової інформаційної системи, ER-діаграми інформаційної бази даних, Розгортання системи, Екранні форми розробки, Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Меліксетов О.І.	03.02.2025	22.02.2025
2	Меліксетов О.І.	22.02.2025	21.03.2025
3	Меліксетов О.І.	21.03.2025	25.04.2025
4	Меліксетов О.І.	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проекту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент



(підпис)

Керівник роботи



(підпис)

Трибрат Д.А.

(ПБ)

Меліксетов О.І.

(ПБ)

АНОТАЦІЯ

Бакалаврська робота на тему «Розробка інформаційної підсистеми голосового управління для розпізнавання мовних команд у медичній сфері» присвячена аналізу існуючих методів розпізнавання мовлення, оцінці методів розпізнавання та вибору найбільш адекватного для побудови підсистеми розпізнавання.

Виконано аналіз методів розпізнавання: штучно-нейронної мережі, прихованих моделей Маркова, спектрально-часовий, формантний та безпосередньо за реалізацією. За експертною оцінкою було обрано формантний метод як найбільш адекватний щодо вимог підсистеми розпізнавання.

У роботі було розроблено концепцію підсистеми та сформульовані вимоги до підсистеми голосового керування. Ключовими було обрано вимоги гнучкості, масштабованості та низьку потребу до ресурсів.

Також у роботі виконані розділи охорони праці.

Бакалаврська робота викладена на 115 сторінках друкованого тексту, містить 19 рисунків, 12 таблиць, 3 додатки та список використаних джерел з 7 найменувань.

Робота виконана українською мовою.

Ключові слова: голосове керування, медична сфера, нейронні мережі, модель Маркова.

ABSTRACT

The bachelor's thesis on the topic "Development of an information subsystem of voice control for recognition of speech commands in the medical field" is devoted to the analysis of existing speech recognition methods, evaluation of recognition methods and selection of the most adequate one for building a recognition subsystem.

The analysis of recognition methods was performed: artificial neural network, hidden Markov models, spectral-temporal, formant and directly by implementation. According to expert assessment, the formant method was chosen as the most adequate in terms of the requirements of the recognition subsystem.

The work developed the concept of the subsystem and formulated the requirements for the voice control subsystem. The key requirements were chosen as flexibility, scalability and low resource requirements.

The work also includes sections on labor protection.

The bachelor's thesis is set out on 115 pages of printed text, contains 19 figures, 12 tables, 2 appendices and a list of used sources from 7 names.

The work is done in Ukrainian.

Keywords: voice control, medical field, neural networks, Markov model.

ЗМІСТ

Вступ	4
РОЗДІЛ 1. Дослідження існуючих методів розпізнавання мовних команд	6
1.1 Загальні положення	6
1.2 Задача класифікації та ідентифікації	8
1.3 Існуючі системи розпізнавання мовних команд та концепції їх побудови	8
1.4 Існуючі методи розпізнавання мовлення	10
1.5 Методи, що використовують безпосереднє порівняння сигналів	11
1.6 Штучні нейронні мережі	12
1.7 Методи, основані на застосуванні прихованої марківської моделі	15
1.8 Методи, що базуються на спектральному аналізі мовного сигналу	20
1.9 Метод отримання оцінки спектру сигналу	22
1.10 Аналіз ефективності методів	26
1.11 Вибір методу розпізнавання голосових команд	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ Інформаційної підсистеми голосового керування процесами у операційній системі	32
2.1 Опис предметної галузі	32
2.2 Формування вимог до підсистеми голосового керування процесами у операційній системі	33
2.3 Цілі, критерії та обмеження створення підсистеми голосового керування	34
2.4 Функції та задачі підсистеми голосового керування, що створюється	35
2.5 Розробка концепції підсистеми голосового керування процесами	35
2.6 Функціонально-вартісний аналіз	38
2.7 Постановка задачі на розробку підсистеми	46
Висновки з розділу 2	46
РОЗДІЛ 3. Розробка проектних рішень щодо підсистеми голосового керування процесами у операційній системі	48
3.1 Загальносистемні рішення	48
3.2 Аналіз концептуальної моделі та функціональних вимог	49
3.3 Рішення з організаційного забезпечення	53
3.4 Організаційні рішення	55
3.5 Рішення з інформаційного забезпечення	56
3.6 Діаграма потоків даних	56
3.7 Концептуальна модель даних	57
3.8 Логічна модель даних	57
3.9 Фізична модель даних	58
3.10 Діаграма класів	61
3.11 Діаграма компонентів	64
3.11 Рішення з технічного забезпечення	67
3.12 Модель розгортання системи	68
3.13 Рішення з математичного забезпечення	69
3.14 Рішення з програмного забезпечення	74
3.15 Алгоритмічне рішення	76
3.14 Рішення стосовно тестування підсистеми	84
Висновки з розділу 3	86
РОЗДІЛ 4. ОХОРОНА ПРАЦІ	87
4.1 Загальні принципи охорони праці	87
Висновки до розділу 4	96
Висновки	98
Список використаних джерел	101
Додаток А – Інструкція користувача	102
Додаток Б - Технічне завдання на розробку	105
Додаток В – Фрагмент програмного коду	108

ВСТУП

Голосове керування різноманітними процесами та діями є зручним для людини, оскільки є більш природнім ніж натискання клавіш клавіатури або використання маніпуляторів. Подібна функція є життєво необхідною для людей з фізичним обмеження для комфортної роботи з комп'ютером.

На даний момент часу існують досить багато систем, що орієнтовані на вирішення задачі голосового керування процесами у операційній системі. Але в багатьох випадках структура та принцип роботи системи є комерційною таємницею та не розголошується. Така поведінка дозволяє стримувати конкурентів у цієї галузі, що дає можливість комерційним підприємствам отримувати значний прибуток. Але подібне утримування, відповідно стримує подальший розвиток галузі як такої.

До складу операційних систем Vista та Seven від корпорації Microsoft входить модуль голосового керування та набору тексту. Але зазначений модуль підтримує лише англійську мову, тобто для користувачів, котрі не знають англійської, використання цієї підсистеми є неможливим. Стосовно інших операційних систем, наприклад операційної системи Linux, вони не мають належного рівня системи голосового керування.

Також голосове керування використовується у керування різноманітними приладами, наприклад найбільш яскравий приклад використання подібної системи уявляє собою використання у мобільних приладах. Багато мобільних телефонів мають функцію голосового набору телефону з контактної книжки.

Існує тенденція до впровадження системи голосового до складу автомобілю. Так виробники автомобілів прагнуть підвищити безпеку та комфорт водію, тому голосове керування використовується для керування бортовою електронікою, наприклад керування радіоприймачем. Це дозволить не відводити очі від дороги та не займати руки водія зайвими маніпуляціями.

Слід зазначити, що при проектуванні підсистеми голосового керування слід брати до уваги такі аспекти як умови застосування, тобто можливу

наявність шуму при вимовленні голосової команди, обсяг словника команд та ін. Для більш складних систем має значення контекст вимовляння команди, що потребує побудови мовної моделі окремої мови, або групи мов. Також суттєвим є мовлення користувача, оскільки деякі методи розпізнавання потребують вимовляння команди рівним голосом та з необхідною швидкістю.

Отже існує проблема побудови підсистеми розпізнавання мовлення, котра могла б використовуватися у складі операційної системи або окремого приладу. Оскільки у результаті дослідження предметної галузі не було знайдено рішення, котре дозволяє масштабувати підсистему розпізнавання, при розробці подібної системи слід відвести окрему увагу можливості масштабування системи, тобто підвищити гнучкість системи. Подібне рішення дозволить збільшити галузі у котрих можна застосовувати подібну підсистему.

Сучасний розвиток технологій програмування дозволяє розробляти системи, котрі можуть виконуватися у більшості операційних систем. Оскільки сфера використання системи у багатьох випадках безпосередньо залежить від операційної системи, де розгортається підсистема, то збільшення кількості операційних систем, що можуть підтримувати підсистему, що розроблюється, збільшить кількість сфер використання цієї підсистеми.

Як було зазначено вище, деякі операційні системи мають у своєму складі системи розпізнавання, та інші таких засобів не мають. Та засоби, що використовуються у операційних системах для розпізнавання мовлення неможливо перенести на іншу операційну систему, що диктується комерційною політикою розробника.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ РОЗПІЗНАВАННЯ МОВНИХ КОМАНД

1.1 Загальні положення

Акустичний мовний сигнал являє собою акустичні коливання повітряного середовища, що породжуються артикуляційними органами[1]. Кожний звук, що вимовляє людина має свої унікальні амплітудно-частотні характеристики, за якими інша людина може його розрізнити. Мовний апарат уявляє собою багато-резонансну систему, тому мовний сигнал уявляє собою суму декількох гармонійних коливань.

Мовний акустичний сигнал може бути представлено у аналоговому вигляді, тобто неперервної зміни амплітуди коливання, що задається напругою. Такий спосіб представлення сигналу застосовується на даний момент часу досить рідко, оскільки не має необхідної масштабованості. Але це представлення сигналу найбільш відповідає фізиці процесу. Для відображення звукових коливань у аналогову форму використовується досить простий пристрій – мікрофон. Слід зазначити, що так або інакше аналогове представлення використовується у якості інструменту при роботі з сигналами, але зберігання сигналу виконується за допомогою цифрового представлення сигналу.

Найбільш сучасним способом представлення сигналу є цифровий. Даний засіб представлення передбачає зберігання не неперервного сигналу, а дискретного, тобто сукупності амплітуд коливань (відліках), що отримані у часових зразах. Величина, що є зворотною до часу відстані між двома послідовними часовими зрізами, називається частотою дискретизації. Частота дискретизації обирається виходячи з фізичного процесу, що описується аналоговим сигналом та теореми Котельникова (Найквіста-Шеннона), котра каже, що, якщо аналоговий сигнал має обмежений спектр, то він може бути однозначно відновлено без втрат за своїми дискретними

відліками, що взято строго більше подвійної максимальної частоти спектра сигналу.

У якості перетворювача аналогової амплітуди у цифровий вигляд використовується спеціальний електронний пристрій – аналогово-цифровий перетворювач(АЦП), він перетворює вхідну напругу у цифровий код. Цей пристрій має декілька ключових характеристик: частоту дискретизації та розподільну здатність. Розподільна здатність – це кількість двійкових розрядів, що відводиться для представлення одного відліку. У сучасних звукових картах, що використовуються у персональних комп'ютерах використовуються АЦП, котрі підтримують частоту дискретизації до 192кГц та мають розподільна здатність рівну 24 бита.

Слід зазначити, що людське мовлення знаходиться у частотному діапазоні до 3500 Гц, тобто для представлення людської мови у чисельному вигляді достатньо частоти дискретизації рівної 8000Гц. Подана частота використовується у телекомунікація для кодування людського мовлення. На Рисунку 1.1 наведено амплітудно-часову діаграму, що відображає коливання при мовленні слова «один».

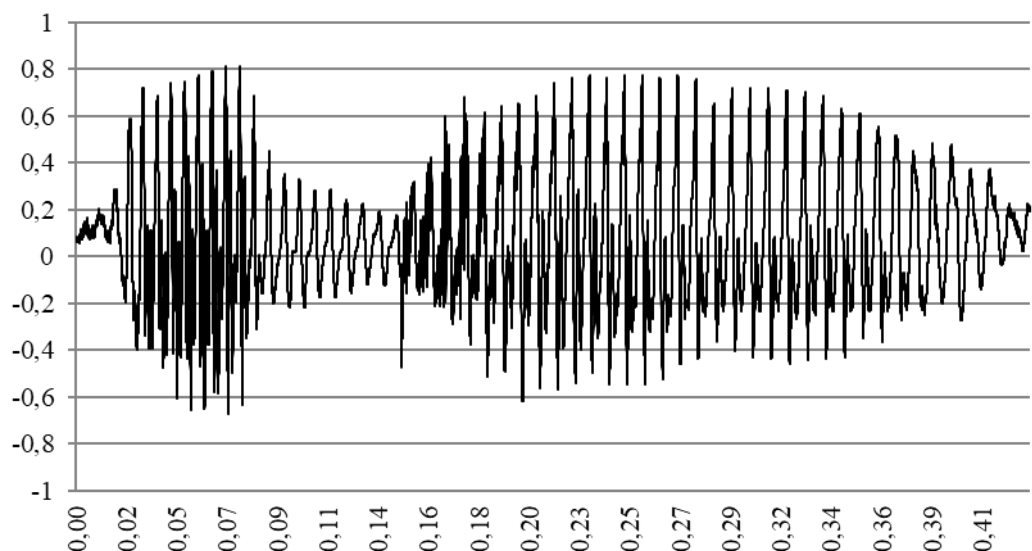


Рисунок 1.1 – Реалізація слова «один»

1.2 Задача класифікації та ідентифікації

Задача класифікації це формалізована задача, у котрій є сукупність об'єктів, що розподілені деяким чином на класи. Задано кінцеву множину об'єктів(вибірку), для котрих відомо, к яким класам вони належать. Таку множину називають. Класова приналежність інших об'єктів не відома. Задача класифікації об'єкту полягає у тому щоб коректно виявити приналежність окремого об'єкту до одного з класів.

Опишемо задачу математично. Нехай X – множина об'єктів, Y – множина класів об'єктів. Існує деяка цільова залежність-відображення $y^*: X \rightarrow Y$, значення котрої відомі тільки для кінцевої навчальної вибірки. Необхідно побудувати алгоритм $a: X \rightarrow Y$, що може класифікувати довільний об'єкт $x \in X$.

Побідну концепцію переносять на мовні команди, де виділяються сукупність класів команд, та вибірка реалізацій, що відповідають одному з класів мовних команд. Тобто у якості об'єктів виступають звукові реалізації мовних команд, а класи команд уявляють собою значення, що вкладається у мовну команду, наприклад команду-дію.

1.3 Існуючі системи розпізнавання мовних команд та концепції їх побудови

Перший пристрій для розпізнавання мови з'явився в 1952 році, він міг розпізнавати вимовлені людиною цифри. В 1964 році на ярмарку комп'ютерних технологій у Нью-Йорку було представлено пристрій IBM Shoebox.

Комерційні програми з розпізнавання голосу з'явилися на початку дев'яностих років. Зазвичай їх використовують люди, які через травми рук не в змозі набирати велику кількість тексту. Ці програми (наприклад, Dragon Naturally Speaking, VoiceNavigator) переводять голос користувача в текст,

таким чином, розвантажуючи його руки. Надійність перекладу у таких програм не дуже висока, але з роками вона поступово покращується.

Збільшення обчислювальних потужностей мобільних пристроїв дозволило і для них створити програми з функцією розпізнавання мови. Серед таких програм варто відзначити Microsoft Voice Command, яка дозволяє працювати з багатьма прикладними програмами за допомогою голосу. Наприклад, можна включити відтворення музики в плеєрі або створити новий документ.

У комп'ютерах Apple Macintosh у налаштуваннях системи є вбудована функція Speech, що здатна аналізувати команди користувача при натисканні певної клавіші, або якщо команді користувача передують ключові слова.

Ще однією цікавою програмою є Speereo Voice Translator — голосовий перекладач. SVT здатна розпізнавати фрази, вимовлені англійською мовою, і "говорити" у відповідь переклад на одній з вибраних мов.

Інтелектуальні мовні прикладні програми, що дозволяють автоматично синтезувати і розпізнавати людську мову, є наступним етапом розвитку інтерактивних голосових систем IVR. Використання інтерактивного телефонного програмного забезпечення в даний час є не даниною моді, а життєвою необхідністю. Зниження навантаження на операторів контакт-центрів і секретарів, скорочення витрат на оплату праці і підвищення продуктивності систем обслуговування — ось тільки деякі переваги, що доводять доцільність подібних програм.

Однак, прогрес не стоїть на місці і останнім часом у телефонних інтерактивних програмах все частіше використовують системи автоматичного розпізнавання і синтезу мови. У цьому випадку спілкування з голосовим порталом стає природнішим, оскільки вибір в ньому може бути здійснений не тільки за допомогою тонового набору, але і за допомогою голосових команд. При цьому системи розпізнавання є незалежними від дикторів, тобто розпізнають голос будь-якої людини. Основною перевагою голосових систем є доброзичливість до користувача — він позбавляється від

необхідності продиратися крізь складні і заплутані лабіринти голосових меню. Тепер достатньо вимовляння мети дзвінка, після чого голосова система автоматично перемістить абонента в потрібний пункт меню.

Наступним кроком технологій розпізнавання мови можна вважати розвиток так званих Silent Speech Interfaces (SSI) (Інтерфейсів Безмовного Доступу). Ці системи обробки мовлення базуються на одержанні й обробці мовних сигналів на ранній стадії артикуляції. Даний етап розвитку розпізнавання мови викликаний двома істотними вадами сучасних систем розпізнавання: надмірна чутливість до шумів, а також необхідність чіткої і ясної вимови при звертанні до системи розпізнавання. Підхід, заснований на SSI, полягає в тому, щоби використовувати нові сенсори, що не піддаються впливу шумів, як доповнення до оброблених акустичних сигналів.

Розвиток цієї технології обумовлено таким феноменом сприйняття як феномен Мак-Гурка. Даний ефект демонструє взаємодію між слухом та зором людини при сприйнятті мовлення. Так поданий ефект передбачає, у разі людської співбесіди недолік звукової інформації можна компенсувати артикуляційними проявами мовлення.

Існуючі системи розпізнавання розподіляються за своєю архітектурою на системи, що працюють на клієнті, та «клієнт-серверні» системи. Перший вид системи передбачає обробку сигналу і вирішення задачі класифікації на тому електронному пристрої на якому проводиться запис звуку. Друга архітектура системи передбачає обробку та вирішення задачі класифікації на сервері. Кожен з підходів має свої переваги і недоліки, наприклад, перший метод організації потребує значної продуктивності системи, а другий необхідність організації каналу зв'язку з сервером.

1.4 Існуючі методи розпізнавання мовлення

Існує сукупність підходів або методів направлених на вирішення проблеми класифікації голосових команд. Слід зазначити, що більшість методів, що використовуються у системах голосового керування досить

часто комбінуються з іншими. Більшість методів розпізнавання мовлення не конкурують з іншими, тобто використання окремого підходу не виключає можливості використання іншого для покращення розпізнавання.

Кожен з методів розпізнавання має свої переваги та недоліки, та також потребує специфічних даних для аналізу щоб вирішити задачу класифікації.

Окремий метод уявляє собою сукупність рекомендації щодо обору опорних даних або організації системи.

Отже перейдемо до огляду кожного з існуючих методів та рекомендацій щодо його впровадження.

1.5 Методи, що використовують безпосереднє порівняння сигналів

Одним з базових методів порівняння сигналів є порівняння сигналів за реалізацією. Такі методи використовують два сигнали еталонний та вхідний та розглядають їх як послідовну сукупність точок (вибірку), котра далі аналізується базовими методами статистики або порівняння сигналів, наприклад обчислення середньої похибки за реалізаціями. Тобто при порівнянні сигналів не використовується частотні характеристики сигналу.

Для того щоб вважати, що сигнал з деякою похибкою ідентичний еталонному використовують формулу:

$$E = \sum_{i=1}^N f(a_i, b_i)$$

Де: a_i – еталонний сигнал;

b_i – вхідний сигнал (сигнал, що порівнюється)

$f(x, y)$ – деяка функція, що реалізує порівняння значень точок, здебільше використовують наступну функцію: $f(x, y) = |x - y|$ або $f(x, y) = (x - y)^2$.

Тобто сигнал вважається прийнятий якщо отримана помилка менше деякого критичного значення: $E < E_{кр}$. Де критичне значення обирається для окремого сигналу відповідно до кількості точок та допуску.

Також для реалізації цього методу використовують функцію:

$$E = \frac{1}{N} \sum_{i=1}^N f(a_i, b_i)$$

Тобто значення похибки нормується відповідно до кількості точок та характеризує середню похибку.

Також у якості додаткових критеріїв використовуватися значення максимальної похибки та інш.

Подані методи порівняння сигналів мало ефективні при порівнянні звукових сигналів, оскільки не вважають можливість зсуву окремих фонем команди за часом, інтенсивності вимовляння, швидкості мовлення та багатьох інших факторів, що відповідають людській мові.

1.6 Штучні нейронні мережі

Виходячи з назви подібний клас методів використовує у якості інструментарію штучну нейронну мережу. Як відомо, штучна нейронна мережа являє собою сукупність математичних моделей, та також їх програмні або апаратні реалізації, що побудовані за принципом організації біологічних нейронних мереж.

На сьогоднішній момент часу застосування штучних нейронних мереж охоплює велику кількість різноманітних технічних та практичних галузей. Окрім того штучні нейронні мережі часто використовують у якості інструменту ідентифікації образів, у тому числі звукового зображення. Існує окрема великий науково-практичний напрямок що присвячено розробці, навчанню та експлуатації систем побудованих з використанням технології штучної нейронної мережі.

Концепція побудови типової штучної нейронної мережі полягає у побудові системи з окремих вузлів (нейронів), що з'єднані між собою та організовані шарами. Це є типовою концепцією побудови, але можуть також використовуватись більш складні конструкції з нейронів.

На Рисунку 1.1 наведено простий приклад нейронної мережі, котра організована у чотирьох шарах: де:

- перший шар. Вхід до нейронної мережі, до цих вузлів подається зовнішній вплив. Наведений приклад має 2 входу.
- другий та третій. Ці шари уявляють собою внутрішні шари нейронної мережі, кожен з котрих має 3 нейрону у своєму складі.
- четвертий шар. Уявляє собою вихід нейронної мережі.

Приведений приклад має один вихід.

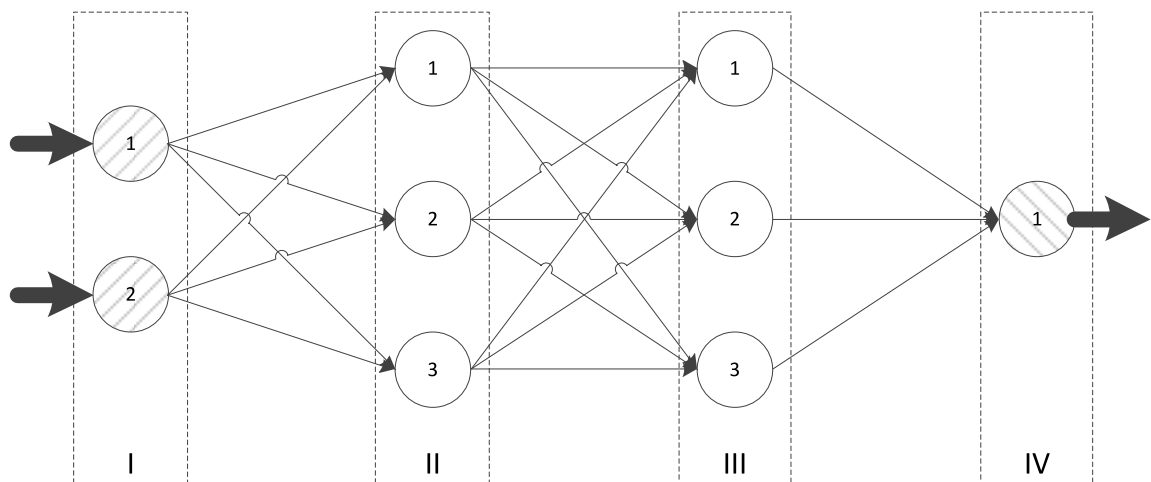


Рисунок 1.2 – Приклад нейронної мережі

Окремо взятий вузол нейронної мережі уявляє собою модель нейрону. На Рисунку 1.2 наведено схематичне зображення нейрону. Кожен окремий нейрон має сукупність входів $x_1..x_n$, що передають сигнали з зовнішнього джерела або з попереднього слою нейронної мережі. Кожному з входів відповідає ваговий коефіцієнт $k_1..k_n$, завдяки котрому корегується вплив вхідних сигналів на окремий нейрон. Далі аналізується сумарний вплив зважених входів завдяки активаційній функції, значення якої обчислюється як:

$$y = f\left(\sum_{i=1}^n k_i x_i\right)$$

Отримане значення активаційної функції, відповідно до слою у котрому розташовано нейрон, передається наступному слою нейронів або

вважається вихідним значенням штучної нейронної мережі. У разі якщо нейрон розташований у внутрішньому або вхідному шарі вихідний сигнал нейрона передається до всіх нейронів наступного шару. У разі якщо нейрон розташований у вихідному шарі – вихід нейрону уявляє собою вихід нейронної мережі та значення на виході може бути інтерпретовано зовнішньою системою відповідно до задачі.

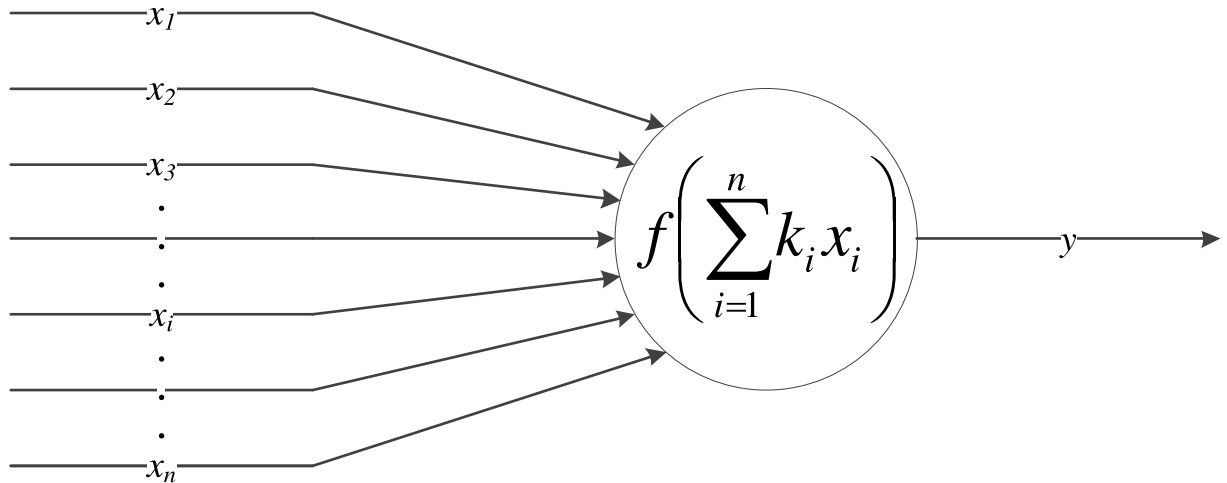


Рисунок 1.3 – Просте зображення нейрону

Ключовим етапом експлуатації нейронної мережі є етап навчання, при якому підбираються значення коефіцієнтів k для кожного нейрону. В залежності від обраної структури нейронної мережі для вирішення питання навчання використовується відповідний метод навчання. Для типової структури, що наведена на Рисунку 1.1, можна використати метод зворотного поширення похибки для навчання, котрий ґрунтується на концепції навчання з вчителем.

Первинним етапом розробки концепції системи (підсистеми), що використовує штучну нейронну мережу є вибір базових даних, що подається до нейронної мережі. В застосуванні до розпізнавання мовлення досить тяжко обрати відповідну сукупність ознак, котрі були б здатні класифікувати мовні команди з достатньою малою похибкою.

Також недоліком системи є погана масштабованість оскільки при доданні нової команди, обов'язково повинна виконуватись досить

ресурсоємний процес навчання мережі, та також можливий випадок при котрому буде необхідна перебудова топології нейронної мережі.

Але штучна нейронна мережа проявляє досить добрий потенціал, тобто має малу похибку при вирішенні задачі класифікації за обраними числовими ознаками.

1.7 Методи основані на застосуванні прихованої марківської моделі

Прихована марківська модель – статистична модель, що імітує роботу процесу, схожого на марківський процес з невідомими параметрами. Задачею являється розгадування невідомих параметрів на основі параметрів, що спостерігаються. Отримані параметри можуть бути використані у подальшому аналізі, наприклад для розпізнавання образів та класифікації.

Основні напрями використання приховані марківські моделі є розпізнавання мовлення, письма, дій та у біо-інформатиці.

Діаграма наведена на Рисунку 1.3 відображає структуру прихованої марківської моделі. Овали уявляють собою змінні з випадковим значенням. Випадкова змінна $x(t)$ уявляє собою значення прихованої змінної у момент часу t . Випадкова змінна $y(t)$ – це значення змінної, що спостерігається у момент часу t . Стрілки зображають умовні залежності.

З діаграми ясно, що значення прихованої змінної $x(t)$ (у момент часу t) залежить тільки від значення прихованої змінної $x(t-1)$ (у момент часу $t-1$). Це називається властивістю Маркова. Хоча у той же час значення змінної $y(t)$, що спостерігається, залежить тільки від значення прихованої змінної $x(t)$ (обидві в один момент часу).

Тобто ймовірність побачити послідовність $Y = y(0), y(1), \dots, y(L-1)$ довжиною L дорівнює:

$$P(Y) = \sum_X P(Y|X)P(X),$$

де сума проходить по всім можливим послідовностям прихованих вузлів $X = x(0), x(1), \dots, x(L-1)$. Метод підрахунку повним перебором

значень $P(Y)$ дуже трудомісткий для багатьох задач з реального життя у силу того, що кількість можливих послідовностей прихованих вузлів дуже велика. Але використання процедури «вперед-назад» дозволяє суттєво підвищити швидкість обчислень.

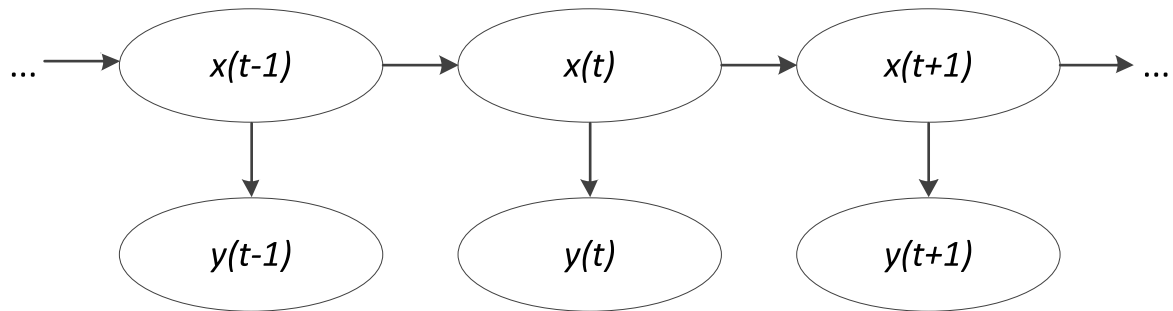


Рисунок 1.4 – Загальний вигляд прихованої марківської моделі

Існують три основні задачі та відповідно алгоритми пов'язані з прихованими марківськими моделями:

- Задача обчислення ймовірності виникнення послідовності, при заданих параметрах моделі та послідовності. Задачу вирішує алгоритм «вперед-назад».
- Задача виявити найбільш підходящу послідовність прихованих вузлів, котра найбільш точно описує задану параметрами модель. Задачу вирішує алгоритм Вітербі.
- Задача «тренування» прихованої марківської моделі відповідно до вхідної та вихідної послідовностей. Задачу вирішує алгоритм Баума-Велша.

Отже для вирішення задачі класифікації голосових команд використовуючи метод прихованих марківських моделей необхідно розглянути принцип побудови алгоритмів Вітербі та Баума-Велша, оскільки перший відповідає безпосередньо за класифікацію, та другий використовується для підбору необхідних коефіцієнтів.

Алгоритм Вітербі належить до класу алгоритмів динамічного програмування. У 1967 році Вітербі розробив та проаналізував алгоритм, у котрому реалізується перетворення (декодування) вхідних параметрів у

вихідні, побудовані за принципом максимальної правдоподібності. У алгоритмі зменшується обчислювальне навантаження за допомогою застосування особливої структури – решітки коду. Алгоритм включає у себе обчислення міри подібності між сигналом, отриманим у момент відповідний момент часу та усіма путями решітки, що входять до складу стану у цій момент часу. Алгоритм Вітербі базується на так званому принципі «жадібності», тобто на кожній ітерації обирається найбільш вірогідний шлях.

Роботу цього алгоритму можна представити діаграмою наведеною на Рисунку 1.4. На кожному етапі обирається перехід з найкращим показником правдоподібності. Таким чином будується ланцюг переходів, що дозволяє за цією послідовністю давати оцінку приналежності окремої команди до відповідного класу.

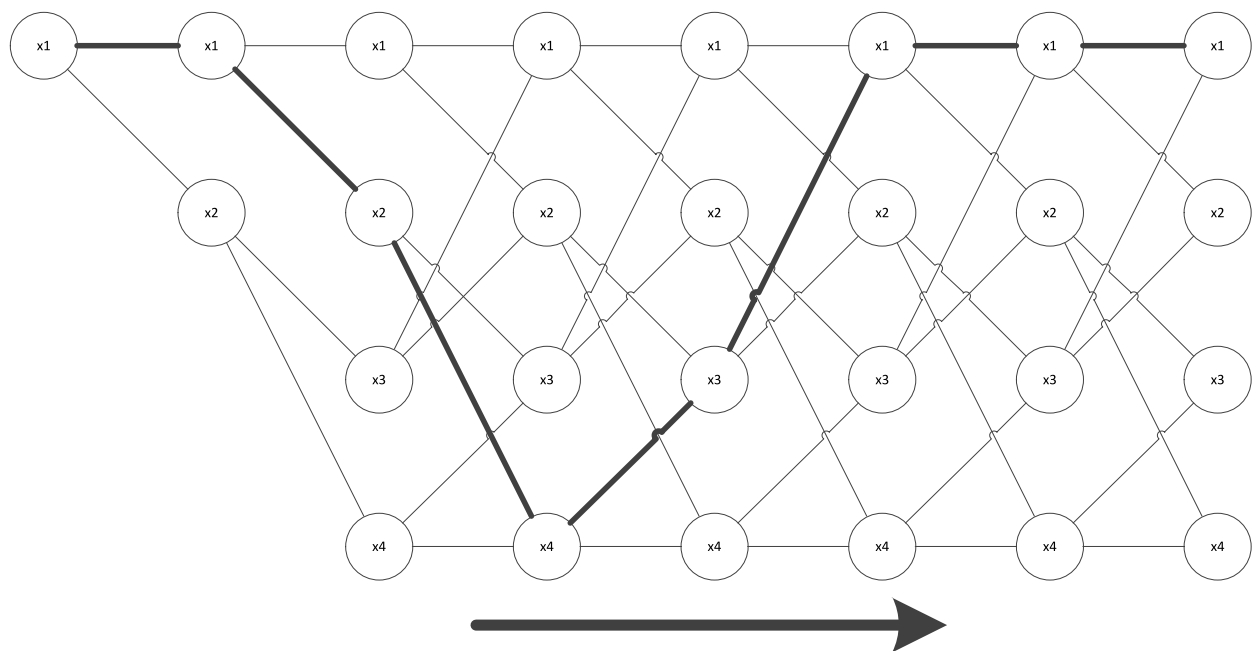


Рисунок 1.5 – Загальний вигляд прихованої марківської моделі

Отже наведеній алгоритм Вітербі потребує визначених прихованої марківської моделі: кінцевої множини станів та параметрів переходів. Для вирішення цієї задачі використовується алгоритм Баума-Велша.

Прихована модель Маркова – це ймовірнісна модель випадкових змінних $\{Y_1, \dots, Y_t, X_1, \dots, X_t\}$. Змінні Y_i – відомі дискретні спостереження, а X_i – «приховані» дискретні величини. У рамках прихованої моделі Маркова

існує два незалежних твердження, що забезпечують збіжність цього алгоритму.

- i -та прихована змінна при відомій $(i-1)$ -ой змінній, незалежна від усіх попередніх $(i-1)$ змінних тобто:

$$P(X_i | X_{i-1}, Y_{i-1}, \dots, X_1, Y_1) = P(X_i | X_{i-1})$$

- i -е відоме спостереження залежить тільки від i -ого стану, тобто не залежить від часу:

$$P(Y_i | X_i, X_{i-1}, Y_{i-1}, \dots, X_1, Y_1) = P(Y_i | X_i)$$

Алгоритм, що описується, використовує концепцію «припущень та максимізацій» для пошуку максимальної ймовірнісної оцінки параметрів прихованій моделі Маркова.

Змінна X_i – дискретна змінна, котра може приймати одно з N значень $(1..N)$. Будемо вважати, що дана модель Маркова визначається як $P(X_i | X_{i-1})$, однорідна за часом, тобто не залежна від відліку часу. Тоді можна задати $P(X_i | X_{i-1})$ як незалежну від часу стохастичну матрицю переміщень $A = \{a_{ij}\} = p(X_t = j | X_{t-1} = i)$. Окремий випадок для часу $t=1$ визначається начальним розподілом $\pi_i = P(X_1 = i)$.

Припустимо, що система знаходиться у стані j у момент часу t якщо $X_t=j$. Послідовність заданих станів визначається як $x = (x_1, \dots, x_T)$, де $q_t \in \{1 \dots N\}$ є станом у момент часу t .

Спостереження може мати одно з L можливих значень, $Y_t \in \{y_1, \dots, y_L\}$. Ймовірність заданого вектору спостережень у момент часу t для стану j визначається як $b_j(y) = P(Y_t = o_t | X_t = j)$. Де $B = \{b_{ij}\}$ – матриця L на N . Задана послідовність спостережень у визначається як $Y = (Y_1 = y_1, \dots, Y_T = y_T)$.

Відповідно, ми можемо описати приховану модель Маркова за допомогою $\lambda = (A, B, \pi)$. При заданому векторі спостережень у алгоритм Баума-Велша знаходить $\lambda^* = \max_{\lambda} P(Y | \lambda)$. λ – максимізує ймовірність спостережень у.

Початковими даними є $\lambda = (A, B, \pi)$ з випадковим початковими умовами. Алгоритм виконує ітеративне оновлення параметру λ до сходження у одній точці.

Алгоритм використовує дві процедури: пряму та зворотною.

Пряма процедура виконується наступним чином. Визначимо $\alpha_i(t) = p(Y_1 = y_1, \dots, Y_t = y_t, X_t = i | \lambda)$, що являється ймовірністю отримання заданої послідовності $y_1, \dots, y_t$ для станату i у момент часу t .

$\alpha_i(t)$ можна отримати рекурсивним методом:

- $\alpha_i(1) = \pi_i \cdot b_i(O_1)$;
- $\alpha_j(t+1) = b_j(O_{t+1}) \sum_{i=1}^N \alpha_i(t) a_{ij}$.

Зворотний процес дозволяє обчислити ймовірність кінцевої заданої послідовності $y_1, \dots, y_t$, за умов, якщо процес було почато з початкового стану i , у момент часу t .

Можна обчислити $\beta_i(t)$:

- $\beta_i(T) = 1$;
- $\beta_i(t) = \sum_{j=1}^N \beta_j(t+1) a_{ij} b_j(Y_{t+1})$.

Використовуючи α та β можна обчислити наступні значення:

$$\gamma_i(t) \equiv p(X_t = i | Y, \lambda) = \frac{\alpha_i(t) \beta_i(t)}{\sum_{j=1}^N \alpha_j(t) \beta_j(t)}$$

$$\xi_{ij}(t) \equiv p(X_t = i, X_{t+1} = j | Y, \lambda) = \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}{\sum_{i=1}^N \sum_{j=1}^N \alpha_i(t) a_{ij} \beta_j(t+1) b_j(Y_{t+1})}$$

Маючи γ та ξ , можна визначити:

$$\bar{\pi}_i = \gamma_i(1)$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$\bar{b}_i(k) = \frac{\sum_{t=1}^T \delta_{Y_t, y_k} \gamma_i(t)}{\sum_{t=1}^T \gamma_i(t)}$$

Використовуючи нові значення A , B , π , ітерації продовжуються до сходження.

1.8 Методи, що базуються на спектральному аналізі мовного сигналу

До цього класу належить досить велика кількість методів, але кожен з них у якості основного інструменту використовує спектр сигналу.

Найбільш уживаними є два методи, що використовують спектр, це спектрально-часовий метод та формантний метод.

Спектрально-часовий метод аналізу

Спектрально-часовий метод аналізу, призводить аналіз спектру на досить малому проміжку часу та аналізує зміну спектру за унікальними ознаками:

- Частоти первой, второй, третьей форманты, их энергии
- Продолжительность формантных участков
- Наклон соответственного формантного участка
- Уровни максимальной и минимальной энергии
- Моментальная мощность сигнала
- Продолжительность переходных участков
- Частота основного тона

На Рисунку 1.6 наведено спектрально-часову діаграму, що являє собою об'єкт дослідження спектрально-часового методу розпізнавання мовлення.

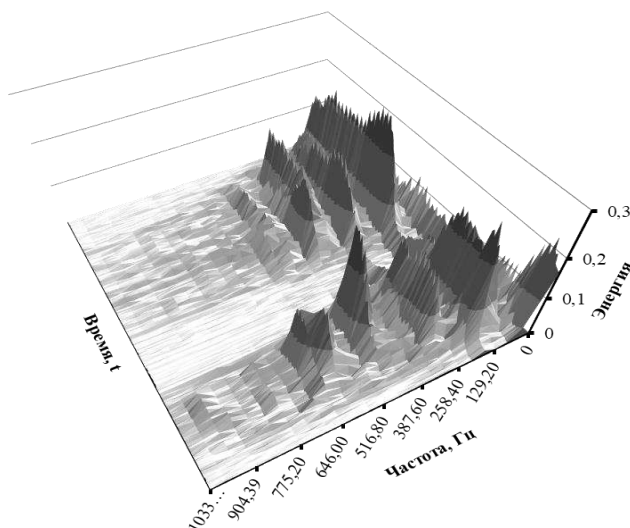


Рисунок 1.6 – Спектрально-часова діаграма слова «один»

Для різних задач розпізнавання спектрально-часовий аналіз проводять за різними набором відмінних ознак.

Даний метод потребує значних обчислювальних ресурсів, що необхідні для побудови миттєвих зрізів спектру.

Форматний метод

Формантний метод за оцінкою спектру усієї звукової реалізації призводить аналіз ключових гармонік за частотами та інтенсивністю.

Форманта – це термін фонетики, котрий визначає акустичну характеристику звуків мовлення (здебільше голосних), зв'язану з рівнем частоти голосового тону, що утворюють тембр звуку.

У термінології розпізнавання мовлення форманта, це імпульсний пік на спектрограмі сигналу на відповідній частоті та енергією, що відповідає наявності відповідної гармоніки.

Формантною зоною називається зона, котра зайнята окремою формантою. Ця зона визначається тривалістю, максимальною енергією, площею та частотою.

На Рисунку 1.7 та 1.8 зображено спектрограми мовних команд «один» та «два», відповідно.

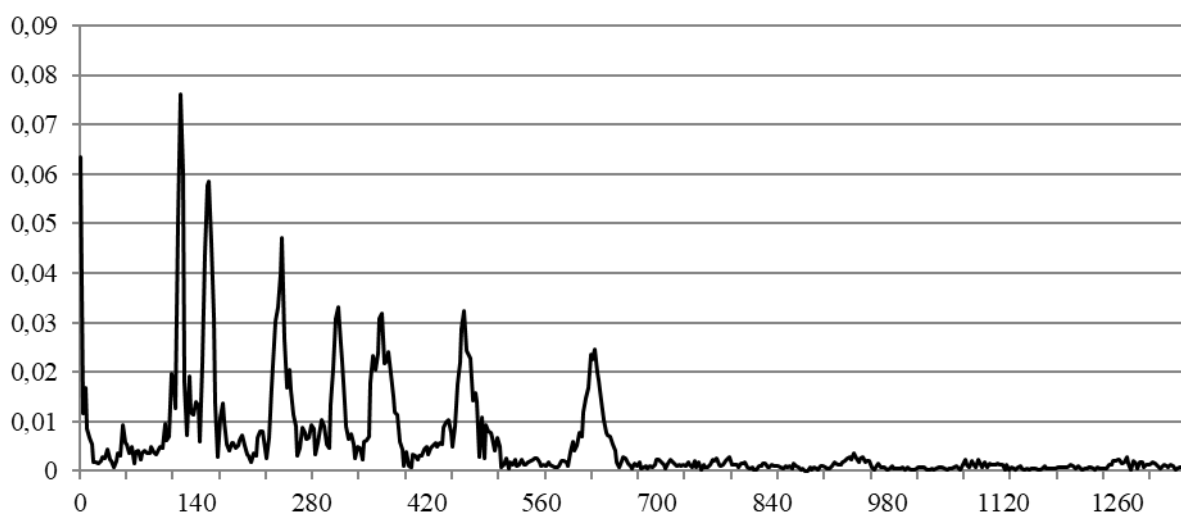


Рисунок 1.7 – Спектр слова «один»

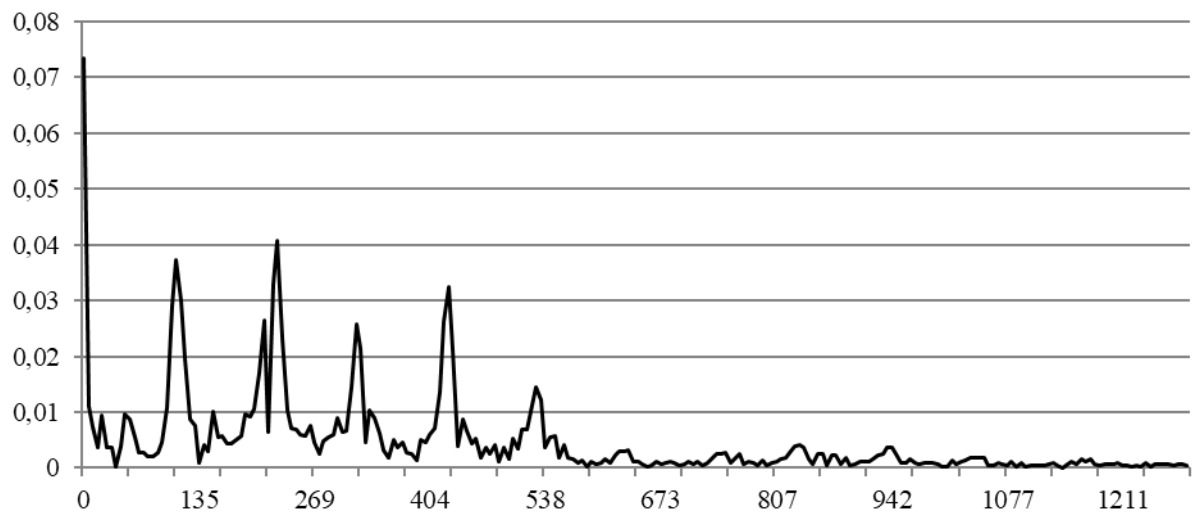


Рисунок 1.8 – Спектр слова «два»

На основі декількох реалізацій однієї команди будується усереднений спектр команди. Далі виділяються основні форманти спектру, стільки скільки виявлено на спектрограмі. Як було встановлено експериментально кількість формантних зон варіюється від 3 до 8.

На основі формантних зон будується еталонне значення команди у наступному вигляді:

$$Q: S(\nu_i \pm dx_i) = s_i \pm dy_i, \quad i \in [1..n] \quad (1)$$

Де n – кількість значущих формант

Тобто кожна формантна зона має центральну частоту ν_i та допустиме відхилення dx_i , та ця формантна зона у спектрі повинна мати енергію s_i з допустимим відхиленням dy_i .

Команда вважається прийнятою якщо спектр піддослідного сигналу має точки, що задовольняють всі еталонні формантні зони, тобто якщо виконується вираз (1).

1.9 Метод отримання оцінки спектру сигналу

Для вирішення задачі обчислення оцінки спектру використовують перетворення Фур'є. Подане перетворення дозволяє отримати розкладання початкової функції на елементарні складові частини – гармонійні коливання з різними частотами. Таке розкладання на частоти є досить зручним для

аналізу мовного сигналу, оскільки дозволяє визначити ключові гармоніки сигналу.

Дискретне перетворення Фур'є перетворює сукупність чисел $a_0, a_1, \dots, a_{n-1}$ у сукупність чисел $b_0, b_1, \dots, b_{n-1}$, таких що $b_k = \sum_{j=0}^{n-1} a_j \varepsilon^{kj}$, де $\varepsilon^n = 1$ та $\varepsilon^k \neq 1$, якщо $0 < k < n$.

Швидке перетворення Фур'є (ШПФ) дозволяє обчислити дискретне перетворення Фур'є (ДПФ) зі складністю алгоритму меншу за $O(N^2)$, що потребуються для обчислення ДПФ за початковою формулою. Формула необхідна для обчислення швидкого перетворення Фур'є має вигляд:

$$X_m = \sum_{n=0}^{M-1} x_{2n} a_M^{nm} + e^{-2\pi i \frac{m}{N}} \sum_{n=0}^{M-1} x_{2n+1} a_M^{nm}$$

де:

x – вибірка для якої використовується перетворення Фур'є;

M – цілове значення перетворення Фур'є;

a_M^{nm} – коефіцієнт, що обчислюється за формулою $a_M^{nm} = e^{-2\pi i \frac{m}{N}}$.

Оскільки отриманий спектр сигналу після швидкого перетворення Фур'є є досить зашумленим (Рисунки 1.6 та 1.7) та для аналізу найчастіше потрібен згладжений спектр то необхідно отримати застосувати методи згладжування оцінки спектра. Одним з найбільш уживаних та ефективних методів згладжування спектру – це застосування методу ковзкого вікна, що виконується за наступною формулою:

$$\bar{S}_i = \frac{1}{n} \sum_{j=-\frac{n}{2}}^{\frac{n}{2}} S_{j+i} W(j)$$

де:

S – початковий спектр

$W(j)$ – функція-коефіцієнт, котра відповідає окремому методу згладжування.

У Таблиці 1.1 наведені найбільш уживані функції $W(j)$.

Таблиця 1.1 – Вираження для деяких віконних функцій

Назва вікна	Функція
Прямокутне вікно	$W(j) = 1$
Синус-вікно	$W(j) = \sin\left(\frac{\pi \cdot j}{n-1}\right)$
Вікно Ланкзоса	$W(j) = \text{sync}\left(\frac{2j}{n-1} - 1\right) \text{sync}(x) = \frac{\sin(\pi x)}{\pi x}$
Вікно Бартлету	$W(j) = 1 - \left \frac{2j}{n-1} - 1\right $
Вікно Ханна	$W(j) = 0.5 - 0.5 \cos\left(\frac{2\pi \cdot j}{n-1}\right)$
Вікно Бартлета-Ханна	$W(j) = a_0 - a_1 \cdot \left \frac{j}{n-1} - 0.5\right - a_2 \cos\left(\frac{2\pi \cdot j}{n-1}\right)$ $a_0 = 0.62 \quad a_1 = 0.48 \quad a_2 = 0.38$
Вікно Хеммінгу	$W(j) = 0.54 - 0.46 \cos\left(\frac{2\pi \cdot j}{n-1}\right)$
Вікно Блекмана	$W(j) = a_0 - a_1 \cos\left(\frac{2\pi \cdot j}{n-1}\right) + a_2 \cos\left(\frac{4\pi \cdot j}{n-1}\right)$ $a_0 = 0.42 \quad a_1 = 0.5 \quad a_2 = 0.08$
Вікно Блекмана-Харрису	$W(j) = a_0 - a_1 \cos\left(\frac{2\pi \cdot j}{n-1}\right) - a_2 \cos\left(\frac{4\pi \cdot j}{n-1}\right) - a_3 \cos\left(\frac{6\pi \cdot j}{n-1}\right)$ $a_0 = 0.35875 \quad a_1 = 0.48829 \quad a_2 = 0.14128 \quad a_3 = 0.01168$
Вікно Наталла	$W(j) = a_0 - a_1 \cos\left(\frac{2\pi \cdot j}{n-1}\right) - a_2 \cos\left(\frac{4\pi \cdot j}{n-1}\right) - a_3 \cos\left(\frac{6\pi \cdot j}{n-1}\right)$ $a_0 = 0.355768 \quad a_1 = 0.48396 \quad a_2 = 0.144232 \quad a_3 = 0.012604$
Віно Блекмана-Наталла	$W(j) = a_0 - a_1 \cos\left(\frac{2\pi \cdot j}{n-1}\right) - a_2 \cos\left(\frac{4\pi \cdot j}{n-1}\right) - a_3 \cos\left(\frac{6\pi \cdot j}{n-1}\right)$ $a_0 = 0.3635819 \quad a_1 = 0.4891775 \quad a_2 = 0.1365995 \quad a_3 = 0.0106411$
Вікно з плоскою вершиною	$W(j) = \sum_{i=0}^4 a_i \cos\left(\frac{2i \cdot \pi \cdot j}{n-1}\right) \quad a_0 = 1.0 \quad a_1 = 1.93$ $a_2 = 1.29 \quad a_3 = 0.388 \quad a_4 = 0.032$
Вікно Гауса	$W(j) = e^{-0.5 \left(\frac{j - \frac{n-1}{2}}{\sigma \frac{n-1}{2}}\right)^2}$

Порівняльний аналіз існуючих методів розпізнавання

Кожен з описаних вище методів має свою унікальну концепцію тобто кожен з них потребує детального дослідження, оскільки наведені методи не являються тривіальними та можуть розглядатись як окрема підсистема.

Проблеми пов'язані з впровадженням методів

Виходячи з опису та структури наведених методів розпізнавання можна зробити висновок, що кожен з них має свої характеристики щодо гнучкості та масштабованості.

Отже метод, що базується на штучної нейронної мережі має добрі показники при розпізнаванні але потребує досить складних математичних маніпуляцій при реалізації. Окремою проблемою стоїть розробка ефективної топології штучної нейронної мережі та реалізація можливості нарощення функціоналу.

Методи, що використовують приховані марківські моделі, як і методи штучних нейронних мереж мають досить добрі показники розпізнавання, але слабкою стороною цього методу як и попереднього є погана масштабованість.

Метод спектрально-часового аналізу мовлення дозволяє отримувати характеристики сигналу за часом. Тобто цей метод може бути використано як джерело інформації для двох попередніх методів.

Слід зазначити, що використання знань про структуру окремо взятої мови – кількості фонем та їх зв'язків, може дати необхідні для побудови системи з використанням штучних нейронних мереж та прихованих марковських процесів. Це дослідження являє собою окремий напрямок розвитку систем голосового керування. Головною перевагою цього підходу є значне підвищення ефективності розпізнавання мовлення для окремої мови. Та з іншої сторони впровадження такої системи не передбачає можливості використання команд на інших языках. Також слід зазначити, що розробка відповідної системи знань щодо фонетичної будови окремої мови передбачає колосальний обсяг досліджень, здебільше пов'язаний з лінгвістичною

стороною питання. Масштабованість у цих методах може бути реалізована на більш високому рівні. Тобто розпізнавання ведеться по фонемах та за ними складаються слова, котрі далі аналізуються на предмет виникнення цільових команд. Але цей підхід дозволить використовувати систему розпізнавання мовлення і для більш широкої галузі – набору тексту.

Найбільш невибагливим з точки зору використання ресурсів є формантний метод розпізнавання. Цей метод дозволяє розпізнавати кінцеву кількість команд, але це реалізується з невеликим використанням ресурсів. Головною перевагою цього метода є можливість досить просто реалізувати додавання нових команд до системи. Але при збільшенні команд суттєво збільшується похибка. Тобто головним недоліком цієї системи є невелика кількість команд (до 100) , що можуть бути використані при експлуатації системи.

1.10 Аналіз ефективності методів

Для того щоб обрати найбільш оптимальний метод для розпізнавання використаємо метод експертного оцінювання наведених вище методів за характеристиками по десятибальній шкалі.

Проведемо порівняльний аналіз наведених вище методів. Оцінимо кожен з методів за наступними характеристиками, кожна з характеристик має свій коефіцієнт значущості, котрий було отримано за експертною оцінкою за допомогою попереднього дослідження.

- Розробка. Характеризує складнощі, що можуть виникнути при розробці ($k=0,15$);
- Розмір словника. Характеризує наскільки великим може бути словник команд, щоб не зменшити ефективність системи ($k=0,1$);
- Швидкість роботи методу. Характеризує наскільки швидко метод може вирішити задачу класифікації ($k=0,15$);

- Гнучкість. Характеризує можливість модернізації системи($k=0,1$);
- Експлуатація. Характеризує складність експлуатації системи побудованої на основі методу ($k=0,05$);
- Додання нових команд. Характеризує простоту додання нових команд до системи ($k=0,1$);
- Потреби до ресурсів системи. Характеризують вимоги до продуктивності системи на яку розгортається система($0,1$);
- Надійність розпізнавання. Характеризує ефективність розпізнавання($k=0,3$).

У Таблиці 1.2 наведено експертні оцінки методів за характеристиками.

Таблиця 1.2 – Порівняльний аналіз методів розпізнавання

Характеристика \ Метод	Штучні нейронні мережі	Приховані марківські моделі	Форматний метод	За реалізацією
Розробка	3	2	9	10
Розмір словника	9	9	7	5
Швидкість роботи методу	6	7	9	6
Гнучкість	7	6	9	2
Експлуатація	8	8	8	5
Додання нових команд	6	7	8	2
Потреби до ресурсів системи	5	5	8	5
Надійність розпізнавання	8	8	7	2

Відповідно з наведених експертних оцінок було отримано результуючі значення оцінки кожного з методів.

- Штучні нейронні мережі: 6,7
- Приховані марківські моделі: 6,75
- Форматний метод: 7,95
- За реалізацією: 4,15

1.11 Вибір методу розпізнавання голосових команд

Для вирішення задачі розпізнавання мови у підсистемі голосового керування необхідно обрати один з методів наведених вище. Порівняльний аналіз методів наведено у Таблиці 1.2.

Відповідно до наведених вище результатів експертної оцінки оберемо для використання у підсистемі голосового керування метод, що базується на форматному аналізі спектру, оскільки цей метод вважається досить простим у реалізації та має не великі потреби щодо ресурсів системи, та також додавання нових команд для цього методу є досить простим. Також обраний метод базується на використанні найбільш точних характеристиках сигналу – оцінках спектру, що однозначно характеризують сигнал.

Випробування обраного методу розпізнавання

Для дослідження ефективності розробленої системи розпізнавання оберемо критерій ймовірності розпізнавання. Оцінка коефіцієнту ймовірності отримується апостеріорі, тобто експериментально, наступним чином:

$$P_i = \frac{n_i}{N_i}, \quad (2)$$

де:

n_i – кількість успішних дослідів з ідентифікацій i -ої мовної команд

N_i – загальна кількість дослідів з ідентифікації i -ої мовної команди

P_i – оцінка ймовірності розпізнавання i -ої мовної команди

Отже для досліду було взято три мовні команди – звукові реалізації слів «один», «два» та «три». Для кожної з команд було побудовані еталонні значення спектру у вигляді (1). У таблиці 1.3 наведено еталонні значення формант для команди «один».

Таблиця 1.3 – Еталонні значення формант для команди «один»

Форманта	v_i	dx_i	s_i	dy_i
1	140	48	0,05	0,014
2	245	24	0,03	0,008
3	340	50	0,021	0,0076
4	470	34	0,019	0,0064

Було проведено дослід з розпізнавання приведених вище команд. Для кожної команди було проведено по 50 експериментів з розпізнавання. Дослід проводився у тиші, без додаткових апаратних або програмних засобів вилучення шуму. У Таблиці 1.4 наведені результати досліду.

Таблиця 1.4 – Результати досліду розпізнавання

Команда	n_i	N_i	P_i
«один»	45	50	0,9
«два»	43	50	0,86
«три»	46	50	0,92

Тобто середній показник ймовірності розпізнавання дорівнює 89,3%.

Вибір методу сегментації

Вирішення проблеми сегментації звукового потоку, тобто виділення окремих команд з потоку, є однією з ключових проблем при розробці системи голосового керування.

Точність сегментації у значній мірі визначає надійність системи автоматичного розпізнавання мовлення.

Відомі два типи алгоритмів сегментації мовного сигналу: сегментація за умови, що відома послідовність фонем даної фрази, та сегментація на

сонові ступеня зміни акустичних характеристик сигналу. Іноді виділяють в окремий вид комбінований алгоритм.

Існує досить багато методів, що дозволяють сегментувати не тільки за наявністю команди мовних даних чи ні, але ще і розділяти окремі фонemi. Оскільки не планується виділяти окремі фонemi з мовного потоку, досить методу, що дозволить встановити наявність команди у мовному потоку. Існують наступні методи сегментації мовлення:

- Сегментація за усередненим нормованим спектром
- Сегментація динамічним детектором
- Сегментація за кореляції між рівновіддалених спектрах

Оберемо для впровадження у підсистему перший з наведених методів. Він дозволяє з необхідною точністю вирішити задачу сегментації та являється найбільш простим у реалізації.

Тобто цей метод вважає що сигнал має деяку команду якщо спектр сигналу має хоча б одну форманту з цільовим значенням енергії.

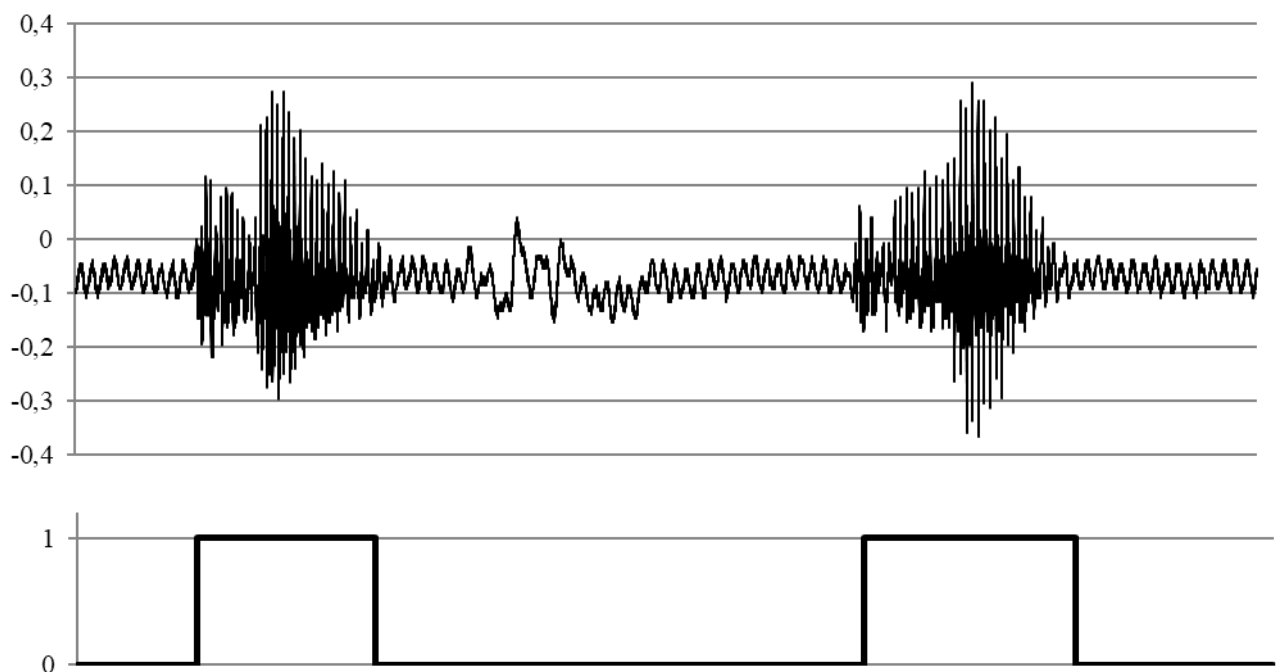


Рисунок 1.9 – Сегментація команд

Висновки з розділу 1

В даному розділі було проведено дослідження існуючих методів розпізнавання мовлення.

Існуючі методи розпізнавання можна умовно розділити на 3 групи.

Методи першої групи будуються на штучних нейронних мережах. Ці методи мають досить добрий показник розпізнавання та головними недоліками є необхідність дослідження пов'язані з розробкою структури нейронної мережі та досить низький показник гнучкості.

Методи другої групи використовують приховані марківські моделі для вирішення задачі класифікації. Ця група методів дозволяє побудувати лінгвістичну модель окремої мови. Недоліком цього класу методів є складна реалізація структури та велика обчислювальна складність.

Методи, що базуються на спектральному аналізі голосового сигналу показали, що мають досить не поганий потенціал при побудові систем розпізнавання. Оскільки спектрально-часовий метод аналізу сигналу надає багато інформації щодо поведінки сигналу за часом його можна використати як джерело необхідних ознак для перших двох груп методів. Цей метод має недолік – він потребує значної обчислювальної спроможності системи.

Отже при розробці підсистеми розпізнавання мовлення будемо використовувати формантний метод, оскільки він зарекомендував себе досить ефективним та гнучким.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ГОЛОСОВОГО КЕРУВАННЯ ПРОЦЕСАМИ У ОПЕРАЦІЙНІЙ СИСТЕМІ

2.1 Опис предметної галузі

Голосове керування різноманітними процесами та діями є зручним для людини, оскільки є більш природнім ніж натискання клавіш клавіатури або використання маніпуляторів. Подібна функція є життєво необхідною для людей з фізичним обмеження для комфортної роботи з комп'ютером.

На даний момент часу існують досить багато систем, що орієнтовані на вирішення задачі голосового керування процесами у операційній системі. Але в багатьох випадках структура та принцип роботи системи є комерційною таємницею та не розголошується. Така поведінка дозволяє стримувати конкурентів у цієї галузі, що дає можливість комерційним підприємствам отримувати значний прибуток. Але подібне утримування, відповідно стримує подальший розвиток галузі як такої.

До складу операційних систем Vista та Seven від корпорації Microsoft входить модуль голосового керування та набору тексту. Але зазначений модуль підтримує лише англійську мову, тобто для користувачів, котрі не знають англійської, використання цієї підсистеми є неможливим. Стосовно інших операційних систем, наприклад операційної системи Linux, вони не мають належного рівня системи голосового керування.

Також голосове керування використовується у керування різноманітними приладами, наприклад найбільш яскравий приклад використання подібної системи уявляє собою використання у мобільних приладах. Багато мобільних телефонів мають функцію голосового набору телефону з контактної книжки.

Існує тенденція до впровадження системи голосового до складу автомобілю. Так виробники автомобілів прагнуть підвищити безпеку та комфорт водію, тому голосове керування використовується для керування

бортовою електронікою, наприклад керування радіоприймачем. Це дозволить не відводити очі від дороги та не займати руки водія зайвими маніпуляціями.

Слід зазначити, що при проектуванні підсистеми голосового керування слід брати до уваги такі аспекти як умови застосування, тобто можливу наявність шуму при вимовленні голосової команди, обсяг словника команд та ін. Для більш складних систем має значення контекст вимовляння команди, що потребує побудови мовної моделі окремого мови, або групи мов. Також суттєвим є мовлення користувача, оскільки деякі методи розпізнавання потребують вимовляння команди рівним голосом та з необхідною швидкістю.

Отже існує проблема побудови підсистеми розпізнавання мовлення, котра могла б використовуватися у складі операційної системи або окремого приладу. Оскільки у результаті дослідження предметної галузі не було знайдено рішення, котре дозволяє масштабувати підсистему розпізнавання, при розробці подібної системи слід відвести окрему увагу можливості масштабування системи, тобто підвищити гнучкість системи. Подібне рішення дозволить збільшити галузі у котрих можна застосовувати подібну підсистему.

2.2 Формування вимог до підсистеми голосового керування процесами у операційній системі

Сучасний розвиток технологій програмування дозволяє розробляти системи, котрі можуть виконуватися у більшості операційних систем. Оскільки сфера використання системи у багатьох випадках безпосередньо залежить від операційної системи, де розгортається підсистема, то збільшення кількості операційних систем, що можуть підтримувати підсистему, що розроблюється, збільшить кількість сфер використання цієї підсистеми.

Як було зазначено вище, деякі операційні системи мають у своєму складі системи розпізнавання, та інші таких засобів не мають. Та засоби, що використовуються у операційних системах для розпізнавання мовлення

неможливо перенести на іншу операційну систему, що диктується комерційною політикою розробника. От же слід реалізувати підтримку найбільш поширених операційних систем, таких як сімейство операційних систем Microsoft Windows та сімейство операційних систем Linux.

Також слід передбачити можливість застосування підсистеми голосового керування у мобільних приладах, для задоволення потреб невеликих систем.

Існує багато підходів для реалізації подібного функціоналу, отже необхідно обрати найбільш ефективний, що задовольнив би інші вимоги до підсистеми.

2.3 Цілі, критерії та обмеження створення підсистеми голосового керування

Інформаційна підсистема голосового керування призначена для підвищення ефективності при роботі з ПК, тобто створення додаткового інтерфейсу взаємодії користувача та ПК.

Підставами для розробки є відсутність уніфікованих рішень щодо голосового розпізнавання у існуючих операційних системах та приладах. Розробка подібної підсистеми дозволить з незначними змінами, що стосуються специфіки роботи, використовувати її у багатьох операційних системах.

Основна мета створення інформаційної підсистеми – підвищення ефективності діяльності користувача персональним комп'ютером, підвищення оперативності керування.

Основна мета використання подібної підсистеми – це підвищення ступеню динамізму при керування комп'ютером та надання можливості працювати з комп'ютером людям з обмеженнями руху.

Основні критерії створення інформаційної підсистеми:

- Соціальний критерій. Інформаційна система розрахована на користувачів з різною комп'ютерною підготовкою. Необхідний зручний та інтуїтивно зрозумілий інтерфейс та принцип роботи з нею;
- Економічний критерій. Автоматизована інформаційна підсистема голосового керування створюється безкоштовно та використання ліцензії GNU GPL до підсистеми, що розроблюється, надає змогу удосконалювати інформацій підсистему у разі необхідності любим бажаючим.

2.4 Функції та задачі підсистеми голосового керування, що створюється

Інформаційна підсистема голосового керування призначена для підвищення ефективності керування операційною системою.

Отже підсистема повинна виконувати наступні задачі:

- класифікувати вимовлену користувачем мовну команду;
- мати можливість збільшити обсяг мовних команд;
- бути гнучкою та масштабованою.

Основними функціями інформаційної підсистеми голосового керування є:

- розрізняння вимовлених мовних команд;
- додавання нових мовних команд;
- можливість підключення специфічних джерел аудіо інформації;
- можливість використання сторонніх методів класифікації.

2.5 Розробка концепції підсистеми голосового керування процесами

Опис результатів аналізу вимог до підсистеми, що розроблюється

Виходячи з описаних вище вимог можна зробити висновок, що для реалізації цільової підсистеми необхідно ретельно підійти до обору концепції підсистеми, що розроблюється. Отже виділимо основні вимоги, що стосуються концептуального моделювання системи:

- Закласти можливість зміни модулів збору аудіо інформації, класифікації;
- Закласти можливість додавання реакцій підсистеми на кожну окрему голосову команду;
- Можливість з найменшими затратами провести інтегрування підсистеми до складу окремої системи (операційної системи, приладу)

Отже перейдемо до розробки концепції підсистеми, що задовольняють вимогам.

Розробка концепції інформаційної підсистеми

Виходячи з вимог наведених у попередніх пунктах, необхідно обрати найбільш ефективну концепцію інформаційної системи, що розроблюється, для реалізації достатньої гнучкості та масштабованості. На Рисунку 2.1 наведено концептуальну модель інформаційної системи розпізнавання, котра задовольняє вимогам попередніх пунктів.

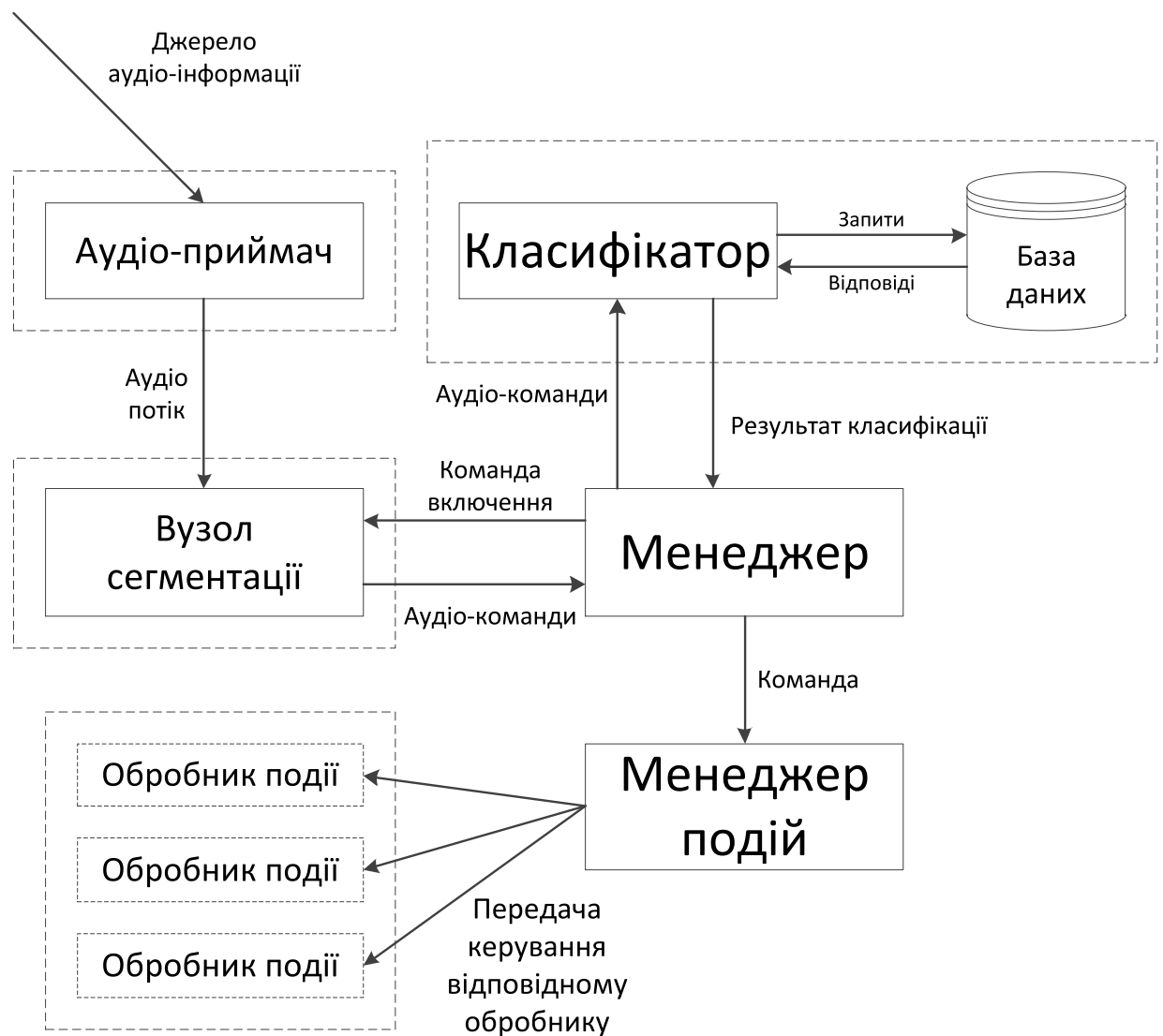


Рисунок 2.1 – Концептуальна модель підсистеми голосового керування

Інформаційна підсистема голосового керування складається з наступних частин:

- **Аудіо-приймач.** Задача цього вузла реалізовувати взаємодію між апаратними пристроями, наприклад мікрофоном, та блоком сегментації. Цей вузол може бути замінено на інший, відповідно до системи у яку підсистема інтегрується;
- **Вузол сегментації.** Задача цього вузла полягає у аналізі вхідного потоку аудіо-інформації та виділення з нього окремих команд за деяким алгоритмом. Слід зазначити, що цей вузол також може бути замінено у разі необхідності;

– **Класифікатор.** Основна задача цього вузла – вирішення задачі класифікації. Тобто по вхідної аудіо-команді та існуючої бази, що входить до складу класифікатору, блок вирішує належить вхідна мовна команда до якогось з існуючих класів або ні. Та передає відповідь-результат до менеджера. Виходячи з вимоги масштабованості цей вузол може бути заміщено на більш ефективний, а бо більш відповідний до потреб системи, до якої проводиться інтеграція;

– **База даних.** Цей елемент входить до складу класифікатора, оскільки окрема реалізація класифікатору може використовувати різноманітні еталони даних. Тобто класифікатор перероблює вхідні еталони мовні команди до формату необхідному окремій реалізації класифікатору;

– **Менеджер.** Задача цього вузла організація взаємодії між класифікатором, вузлом сегментації та менеджером подій. Оскільки до вузла сегментації та класифікатору застосовуються вимоги гнучкості, поточний вузол має вміти працювати з наведеними вузлами за деяким внутрішнім принципом. Головна задача цього вузла маршрутизування даних між вузлами, та надання зовнішнього інтерфейсу користувачеві.

– **Менеджер подій.** На основі команди, переданої з менеджера менеджер подій виконує обробку команди передаючи керування окремому обробнику подій.

– **Обробник подій.** Сукупність цих блоків реалізують безпосередню взаємодію з зовнішньою системою виконуючи відповідні дії, приписані до цього обробника. Таке рішення дозволяє реалізувати масштабування системи при доданні нових команд та відповідних дій.

2.6 Функціонально-вартісний аналіз

Для вибору найбільш ефективного варіанту концепції інформаційної підсистеми голосового керування використовується метод функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА, Activity Base Costing, ABC) – метод визначення вартості та інших характеристик виробів, послуг і споживачів, що використовують як основу функції та ресурси, що задіяні у виробництві, маркетингу, продажі, доставці, технічній підтримці, наданні послуг, обслуговуванні клієнтів, а також забезпечення якості. В основі методу функціонального аналізу лежить функціональний підхід, відповідно до якого об’єктом аналізу є не сам продукт (система, підсистема, програмне забезпечення), а функції, що він виконує.

ФВА – один з методів, що дозволяє вказати на можливі шляхи поліпшення вартісних показників. Ціль створення ФВА – моделі для удосконалення діяльності підприємств – досягти поліпшення у роботі підприємств по показниках вартості, трудомісткості та продуктивності. Проведення розрахунків по ФВА – моделі дозволяє отримати великий обсяг ФВА-інформації для ухвалення рішення.

Функціонально-вартісний аналіз складається з двох основних етапів:

- функціональний аналіз;
- вартісний аналіз.

Функціональний аналіз

Виходячи з вимог до компонентів інформаційної підсистеми та засобів розробки приведемо можливі варіанти реалізації системи – Таблиця 2.1.

У таблиці наведено дві апаратні конфігурації, при цьому наводити більш продуктивні конфігурації немає сенсу. Щодо програмного забезпечення то можна зазначити, що усі наведені операційні системи є безкоштовними, операційна система MS Windows 7/XP була отримана за ліцензією MSDN AA.

Таблиця 2.1 – Варіанти засобів розробки

Варіант	Технічне забезпечення	Програмне забезпечення		
	Платформа	Операційна система	Мова програмування	Середовище розробки
1	CPU: Intel Core Duo T2390 RAM: 2Gb	MS Windows XP	Delphi	RAD Studio 2010

2	HDD: 160 Gb		C++	RAD Studio 2010
3			C++	Eclipse
4			Java	NetBeans
5				Eclipse
6		MS Windows 7	C++	RAD Studio 2010
7			Java	Eclipse
8			C#	Visual Studio
9		Ubuntu 10.10	C++	Eclipse
10			Java	NetBeans
11				Eclipse
12	CPU: Intel Core 2 Duo E8400 RAM: 4Gb HDD: 500Gb	MS Windows 7	C#	Visual Studio
13			C++	Eclipse
14			Java	NetBeans
15				Eclipse

У Таблиці 2.2 наведено експертну оцінку наведених варіантів за критеріями.

Таблиця 2.2 Експерта оцінка варіантів реалізації

Варіант	Функціональність	Надійність	Адміністрування	Продуктивність	Ефективність
1	7	8	7	8	9
2	5	8	7	8	8
3	7	9	9	8	10
4	8	9	9	8	9
5	8	9	10	8	10
6	8	8	8	7	9
7	7	10	9	7	9
8	7	9	9	7	8
9	8	10	7	7	10
10	7	10	8	8	9
11	8	10	8	8	10
12	9	8	9	8	8
13	10	9	9	10	10
14	9	10	9	10	9
15	10	10	9	10	10

Наступним етапом є визначення вагових коефіцієнтів для кожного з параметрів. У Таблиці 2.3 наведені вагові коефіцієнти для описаних критеріїв, значення коефіцієнтів обираються так щоб сума усіх коефіцієнтів дорівнювала одиниці.

Таблиця 2.3 – Значення вагових коефіцієнтів

Параметр, F_i	Коефіцієнт, φ_i
Функціональність	0,3
Надійність	0,15
Адміністрування	0,1
Продуктивність	0,3
Ефективність	0,15

Обчислимо коефіцієнт технічного рівня для кожного з варіантів за формулою:

$$K[F_i] = \sum_{j=1}^n \varphi_{ij} B_j$$

У таблиці 2.4 наведено результати обчислення коефіцієнтів по кожному з варіантів.

Таблиця 2.4 – Коефіцієнти технічного рівня для кожного з варіантів

Варіант	Коефіцієнт технічного рівня
1	7,75
2	7
3	8,25
4	8,4
5	8,65
6	7,85
7	7,95
8	7,65
9	8,2
10	8,15
11	8,6
12	8,4
13	9,75
14	9,45
15	9,9

Виходячи з отриманих результатів можна судити о ступені технічного рівня кожного з варіантів. Якнайкращим є той варіант, якому відповідає найбільший коефіцієнт технічного рівня.

Таким чином можна сказати, що 15-ий варіант є найкращім з технічної точки зору:

– Апаратна частина:

CPU: Intel Core 2 Duo E8400

RAM: 4Gb

HDD: 500Gb

– Програмна частина:

Операційна система: MS Windows 7

Мова програмування: Java

Середовище розробки: Eclipse

Вартісний аналіз

Аналіз функцій, що реалізуються підсистемою, повинен бути доповнений вартісним аналізом. Для цього розглядаються зведені витрати Z_j по кожному j -ому варіанту.

$$Z_j = C_j + E_n K_j,$$

де C_j – поточні витрати по j -ому варіанту;

K_j – капітальні вкладення по j -ому варіанту;

E_n – нормативний коефіцієнт економічної ефективності капітальних вкладень, що дорівнює 0,15.

Зведені витрати Z_j розраховують у сфері виробництва (у розробника) чи в сфері експлуатації (у користувача). У нашому випадку доцільно розраховувати витрати у сфері виробництва.

Поточні витрати при розробці інформаційної системи(підсистеми) C_j – це собівартість створення системи. Для усіх варіантів будемо вважати, що собівартість створення системи однакова. Собівартість системи розрахована у Розділі 4 даної роботи та складає 12000 грн.

Капітальні вкладення K_j – це витрати, що носять одноразовий характер. У даному випадку внесемо сюди вартість придбання ліцензійного програмного забезпечення та апаратного забезпечення, необхідного для створення системи.

Таким чином, приведемо вартості необхідні для подальшого обчислення:

Вартість апаратних а програмних рішень з необхідними коментарями наведена у Таблиці 2.5.

Таблиця 2.5 – Вартість апаратних або програмних рішень

Рішення	Вартість	Коментар
CPU: Intel Core Duo T2390 RAM: 2Gb HDD: 160 Gb	2500	Апаратне рішення
CPU: Intel Core 2 Duo E8400 RAM: 4Gb HDD: 500Gb	3200	Апаратне рішення
MS Windows XP та MS Windows 7	0	Програмне рішення. Комерційна операційна система, що використовується у освітніх цілях. Ліцензія MSDN AA.
RAD Studio 2010	11000	Програмне рішення. Комерційне середовище розробки.
Visual Studio	0	Програмне рішення. Комерційне середовище розробки, що використовується у освітніх цілях. Ліцензія MSDN AA.
NetBeans	0	Програмне рішення. Вільне середовище розробки, безкоштовне. Ліцензія CDDL.
Eclipse	0	Програмне рішення. Вільне середовище розробки, безкоштовне. Ліцензія Eclipse Public License.

Відповідно до наведеної вартості можна обчислити капітальні вкладення та відповідно до них розрахувати зведені витрати по кожному варіанту. За результатами функціонального та вартісного аналізу можна розрахувати коефіцієнт техніко-економічного рівня по кожному з варіантів за формулою:

$$K_{TEP_i} = \frac{K_{TP_i}}{Z_i},$$

де K_{TEP} – показник технічного рівня для i -ого варіанту;

Z_i – зведені витрати для i -ого варіанту.

Результати розрахунків наведені у Таблиці 2.6

Таблиця 2.6

Варіант	Вартість, грн	Зведені витрати, $З_i$ (грн)	Коефіцієнт, K_{TEP_i}
1	13500	14025	0,000553
2	13500	14025	0,000499
3	2500	12375	0,000667
4	2500	12375	0,000679
5	2500	12375	0,000699
6	13500	14025	0,00056
7	2500	12375	0,000642
8	2500	12375	0,000618
9	2500	12375	0,000663
10	2500	12375	0,000659
11	2500	12375	0,000695
12	3200	12480	0,000673
13	3200	12480	0,000781
14	3200	12480	0,000757
15	3200	12480	0,000793

Найкращій варіант визначається за максимальним значенням коефіцієнту K_{TEP_i} . Таким чином, визначимо, що найкращій варіант під номером 15.

Отже підсумковий варіант конфігурації системи має вигляд:

– Апаратна частина:

CPU: Intel Core 2 Duo E8400

RAM: 4Gb

HDD: 500Gb

– Програмна частина:

Операційна система: MS Windows 7

Мова програмування: Java

Середовище розробки: Eclipse

2.7 Постановка задачі на розробку підсистеми

Виходячи з описаних вище вимог можна сформулювати задачу на розробку підсистеми.

Отже підсистема, що розроблюється, повинна мати наступний функціонал:

- класифікувати вимовлену користувачем мовну команду;
- мати можливість збільшити обсяг мовних команд;
- бути гнучкою та масштабованою;
- розрізняння вимовлених мовних команд;
- додавання нових мовних команд;
- можливість підключення специфічних джерел аудіо інформації;
- можливість використання сторонніх методів класифікації.

Виходячи з вимог до реалізації та впровадження підсистеми можна звести наступні вимоги:

- підсистема повинна мати можливість інтегрування у більшість операційних систем;
- малу потребу у ресурсах: ресурсів центрального процесору та пам'яті;
- можливість нарощувати функціонал без суттєвих змін у архітектурі

Висновки з розділу 2

Інформаційна підсистема голосового керування призначена для спрощення процесу керування операційною системою завдяки створенню додаткового звукового інтерфейсу.

Підставами для розробки підсистеми є підвищення ефективності роботи людини за персональним комп'ютером, що у свою чергу підвищує ефективність та зберігає ресурс часу для працівника.

Розробка відповідної системи необхідна не тільки користувачам персональних комп'ютерів але й користувачам різноманітних приладів, що мають у своєму складі операційну систему та аудіо-підсистему.

Булi розробленi вимоги щодо функціоналу та задач, що підсистема повинна вирішувати. Головними вимогами є:

- розрізняння вимовлених мовних команд;
- додавання нових мовних команд;
- можливість підключення специфічних джерел аудіо інформації;
- можливість використання сторонніх методів класифікації.

Та головними вимогами до концепції системи:

- Закласти можливість зміни модулів збору аудіо інформації, класифікації;
- Закласти можливість додавання реакцій підсистеми на кожен окрему голосову команду;
- Можливість з найменшими затратами провести інтегрування підсистеми до складу окремої системи (операційної системи, приладу)

Створення подібної підсистеми потребує вироблення концепції щодо будови та структури підсистеми, що створюється. Отже було розроблено концепцію підсистеми та описані функції кожного з вузлів системи.

На основі функціонально-вартісного аналізу було визначено найбільш оптимальну концепцію системи для розробки цільової підсистеми:

- Апаратна частина:
CPU: Intel Core i7 vPro
RAM: 8Gb
HDD: 500Gb
- Програмна частина:
Операційна система: MS Windows 2020
Мова програмування: Java
Середовище розробки: Eclipse

РОЗДІЛ 3. РОЗРОБКА ПРОЕКТНИХ РІШЕНЬ ЩОДО ПІДСИСТЕМИ ГОЛОСОВОГО КЕРУВАННЯ ПРОЦЕСАМИ У ОПЕРАЦІЙНІЙ СИСТЕМІ

3.1 Загальносистемні рішення

Існують багато підходів щодо проектування інформаційної системи (підсистеми). Одним з найбільш ефективних підходів до проектування є шаблонний підхід до проектування архітектури програмного забезпечення MVC. Даний підхід дозволяє проектувати досить гнучкі рішення.

Шаблон проектування MVC (Model-view-controller, «Модель-представлення-поведінка») – це підхід до проектування інформаційних систем, у котрому модель даних інформаційної системи, інтерфейс користувача та керуюча логіка розділені на три окремих компоненти, так що модифікування одного з компонентів має мінімальний вплив на інші компоненти.

Цей шаблон проектування дозволяє поділити дані, представлення та обробку дій користувача на три окремі компоненти:

- Модель (Model). Модель надає дані (зазвичай для Представлення), а також реагує на запити (зазвичай від Поведінки), змінюючи свій стан;
- Представлення (View). Відповідає за відображення інформації (інтерфейс користувача);
- Поведінка (Controller). Інтерпретує дані, введені користувачем, та інформує модель та представлення о необхідності відповідної реакції.

Важно зазначити, що як представлення так и поведінка залежать від моделі. Однак модель не залежить ні від представлення, ні від поведінки. Це одна з ключових переваг подібного розподілу. Воно дозволяє будувати модель незалежно від візуального представлення, а також створювати декілька різних представлень для одної моделі.

Слід також зазначити, що основна бізнес-логіка як і сховище даних належить до модулю Модель.

На Рисунку 3.1 наведено типову структуру системи, що проектується за шаблоном MVC та взаємодію між модулями системи. Суцільною лінією позначено прямі зв'язки між модулями, та пунктирними – непрямі.

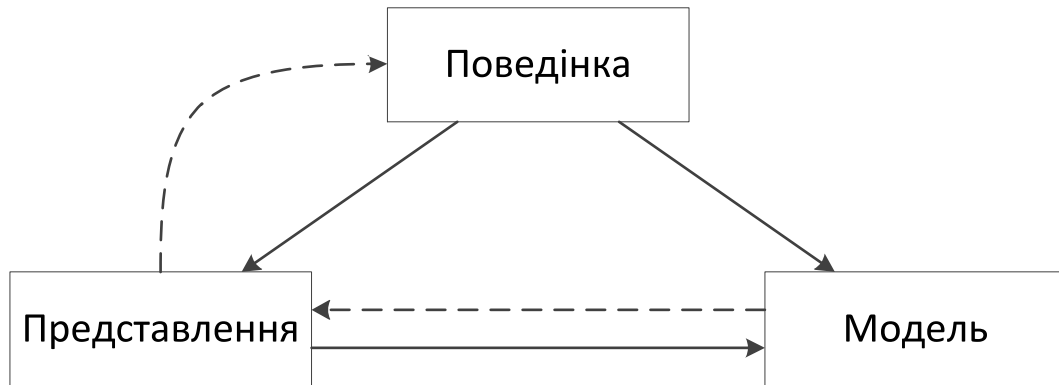


Рисунок 3.1 – Концепція MVC

3.2 Аналіз концептуальної моделі та функціональних вимог

Виходячи з вищеописаної концепції побудови необхідно провести деталізацію наведеної у Розділі 2 концептуальної моделі підсистеми, що розроблюється.

Виходячи з вимог до підсистеми та концепції її побудови існує можливість використання зовнішніх модулів для вирішення задачі сегментації, класифікації та отримання аудіо-потoku та необхідності додання довільного обробника події. Крім задоволення цієї вимоги необхідно реалізувати стандартні модулі, котрі використовувалися би при збоях зовнішніх модулів або при первинному запуску програми.

На Рисунку 3.2 наведено цільова концептуальна модель підсистеми. Отже опишемо детально функції котрі повинен виконувати кожен з модулів:

Модуль Аудіо-приймач. Модуль повинен встановлювати з'єднання з програмно-апаратною системою та перетворювати деякий абстрактний аудіо-потік до потоку, що відповідає вхідному інтерфейсу *Вузла сегментації*. До безпосередніх обов'язків цього модулю є організація деякого буферу, щоб

вилучити можливість втрати даних з аудіо-потoku при критичній загрузці програмно-апаратних засобів. Потік, що реалізує інтерфейс взаємодії аудіо-приймача з вузлом сегментації повинен бути стандартизованим, щоб спростити підтримку підсистеми та надати можливість розробки зовнішніх модулів.

Модуль *сегментації*. Повинен виділяти з вхідного стандартизованого потоку аудіо-інформації за деяким математичним алгоритмом окремі мовні команди та передавати команди до *Менеджеру*. Також сегментатор повинен мати деякий набір методів, котрі дозволили би йому працювати в додаткових режимах. Наприклад, режим при котрому аудіо-потік ігнорується, тобто сегментатор «спить», необхідно при призупиненні функціонуванні підсистеми.

Модуль *Класифікатор*. Цей модуль повинен по вхідній команді, що надається йому з *Менеджеру* проводити класифікацію команди відповідно до існуючих правил класифікації, що залежать від окремої реалізації методу класифікації. Також *Класифікатор* повинен мати свою власну базу даних у котрій міститься інформація щодо еталонних (опорних) даних класифікації. Слід зазначити, що опорна база даних може змінюватися відповідно до методу класифікації, оскільки кожен окремий метод потребує своїх даних, тобто реалізація уніфікованої бази даних є нераціональною, оскільки буде потребувати значних ресурсів та зайвого часу при проектуванні та реалізації.

Модуль *Менеджер*. Цей модуль уявляє собою головний модуль підсистеми, що реалізує функціонування всієї системи та координує взаємодію між іншими модулями системи. Диктує жорсткі вимоги щодо інтерфейсу взаємодії з зовнішніми модулями. Також до складу цього модуля входить функція організація додавання до підсистеми нових команд, та відповідно реалізація інтуїтивно-зрозумілого інтерфейсу користувача. Також на стадії розпізнавання окремих команд менеджер повинен надавати користувачеві мінімальний інтерфейс для відображення стану системи та включення/призупинення/відновлення/відключення системи розпізнавання.

Менеджер подій. Задача менеджера подій при отриманні окремої команди передати керування відповідному обробнику подій. Обробник подій уявляє собою зовнішній модуль, що реалізує необхідну реакцію системи. Подібний підхід до реалізації керування системою, до якої інтегрується підсистема є найбільш ефективний оскільки дозволяє виконувати практично будь-яку дію, що підтримується мовою програмування, що реалізує обробний подій.



Рисунок 3.2 – Концептуальна модель підсистеми голосового керування

Отже виходячи з вимог описаних вище необхідно провести проектування підсистеми відповідно до шаблону проектування MVC.

Визначимо приналежність кожного з модулів до окремої частини моделі MVC – Таблиця 3.1.

Таблиця 3.1 – Приналежність елементів підсистеми до частин моделі MVC

Компонент MVC	Контекстний модуль	Коментар
Модель	Аудіо-приймач	Бізнес-логіка підсистеми
	Вузол сегментації	Бізнес-логіка підсистеми
	Класифікатор	Бізнес-логіка підсистеми
	База даних класифікатору	Модуль DAO
	Обробник події	Бізнес-логіка підсистеми з обробки подій
Поведінка	Менеджер	Частина що відповідає за контроль підключених модулів
	Менеджер подій	Контролер обробників подій
Представлення	Менеджер	Частина відповідна за взаємодію з користувачем через графічний або аудіо інтерфейс

Як видно з наведеної таблиці первина контекстна діаграма не повністю відповідає початковій контекстній діаграмі, тобто необхідно провести декомпозицію Менеджеру для задоволення шаблону проектування MVC. Подібні дії будуть виконані на рівні проектування діаграми класів, оскільки потребують більш детального знання щодо реалізації підсистеми.

Також виходячи з проаналізованого функціоналу необхідно зробити інтерфейси взаємодії між окремими модулями виходячи з вимог до гнучкості. Тобто можливо зробити інтерфейси взаємодії надлишковими, але у рамках потрібного.

Також для реалізації стандартних (за замовченням) модулів підсистеми Аудіо-приймача, Класифікатору та Вузла сегментації необхідно обрати відповідні алгоритмічні рішення. Для останніх двох модулів рішення стосовно обору методів було виконано у Розділі 1. У якості рішення стосовно Аудіо-приймача необхідно обрати найбільш кросплатформову реалізацію, оскільки цей модуль найбільш пов'язаний з взаємодією з операційною системою.

3.3 Рішення з організаційного забезпечення

Одним з важливих питань щодо організації проектування, розробки та модернізації програмного забезпечення є питання ліцензування, що регламентує правові норми щодо цих дій. Існують багато різноманітних ліцензій, але кожен з них можна поділити на комерційну та вільну, що регламентують права користувача та програміста.

Відповідно комерційна ліцензія передбачає власником розробника, та усі права щодо модернізації інформаційної системи належать йому. Користувачеві розроблена інформаційна система надається за окрему платню.

Термін вільне програмне забезпечення ввів Річард Столмен, засновник проекту GNU, для опису програмного забезпечення, котре можна безперешкодно використовувати, вивчати та змінювати і котре може копіюватись та поширюватись у змінній чи незмінній формі без будь-яких обмежень, з тим щоб наступний користувач також мав всі перелічені права.

Будь-хто може продавати копії вільного програмного забезпечення, використовувати його в комерційних чи інших цілях. Проте, оскільки кожен, хто володіє вільним програмним забезпеченням, може передавати його будь-кому безкоштовно, то в цілому ВПЗ є безкоштовним. Бізнесові моделі ВПЗ базуються на додаткових послугах на кшталт технічної підтримки, навчанні, сертифікації чи інтеграції. Проте ВПЗ забороняє бізнесові моделі, засновані на абсолютній відсутності будь-яких прав у користувача і вимозі оплачувати право використання ПЗ.

Отже досить важливим питанням є питання щодо обору ліцензії під якою підсистема, що розробляється, буде використовуватися надалі. Оскільки підсистема, що розроблюється, уявляє собою академічну задачу, тобто створюється для навчання та будь яка пряма комерційна вигода не ставиться ціллю слід обрати ліцензію, що регламентує права вільного програмного забезпечення. Також оскільки однією з головних вимог є вимога гнучкості та можливості модифікацій у системі використання ліцензії на

вільне програмне забезпечення обумовлено не тільки економічно, але й технологічно. Оберемо у якості ліцензії було обрано ліцензію GNU GPLv2.

GNU General Public License (Громадська ліцензія GNU або Загальна громадська ліцензія GNU) — одна з найпопулярніших ліцензій на вільне програмне забезпечення, створена Річардом Столменом для проекту GNU. Часто її скорочено називають GNU GPL чи просто GPL, якщо з контексту зрозуміло, про яку ліцензію йде мова (існує чимало інших ліцензій зі словами «general public license» у назві).

Мета GNU GPL — надання користувачеві прав копіювати, змінювати та розповсюджувати програми, і зобов'язань, згідно з якими користувачі всіх похідних програм теж отримають перераховані вище права. Принцип «спадковості» таких прав називають «копілефт» (транслітерація англійського «copyleft»), такий термін запропонував Річард Столмен. На відміну від GPL, ліцензії ПЗ власника дуже рідко надають користувачеві такі права і переважно намагаються, навпаки, обмежити їх, наприклад, встановивши заборону на відновлення вихідного коду.

GPL — приклад сильної копілефт ліцензії, яка вимагає, щоб всі похідні роботи були доступні під тими ж умовами, як і оригінал. GPL надає одержувачам комп'ютерної програми права відповідно визначенню вільного програмного забезпечення і використовує копілефт, щоб гарантувати, що ці права будуть збережені, навіть, коли робота значно змінена або до неї будуть додані будь-які частини. Це відрізняє її від дозволяючих ліцензій на безкоштовне програмне забезпечення, прикладом яких є ліцензія BSD.

Ліцензія GNU GPL регламентує наступні свободи:

- свободу запуску програми, з будь-якою метою;
- свободу вивчення того, як програма працює, і її модифікації (попередньою умовою для цього є доступ до програмного коду);
- свободу розповсюдження копій;

- свободу поліпшення програми, і випуску поліпшень в публічний доступ (попередньою умовою для цього є доступ до програмного коду).

Для забезпечення прав розробників і користувачів GPLv2 накладає обов'язок поширювати початковий код разом з програмою і передавати користувачеві всі права, які захищає ліцензія.

Ці обмеження є предметом довгих суперечок. Альтернативною є точка зору, що виражається в різних ліцензіях, що базуються на ліцензії BSD, яка дозволяє користувачеві те, що забороняє GPL — використання коду в невільних програмних продуктах. Ліцензії такого типу використовує значна частина вільного ПЗ, включаючи FreeBSD, Apache, BIND і інші проекти.

Разом з тим, GPL 2 також використовується безліччю програмних проектів. Парадокс полягає в тому, що, незважаючи на суперечки, які не стихають у вузьких колах, в результаті 15-річної історії розвитку в будь-якому репозиторії Linux або колекції портів FreeBSD завжди можна знайти програми, що розповсюджуються під ліцензіями обох типів.

3.4 Організаційні рішення

Підсистема голосового керування розробляється для інтеграції до складу операційних систем або вбудованих рішень.

Користувачами підсистеми, що розроблюється будуть користувачі операційних систем, зокрема люди з обмеженнями руху.

Програмний продукт розгортається на комп'ютері користувача з попереднім встановленням віртуальної машини Java. Слід відмітити, що користувач повинен перед використанням підсистеми бути згоден з ліцензіями поширення підсистеми та з ліцензіями GNU General Public License та Java Community Process, що відповідають ліцензуванню JRE.

Користувач має можливість додавати нові мовні команди та встановлювати відповідні обробники подій до мовної команди; використовувати засоби підсистеми для розпізнавання мовних команд.

Адміністрування підсистеми проводиться адміністратором, що обслуговує операційну систему та ПК користувача. Адміністратор має можливість змінювати сукупність обробників подій, що надаються користувачеві; конфігурувати систему відповідно до операційної системи, зокрема змінювати джерело даних, точніше метод отримання звукових даних, підключати зовнішні модулі сегментування, класифікації.

До розробки зовнішніх модулів сегментування, класифікації та обробників подій, що не входять до початкового набору модулів залучається програміст. Програміст має право відповідно до встановленого програмного інтерфейсу розробити окремий вузол(модуль) підсистеми, та розробити відповідну документацію для впровадження окремого модуля до складу підсистеми, за котрою буде діяти адміністратор.

Слід зазначити, що усі дії адміністратора та програміста не повинні суперечити умовам ліцензування складових підсистеми.

3.5 Рішення з інформаційного забезпечення

Відповідно до обраних методів класифікації/сегментації можна розробити відповідні моделі даних, котрі уявляють окремою реалізацією, що підтримує обраний метод класифікації/сегментації даних. Також слід розробити рішення щодо взаємодії між моделями підсистеми наведеними у концептуальній моделі.

3.6 Діаграма потоків даних

Діаграма потоків даних (англ. Data Flow Diagram) — графічне представлення «потоків» даних в інформаційній системі. Діаграма потоків даних також може використовуватись для представлення обробки даних (структурна розробка). Вважається звичайним, для розробника, з початку креслити ДПД рівня контексту, завдяки чому буде показано взаємодію системи із зовнішніми модулями. Ця ДПД рівня контексту, потім

розширюється, для того, щоб показати розроблювану систему детальніше. Діаграми потоків даних містять чотири типи графічних елементів: процеси, що представляють собою трансформацію даних в рамках системи, що описується, репозиторії (сховище даних), зовнішні по відношенню до системи сутності та потоки даних між елементами трьох попередніх типів.

Наведена на Рисунку 3.2 концептуальна модель відповідає ДПД рівня контексту, отже потребує деталізації відповідно до нотації, отже на Рисунку 3.3 зображено DFD-діаграму підсистеми, що розроблюється.

3.7 Концептуальна модель даних

Оскільки головною задачею підсистеми голосового керування процесами у операційній системі не є оперування даними концептуальна модель досить примітивна, але потребує уваги при розробці. Отже на Рисунку наведено концептуальну модуль даних, котрими оперує формантний метод при виконання завдання класифікації.

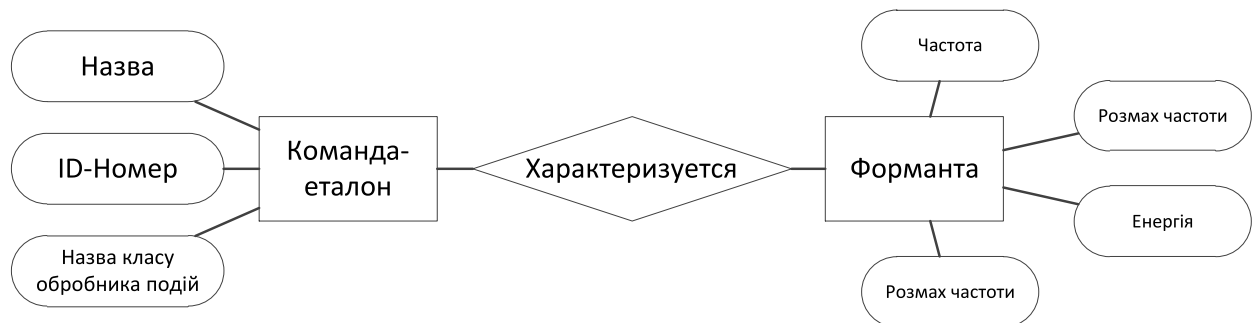


Рисунок 3.3 – Діаграма класів для підсистеми, що розробляється

3.8 Логічна модель даних

Виходячи з наведеної вище концептуальної моделі даних слід розробити логічну модель даних. На Рисунку 3.3 наведено логічну модель даних, котра описує структури, що використовуються при реалізації інформаційної системи. Окремі дві сутності пов'язані з собою зовнішнім ключем «ID-номер команди».

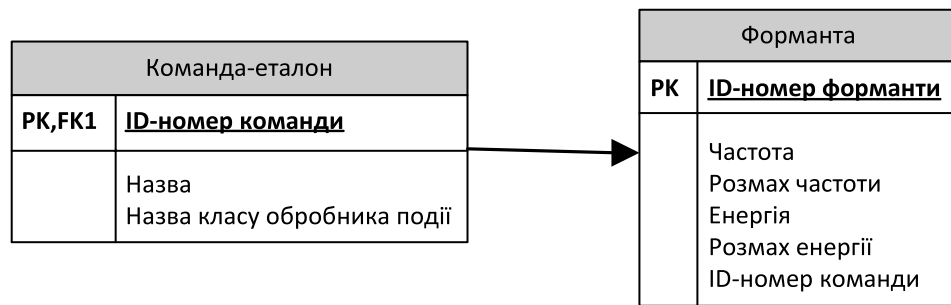


Рисунок 3.4 – Діаграма класів для підсистеми, що розробляється

3.9 Фізична модель даних

Фізична модель даних описує фактичні проектні рішення щодо реалізації окремої логічної моделі даних. Отже перше рішення, котре необхідно розробити, це рішення щодо обору СУБД. З цієї точки зору слід зазначити, що використання СУБД накладає ряд обмежень та робить інформаційну систему більш масивною. Використання СУБД необхідно коли існує достатньо великий обсяг інформації котру необхідно обробляти та зберігати найбільш ефективно. Отже для реалізації наведеної на Рисунку 3.4 логічної моделі даних не слід обирати громіздкі рішення, отже реалізовувати сховище даних будемо на рівні файлової системи, тобто окремими файлами з відповідною структурою. Таке рішення значно зменшить ресурси, що потрібні для використання підсистеми, оскільки використовуються стандартні дії с файловою структурою. Також це рішення зумовлене вимогами много-платформового рішення, зокрема можливості впровадження до складу мобільних приладів. Крім того для рішення даної задачі немає необхідності використання головної переваги СУБД – масштабованості, оскільки головна бізнес-логіка не зав'язана на збереження даних.

Для вирішення задачі реалізації власного обмеженого прототипу системи керування базою даних візьмемо у якості формату збереження формат XML, котрий дозволяє реалізувати досить елегантне рішення зі збереження даних.

Розширювана мова розмітки (англ. Extensible Markup Language, скорочено XML) — запропонований консорціумом World Wide Web (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML. XML документ складається із текстових знаків, і придатний до читання людиною.

Стандарт XML (англ. Recommendation, перше видання від 10 лютого 1998, останнє, четверте видання 29 вересня 2006) визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої, шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних.

XML документ має ієрархічну логічну структуру, і може представлятись у вигляді дерева. Вузлами цього дерева можуть бути:

- Елементи, фізична структура яких складається із:
коректної пари відкриваючого та закриваючого тегів (<Назва-тега>) та (</Назва-тега>), або
тега порожнього елемента (<Назва-тега />),
- Атрибути, що мають вигляд пар ключ-значення (назва атрибута="значення атрибута") і знаходяться або у відкриваючому, або у порожньому тезі (подібно до метаданих),
- Вказівки щодо обробки документа (англ. Processing Instruction) (<?Обробник параметр ?>)
- Коментарі (<!-- Текст коментаря -->)
- Текст, або у вигляді простого тексту, або фрагментів CDATA (<![CDATA[довільний текст]]>).

XML-документ повинен мати лише один кореневий елемент. Решта елементів є піделементами цього кореневого елемента.

Виходячи з вимог до структури XML-файлу фізична модель бази даних може біти представлена у наступному вигляді:

- Кожна форманта описується наступним блоком:

```
<formant>
    <frequency>Значення частоти</frequency>
    <dfrequency>Розмах частоти</dfrequency>
    <energy>Значення енергії</energy>
    <denergy>Розмах енергії</denergy>
</formant>
```

- Кожна еталона команда уявляє собою блок наступного вигляду:

```
<etalon>
    <etalonName>Назва команди</etalonName>
    <eventHandler>Обробник.class</eventHandler>
    <formant>...</formant>
    <formant>...</formant>
</etalon>
```

Тобто кожен з блоків що описують окремий еталон використовується сукупність полів, котрі відображають відповідні поля з логічної структури. Слід відзначити, що приналежні до окремого еталону значення формант зберігаються у блоку еталону та мають власну структуру, та кількість таких блоків залежить від окремого еталону.

Для розуміння структури сховища еталонів наведемо приклад описаної вище структури, відповідно до вимог оформлення файлів XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<etalonStorage>
    <etalon>
        <etalonName>Відкрити текстовий редактор</etalonName>
        <eventHandler>EventTextEditor.class</eventHandler>
        <formant>
            <frequency>140</frequency>
            <dfrequency>40</dfrequency>
```

```

        <energy>0.05</energy>
        <denergy>0.014</denergy>
    </formant>
    <formant>
        <frequency>245</frequency>
        <dfrequency>24</dfrequency>
        <energy>0.03</energy>
        <denergy>0.008</denergy>
    </formant>
    <formant>
        <frequency>340</frequency>
        <dfrequency>40</dfrequency>
        <energy>0.021</energy>
        <denergy>0.0076</denergy>
    </formant>
</etalon>
</etalonStorage>

```

Наведений приклад має сховища має одну команду, котра відповідає за відкриття текстового редактору, та описується чотирма формантами, значеннями котрих оперує класифікатор при виконанні класифікації.

3.10 Діаграма класів

Діаграма класів — статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак, їх динаміку розкрито в діаграмах інших типів.

Представлення діаграми класів прийнято виконувати за допомогою нотації UML.

UML (англ. Unified Modeling Language - уніфікована мова моделювання) - мова графічного опису для об'єктного моделювання в області

розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, який використовує графічні позначення для створення абстрактної моделі системи, званої UML-моделлю.

UML був створений для визначення, візуалізації, проектування та документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретується код можлива кодогенерації.

Для підсистеми, що розробляється необхідно побудувати діаграму класів, котра буде відповідати концептуальній моделі підсистеми та необхідному функціоналу.

На Рисунку 3.5 наведено діаграму класів, котра у нотації UML зображає взаємодію між окремими класами підсистеми.

Слід зазначити, що відповідно до шаблону проектування MVC можна диференціювати класи, що належать до окремої частини MVC.

До частини *View* належить клас *UserInterface*, котрий реалізує відображення необхідної інформації користувачеві.

Клас *Handler* реалізує реакції на дії користувача та передає повідомлення до *Model*, тобто цей клас реалізує функції модулю *Controller*.

Наведені класи не містять бізнес-логіки, а реалізують взаємодію з користувачем. Отже вони цікавлять розробника системи менше ніж основна бізнес-логіка, що міститься у модулі *Model*. Для цього модуля проведена більш ретельна деталізація, оскільки необхідна ретельна розробка кожного з елементів.

Перейдемо до опису архітектури інформаційної підсистеми, що розроблюється. Підсистема має три програмних інтерфейси, що регламентують вимоги окремих класів щодо обміну даними.

Тобто інтерфейс *Segmentable* описує перелік методів(повідомлень) за котрими клас *Manager* взаємодіє з окремою реалізацією методу сегментації. Тобто цей інтерфейс має бути використано для реалізації класу, що

відповідає за сегментування команд. Метод *getAudioCommand()* повинен повертати реалізацію окремої команди.

Наступний інтерфейс *Classifiable* визначає вимоги щодо реалізації класу, що відповідає за класифікацію команд. Отже він має один метод, котрий отримує у якості параметру звукову реалізацію команди та повертає відповідну команду.

Останній інтерфейс *Executable* визначає необхідні повідомлення, котрі може отримувати реалізація цього інтерфейсу, тобто у нашому випадку цей інтерфейс регламентує обов'язкову реалізацію методу *execute()*.

Виділення описаних вище інтерфейсів дозволяє реалізовувати гнучкість без перекомпіляції всієї підсистеми. Тобто зовнішні модулі сегментування при необхідності можуть бути відкомпільовані окремо та підключатися до підсистеми динамічно. Взаємодія між такими модулями буде проходити через відповідний інтерфейс.

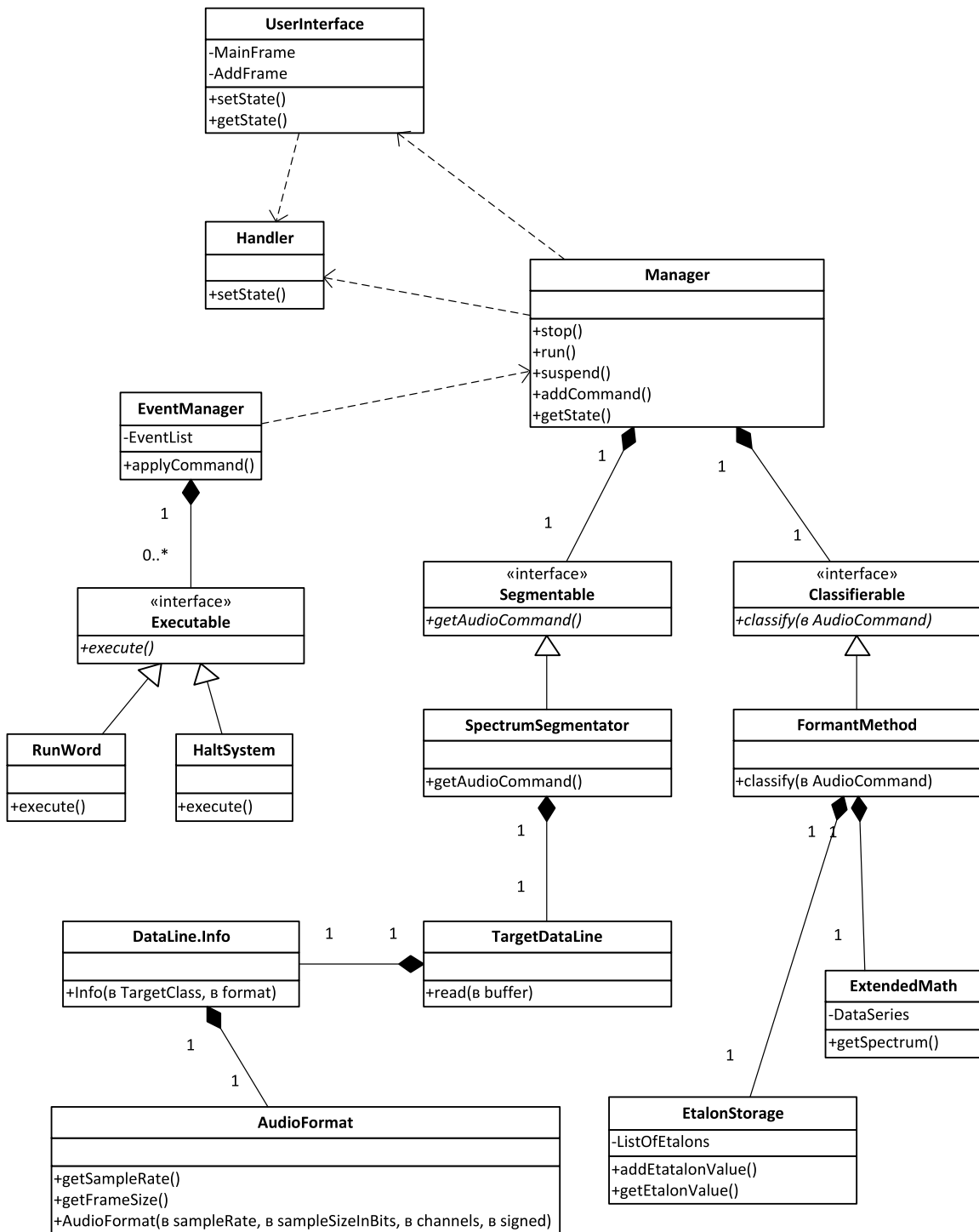


Рисунок 3.5 – Діаграма класів для підсистеми, що розробляється

3.11 Діаграма компонентів

Діаграма компонентів — в UML, діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі — під час компоновки, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгорткування.

Компоненти з'єднуються разом з використанням структурного з'єднання англ. *assembly connector* щоб з'єднати інтерфейси двох компонент. Це ілюструє тип зв'язку "клієнт-сервер".

Структурне з'єднання - "з'єднання між двома компонентами, що передбачає те, що один з них надає послугу, що потребує інший компонент."

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Отже виходячи з цих зауважень розробимо діаграму компонентів для підсистеми, що розробляється (Рисунок 3.6).

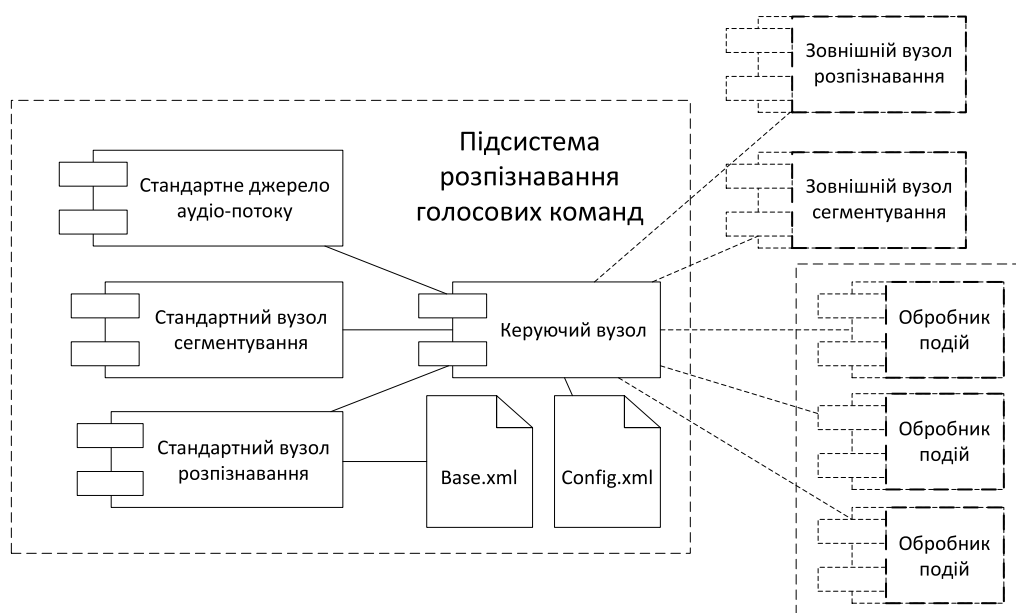


Рисунок 3.6 – Діаграма компонентів для підсистеми, що розробляється

Дії та функції деяких компонентів-вузлів вже відомі, але на даний діаграмі також наведено два файли

Перший файл (*Base.xml*) уявляє собою фізичну реалізацію сховища еталонних команд. Має структуру наведену у пункті 3.3.4. Другий файл

Другий файл (*Config.xml*) уявляє собою конфігураційний файл, котрий регламентує підключення зовнішніх модулів сегментування та класифікації. Виходячи з вимог до файлів формату XML, конфігураційний файл має наступну структуру:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<configuration>
    <segmentator>Default</segmentator>
    <classification>Default</classification>
</configuration>
```

Наведений конфігураційний файл регламентує використання стандартних засобів розпізнавання, та може генеруватися при відсутності конфігураційного файлу.

У разі необхідності можна вказати зовнішні модулі, на котрі перекладаються відповідні функції, наприклад:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<configuration>
    <segmentator>./ExternalSegmentaror.class</segmentator>
    <classification>./Classifier.class</classification>
</configuration>
```

Тобто функції сегментування перекладають на зовнішній модуль сегментування, що розташовано у *ExternalSegmentaror.class*, та має відповідний програмний інтерфейс *Segmentable*. Функції розпізнавання (класифікації) перекладаються на зовнішній модуль *Classifier.class*, котрий використовує відповідний комунікаційний інтерфейс *Classifiable*.

3.11 Рішення з технічного забезпечення

Опис комплекту технічних засобів

Виходячи з вимог до гнучкості системи та много-платформності та вимоги до низьких потреб щодо ресурсів системи можна зробити опис технічних засобів.

Рішення з технічного забезпечення можуть варіюватися відповідно цільової системи до котрої проводиться інтеграція підсистеми.

Наведемо альтернативні комплекти технічних засобів з коментарями.

Підсистема розгортається на персональному комп'ютері під керівництвом операційної системи Windows 7, котра має досить суттєві вимоги до апаратної частини.

Комплект апаратних засобів:

- Процесор: Intel Core 2 Duo E
- Оперативна пам'ять: 2Гб
- Жорсткий диск: 160Гб
- Звуковий адаптер з підключеним мікрофоном

Комплект програмних засобів:

- Windows 7 Professional
- SoundCard Drivers
- Java Runtime Environment

Наведена апаратна конфігурація дозволяє ефективно працювати користувачеві, але наведена конфігурація не є найкращім рішенням з точки зору продуктивності, але є достатньою та ефективною за економічним показником.

Відповідно до вимог розгортання системи на мобільних рішеннях можна навести альтернативний комплект технічних засобів.

Комплект апаратних засобів:

- Процесор: AT91SAM9260 або AT91SAM9XEXXX.

- Оперативна пам'ять: 32-256 Мб
- ПЗУ: NANDFlash 32Мб
- Встановлений аудіо-декодер та аудіо-вихід та підключений до нього мікрофон.

Комплект програмних засобів:

- Первинний загрузчик Bootstrap
- Вторинний загрузчик U-boot
- Операційна система Linux
- Java Runtime Environment

3.12 Модель розгортання системи

Відповідно до комплекту технічних засобів реалізується окрема модель розгортання системи. Типовою дією є задоволення умов комплекту технічних програмно-апаратних засобів та відповідно копіювання дистрибутиву підсистеми до файлової системи цільової системи.

Розпишемо дії стосовно розгортання системи для першого з комплекту технічних засобів. Дії можна умовно поділити на два етапи: задоволення технічних умов щодо програмних засобів та безпосередньо розгортання підсистеми. Отже:

Задоволення технічних умов:

- Встановлення дистрибутиву операційної системи
- Встановлення (якщо необхідно) драйверів, зокрема на аудіо-адаптер
- Встановлення Java Runtime Environment. Цю дію можна виконати завантаживши дистрибутив з адреси:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Розгортання підсистеми

- Копіювання дистрибутиву (зокрема файлу *.jar та *.bat).

- Якщо необхідно – створення необхідних ярликів та настроювання автоматичної загрузки

Відповідно друга модель розгортання описує розгортання підсистеми на мобільну платформу, конфігурація котрої наведена вище, дії також можна поділити на два етапи як і у попередньому способі. Деталізацію окремих пунктів опустимо.

Задоволення технічних умов:

- Конфігурація та компіляція загрузчиків.
- Конфігурація та компіляція ядра Linux та кореневої системи.
- Встановлення Java Runtime Environment. Цю дію можна виконати завантаживши дистрибутив з адреси:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Розгортання підсистеми

- Копіювання дистрибутиву (зокрема файлу *.jar та *.bat).
- Якщо необхідно – настроювання автоматичної загрузки

3.13 Рішення з математичного забезпечення

Рішення щодо обору методу сегментації

Вирішення проблеми сегментації звукового потоку, тобто виділення окремих команд з потоку, є однією з ключових проблем при розробці системи голосового керування.

Точність сегментації у значній мірі визначає надійність системи автоматичного розпізнавання мовлення.

Відомі два типи алгоритмів сегментації мовного сигналу: сегментація за умови, що відома послідовність фонем даної фрази, та сегментація на сонові ступеня зміни акустичних характеристик сигналу. Іноді виділяють в окремий вид комбінований алгоритм.

Існує досить багато методів, що дозволяють сегментувати не тільки за наявністю команди мовних даних чи ні, але ще и розділяти окремі фонemi. Оскільки не планується виділяти окремі фонemi з мовного потоку, досить

методу, що дозволить встановити наявність команди у мовному потоку.

Існують наступні методи сегментації мовлення:

- Сегментація за усередненим нормованим спектром
- Сегментація динамічним детектором
- Сегментація за кореляції між рівновіддалених спектрах

Оберемо для впровадження у підсистему перший з наведених методів. Він дозволяє з необхідною точністю вирішити задачу сегментації та являється найбільш простим у реалізації.

Тобто цей метод вважає що сигнал має деяку команду якщо спектр сигналу має хоча б одну форманту з цільовим значенням енергії.

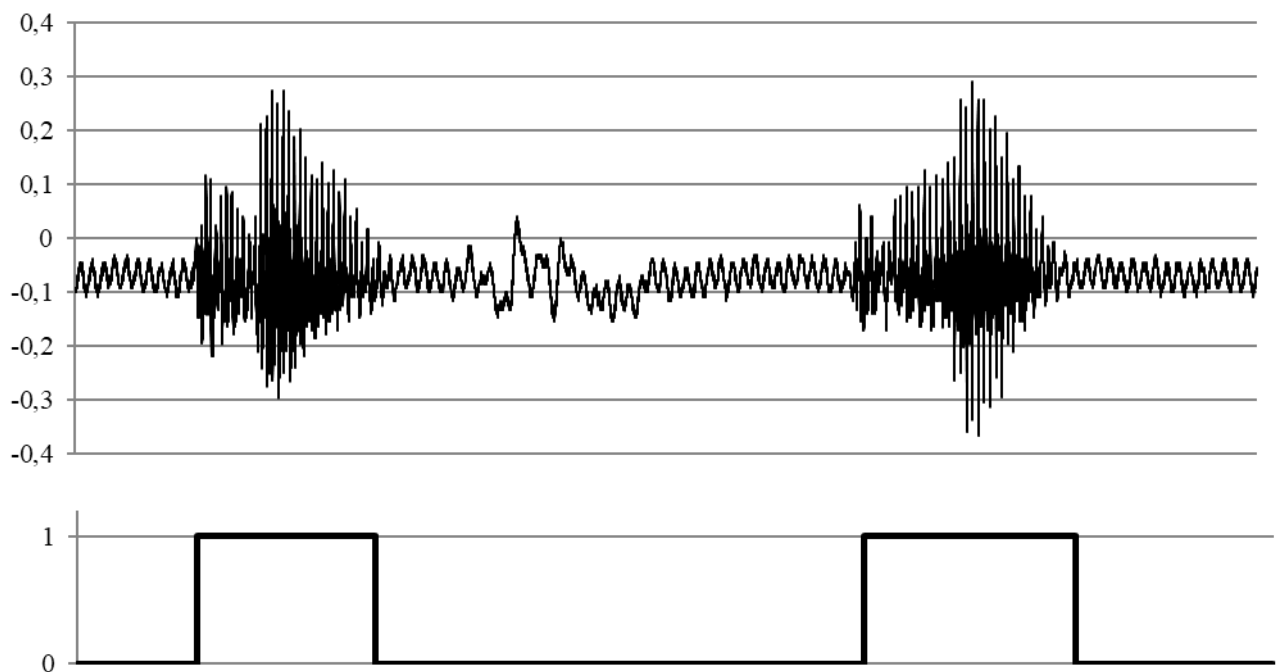


Рисунок 3.7 – Сегментація команд

Вибір методу класифікації

У якості методу класифікації будемо використовувати форматний метод.

Формантний метод за оцінкою спектру усієї звукової реалізації призводить аналіз ключових гармонік за частотами та інтенсивністю.

Форманта – це термін фонетики, котрий визначає акустичну характеристику звуків мовлення (здебільше голосних), зв’язану з рівнем частоти голосового тону, що утворюють тембр звуку.

У термінології розпізнавання мовлення форманта, це імпульсний пік на спектрограмі сигналу на відповідній частоті та енергією, що відповідає наявності відповідної гармоніки.

Формантною зоною називається зона, котра зайнята окремою формантою. Ця зона визначається тривалістю, максимальною енергією, площею та частотою.

На основі декількох реалізацій однієї команди будується усереднений спектр команди. Далі виділяються основні форманти спектру, стільки скільки виявлено на спектрограмі. Як було встановлено експериментально кількість формантних зон варіюється від 3 до 8.

На основі формантних зон будується еталонне значення команди у наступному вигляді:

$$Q: S(v_i \pm dx_i) = s_i \pm dy_i, \quad i \in [1..n] \quad (1)$$

Де n – кількість значущих формант

Тобто кожна формантна зона має центральну частоту v_i та допустиме відхилення dx_i , та ця формантна зона у спектрі повинна мати енергію s_i з допустимим відхиленням dy_i .

Команда вважається прийнятою якщо спектр піддослідного сигналу має точки, що задовольняють всі еталонні формантні зони, тобто якщо виконується вираз (1).

Вибір методу побудови оцінки спектру сигналу

Для вирішення задачі обчислення оцінки спектру використовують перетворення Фур’є. Подане перетворення дозволяє отримати розкладання початкової функції на елементарні складові частини – гармонійні коливання з різними частотами. Таке розкладання на частоти є досить зручним для

аналізу мовного сигналу, оскільки дозволяє визначити ключові гармоніки сигналу.

Дискретне перетворення Фур'є перетворює сукупність чисел $a_0, a_1, \dots, a_{n-1}$ у сукупність чисел $b_0, b_1, \dots, b_{n-1}$, таких що $b_k = \sum_{j=0}^{n-1} a_j \varepsilon^{kj}$, де $\varepsilon^n = 1$ та $\varepsilon^k \neq 1$, якщо $0 < k < n$.

Швидке перетворення Фур'є (ШПФ) дозволяє обчислити дискретне перетворення Фур'є (ДПФ) зі складністю алгоритму меншу за $O(N^2)$, що потребуються для обчислення ДПФ за початковою формулою. Формула необхідна для обчислення швидкого перетворення Фур'є має вигляд:

$$X_m = \sum_{n=0}^{M-1} x_{2n} a_M^{nm} + e^{-2\pi i \frac{m}{N}} \sum_{n=0}^{M-1} x_{2n+1} a_M^{nm}$$

де:

x – вибірка для якої використовується перетворення Фур'є;

M – цілове значення перетворення Фур'є;

a_M^{nm} – коефіцієнт, що обчислюється за формулою $a_M^{nm} = e^{-2\pi i \frac{m}{n}}$.

Оскільки отриманий спектр сигналу після швидкого перетворення Фур'є є досить зашумленим (Рисунки 1.6 та 1.7) та для аналізу найчастіше потрібен згладжений спектр то необхідно отримати застосувати методи згладжування оцінки спектра. Одним з найбільш уживаних та ефективних методів згладжування спектру – це застосування методу ковзкого вікна, що виконується за наступною формулою:

$$\bar{S}_i = \frac{1}{n} \sum_{j=-\frac{n}{2}}^{\frac{n}{2}} S_{j+i} W(j)$$

де:

S – початковий спектр

$W(j)$ – функція-коефіцієнт, котра відповідає окремому методу згладжування.

У Таблиці 1.3 наведені найбільш уживані функції $W(j)$.

Таблиця 3.3 – Вираження для деяких віконних функцій

Назва вікна	Функція
Прямокутне вікно	$W(j) = 1$
Синус-вікно	$W(j) = \sin\left(\frac{\pi \cdot j}{n-1}\right)$
Вікно Ланкзоса	$W(j) = \text{sync}\left(\frac{2j}{n-1} - 1\right) \text{sync}(x) = \frac{\sin(\pi x)}{\pi x}$
Вікно Бартлету	$W(j) = 1 - \left \frac{2j}{n-1} - 1\right $
Вікно Ханна	$W(j) = 0.5 - 0.5 \cos\left(\frac{2\pi \cdot j}{n-1}\right)$
Вікно Бартлета-Ханна	$W(j) = a_0 - a_1 \cdot \left \frac{j}{n-1} - 0.5\right - a_2 \cos\left(\frac{2\pi \cdot j}{n-1}\right)$ $a_0 = 0.62 \quad a_1 = 0.48 \quad a_2 = 0.38$
Вікно Хеммінгу	$W(j) = 0.54 - 0.46 \cos\left(\frac{2\pi \cdot j}{n-1}\right)$
Вікно Блекмана	$W(j) = a_0 - a_1 \cos\left(\frac{2\pi \cdot j}{n-1}\right) + a_2 \cos\left(\frac{4\pi \cdot j}{n-1}\right)$ $a_0 = 0.42 \quad a_1 = 0.5 \quad a_2 = 0.08$
Вікно Блекмана-Харрису	$W(j) = a_0 - a_1 \cos\left(\frac{2\pi \cdot j}{n-1}\right) - a_2 \cos\left(\frac{4\pi \cdot j}{n-1}\right) - a_3 \cos\left(\frac{6\pi \cdot j}{n-1}\right)$ $a_0 = 0.35875 \quad a_1 = 0.48829 \quad a_2 = 0.14128 \quad a_3 = 0.01168$
Вікно Наталла	$W(j) = a_0 - a_1 \cos\left(\frac{2\pi \cdot j}{n-1}\right) - a_2 \cos\left(\frac{4\pi \cdot j}{n-1}\right) - a_3 \cos\left(\frac{6\pi \cdot j}{n-1}\right)$ $a_0 = 0.355768 \quad a_1 = 0.48396 \quad a_2 = 0.144232 \quad a_3 = 0.012604$
Вікно Блекмана-Наталла	$W(j) = a_0 - a_1 \cos\left(\frac{2\pi \cdot j}{n-1}\right) - a_2 \cos\left(\frac{4\pi \cdot j}{n-1}\right) - a_3 \cos\left(\frac{6\pi \cdot j}{n-1}\right)$ $a_0 = 0.3635819 \quad a_1 = 0.4891775 \quad a_2 = 0.1365995 \quad a_3 = 0.0106411$
Вікно з плоскою вершиною	$W(j) = \sum_{i=0}^4 a_i \cos\left(\frac{2i \cdot \pi \cdot j}{n-1}\right) \quad a_0 = 1.0 \quad a_1 = 1.93$ $a_2 = 1.29 \quad a_3 = 0.388 \quad a_4 = 0.032$
Вікно Гауса	$W(j) = e^{-0.5 \left(\frac{j - \frac{n-1}{2}}{\sigma \frac{n-1}{2}}\right)^2}$

3.14 Рішення з програмного забезпечення

Вибір мови програмування

При побудові підсистеми розпізнавання голосових команд необхідно обрати відповідну мову програмування таким чином, щоб зменшити витрати на реалізацію та впровадження підсистеми та щоб обрана мова програмування потенційно задовольняла вимогам поставленим до підсистеми.

При проведенні функціонально-вартісного аналізу була обрана концепція розробки, що базується на використанні мови програмування Java.

Проведемо обґрунтування обору відповідної мови програмування.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім, Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. За необхідності таких дій java дозволяє викликати підпрограми, написані іншими мовами програмування.

Для запуску програми написаної на мові Java потребується використання інтерпретатора – віртуальної машини Java.

Віртуальна машина Java (англ. Java Virtual Machine; JVM) — набір комп'ютерних програм та структур даних, що використовують модель віртуальної машини для виконання інших комп'ютерних програм чи скриптів.

JVM використовує байткод Java, який як правило, але не завжди генерується з вихідних кодів мови програмування Java; віртуальну машину також застосовують для виконання коду, згенерованого з інших мов

програмування. Наприклад, вихідний код Ada можна скомпілювати у Java байткод.

Віртуальна машина Java — основний компонент Java платформи. JVM доступна для всіх основних сучасних платформ, тому про програми, що скомпільовані у Java байткод теоретично можна сказати «Написано один раз, працює скрізь» (англ. "Write once, run anywhere").

Отже інтерпретована мова програмування Java дозволяє без перекомпіляції організувати переносимість програми до будь-якої платформи, що підтримує віртуальну машину Java. Тобто це дозволить задовольнити вимогу щодо переносимості програмного продукту. Не складно реалізувати використання додаткових модулів розпізнавання/сегментації, котрі будуть уявляти собою окремі Java класи, котрі будуть реалізовувати відповідні інтерфейси.

Також досить цікавим з точки зору впровадження підсистеми до вбудованих рішень є наявність у сучасних мікропроцесорах апаратного інтерпретатора, тобто технології, що суттєво підвищує виконання програми на мові програмування Java. Застосовно до наведених вище мікропроцесорів, заснованих на технології ARM, ця технологія має назву Jazelle.

Jazelle - сімейство технологій, розроблених і просуваються компанією ARM, призначених для апаратного прискорення виконання програм, написаних на мові Java. Jazelle розроблена для архітектури ARMv5TEJ, і першим процесором, що використовують технологію Jazelle, був ARM926EJ-S.

В даний час на ринку присутня різновид Jazelle DBX (Dynamic Bytecode eXecution - динамічне виконання байт-кодів). Основним елементом цієї технології є спеціальний співпроцесор, що вбудовується в стандартну архітектуру мікропроцесорів ARM. Співпроцесор апаратно перетворює байт-код Java у відповідну послідовність команд основного процесора, який негайно їх виконує. Таким чином програмна інтерпретація байт-кодів замінюється апаратної, що призводить до істотного прискорення обробки

байт-кодів, і як наслідок, прискорення виконання програм на Java. Іншими словами, Jazelle DBX виробляє апаратне перетворення байт-кодів у команди основного процесора «на льоту». Звідси і назва DBX - «динамічне виконання».

Розширення Jazelle використовує низькорівневий бінарний переклад, виконаний у вигляді додаткового ступеня між вибіркою і декодуванням етапу інструкцій процесора pipeline. Пізнаний байт-код перетворюється в рядок з одного або більше рідних інструкцій ARM.

3.15 Алгоритмічне рішення

Для вирішення задачі прийняття відповідної команди за формантним методом біло розроблено підпрограму(метод), що відповідає чи належить вхідна команда(Y_i, E_i) до окремого еталону(v_i, e_i, dv_i, de_i)

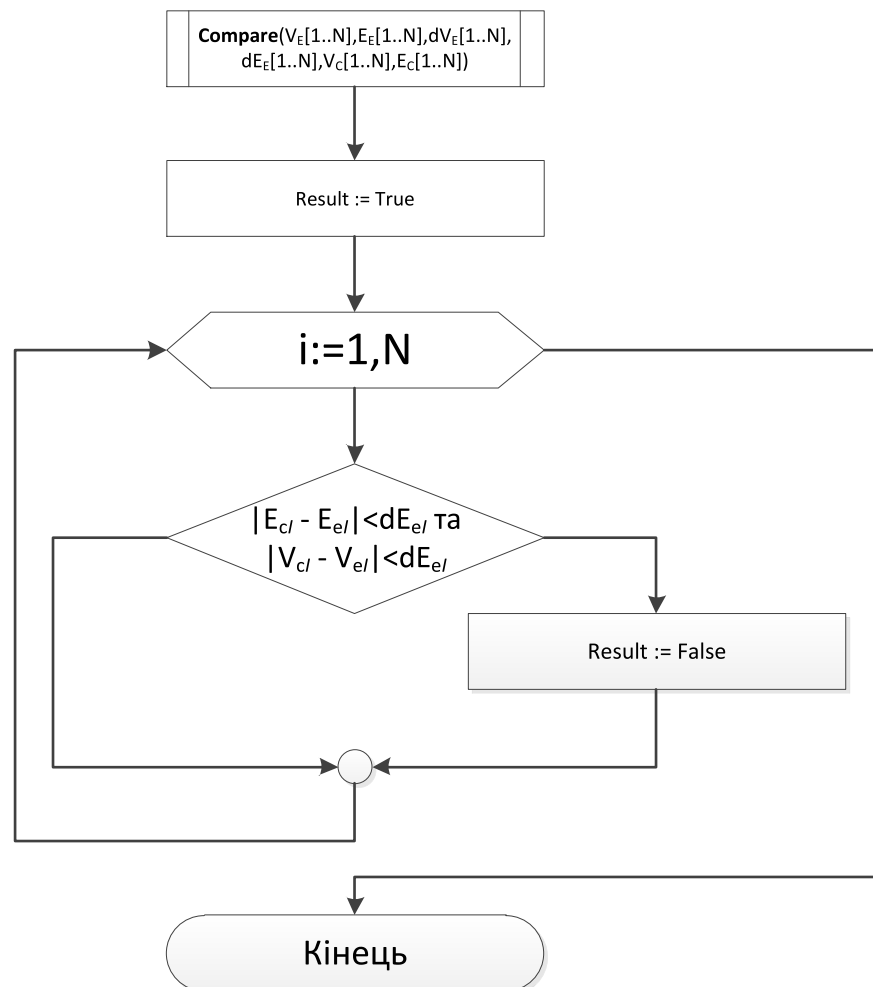


Рисунок 3.8 – Сегментація команд

Вибір додаткових засобів реалізації

Шаблони проектування та розробки

Відповідно до вимог до підсистеми та вимог до ООП концепції побудови інформаційної системи, а також виходячи з обору мови програмування Java необхідно прийняти сукупність шаблонів, котрі дозволять зробити інформаційну підсистему більш структурованою та гнучкою.

Шаблони проектування програмного забезпечення (або інформаційної системи) – ефективні способи вирішення задач проектування програмного забезпечення. Шаблон не є закінченим зразком, який можна безпосередньо транслювати в програмний код. Об'єктно-орієнтований шаблон найчастіше є зразком вирішення проблеми і відображає відношення між класами та об'єктами, без вказівки на те, як буде зрештою реалізоване це відношення.

Головна користь кожного окремого шаблону полягає в тому, що він описує рішення цілого класу абстрактних проблем. Також той факт, що кожен шаблон має своє ім'я, полегшує дискусію про абстрактні структурах даних (ADT) між розробниками, так як вони можуть посилатися на відомі шаблони. Таким чином, за рахунок шаблонів проводиться уніфікація термінології, назв модулів і елементів проекту.

Правильно сформульований шаблон проектування дозволяє, відшукавши вдале рішення, користуватися ним знову і знову.

Одним з фундаментальних шаблонів є шаблон «Ланцюжок відповідальностей», оскільки він регламентує взаємодію між об'єктами.

Ланцюжок відповідальностей – шаблон об'єктно-орієнтованого дизайну у програмуванні.

В об'єктно-орієнтованому дизайні, шаблон «ланцюжок відповідальностей» є шаблоном, який складається з об'єктів «команда» і серії об'єктів-виконавців. Кожен об'єкт-виконавець має логіку, що описує типи об'єктів «команда», які він може обробляти, а також як передати далі

ланцюжком ті об'єкти-команди, що він не може обробляти. Крім того існує механізм для додавання нових призначених для обробки об'єктів у кінець ланцюжка.

У варіаціях стандартного ланцюжка відповідальностей, деякі обробники можуть бути в ролі диспетчерів, які здатні відсилати команди в різні напрямки формуючи Дерево відподальності. У деяких випадках це можна організувати рекурсивно, коли оброблюваний об'єкт викликає об'єкт вищого рівня обробки з командою що пробує вирішити меншу частину проблеми; у цьому випадку рекурсія продовжує виконуватися поки команда не виконається, або поки дерево повністю не буде оброблене. XML-інтерпретатор (проаналізований, але який ще не було поставлено на виконання) може бути хорошим прикладом.

Цей шаблон застосовує ідею слабого зв'язку, який розглядається як програмування у найкращих практиках.

Система автоматизованого документування

Слід відмітити, якщо підсистему планується підтримувати за ліцензією GNU GPL то необхідно провести документування модулів, що реалізуються. Мова програмування Java має у своєму складі інтегровані засоби для створення документації, вони називаються JavaDoc.

Javadoc – генератор документації в HTML-форматі з коментарів вихідного коду на Java від Sun Microsystems. Javadoc – стандарт для документування класів Java. Більшість середовища життє розробки програмного забезпечення автоматично генерують HTML-документацію, використовуючи Javadoc.

Javadoc також надає API для створення доклетів і теглетів, які дозволяють програмісту аналізувати структуру Java-додатки.

Засоби збірки інформаційної підсистеми

Існують декілька утиліт для автоматичної збірки програмного забезпечення написаного на мові програмування Java. Не завжди є можливість використання засобів IDE для отримання виконуваних файлів. Та у багатьох випадках немає необхідності використання середі розробки. Для збірки інформаційної системи реалізованої мовою програмування Java існує декілька засобів, найбільш уживані це Ant та Maven. Обидві утиліти від розробника мови програмування Java, компанії Sun Microsystems, тепер Oracle.

Apache Ant (англ. ant — мураха і водночас акронім — «Another Neat Tool») — java-утиліта для автоматизації процесу збирання програмного продукту.

Ant — платформно-незалежний аналог UNIX-утиліти *make*, але з використанням мови Java, він вимагає платформи Java, і краще пристосований для Java-проектів. Найпомітніша безпосередня різниця між Ant та Make те, що Ant використовує XML для опису процесу збирання і його залежностей, тоді як Make має свій власний формат Makefile. За замовчуванням XML-файл називається build.xml.

Ant був створений в рамках проекту Jakarta, сьогодні — самостійний проект першого рівня Apache Software Foundation.

Перша версія була розроблена інженером Sun Microsystems Джеймсом Девідсоном (James Davidson), який потребував утиліти подібної make, розробляючи перший референтну реалізацію J2EE.

На відміну від make, утиліта Ant повністю незалежна від платформи, потрібно лише наявність на застосовуваній системі встановленої робочого середовища Java — JRE. Відмова від використання команд операційної системи і формат XML забезпечують переносимість сценаріїв.

Управління процесом складання відбувається за допомогою XML-сценарію, який також називають Build-файлом. У першу чергу цей файл містить визначення проекту, що складається з окремих цілей (Targets). Цілі порівняні з процедурами в мовах програмування і містять виклики команд-

завдань (Tasks). Кожне завдання являє собою неподільну, атомарному команду, що виконує певну елементарну дію.

Між цілями можуть бути визначені залежності — кожна мета виконується тільки після того, як виконані всі цілі, від яких вона залежить (якщо вони вже були виконані раніше, повторного виконання не здійснюється).

Типовими прикладами цілей є `clean` (видалення проміжних файлів), `compile` (компіляція всіх класів), `deploy` (розгортання програми на сервері). Конкретний набір цілей та їхнього взаємозв'язку залежать від специфіки проекту.

Ant дозволяє визначати власні типи завдань шляхом створення Java-класів, що реалізують певні інтерфейси.

Maven програмний інструмент для управління Java проектами `management` та `build` програм, створений Джейсоном ван Зилом (Jason van Zyl) у 2002 році. У функціонуванні подібний до Apache Ant але має простішу `build`-модель конфігурації, яка базується на форматі XML. Maven хоститься Apache Software Foundation, де він був частиною Jakarta Project.

Maven використовує конструкцію відому як Project Object Model (POM) для опису програмного проекту, який потрібно побудувати (`build`), залежності від зовнішніх модулів та компонентів та порядку побудови. Надходить з визначеними заделегідь цілями для виконання певних чітко визначених задач, таких як компіляція коду та пакетування.

Ключовою особливістю Maven є його мережева готовність (`network-ready`).

Двигун ядра може динамічно завантажувати плагіни з репозиторію, того самого репозиторію що забезпечує доступ до багатьох версій різних джава-проектів з відкритим кодом, від Апач та інших організацій та розробників. Цей репозиторій та його реорганізований наступник, - Maven 2 репозиторій, - намагається бути де-факто механізмом для дистрибуції джава програм, але прийняття його в такій якості йде повільно.

Maven забезпечує підтримку побудови не просто перебираючи файли з цього репозиторію, але й завантажуючи назад артефакти у кінці побудови. Локальний кеш скачаних артефактів діє як первісний засіб синхронізації виходу проєктів на локальній системі.

Maven базується на плагін-архітектурі, що дозволяє зробити використання будь-якої програми контрольованим через стандартний вхід. Теоретично, це могло б дозволити будь-кому писати плагіни для інтерфейсу з інструментами для побудови (компілятори, тестери і т.п.) для будь-якої мови.

Отже для збірки цільової системи оберемо утиліту Ant. У Доданку А наведені інструкції по зборці.

Модель переходів ставні

Однією з характерних особливостей систем різної природи і призначення є взаємодія між собою окремих елементів, з яких утворені ці системи. Йдеться про те, що різні складові елементи систем не існують ізольовано, а роблять певний вплив один на одного, що і відрізняє систему як цілісне утворення від простої сукупності елементів.

У мові UML взаємодія елементів розглядається в інформаційному аспекті їх комунікації, тобто взаємодіючі об'єкти обмінюються між собою деякою інформацією. При цьому інформація приймає форму закінчених повідомлень. Іншими словами, хоча повідомлення і має інформаційний зміст, воно має додаткову властивість робити направлений вплив на свого одержувача. Це повністю узгоджується з принципами ООП, коли будь-які види інформаційної взаємодії між елементами системи повинні бути зведені до відправки і прийому повідомлень між ними.

Взаємодії об'єктів можна розглядати в часі, і тоді для представлення тимчасових особливостей передачі і прийому повідомлень між об'єктами використовується діаграма послідовності.

На діаграмі послідовності зображуються виключно ті об'єкти, які безпосередньо беруть участь у взаємодії і не показуються можливі статичні

асоціації з іншими об'єктами. Для діаграми послідовності ключовим моментом є саме динаміка взаємодії об'єктів в часі. При цьому діаграма послідовності має як би два вимірювання. Одне – зліва направо у вигляді вертикальних ліній, кожна з яких зображає лінію життя окремого об'єкту, що бере участь у взаємодії. Графічно кожен об'єкт зображується прямокутником і розташовується у верхній частині своєї лінії життя.

Друге вимірювання діаграми послідовності — вертикальна тимчасова вісь, направлена зверху вниз. Початковому моменту часу відповідає сама верхня частина діаграми. При цьому взаємодії об'єктів реалізуються за допомогою повідомлень, які посилаються одними об'єктами іншим. Повідомлення зображуються у вигляді горизонтальних стрілок з ім'ям повідомлення і також утворюють порядок за часом свого виникнення. Іншими словами, повідомлення, розташовані на діаграмі послідовності вище, ініціюються раніше тих, які розташовані нижче. При цьому масштаб на осі часу не вказується, оскільки діаграма послідовності моделює лише тимчасову впорядкованість взаємодій типу "раніше-пізніше".

Діаграма послідовності підсистеми приведена на Рисунку 3.8.

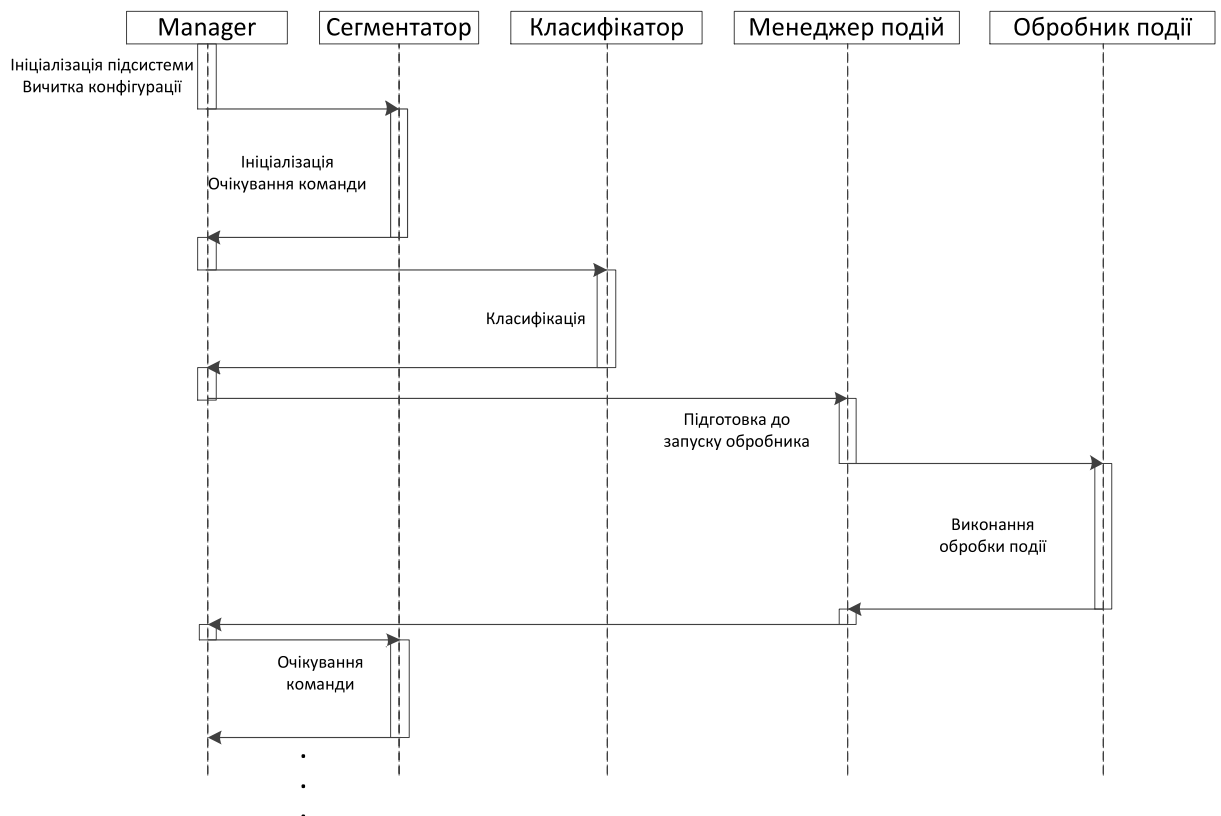


Рисунок 3.9 – Діаграма класів для підсистеми, що розробляється

Отже при старті підсистеми стартує головний модуль підсистеми – менеджер, котрий реалізує керування іншими модулями. При старті виконується ініціалізація всієї підсистеми відповідно до конфігураційного файлу. Далі керування передається сегментатору, котрий повертає до менеджера команду, котра далі передається класифікатору, та у разі успішного класифікації керування передається до менеджера подій та далі до обробника окремої події. Далі підсистема готова отримувати наступну команду.

Опис функцій модулів програмного забезпечення інформаційної підсистеми

Інформаційна підсистема голосового керування складається з наступних частин:

- **Аудіо-приймач.** Задача цього вузла реалізовувати взаємодію між апаратними пристроями, наприклад мікрофоном, та блоком сегментації. Цей вузол може бути замінено на інший, відповідно до системи у яку підсистема інтегрується;
- **Вузол сегментації.** Задача цього вузла полягає у аналізі вхідного потоку аудіо-інформації та виділення з нього окремих команд за деяким алгоритмом. Слід зазначити, що цей вузол також може бути замінено у разі необхідності;
- **Класифікатор.** Основна задача цього вузла – вирішення задачі класифікації. Тобто по вхідної аудіо-команді та існуючої бази, що входить до складу класифікатору, блок вирішує належить вхідна мовна команда до якогось з існуючих класів або ні. Та передає відповідь-результат до менеджера. Виходячи з вимоги масштабованості цей вузол може бути замінено на більш ефективний, а бо більш відповідний до потреб системи, до якої проводиться інтеграція;

- **База даних.** Цей елемент входить до складу класифікатора, оскільки окрема реалізація класифікатора може використовувати різноманітні еталони дані. Тобто класифікатор перероблює вхідні еталони мовні команди до формату необхідному окремій реалізації класифікатора;
- **Менеджер.** Задача цього вузла організація взаємодії між класифікатором, вузлом сегментації та менеджером подій. Оскільки до вузла сегментації та класифікатора застосовуються вимоги гнучкості, поточний вузол має вміти працювати з наведеними вузлами за деяким внутрішнім принципом. Головна задача цього вузла маршрутизування даних між вузлами, та надання зовнішнього інтерфейсу користувачеві.
- **Менеджер подій.** На основі команди, переданої з менеджеру менеджер подій виконує обробку команди передаючи керування окремому обробнику подій.
- **Обробник подій.** Сукупність цих блоків реалізують безпосередню взаємодію з зовнішньою системою виконуючи відповідні дії, приписані до цього обробника. Таке рішення дозволяє реалізувати масштабування системи при доданні нових команд та відповідних дій.

3.14 Рішення стосовно тестування підсистеми

Тестування підсистеми будемо виконувати на двох рівнях: модульно-функціональне тестування та фінальне тестування, котре виконується при здачі проекту заказникам. Оскільки за прийомне тестування (фінальне) відповідає заказник та тестування функцій підсистеми є важливим кроком створення підсистеми то приділимо цьому більшу увагу. При розробці підсистеми, реалізації окремих класів, для кожного з розроблених класів будемо будувати функціональні тести, кожен з котрих буде тестувати окремий метод класу. Такий метод тестування належить до методології тестування чорним ящиком. Тобто виділяються декілька класів

еквівалентностей, на котрі можна розбити всі варіанти використання методу, та за кожним з класів еквівалентності будується окремий тест.

Для мови програмування Java є окрема бібліотека JUnit, котра дозволяє організовувати тестування у автоматизованому режимі. На даний момент часу актуальною версією цього пакету є версія 4.0.

Опит, отриманий при роботі с JUnit, дуже важливий при розробці концепцій тестування програмного забезпечення.

У Доданку А наведено тестові модулі для тестування класів підсистеми.

Рішення стосовно розробки документації

Документування при розробці та експлуатації програмного забезпечення або інформаційної системи (підсистеми) являє етап, котрий дозволяє спростити відповідно розробку та експлуатацію.

Можна умовно поділити документування на два етапи, це етап розробки, до котрого можна віднести різноманітні ескізи екранних форм, деталізації окремих частин системи, наприклад діаграми класів відповідно до модулю, реалізації окремих класів у термінах окремої мови.

Відповідно до вимог до підсистеми необхідно розробити наступну документацію:

- Керівництво користувача

Повинно надаватися користувачеві при встановленні системи.

Може бути як у електронному вигляді (PDF) або у печатному.

- Керівництво програміста

Повинно надаватися програмістові при доробці інформаційної підсистеми, може бути у форматі PDF або у більш зручному для програміста Java – у форматі JavaDoc.

Керівництво користувача та програміста наведено у Доданку Б та Доданку В, відповідно.

Висновки з розділу 3

У процесі виконання даного розділу було отримано ряд проектних рішень щодо розробки, експлуатації та підтримки підсистеми, що розробляється. Крім того було обрано мову програмування (Java), методику тестування(JUnit) та утиліту для збірки підсистеми (Ant).

Для документування класів, що реалізують функціонал підсистеми було використано систему документування інтегровану до мови програмування Java – JavaDoc. Це дозволяє суттєво зменшити час на розробку специфічної документації та при розробці зовнішніх модулів до підсистеми.

Було розроблено концепцію та шаблони, що необхідні для реалізації цільової підсистеми. Розроблені програмні інтерфейси та класи.

У якості алгоритму розпізнавання було обрано формантний метод, оскільки він дозволяє досягнути необхідного рівня гнучкості, та потребує не значних ресурсів для розпізнавання.

Проектні рішення відповідають вимогам, що було висунуто у Розділі 2, зокрема гнучкості, незалежності від платформи, низького використання ресурсів (оптимальності виконання). Відповідно до можливості використання – підсистема голосового розпізнавання може використовуватися у мобільних пристроях, зокрема які побудовані на архітектурі ARM9, та найбільш ефективності досягають на вбудованих рішеннях, що підтримують технологію Jazelle.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1 Загальні принципи охорони праці

Згідно до ст.1 Закону "Про охорону праці", охорона праці є системою збереження життя та здоров'я працівників у процесі трудової діяльності, що включає у себе правові, соціально-економічні, організаційно-технічні, санітарно-гігієнічні, лікувально-профілактичні, реабілітаційні та інші засоби та заходи. Як правовий інститут охорона праці визначає цілий комплекс правових норм.

Необхідно відмітити, що охорону праці не можна ототожнювати з технікою безпеки, виробничій санітарією, гігієною праці, оскільки вони є елементами охорони праці, її складовими частинами. Таким чином до складу системи охорони праці входять наступні елементи:

- **Техніка безпеки**

Звід правил та положень, що направлені на забезпечення умов безпечної праці та/або проведення будь-яких інших робіт.

- **Виробнича санітарія**

Система організаційних заходів та технічних засобів, що запобігають або зменшують вплив на працюючих шкідливих виробничих факторів.

- **Гігієна праці**

Профілактична медицина, що вивчає умови та характер праці, їх вплив на здоров'я на функціональний стан людини та що розроблює наукові основи та практичні методи щодо профілактики шкідливих та небезпечного впливу факторів виробничого середовища та трудового процесу на працюючих

- **Електробезпека**

Описує стан захищеності працівника від шкідливого та небезпечного впливу електричного току або дуги, електромагнітного поля та статичної електрики.

- **Пожежна безпека**

Описує стан захищеності людини, майна, суспільства та держави від пожежі.

- **Виробнича безпека**

Описує стан захищеності життєво важливих інтересів людини та суспільства щодо аварій на небезпечних виробничих об'єктах та наслідків зазначених аварій. У свою чергу охорона праці, електробезпека виробнича безпека та пожежна безпека є складовими частинами безпеки життєдіяльності.

- **Безпека життєдіяльності**

Наука о комфортної та безпечної взаємодії людини з техносферою.

- **Керування безпекою труда**

Система націлена на організацію робота по забезпеченню безпеки, зниженню травматизму та аварійності, професійних захворювань, покращенню умов праці на основі комплексу задач по створенню безпечних та без шкідливих умов праці. Будується на застосуванні законодавчих нормативних актів у області охорони праці.

Ст.4 Закону "Про охорону праці" формулює принципи державної політики у сфері охорони праці. Ці принципи в силу вказаного їх визначення, слід кваліфікувати і як принципи інституту охорони праці, які підлягають застосуванню при відсутності відповідних конкретних правових норм, і як такі, що визначають напрямки подальшого розвитку інституту охорони праці, і обов'язки держави, які вона взяла на себе і має додержувати при здійсненні своїй політики у майбутньому. З урахуванням викладеного, принципи, зазначені ст. 4 Закону "Про охорону праці", мають певне регулятивне значення, названій статті закріплені такі принципи:

- пріоритет життя і здоров'я працівників відносно результатів виробничої діяльності підприємства, повна відповідальність власника за створення безпечних і нешкідливих умов праці;

- комплексне вирішення завдань охорони праці на засадах національних програм з цих питань, а також з урахуванням інших напрямків економічної і

соціальної політики, досягнень в галузі науки, техніки охорони здоров'я;

- соціальний захист працівників, повне відшкодування шкоди особам потерпілим від нещасних випадків на виробництві та професійних захворювань;

- установлення єдиних нормативів з охорони праці для всіх підприємств незалежно від форм власності і видів їх діяльності;

- використання економічних методів управління охороною праці. Проведення політики пільгового оподаткування, що сприяє створенню безпечних та нешкідливих умов праці, участь держави в фінансуванні заходів з охорони праці;

- здійснення навчання населення, професійної підготовки та підвищення кваліфікації працівників з питань охорони праці;

- забезпечення координації діяльності державних органів, установ, організацій та об'єднань громадян, що вирішують різні проблеми охорони здоров'я, гігієни та безпеки праці, а також співробітництво.

Умови праці на робочому місці, безпека технологічних процесів, машин, механізмів, устаткування та інших засобів виробництва, стан засобів колективного та індивідуального захисту, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. Власник або уповноважений ним орган повинен впроваджувати сучасні засоби техніки безпеки, які запобігають виробничому травматизмові, і забезпечувати санітарно-гігієнічні умови, що запобігають виникненню професійних захворювань працівників.

На власника або уповноважений ним орган покладається систематичне проведення інструктажу (навчання) працівників з питань охорони праці, протипожежної охорони.

Охорона праці – один з центральних інститутів трудового права. Він має виключно практичне значення. Недодержання вимог охорони праці створює небезпеку для здоров'я і життя працівників. У свою чергу і ті, кого законодавець називає власником або уповноваженим ним органом, несуть

сувору, у тому числі й кримінальну, відповідальність за порушення правил охорони праці.

Аналіз небезпечних і шкідливих факторів при роботі з ПК у відділі програмного забезпечення

З точки зору впливу небезпечних і шкідливих факторів при роботі з ПК у відділі програмного забезпечення можна виділити наступні:

1) Вплив електромагнітного та електростатичного випромінювань

При роботі ПК виникають два типи випромінювань: електростатичне та електромагнітне. Навколо електростатично зарядженого монітора підвищується концентрація пилу. Електромагнітне випромінювання створюється магнітними котушками, що знаходяться біля цокольної частини ЕПТ. Результати медичних досліджень показують, що електризований пил викликає запалення шкіри, а електромагнітне випромінювання приводить до порушення біологічних процесів (порушення сну, зниження кров'яного тиску, уповільнення серцевої діяльності, зміни складу крові).

2) Вплив недостатньої освітленості робочого місця.

Напружена зорова робота внаслідок нераціонального освітлення може бути причиною функціональних порушень у зоровому аналізаторі і привести до розладу зору, а у важких випадках – і до повної втрати. Втома органів зору залежить від ступеня напруженості процесів, що супроводжують зорове сприйняття. При роботі оператора ПК одним з основних навантажень на організм є навантаження на зір, тому дуже важливо приділити особливу увагу освітленості робочого місця.

У залежності від джерела світла виробниче висвітлення може бути трьох видів: природним, штучним і сполученим. При роботі з ПК вдень у ясну погоду використовується природне освітлення, яке здійснюється через світлові віконні прорізи. При нестачі природного світла, а також при роботі у вечірній час використовується штучне освітлення, джерелами якого є газорозрядні лампи. При такому освітленні величезне значення має правильне розташування джерел світла, їхній тип і кількість.

Правильна освітленість робочого місця сприяє високій працездатності користувача ПК і зводить до мінімуму шкідливий вплив на органи зору. Освітленість робочого місця, що рекомендується для користувача ПК складає 400–750 лк.

3) Вплив електричного струму.

Робота комп'ютерів і периферійних пристроїв можлива завдяки електричному струмові, до джерел якого вони підключаються, тому існує небезпека ураження користувача струмом. Щоб уникнути нещасних випадків варто строго дотримуватися правил електробезпеки.

Небезпека електричного струму на відміну від інших небезпек збільшується тим, що людина не в змозі без спеціальних приладів виявити напругу дистанційно, а також швидкоплинністю поразки – небезпека виявляється, коли людина вже уражена. Ураження струмом приводить до різних порушень в організмі, викликаючи як місцеву поразку тканин і органів, так і загальне ураження організму. Людина відчуває дратівну дію перемінного струму промислової частоти силою 0,6–1,5 мА і постійного струму 5–7 мА. Ці струми не представляють серйозної небезпеки для організму людини, тому що при такій силі струму можливе самостійне звільнення людини від контакту зі струмоведучими частинами. Для перемінного струму промислової частоти сила невідпускаючого струму знаходиться в межах 6–20 мА. Постійний струм не викликає невідпускаючого ефекту, але приводить до сильних болючих відчуттів, сила такого струму 15–80 мА і більше.

4) Вплив шуму.

Шумом прийнято називати всякий небажаний для людини звук, що заважає сприйняттю корисних сигналів. Шум являє собою безладне сполучення звуків різної інтенсивності і частоти. Він викликає труднощі з розпізнанням колірних сигналів, знижує швидкість сприйняття кольору, гостроту зору, зорову адаптацію, порушує сприйняття візуальної інформації,

зменшує на 5–12% продуктивність праці. Тривалий вплив шуму з рівнем звукового тиску 90 дБ знижує продуктивність праці на 30–60%.

Робота ПК може супроводжуватися шумами, викликаними деякими рухливими пристроями усередині комп'ютера, а також підключеними до нього. До таких пристроїв можна віднести вентилятори, пристрої читання дисків, принтери. Виникаючий шум, у залежності від інтенсивності, може привести до підвищення нервової напруги і зниження працездатності.

5) Вплив несприятливих умов мікроклімату.

Мікрокліматичні параметри (температура, відносна вологість і швидкість руху повітря) впливають на самопочуття і здоров'я людини, а також на надійність роботи засобів обчислювальної техніки.

При підвищеній температурі і вологості в приміщенні відбувається перегрівання тіла, що погіршує самопочуття і працездатність. Варто враховувати, що ПК, які знаходяться в приміщенні, сприяють збільшенню температури.

Для комфортної роботи приймається температура $23 (\pm 2)^{\circ}\text{C}$ й $18 (\pm 2)^{\circ}\text{C}$ в теплий і холодний час року відповідно, при відносній вологості $55 (\pm 5)\%$.

6) Вплив нераціонального розташування устаткування і неправильної організації робочого місця

Робоче місце – це частина простору, у якій користувач ПК здійснює трудову діяльність і проводить велику частину робочого часу. Важливу роль грає планування робочого місця, яке повинне задовольняти вимогам зручності виконання робіт і економії енергії і часу користувача, зручності обслуговувань ЕОМ, дотримання правил охорони праці.

При плануванні робочого місця необхідно враховувати зони досяжності при розташуванні моніторів, клавіатури, маніпулятора «миша». Ці зони встановлюються на підставі антропологічних даних людського тіла. Під час роботи користувач ПК не повинен піддаватися впливові сторони подразників, якими можуть бути неадекватне фарбування пристроїв ЕОМ або приміщення.

Тривале перебування людини в зоні комбінованого впливу різних несприятливих факторів може привести до професійного захворювання.

Розрахунок системи загального освітлення приміщення відділу програмного забезпечення

В приміщенні відділу програмного забезпечення необхідно організувати змішане освітлення (комбінування природного і штучного освітлення). Природне освітлення створюється бічним освітленням через вікно. Штучне освітлення використовується при недостатньому природному освітленні. У даному приміщенні використовується загальне штучне освітлення. Розрахунок виконаємо за методом світлового потоку з урахування потоку, віддзеркаленого від стін та стелі.

Необхідно зробити розрахунок загального освітлення виробничої приміщення. Довжина $A = 15$ м, ширина $B = 10$ м, висота $H = 3,5$ м. Висота робочої поверхні $h_p = 1$ м. Для освітлення використовується світильні УТІМ-5. Мінімальна освітленість лампи наколювання по нормах $E_{min}=100$ лк. Коефіцієнт відображення стелі $S_n=50\%$, стін $S_c=30\%$, робочої поверхні $S_p=10\%$. Напруга мережі 220 В.

Відстань від стелі до робочої поверхні:

$$H_0 = H - h_p = 3,5 - 1 = 2,5 \text{ м.}$$

Відстань від стелі до світильника:

$$h_c = 0,2H_0 = 0,2 * 2,5 = 0,5 \text{ м.}$$

Висота підвісу світильника над освітленою поверхнею:

$$h_c = 0,2H_0 = 0,2 * 2,5 = 0,5 \text{ м.}$$

Висота підвісу світильника над підлогою:

$$H_n = h + h_p = 2 + 1 = 3 \text{ м.}$$

Для досягнення найбільшої рівномірності освітлення приймаємо відношення $L/h = 1,5$.

Відстань між центрами світильників:

$$L = 1,5h = 1,5 * 2 = 3 \text{ м.}$$

Необхідна кількість світильників:

$$N = S / L^2 = (15 * 10) / 3^2 \approx 17.$$

Приймаємо кількість світильників: 18 (3 ряди по 6 світильників).

Індекс приміщення:

$$i = (A * B) / (h * (A + B)) = (15 * 10) / (2 * (15 + 10)) = 3.$$

Коефіцієнт використання світлового потоку $\eta = 0,66$.

Світловий потік однієї лампи:

$$\Phi = (E_{\min} S K_z Z) / (N \eta) = (100 * 150 * 1,3 * 1,15) / (18 * 0,66) = 1887 \text{ лм.}$$

Вибираємо (у таблиці) лампу, що має світловий потік 1995 лм:

ЛДЦ 40-4.

Фактична освітленість:

$$E_{\phi} = E_{\min} \Phi_{\lambda} / \Phi_p = 100 * 1995 / 1887 = 106 \text{ лк.}$$

Загальна потужність:

$$P_{\text{общ}} = P_{\lambda} N = 40 * 18 = 720 \text{ Вт.}$$

Таким чином, для оптимальної роботи в приміщенні необхідно встановити 3 ряди по 6 світильників типу ЛДЦ 40-4. Загальна потужність усіх світильників складе 720 Вт.

Розробка заходів щодо зменшення впливу небезпечних і шкідливих факторів

Невиконання вимог безпеки призводить к тому, що через деякий час роботи за комп'ютером працівник починає відчувати дискомфорт: у нього виникають головні болі та різь у очах, виникають втома та дратівливість. У деяких людей порушується сон, погіршується зір, починають боліти руки, шия, поясниця та таке інше.

З законодавства виходить наступне:

- Максимальний час роботи за комп'ютером не повинно перевищувати за зміну;
- Необхідно робити перерви у роботі з ПК тривалістю 10 хвилин кожні 45 хвилин роботи;
- Тривалість неперервної роботи з комп'ютером без регламентної перерви не повинна перевищувати 1 час;

- Під час регламентної перерви з ціллю зниження нервово-емоційного напруження та стомлення зору, запобігання розвитку позотонічної втоми доцільно виконувати комплекси спеціальних вправи.

Відповідно до законодавства існують деякі нормативи, що регламентують вимоги щодо робочого місця користувача ПК. Тобто площа робочого місця не повинна бути менше 4,5м². В приміщенні повинна проводитися щоденні вологі прибирання та систематичні провітрювання після кожної години роботи. Обладнання, що шумить (друкуючі пристрої, сканери, сервера, та інше), тобто рівні шуму для котрого перевищують нормативні показники, повинно розміщуватися поза робочих місць співробітників.

Робочі столи слід розміщувати таким чином, щоб монітори були орієнтовані бічною стороною до світлових отворів, щоб природне світло падало переважно ліворуч.

При розміщенні робочих місць відстань між робочими столами повинно бути не менше 2,0 м, а відстань між бічними поверхнями відеомоніторів – не менше 1,2 м. Робочі місця співробітників, що виконують творчу роботу і вимагає значного розумового напруження або високої концентрації уваги, рекомендується ізолювати один від одного перегородками висотою від 1,5 м.

Конструкція робочого столу повинна забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання. Висота робочої поверхні столу повинна бути 725 мм, робоча поверхня столу повинна мати ширину 800..1400 мм і глибину 800..1000 мм. Робочий стіл повинен мати простір для ніг висотою не менше 600 мм, шириною – не менше 500 мм, глибиною на рівні колін – не менше 450 мм і на рівні простягнутої ноги – не менше 650 мм.

Конструкція робочого стільця або крісла повинна забезпечувати підтримку раціональної робочої пози працівника і дозволяти змінювати позу з метою зниження статичного напруження м'язів шийно-плечової області і

спини. Робочий стілець або крісло повинні бути підйомно-поворотним, регульованим за висотою і кутах нахилу сидіння і спинки, а також відстані спинки від переднього краю сидіння, при цьому регулювання кожного параметра має бути незалежною, легко здійснюваною мати надійну фіксацію.

Клавіатуру слід розташовувати на поверхні столу на відстані 100..300мм від краю, зверненого до користувача, або на спеціальній поверхні, відокремленої від основної стільниці.

Екран відеомонітору повинен знаходитися від очей користувача на відстані 600..700 мм, але не ближче 500.

При виконанні наведених вище рекомендацій щодо організації режиму роботи та робочого місця співробітника суттєво зменшуються ризики виникнення професійних захворювань. Крім того необхідно дотримуватися загальної техніки безпеки при користуванні електричною технікою.

Висновки до розділу 4

У рамках дослідження та проєктування інформаційної підсистеми голосового управління для медичної галузі було проведено комплексний аналіз умов праці, небезпек і шкідливих факторів, що можуть виникати під час розробки, тестування та експлуатації системи.

З'ясовано, що ключовими аспектами охорони праці для ІТ-фахівців, залучених до створення голосових систем, є:

- дотримання ергономічних вимог до робочого місця (освітлення, мікроклімат, шум, електробезпека),
- профілактика психоемоційного та зорового перенапруження,
- організація безпечної роботи з комп'ютерною технікою та периферійними пристроями.

Особливу увагу приділено медичному середовищу як місцю впровадження голосового інтерфейсу:

було враховано вимоги санітарно-гігієнічних норм, забезпечення стерильності, а також безконтактної взаємодії з системою, що знижує ризик перехресного зараження.

Підсистема голосового управління покликана не лише підвищити ефективність праці медичного персоналу, а й **зменшити фізичне навантаження**, скоротити час взаємодії з інформаційними системами та **забезпечити додатковий рівень безпеки** під час виконання клінічних процедур.

Таким чином, запропоновані організаційно-технічні рішення відповідають чинним вимогам охорони праці та створюють безпечні умови як для розробників, так і для кінцевих користувачів у медичній сфері.

ВИСНОВКИ

У ході виконання магістерської роботи було виконано дослідження існуючих методів розпізнавання мовних команд. Існуючі методи розпізнавання можна умовно розділити на 3 групи.

Методи першої групи будуються на штучних нейронних мережах. Ці методи мають досить добрий показник розпізнавання та головними недоліками є необхідність дослідження пов'язані з розробкою структури нейронної мережі та досить низький показник гнучкості.

Методи другої групи використовують приховані марківські моделі для вирішення задачі класифікації. Ця група методів дозволяє побудувати лінгвістичну модель окремої мови. Недоліком цього класу методів є складна реалізація структури та велика обчислювальна складність.

Методи, що базуються на спектральному аналізі голосового сигналу показали, що мають досить не поганий потенціал при побудові систем розпізнавання. Оскільки спектрально-часовий метод аналізу сигналу надає багато інформації щодо поведінки сигналу за часом його можна використати як джерело необхідних ознак для перших двох груп методів. Цей метод має недолік – він потребує значної обчислювальної спроможності системи.

Отже при розробці підсистеми розпізнавання мовлення було вирішено використовувати формантний метод, оскільки він зарекомендував себе досить ефективним та гнучким.

Також у ході виконання були розроблені вимоги до інформаційної підсистеми голосового керування, котра призначена для спрощення процесу керування операційною системою завдяки створенню додаткового звукового інтерфейсу.

Підставами для розробки підсистеми є підвищення ефективності роботи людини за персональним комп'ютером, що у свою чергу підвищує ефективність та зберігає ресурс часу для працівника.

Розробка відповідної системи необхідна не тільки користувачам персональних комп'ютерів але й користувачам різноманітних приладів, що мають у своєму складі операційну систему та аудіо-підсистему.

Булi розробленi вимоги щодо функціоналу та задач, що підсистема повинна вирішувати. Головними вимогами є:

- розрізняння вимовлених мовних команд;
- додавання нових мовних команд;
- можливість підключення специфічних джерел аудіо інформації;
- можливість використання сторонніх методів класифікації.

Та головними вимогами до концепції системи:

- Закласти можливість зміни модулів збору аудіо інформації, класифікації;
- Закласти можливість додавання реакцій підсистеми на кожен окрему голосову команду;
- Можливість з найменшими затратами провести інтегрування підсистеми до складу окремої системи (операційної системи, приладу)

На основі функціонально-вартісного аналізу було визначено найбільш оптимальну концепцію системи для розробки цільової підсистеми.

У процесі виконання магістерської роботи було отримано ряд проектних рішень щодо розробки, експлуатації та підтримки підсистеми, що розробляється. Крім того було обрано мову програмування (Java), методику тестування(JUnit) та утиліту для збірки підсистеми (Ant).

Для документування класів, що реалізують функціонал підсистеми було використано систему документування інтегровану до мови програмування Java – JavaDoc. Це дозволяє суттєво зменшити час на розробку специфічної документації та при розробці зовнішніх модулів до підсистеми.

Було розроблено концепцію та шаблони, що необхідні для реалізації цільової підсистеми. Розроблені програмні інтерфейси та класи.

Відповідно до можливості використання – підсистема головусого розпізнавання може використовуватися у мобільних пристроях, зокрема які побудовані на архітектурі ARM9, та найбільш ефективності досягають на вбудованих рішеннях, що підтримують технологію Jazelle.

Також у ході виконання роботи було виконано розділи з охорони праці, охорони навколишнього середовища та цивільної оборони та організаційно-економічний розділ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Владика Н. М Розробка системи розпізнавання голосових даних із використанням алгоритмів машинного навчання
http://repository.lnup.edu.ua:8080/jspui/bitstream/123456789/978/1/%D0%92%D0%BB%D0%B0%D0%B4%D0%B8%D0%BA%D0%B0_%D0%9D_%D0%9C.pdf
2. Абдель-Хамід О., Абдель-Рахман М., Хуї Дж., Лі Д., Пенн Г., Ю Д. Транзакції IEEE/ACM щодо обробки аудіо, мовлення та мови. Том 22. Випуск 10: Згорткові нейронні мережі для розпізнавання мовлення. 2017 р
3. Катренко А. В. Системний аналіз об'єктів та процесів комп'ютеризації. Учб. посіб. / А. В. Катренко. – Львів, 2018.
4. Архітектури рекурентної нейронної мережі довгострокової пам'яті для великомасштабного акустичного моделювання (2021), Хасім Сак та ін., Google Research
5. Python IDE for Professional Developers. – 2021. – Режим доступу до ресурсу: <https://www.jetbrains.com/pycharm/>
6. Miao Y, Gowayyed M, Metze F. EESSEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In: Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on. IEEE; 2015
7. Патлайчук, О. В. Розробка інформаційної підсистеми сповіщень при надзвичайних ситуаціях = Development of information subsystem for emergency notifications : бакалаврська робота ; спец. 122 “Комп'ютерні науки” / О. В. Патлайчук ; наук. кер. С. М. Божаткін. – Миколаїв : НУК, 2021. – 86 с.

ДОДАТОК А – ІНСТРУКЦІЯ КОРИСТУВАЧА

Інформаційна підсистема голосового управління для розпізнавання мовних команд у медичній сфері

1. Загальні положення

Ця інструкція призначена для користувачів інформаційної підсистеми голосового управління, яка застосовується для розпізнавання та обробки мовних команд медичного персоналу. Підсистема спрощує взаємодію з електронними медичними системами, дозволяючи лікарям та медсестрам працювати без ручного введення даних.

2. Призначення підсистеми

Інформаційна підсистема призначена для:

Розпізнавання голосових команд українською мовою.

Автоматичного виконання дій у медичній інформаційній системі (МИС) на основі голосових запитів.

Підвищення швидкості обробки даних та зменшення навантаження на персонал.

3. Вимоги до користувача

Користувач повинен:

Бути авторизованим працівником медичного закладу.

Мати базові навички роботи з комп'ютером.

Знати основні голосові команди, передбачені системою.

4. Запуск та авторизація

Увімкніть комп'ютер або планшет, підключений до внутрішньої мережі медичного закладу.

Запустіть програму «Голосовий асистент МИС».

Увійдіть у систему за допомогою особистого логіна та пароля або через біометричну авторизацію (якщо доступно).

5. Робота з голосовими командами

Після авторизації активується режим прослуховування. Користувач може озвучувати команди, наприклад:

Команда	Дія
«Відкрити історію хвороби Петрова»	Відкриває електронну картку пацієнта
«Записати температуру 37,5»	Вносить дані до карти пацієнта
«Призначити аналіз крові»	Створює відповідне направлення
«Показати останні результати аналізів»	Виводить на екран результати
«Завершити роботу»	Завершує сеанс і виходить з системи

Примітка: Команди повинні вимовлятися чітко, без фонових шумів.

6. Рекомендації щодо користування

Використовуйте мікрофон високої якості.

Говоріть чітко, стандартною літературною мовою.

Не переривайте команду — вимовляйте її повністю.

За потреби, можна повторити команду або сказати: «Повтори», «Скасувати».

7. Типові помилки та вирішення

Проблема	Рішення
Система не розпізнає команду	Повторіть команду повільніше і чіткіше
Команда виконана неправильно	Скажіть «Скасувати» і повторіть ще раз
Відсутність реакції на голос	Перевірте мікрофон і підключення до мережі

8. Завершення роботи

Для завершення роботи:

Скажіть «Завершити роботу» або натисніть кнопку «Вихід».

Переконайтесь, що ви вийшли з облікового запису.

Закрийте програму.

9. Технічна підтримка

У разі виникнення труднощів звертайтеся до служби технічної підтримки:

Телефон: +38 (097) 987-65-43

Email: support@medvoice.ua

Графік: Пн–Пт, 9:00–18:00

ДОДАТОК Б - ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ

– 1. Загальні відомості

Назва проєкту: Інформаційна підсистема голосового управління для розпізнавання мовних команд у медичній сфері.

Замовник: НКП МЛ 3

Розробник: НУК ім.адм. Макарова / кафедра ІУСТ

Дата: 13.02.25

Мета створення системи - Створити інформаційну підсистему, що дозволяє медичному персоналу керувати медичною інформаційною системою (МІС) за допомогою голосових команд українською мовою. Система має забезпечити зручність, швидкість та безконтактність взаємодії, що особливо актуально в умовах інтенсивного робочого навантаження або стерильного середовища.

3. Завдання підсистеми

Розпізнавати голосові команди медичного працівника у реальному часі.

Перетворювати голос у текст та ініціювати відповідну дію в медичній системі.

Інтегруватися з існуючими МІС (за API).

Підтримувати набір базових та розширених команд для роботи з електронною медичною документацією.

Забезпечувати авторизацію користувача.

Забезпечувати логування дій користувача для безпеки та контролю.

4. Функціональні вимоги

4.1 Розпізнавання команд:

Підтримка української (та за потреби англійської) мови.

Визначення та обробка ключових фраз (словник команд).

Можливість розширення словника.

4.2 Основні команди:

Відкрити/закрити картку пацієнта.

Додати показники (температура, тиск, пульс).

Призначити аналізи/обстеження.

Вивести результати досліджень.

Додати діагноз/призначення.

Завершити сеанс.

4.3 Авторизація:

Логін/пароль або голосова/біометрична ідентифікація.

Рівні доступу до функцій системи.

4.4 Інтерфейс:

Графічний інтерфейс з візуалізацією команд та відповідей.

Адаптація під ПК/планшети/мобільні пристрої.

Підтримка темної/світлої теми.

4.5 Логування:

Збереження журналу дій користувача.

Можливість експорту звітів.

5. Нефункціональні вимоги

Час реакції на голосову команду: не більше 2 секунд.

Точність розпізнавання — не нижче 90% при стандартних умовах.

Підтримка інтеграції з МІС через REST API.

Забезпечення відповідності вимогам захисту персональних даних (GDPR/Закон України «Про захист персональних даних»).

6. Технічне середовище

Операційні системи: Windows 10/11, Android 10+, Linux (опціонально)

Мова реалізації: Python / JavaScript / інші з підтримкою AI SDK

Сторонні сервіси: Google Speech-to-Text, Whisper, або інші сервіси розпізнавання

База даних: PostgreSQL / MongoDB (за потреби)

Інтеграція: з HL7/FHIR-сумісними МІС

7. Структура системи

Модуль розпізнавання мови

Модуль обробки команд

Модуль інтерфейсу користувача

Модуль безпеки та авторизації

Інтеграційний API-модуль

Модуль логування

8. Тестування

Функціональне тестування всіх команд.

Тестування точності розпізнавання мовлення.

Тестування безпеки доступу.

Навантажувальне тестування (до 10 одночасних користувачів — базово).

9. Документація

Інструкція користувача.

Технічна документація API.

Настанови з інсталяції та супроводу.

10. Етапи реалізації

Етап	Зміст	Термін
1	Аналіз вимог, формування технічного завдання	1 тиждень
2	Розробка прототипу	2 тижні
3	Реалізація функціоналу	4 тижні
4	Тестування та налагодження	2 тижні
5	Впровадження	1 тиждень
6	Підготовка документації	1 тиждень

11. Умови приймання

Підсистема вважається прийнятою, якщо:

Успішно пройдені всі типи тестування.

Працюють усі заявлені функції.

Забезпечено точність розпізнавання згідно з вимогами.

Надається вся документація.

12. Додаткові умови

Можливість масштабування та інтеграції з іншими модулями.

Підтримка упродовж 6 місяців після впровадження (або за договором).

Технічне завдання

ДОДАТОК В – ФРАГМЕНТ ПРОГРАМНОГО КОДУ

```
pip install SpeechRecognition pyaudio
```

Якщо pyaudio не встановлюється — можна спробувати альтернативу `pip install pipwin && pipwin install pyaudio`

```
import speech_recognition as sr

# Список відомих команд і відповідних дій
COMMANDS = {
    "відкрити історію хвороби": "open_patient_record",
    "записати температуру": "record_temperature",
    "призначити аналіз крові": "assign_blood_test",
    "показати результати аналізів": "show_test_results",
    "завершити роботу": "exit_system"
}

def execute_command(command_text):
    for keyword in COMMANDS:
        if keyword in command_text.lower():
            action = COMMANDS[keyword]
            print(f"[INFO] Команда розпізнана: '{keyword}' → Дія: {action}")
            # Тут можна викликати відповідну функцію або API
            return action
    print("[WARNING] Команда не розпізнана.")
    return None

def listen_and_recognize():
    recognizer = sr.Recognizer()
    microphone = sr.Microphone()

    print("[SYSTEM] Очікування голосової команди...")

    with microphone as source:
        recognizer.adjust_for_ambient_noise(source)
        audio = recognizer.listen(source)

    try:
        # Розпізнавання української мови (можна змінити на 'en-US' для
        # англійської)
        command_text = recognizer.recognize_google(audio, language="uk-UA")
        print(f"[VOICE INPUT] Ви сказали: {command_text}")
        execute_command(command_text)

    except sr.UnknownValueError:
        print("[ERROR] Не вдалося розпізнати голос.")
    except sr.RequestError:
        print("[ERROR] Помилка сервісу розпізнавання.")

if __name__ == "__main__":
    while True:
        listen_and_recognize()
```