Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Комп’ютерні науки
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи

магістр
(освітній рівень)

на тему: **Інформаційна система перевірки та виправлення розмітки коду програм**

Виконав: студент 6 курсу, групи САмз-61
спеціальності 124 «Системний аналіз»
(шифр і назва спеціальності)

_____ Веселовська В.О.
(підпис) (прізвище та ініціали)

Керівник _____ Дмитроца Л.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Мацюк О.В.
(підпис) (прізвище та ініціали)

Рецензент _____ Пастух О.А.
(підпис) (прізвище та ініціали)

м. Тернопіль – 2019

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)

акультет Комп'ютерно - інформаційних систем і програмної інженерії
афедра Комп'ютерної науки
світній ступінь Магістр
апрям підготовки
(шифр і назва)
пеціальність 124 „Системний аналіз"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри к.т.н. доцент Бондарєв Ю.О.
«____» _____ 2019 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Василовська Вероніка Олександрівна
(прізвище, ім'я, по батькові)

. Тема проекту (роботи) Інформаційна система перевірки та виправлення розмітки коду програм

Керівник проекту (роботи) Дмитрова Леся Павлівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від «29» жовтня 2019 року №417-ЗЗ
2. Термін подання студентом проекту (роботи) 10 грудня 2019р
3. Вихідні дані до проекту (роботи) наукові літературні джерела.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ 1.Аналіз джерел і постановка завдання щодо створення інтелектуальної системи перевірки та виправлення розмітки коду 2. Загальна методика та основні методи дослідження щодо інтелектуальної системи. 3. Результати розробки інтелектуальної системи перевірки та виправлення розмітки коду програм Спеціальна частина 5. Охорона праці та безпека в надзвичайних ситуаціях. 6 Обґрунтування економічної ефективності.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Тема, мета, завдання, наукова новизна, перший розділ, другий розділ, третій розділ, застосування нейромережевої системи - алгоритми, архітектура для розпізнавання.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата завдання видав	Підпис, дата завдання прийняв
Охорона праці	Дмитрюка В.П., доцент		
Економіка	Лисенка В.Я., доц.		
Цивільний захист	Майстренко І.І. доц.		
Безпека в надзвичайних ситуаціях	Стадник І.А. проф.	27.11.19	11.12.19
Спеціальна частина	Ахтирук Г.В., ст. викл.		

7. Дата видачі завдання 10 вересня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Затвердження теми дипломної роботи	29.10.19.	Виконано
2	Аналіз літературних джерел	30.10-02.11.19	Виконано
3	Обґрунтування актуальності дослідження	03.11-05.11.19	Виконано
4	Аналіз предмета дослідження та предметної області	06.11-08.11.19	Виконано
5	Проведення дослідження засобів і методів програмної реалізації для побудови системи.	09.11-11.11.19	Виконано
6	Оформлення розділу „Аналіз джерел і постановка завдання щодо створення інтелектуальної системи перевірки та виправлення розміти коду"	12.11-21.11.19	Виконано
7	Оформити розділ „Результати розробки інтелектуальної системи перевірки та виправлення розміти коду програм"	22.11-27.11.19	Виконано
8	Оформити розділ „Загальні методи та основні методи досліджень щодо інтелектуальної системи"	28.11-01.12.19	Виконано
9	Оформити розділ „Спеціальна частина"	02.12-03.12.19	Виконано
10	Оформити розділ „Обґрунтування економічної ефективності"	04.12-05.12.19	Виконано
11	Оформлення розділу „Економіка"	06.12-07.12.19	Виконано
12	Оформити розділ „Охорона праці та безпеки в надзвичайних ситуаціях"	08.12-10.12.19	Виконано
13	Нормоконтроль	11.12-12.12.19	Виконано
14	Попередній захист дипломної роботи	13.12.19	Виконано
15	Захист дипломної роботи	28.12.19	

Студент _____ (підпис) Вельовська В.О (прізвище та ініціали)

Керівник проекту (роботи) _____ (підпис) Дмитрюка В.П. (прізвище та ініціали)

АНОТАЦІЯ

Інформаційна системи перевірки та виправлення розмітки коду програм // Дипломна робота ОР «Магістр» // Веселовська Вероніка Олександрівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група САмз-61 // Тернопіль, 2019 // С. 101, рис. – 28, табл. – 7, додат. – 2, бібліогр. – 40.

Ключові слова: РОЗМІТКА КОДУ, ВИПРАВЛЕННЯ РОЗМІТКИ КОДУ, НЕЙРОННА МЕРЕЖА, ІНФОРМАЦІЙНА СИСТЕМА.

У дипломній роботі створено інформаційну систему перевірки та виправлення розмітки коду програм.

У першому розділі було проведено аналіз з наукових статей та публікації, по темі дипломної роботи. Досліджено всі переваги, які надають системи розпізнавання і автовиправлення файлів, їх основні мотивації та цілі, проаналізовано та наведено приклади практики виправлення помилок.

В ході виконання другого розділу було проведено аналіз існуючих методів та вирішення основного завдання і формулювання задач дослідження. Створено алгоритм, для отримання високого відсотка точності і швидкості автовиправлення файлів при малих обчислювальних витратах Для побудови діаграм ми використовували ПО AFPM.

В третьому розділі було представлено процесс системи шляхом вдосконалення існуючих алгоритмів розпізнавання і автовиправлення файлів.

ANNOTATION

Information system of program code markup check and correction // Master's Thesis // Veselovskaya Veronica Alexandrovna // Ternopil' Ivan Pul'uj National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Science, group SNms-61 // Ternopil, 2019 //// Pages – 101, Fig. – 28, Tables. – 7, Appendixs – 2, Bibliograms. – 40.

Keywords: CODE MARKUP, CODE MARKUP CORRECTION, NEURAL NETWORK, INFORMATION SYSTEM.

In the diploma work the Infromation system of program code markup check and correction was carried.

In the first section, an analysis of scientific articles and publications on the topic of the thesis was conducted. All the benefits of file recognition and auto-correction systems, their main motivations and goals are explored, and examples of error correction practices are analyzed.

During the implementation of the second section, the analysis of existing methods and the solution of the main task and formulation of the research tasks were carried out. An algorithm was created to obtain a high percentage of accuracy and speed of file auto-correction at low computational cost. We used AFPM software to construct diagrams.

The third section introduced the process of the system by improving existing algorithms for file recognition and auto-correction.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AFPM – All Fusion Process Modeler

UML – UnifiedModelingLanguage
Уніфікована мова моделювання

БД – База даних

ІСПТВРКП – Інтелектуальна система перевірки та виправлення розмітки коду программ

КС – Комп’ютерна система

ОС – Операційна система

ПЗ – Програмне забезпечення

СУБД – Система управління базами даних

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі, коли все навколо знаходиться під впливом нових технологій, постає завдання щодо покращення умов роботи та забезпечення нормальних умов праці за рахунок комп’ютерних автоматизованих та інтелектуальних систем, адже решта методів стають ненадійними та застарілими. З кожним роком зростає зацікавленість вирішення більш складних задач виправлення коду, що обумовлена зростаючою кількістю програмістів. Один з перспективних напрямків вирішення даної проблеми ґрунтується на застосуванні штучних нейронних мереж і нейрокомп'ютерів, як найбільш прогресивних по відношенню проблем класифікації задач розпізнавання образів. У наш час запропоновано велику кількість архітектур нейромереж для застосування у розпізнаванні та виправленні об’єктів. Аналіз запропонованих рішень показує, що й досі не існує такої моделі, яка б була кращою серед усіх результуючих показників роботи. Перспективу в удосконаленні архітектур вбачають у загорткових нейронних мережах.

Одним із завдань, рішення яких неможливе без застосування інтелектуальних електронних засобів автоматизації, є перевірка структур коду на відповідність заданним патернам. Базовим компонентом різноманітних електронних засобів перевірки є системи їавтоматичного розпізнавання паттернів.

Система автоматичного розпізнавання образів на сьогодні – це програма, яка реалізує алгоритми автоматичного розпізнавання патернів, пов’язаних зі структурами цих образів, з метою автоматизації отримання та подальшої обробки даних у відповідності до потреб. Автоматичне розпізнавання патернів широко використовується при розробці коду програм, оскільки у багатьох IT-компаніях притримуются певного стилю коду.

Об’єкт дослідження штучні нейронні мережі структур коду у певному файлі; та процес виправлення розмітки коду програм самостійно.

Предмет дослідження – програмні засоби перевірки структур коду програм; модель, в основі якої закладена нейронна мережа, для вирішення задачі така як виправлення розмітки коду програм, а також методи і алгоритму реалізацій архітектурі нейронних мереж.

Мета дослідження – створення інформаційної системи для розпізнавання та автовиправлення коду на основі штучної нейронної мережі; дослідження залежності часу обробки файлу з кодом від степені схожості зразкового файлу з оброблюваним; підвищення ефективності та точності виправлень.

Науковою новизною роботи є вдосконалення методів розпізнавання структур коду з файлів, використовуючи нейронну мережу власної архітектури, що дозволяє покращити метод перевірки та підвищити показник точності виправлень. Вдосконалено оптимізований алгоритм розпізнавання тексту у системах перевірки та виправлення розмітки коду програм, який відрізняється від інших підвищеною точністю та самостійного виправлення.

Практичне значення отриманих результатів. Розроблена інформаційна система дозволить значно спростити управління якістю коду та його чистотою. Таким чином можна привести у компанію стажера та не боятись довірити йому проект. Звичайно система буде йому давати підказки. Запропонована інформаційна система для перевірки та виправлення розмітки коду програм може бути застосована на різного роду IT-підприємствах

Для досягнення мети необхідно:

1. проаналізувати методи та моделі розпізнавання символьної інформації;

2. розробити структуру інтелектуальної системи нейромережевої перевірки структур коду;

3. розробити нейронну мережу для написання розмітки коду;

4. здійснити програмну реалізацію інтелектуальної системи нейромережевого розпізнавання структур коду;

5. провести тестування та аналіз результатів.

1 АНАЛІЗ ДЖЕРЕЛ І ПОСТАНОВКА ЗАВДАННЯ ЩОДО СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПЕРЕВІРКИ ТА ВИПРАВЛЕННЯ РОЗМІТКИ КОДУ ПРОГРАМ

1.1 Аналіз основних понять поставленого завдання

Інформаційна система забезпечує приймання інформації, її перетворення, опрацювання, збереження і передачу результатів опрацювання споживачу: людині, машині, іншій інформаційній системі. Прикладом сучасної інформаційної система може бути редакція газета або журналу, оснащена комп'ютерною технікою.

Інформаційна система – це комплекс інформаційних, технічних, програмних та організаційних засобів, необхідних для автоматизованого опрацювання інформації.

В інформаційній системі можуть відбуватися одночасно один, два чи кілька процесів

Опрацювання інформації залежить від змісту вхідної інформації, але під час самого опрацювання інформація не осмислюється, а лише перетворюється згідно з попередньо розробленими алгоритмами[1].

Розмітка коду - штучна мова, що використовує набір анотацій до тексту, що надає інструкції стосовно структури тексту чи його відображення.

Мови розмітки використовувалися століттями, а в останні роки почали використовуватися в системах комп'ютерної верстки та системах обробки текстової інформації.

Добре відомий сучасний приклад мови розмітки, котра використовується вкомп'ютері- це HTML, одна з найпоширеніших мов в Всесвітній мережі. HTML перейняла деякі з домовленостей щодо розмітки в видавничій індустрії[2].

Hypertext Markup Language(HTML) (укр.Мова розмітки гіпертекстових документів) - стандартна мова розмітки для створення веб-

сторінок і веб-додатків. 3Cascading Style Sheets (CSS) і JavaScript, вона утворює тріаду основних технологій для World Wide Web.

HTML може вбудовувати програми, написані на мові сценаріїв, наприклад JavaScript, що впливає на поведінку та вміст веб-сторінок.

Неронні мережі – це математична програмна модель.

Останнім часом з’являється багато новин про нові розробки технології штучних нейронних мереж. На їх основі створюються комп’ютерні програми штучного інтелекту, яки вже зараз здатні обігрувати найкращих гравців в покер[3].

Найпоширеніші застосування нейронних мереж:

Розпізнавання образів та класифікація. В якості образів можуть виступати різні за своєю природою об’єкти: символи тексту, зображення, зразки звуків і т. д. При навчанні мережі пропонуються різні зразки образів із зазначенням того, до якого класу вони відносяться. Коли мережі пред’являється якийсь образ, на одному з її виходів повинна з’явитися ознака того, що образ належить цьому класу. У той же час на інших виходах повинна бути ознака того, що образ до даного класу не належить.

Прийняття рішень та управління. Це завдання близьке до задачі класифікації. Класифікації підлягають ситуації, характеристики яких надходять на вхід нейронної мережі. На виході мережі повинна з’явитися ознака рішення, яке вона прийняла.

Кластеризація. Під кластеризацією розуміється розбиття множини вхідних сигналів на класи, при тому, що ні кількість, ні ознаки класів заздалегідь не відомі. Після навчання така мережа здатна визначати, до якого класу належить вхідний сигнал.

Прогнозування. Здібності нейронної мережі до прогнозування безпосередньо випливають з її здатності до узагальнення та виділення прихованих залежностей між вхідними та вихідними даними. Після навчання мережа здатна передбачити майбутнє значення якоїсь послідовності на основі декількох попередніх значень або якихось існуючих зараз чинників.

Стиснення даних і асоціативна пам’ять. Здатність нейромереж до виявлення взаємозв’язків між різними параметрами дає можливість висловити дані великої розмірності більш компактно, якщо дані тісно взаємопов’язані між собою. Зворотній процес – відновлення вихідного набору даних з частини інформації – називається асоціативною пам’яттю.

1.2 Тестування програмного забезпечення

Тестування програмного забезпечення (англ.Software Testing - це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки.

Тестування програмного забезпечення – процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином.

Може оцінюватись:

- відповідність вимогам, якими керувалися проектувальники та розробники;
- правильна відповідь для усіх можливих вхідних даних;
- виконання функцій за прийнятний час;
- практичність;
- сумісність з програмним забезпеченням та операційними системами;
- відповідність задачам замовника;

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Як

результат програмне забезпечення (ПЗ) тестується стандартним виконанням програми з метою виявлення багів(помилок або інших дефектів).

Тестування ПЗ може надавати об'єктивну, незалежну інформацію про якість ПЗ, ризики відмови, як для користувачів, так і для замовників.

Тестування може проводитись, як тільки створено виконуваний код (навіть частково завершений). Процес розробки зазвичай передбачає, коли та як буде відбуватися тестування. Наприклад, при поетапному процесі, більшість тестів відбувається після визначення системних вимог і тоді вони реалізуються в тестових програмах. На противагу цьому, відповідно до вимог гнучкої розробки ПЗ, програмування і тестування часто відбувається одночасно [1].

Перші програмні системи розроблялися в рамках програм наукових досліджень або програм для потреб міністерств оборони. Тестування таких продуктів проводилося строго формалізовано із записом всіх тестових процедур, тестових даних, отриманих результатів. Тестування виділялося в окремий процес, який починався після завершення кодування, але при цьому, як правило, виконувалося тим же персоналом.

У 1960-х багато уваги приділялося «вичерпному» тестуванню, яке повинно проводитися з використанням усіх шляхів у коді або всіх можливих вхідних даних. Було відзначено, що в цих умовах повне тестування ПЗ неможливе, тому що, по-перше, кількість можливих вхідних даних дуже велика, по-друге, існує безліч шляхів, по-третє, складно знайти проблеми в архітектурі та специфікаціях. З цих причин «вичерпне» тестування було відхилено й визнано теоретично неможливим.

На початку 1970-х тестування ПЗ розглядалося як «процес, спрямований на демонстрацію коректності продукту» або як «діяльність з підтвердження правильності роботи ПЗ». У програмній інженерії, яка в той час зароджувалася, верифікація ПЗ визначалася як «доказ правильності». Хоча концепція була теоретично перспективною, на практиці вона вимагала багато часу й не охоплювала всі аспекти тестування. Було вирішено, що

доказ правильності - неефективний метод тестування ПЗ. Однак, у деяких випадках демонстрація правильної роботи використовується і в наші дні, наприклад, приймально-здавальні випробування. У другій половині 1970-х тестування представлялося як виконання програми з наміром знайти помилки, а не довести, що вона працює. Успішний тест – це тест, який виявляє раніше невідомі проблеми. Даний підхід цілком протилежний попередньому. Зазначені два визначення являють собою «парадокс тестування», в основі якого лежать два протилежних твердження: з одного боку, тестування дозволяє переконатися, що продукт працює добре, а з іншого - виявляє помилки у ПЗ, показуючи, що продукт не працює. Друга мета тестування є більш продуктивною з точки зору поліпшення якості, оскільки не дозволяє ігнорувати недоліки ПЗ.

У 1980-х тестування розширилося таким поняттям як запобіганням дефектам. Проектування тестів – найбільш ефективний з відомих методів запобігання помилок. В цей же час почали висловлюватися думки, що необхідна методологія тестування, зокрема, що тестування повинно включати перевірки впродовж усього циклу розроблення, при цьому це має бути керований процес. В ході тестування треба перевірити не тільки зібрану програму, але й вимоги, код, архітектуру, самі тести. «Традиційне» тестування, яке існувало до початку 1980-х, відносилося тільки до скомпілювати, готової системі (зараз це зазвичай називається системне тестування), але надалі тестувальники стали залучатися в усі аспекти життєвого циклу розроблення. Це дозволяло раніше знаходити проблеми у вимогах та архітектурі й тим самим скорочувати терміни та бюджет розроблення. У середині 1980-х з'явилися перші інструменти для автоматизованого тестування. Передбачалося, що комп'ютер зможе виконати більше тестів, ніж людина, причому зробить це більш надійно. Спочатку ці інструменти були вкрай простими й не мали можливості написання сценаріїв на скриптових мовах.

На початку 1990-х у поняття «тестування» стали включати планування, проектування, створення, підтримку й виконання тестів та тестових оточень, а це означало перехід від тестування до забезпечення якості, що охоплює весь цикл розроблення ПЗ. У цей час починають з'являтися різні програмні інструменти для підтримки процесу тестування: більш просунуті середовища для автоматизації з можливістю створення скриптів і генерації звітів, системи управління тестами, ПЗ для проведення навантажувального тестування. У середині 1990-х з розвитком Інтернету й розробленням великої кількості веб-застосунків особливої популярності стало набувати «гнучке тестування» (за аналогією з гнучкими методологіями програмування).

У 2000-х з'явилося ще більш широке визначення тестування, коли в нього було додано поняття «оптимізація бізнес-технологій». ВТО направляє розвиток інформаційних технологій згідно з цілями бізнесу. Основний підхід полягає в оцінці та максимізації значущості всіх етапів життєвого циклу розроблення ПЗ для досягнення необхідного рівня якості, продуктивності, доступності.

Тестування – це одна з технік контролю якості, що включає в себе

- Планування робіт (Test Management)
- Проектування тестів (Test Design)
- Виконання тестування (Test Execution)
- Аналіз отриманих результатів (Test Analysis).

Верифікація (Verification)- це процес оцінки системи або її компонентів з метою визначити чи задовольняють результати поточного етапу розробки умовам, сформованим на початку цього етапу. Тобто чи виконуються цілі, терміни, завдання з розробки проекту, визначені на початку поточної фази. Валідація (Validation)- це визначення відповідності розроблюваного програмного забезпечення між очікуваннями і потребами користувача, вимогам до системи.

План Тестування (Test Plan)- це документ, що описує весь обсяг робіт з тестування, починаючи з опису об'єкта, стратегії, розкладу, критеріїв початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

Тест дизайн (Test Design) - це етап процесу тестування програмного забезпечення, на якому проектуються і створюються тестові випадки (тест кейси), відповідно до визначених раніше критеріями якості та цілями тестування.

Тестовий випадок (Тест кейс/Test Case) - це документ, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації тестованої функції або її частини.

Баг/Дефект Репорт (Bug Report)- це документ, що описує ситуацію або послідовність дій (Steps), що призвела до некоректної роботи об'єкта тестування (Misbehavior), із зазначенням причин та очікуваного результату (Expected Result).

Тестове Покриття (Test Coverage) - це одна з метрик оцінки якості тестування, що представляє із себе щільність покриття тестами вимог або коду, що виконується.

Деталізація Тест Кейсів (Test Case Specification)— це рівень деталізації опису тестових кроків і необхідного результату, при якому забезпечується розумне співвідношення часу проходження до тестового покриття.

Час Проходження Тест Кейса (Test Case Pass Time) - це час від початку проходження кроків тест кейса до отримання результату тесту.

1.3 Методи тестування

Статичне та динамічне тестування**:**

Тестова діяльність, що пов'язана з аналізом результатів розробки програмного забезпечення, називається статичним тестуванням. Воно передбачає перевірку програмних кодів, контроль та перевірку програми без запуску на комп'ютері. Тестова діяльність, що передбачає експлуатацію програмного продукту, називається динамічним тестуванням. Динамічне та статичне тестування доповнюють одне одного.

На етапі статичного тестування перевіряється вся документація, отримана як результат життєвого циклу програми. Це і технічне завдання, і специфікація, і вихідний текст програми на мові програмування. Вся документація аналізується на предмет дотримання стандартів програмування. У результаті статичної перевірки встановлюється, наскільки програма відповідає заданим критеріям та вимогам замовника. Усунення неточностей та помилок у документації - запорука того, що створюваний програмний засіб має високу якість.

Динамічні методи застосовуються в процесі безпосереднього виконання програми. Коректність програмного засобу перевіряється на безлічі тестів або наборів підготовлених вхідних даних. При прогоні кожного тесту збираються та аналізуються дані про відмови та збої в роботі програми.

1.4 Виявлення та виправлення помилок

Для виявлення помилок використовують коди виявлення помилок, для виправлення - коригувальні коди (коди, що виправляють помилки, коди з корекцією помилок, завадостійкі коди.

К. Шеннон сформулював теорему для випадку передачі дискретної інформації з каналу із завадами, яка стверджує, що ймовірність помилкового декодування прийнятих сигналів може бути забезпечена як завгодно малою шляхом вибору відповідного способу кодування сигналів.

Клод Елвуд Шеннон (англ.Claude Elwood Shannon; 30 квітня,1916 - 24 лютого,2001) американськийелектротехнік і математик, «батько теорії інформації».

Шеннон відомий тим, що запропонував теорію інформації в науковій статті, опублікованій в 1948 році. Йому також приписують винайдення теорії проекту цифрового комп'ютера та цифрового каналу в 1937 році, коли, бувши 21-річним студентом в Массачусетському технологічному інституті, він написав дисертацію, в якій демонструє, що з допомогою електричного застосування Булевої алгебри можна сконструювати та розв'язати будь-які логічні і числові зв'язки.

Під завадостійкими кодами розуміють коди, що дозволяють виявляти або виявляти і виправляти помилки, які виникають у результаті впливу завад.

Завадостійкість кодування забезпечується за рахунок введення надмірності в кодові комбінації, тобто за рахунок того, що не всі символи в кодових комбінаціях використовуються для передачі інформації

У процесі зберігання даних і передачі інформації з мереж зв'язку неминуче виникають помилки. Контроль цілісності даних і виправлення помило - важливі завдання на багатьох рівнях роботи з інформацією (зокрема, фізичному, канальному, транспортному рівнях мережевої моделі OSI).

У системах зв'язку можливі кілька стратегій боротьби з помилками:

- Виявлення помилок у блоках даних і автоматичний запит повторної передачі пошкоджених блоків - цей підхід застосовується, в основному, на канальному і транспортному рівнях;
- Виявлення помилок у блоках даних і відкидання пошкоджених блоків -такий підхід іноді застосовується в системах потокового мультимедіа, де важлива затримка передачі і немає часу на повторну передачу;
- Виправлення помилок (англ. forward error correction) застосовується на фізичному рівні.

Коригувальні коди-коди, які слугують для виявлення або виправлення помилок, що виникають при передачі інформації під впливом завад, а також при її зберіганні.

Для цього при запису (передачі) у корисні дані додають спеціальним чином структуровану надлишкову інформацію (контрольне число), а при читанні (прийомі) її використовують для того, щоб виявити або виправити помилки. Природно, що число помилок, яке можна виправити, обмежена і залежить від конкретного застосовуваного коду.

З кодами, які виправляють помилки, тісно пов'язані коди виявлення помилок. На відміну від перших, останні можуть тільки встановити факт наявності помилки в переданих даних, але не виправити її.

В дійсності, використовувані коди виявлення помилок належать до тих же класів кодів, що і коди, що виправляють помилки. Фактично будь-який код, що виправляє помилки, може бути також використаний для виявлення помилок (при цьому він буде здатний виявити більше число помилок, ніж був здатний виправити).

Всі завадостійкі коди можна розділити на два основних класи: блокові і неперервні (рекуррентні або ланцюгові).Як блокові, так і неперервні коди в залежності від методів внесення надмірності розділяються на роздільні і нероздільні. більшість відомих роздільних кодів складають систематичні коди. У цих кодів перевірні символи визначаються в результаті проведення лінійних операцій над певними інформаційними символами. Для випадку двійкових кодів кожний перевірний символ вибирається таким, щоб його сума за модулем два з певними інформаційними символами стала рівною нулю. Декодування зводиться до перевірки на парність певних груп символів. У результаті таких перевірок дається інформація про наявність помилок, а в разі потреби - про позицію символів, де є помилки. Види тестування програмного забезпечення[2].

1.5 Аналіз існуючих рішень

У даному пункті виконано детальний аналіз предметної області, а також проведений аналіз існуючих рішень і їх порівняльна характеристика.

У цьому розділі проведено аналіз існуючих рішень поставленої задачі в даній предметній області. Також був проведений порівняльний аналіз систем розпізнавання номерних знаків.

Аналіз способів порівняння програмного коду:

Дублікат програмного коду – це послідовність програмних інструкцій, присутніх у вихідному коді програми більше одного разу. Довжина такої послідовності має перевищувати задане мінімальне значення. В загальному випадку, дублікатами є будь-які семантичні одиниці програми, що реалізують один алгоритм однаковим способом, але на практиці дублікати визначаються саме за рахунок порівняння послідовностей певних елементів, у вигляді яких представляється вихідний код програми.

Приклад дублікату коду:

На рисунку 1.1 наведена частина коду яка обчислює середня значення масиву цілих чисел.

На рисунок 1.1 – наступна частина коду обчислює середнє значенн я масиву цілих чисел.

```
extern int array1[];
extern int array2[];

int sum1 = 0;
int sum2 = 0;
int average1 = 0;
int average2 = 0;

for (int i = 0; i < 4; i++) {
    sum1 += array1[i];
}

average1 = sum1/4;

for (int i = 0; i < 4; i++) {
    sum2 += array2[i];
}

average2 = sum2/4;
```

Рисунок 1.1 – середнє значенн я масиву цілих чисел.

На риснку 1.2 знаходяться два цикли котрі можуть будь внесені в окрему функцію

```
int calcAverage (int* Array_of_4)
{
    int sum = 0;
    for (int i = 0; i < 4; i++) {
        sum += Array_of_4[i];
    }

    return sum/4;
}
```

Рисунок 1.2два цикли внесені в окрему функцію.

На наступному малюнку ми бачимо використання цієї як звільнює код від дублікатів.

На рисунок 1.3 використання цієї функції звільнить код від дублікатів

```
extern int array1[];
extern int array2[];

int average1 = calcAverage(array1);
int average2 = calcAverage(array2);
```

Рисунок 1.3 Звільнення код від дублікатів

Проблеми до яких призводять дублікати коду:

Дублювання коду є ознакою низького стилю програмування. Гарний стиль програмування звичайно заснований на повторному використанні коду. Може здаватися, що використання дублікатів дозволить дещо прискорити процес створення програми, так як програмісту не потрібно буде думати над тим, як код використовується і як він може використовуватися надалі. Однак проблема полягає в тому, що написання коду це лише невелика частина

життєвого циклу продукту, і подальший супровід коду з дублікатами буде занадто ускладненим. Ось кілька проблем до яких призводить дублювання коду:

Велика кількість коду ускладнює його розуміння: дублювання коду часто призводить до створення довгих, повторюваних послідовностей коду які відрізняються лише кількома рядками або символами

Приховане значення: важко вловити різницю в повторюваних ділянках коду, і тому стає важче розуміти для чого саме призначена та чи інша частина коду. Найчастіше, єдина різниця полягає в параметрах. У цій ситуації найкраще використовувати процедури і функції

Аномалії оновлення: дублювання коду суперечить основному принципу теорії баз даних: «Уникайте надмірності». Невиконання цього принципу призводить до аномалій оновлення, які сильно збільшують витрати на обслуговування коду. У цьому випадку одну і ту ж зміну потрібно ввести в усі дублікати. І в кращому випадку, час витрачений на внесення змін і тестування коду збільшується пропорційно кількості дублікатів. А в гіршому - деякі місця в коді можуть бути пропущені, і виправлення всіх помилок може зайняти місяці або навіть роки. Намагайтеся використовувати бібліотеки коду у такій ситуації.

Розмір файлу - без застосування будь-якого стиснення, файл початкового коду займатиме більше місця на твердому диску.

Порівняння програмного коду – це задача розрахунку індексу подібності для деякої семантичної одиниці програмного коду. Цією одиницею може бути метод або функція, клас або модуль, файл вихідного коду або програма в цілому. Найбільш зручними варіантами є файл (для будь-яких даних у текстових форматах) та метод (для мов об’єктна орієнтованого програмування).

Задача пошуку дублікатів зводиться до пошуку максимальних послідовностей синтаксичних конструкцій у вихідному коді програми, для яких індекс подібності перевищує задане граничне значення.

Існуючі методи, що призначені для пошуку дублювання програмного коду, умовно можна розділити на два класи: текстові та семантичні. Методи, що відносяться до текстових, виконують аналіз на рівні тексту вихідного коду програми та не розрізняють синтаксичних конструкцій мови програмування. Порівняння виконується посимвольно, за елементарну одиницю порівняння приймається текстовий рядок. Методи даного класу знаходять послідовності однакових рядків. Дані методи 1 2 12 використовуються не лише у системах пошуку дублювання коду, а і у системах контролю версій та у програмах порівняння текстових файлів.

Текстові методи мають наступні переваги:

- найбільша можлива швидкість роботи;
- незалежність від мови програмування;
- можливість порівняння будь-яких текстових даних.

За рахунок того, що методи прямого текстового порівняння не передбачають виконання лексичного та синтаксичного розбору вхідних даних, тому їх універсальність та швидкість роботи не може бути досягнута будь-якими іншими методами.

Серед недоліків можна виділити те, що текстові методи можуть використовуватись лише в межах однієї мови програмування. Навіть у межах однієї мови програмування якість розпізнавання дублікатів може бути.

Наявність коментарів, використання різних синтаксичних конструкцій, зміна значень констант та іменування ідентифікаторів, використання відступів та символів табуляції – це фактори, що значно знижують точність розпізнавання дублікатів при використанні текстових методів розпізнавання дублікатів. Можна зробити висновок, що способи пошуку дублікатів програмного коду, що базуються на аналізі текстового 1 3 13 представлення коду, не забезпечують належної якості порівняння у випадку незначних змін вихідного коду, проте мають високу швидкість роботи.

Семантичні методи виконують більш глибокий аналіз вихідного коду програми, що забезпечує кращу точність визначення дублікатів, у порівнянні з текстовими методами, проте їх ефективність значно знижується на великих обсягах коду.

Способи, що базуються на порівнянні вихідного тексту програмного коду, зазвичай виконують порівняння послідовно для кожного рядку коду, і якщо послідовність рядків одного файлу відповідає послідовності рядків іншого файлу, то частини файлів позначаються як підозрілі. Різні алгоритми використовують різні способи числового розрахунку індексу подібності, а деякі інструменти не розраховують його взагалі.

Більш точні способи порівняння потребують більшої кількості кроків, тому вони не можуть гарантувати таку ж швидкість, як способи текстового порівняння. Початкові кроки їх роботи подібні роботі сучасного компілятора [3].

Може виконуватись наступна послідовність кроків:

- лексичний аналіз;
- синтаксичний аналіз;
- побудова графу потоку керування;
- аналіз потоку даних;
- побудова графу залежностей програми;
- порівняння елементів програми у заданому представленні.

Лексичний аналіз – це перший крок, що є необхідним для будь-яких методів пошуку дублікатів, крім текстових, за умови використання мови програмування, лексика якої відома .

Якість не є абсолютною, це суб'єктивне поняття. Тому тестування, як процес своєчасного виявлення помилок та дефектів, не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукту зі специфікацією. При цьому треба розрізняти тестування програмного забезпечення й забезпечення якості програмного

забезпечення, до якого належать всі складові ділового процесу, а не тільки тестування[4].

2 ЗАГАЛЬНА МЕТОДИКА ТА ОСНОВНІ МЕТОДИ ДОСЛІДЖЕНЬ ЩОДО СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПЕРЕВІРКИ ТА ВИПРАВЛЕННЯ РОЗМІТКИ КОДУ ПРОГРАМ

2.1Особливості побудови ІСПТВРКП

Впровадження системи перевірки та виправлення розмітки програмного коду дозволить суттєво зменшити витрати часу на реформат коду, та не витрачати час на ручне правленняю. Система ручного правлення нам дає такий варіант як користувач бере та вдосконалює код самостійно. Цей метод подходить для дуже невеликих проектов. Основная проблема цієї системе - витрата великой кількості часу. Перевагою цієї системи є абсолютно точне правлення, оскільки ПЗ править код в автоматичному режимі, і не дозволяє редагувати окремі куски коду. Доволі довго ця система використовувалась. В часи асемблеру такого не було, але вже тоді це почало зявлятись. Вже тоді програмісти намагалися притримуватися певних структур коду. Чим вище його ріст програмування тим більше програмісти намагалися придумувати різні патерни програмування. Це було зроблено впершу чергу для покращення читання коду, і так як абсолютна більшість проектів пишутся не з нуля, а підтримуются, то ця сиистема є востребованою.Також у багатьох ІТ-компаніях для роботи притримуються певного стилю коду, але робітникиу цих же компаніях програмісти не завжди хочуть дотримуватися певного стилю коду, що не дуже гарно.

Система Beautifier:

Система покращення якості програмного коду Beautifier - програмне забезпечення для покращення програмного коду у відповідності до патернів заданих у налаштуваннях. Застосовується для реформатування коду програм.

Програмне забезпечення (ПЗ) «Beautifier» встановлюється на комп'ютер в якості плагіна.

Система перевіряє на правильність структури коду і якщо все не так, то виконнується функція реформатування.

Така система є більш зручною, так як дозволяє зекономитти багато часу. Можна налаштувати під себе систему, задавши у налаштуваннях свої патерни реформатування.

Функціонал системи дозволяє:

- створювати патерни для ре форматування
- відслідковувати статистику і формувати звіти
- завантажувати сторонні стилі
- зробити автовиправлення після збереження файлу
- автоматичні підказки при невідповідності патернам

Ми будемо розглядати сперва класичний метод, возмем до уваги всі недоліки. Та при розробки свого дипломного проекту їх будемо уникати як можна найбільші. Щоб система автоматично виконувала виправлення та не допускала помилок.

Розглянемо класичний варіант роботи системи перевірки та виправлення розмітки коду програм: встановлюється ПЗ на комп'ютери компанії, для користувача існує інтерфейс який дозволяє одним кліком(обравши зразковий файл) реформувати код. Таким чином програма самостійно обробляє інформацію без втрачання ручного правлення.

Необхідність здійснювати аналіз структур, пошук і виділення структур зразка накладають додаткові вимоги на обчислювальну потужність сервера і, отже, на його вартість. Ці фактори впливають на кінцеву вартість системи.

2.2 Вибір напрямів дослідження та розробки

На основі проведення порівняльного аналізу існуючих систем перевірки та виправлення розмітки коду програм можна виділити наступні задачі дослідження та розробки:

- Аналіз методів та моделей перевірки символьної інформації;

- Розробка структури інтелектуальної системи нейромережевого розпізнавання структур коду;

- Розробка структури використовуваної нейронної мережі;

- Програмна реалізація інтелектуальної системи перевірки та виправлення розмітки коду програм;

Тестування та аналіз результатів.

Покращення, які варто досягнути в результаті розробки системи пропонуються такі:

– перевірка та виправлення розмітки з 100 строк коду протягом 10-100мс;

– розпізнавання кириличних символів у коді;

– точність розпізнавання – 95-98%;

– забезпечення роботи із базою даних структур;

– забезпечення роботи із інтерфейсом користувача оператора;

– точність розпізнавання в умовах поганого стилю кодування.

– Вимоги до розроблюваної системи:

Виходячи з результатів аналізу, проведеного на першому етапі, можна сформувати базовий перелік критеріїв і вимог, яким повинна відповідати розроблювана система[5]

Вимоги до програмної підсистеми:

Розглянемо основні вимоги до розроблюваної системи і її функції:

- система повинна прискорити процес ПЗ;

- система повинна забезпечити надійність отриманих даних;

– система повинна забезпечити точність реформатування;

– система повинна інформувати оператора про критичні значення параметрів за допомогою сигналу;

– система повинна забезпечити наявність центральної СУБД;

– система повинна забезпечити високу швидкість виправлень.

– Вимоги до інтерфейсу системи:

Оператор повинен увійти в систему задля виконання своїх безпосередніх обов’язків.

Система повинна бути стійкою з огляду на частоту її використання.

Мови інтерфейсу: українська, російська, англійська.

2.3 Інструменти для перевірки розмітки коду

Знаходження структур коду у зразковому файлі це завдання з області обробки тексту, яка вирішується різними підходами, але найчастіше за допомогою, так-званих, згортальних нейронних мереж [6]. Нам потрібно знайти у зразковому файлі потрібні структури та запам’ятати їх у пам’яті.

Другий інструмент, який знадобиться – це бібліотека для розпізнавання текстів, яка б могла працювати з різними мовами і яку можна легко налаштовувати під специфіку текстів, які ми будемо розпізнавати. Тут вибір не такий вже й великий, найрозвинутішою є tesseract від Google.

Так само є ряд менш «глобальних» інструментів, за допомогою яких нам потрібно буде нормалізувати область зі структурами зразкового файлу. Зазвичай для таких перетворень використовують Open CV.

Аналогічно потрібно зберегти ці структури до пам’яті. Після цього слід порівняти ці структури зі структурами файлів виправлюваного проекту [7].

2.4 Опис інструментарію

Для виконання розробки ми взяли таку операційною системою як UbuntuGNU/Linux. В цій операційної системі ми будемо і створювати свій проект, але також і інші інструменти будемо використовувати. Також кожна операційна система входить до якогось сімейства. Наша система буде входити в сімейство Unix.

«Linux – це і буде сімейство Unix-також та також схожих на операційних систем які є на базі ядра Linux, які також будуть включають в той чи інший набір Utility program проекту GNU, а також і інші компоненти. Так само як і ядро Linux, але на основі системи як правильно створюються а також поширюються відповідно до моделі та розробки вільного і розкритого програмного забезпечення. частіші поширюються безкоштовно системи Linux.

У вигляді різноманітних дистрибутивів:

- готової для установки та зручною також для супроводу і оновлень, у формі ;

- мають як вільних, так і власницьких свій набір системних та прикладних компонентів» [11].

«Ubuntu –це і є один дистрибутивів Ubuntu і простоту використання орієнтована на зручність. Utility program udo включає в себе широко поширене використання, та дає можливість виконувати користувачам такі завдання як адміністраторські, та небезпечну сесію суперкористувача які не запускаючи потенційно» [12].

Ubuntu, також крім того всього , має розвинену інтернаціоналізацію, яка забезпечує максимальну доступність та для різних представників мовних груп.

Ubuntu для роботи використовувати рекомендується десь від 512 мегабайт RAM і також при установці на жорсткий диск, десь від п'яти гігабайт вільного простору, та гранично мінімальні норми будуть набагато нижче. Як вже було відомо, для розробки в системи використано таку мову програмування Java.

«Java – це є високорівнева мова для програмування яка орієнтована на підвищення та загального призначення, і продуктивності розробника та також читання коду Синтаксис ядра Python мінімалістичний. Так само стандартна бібліотека в себе включає обсяг корисних функцій» [12].

Код в Java влаштувати в класи та функції, які можуть поєднувати у модулі (вони в свою чергу ще можуть бути об'єднані у пакети).

«Ідеальна реалізацією Java це є інтерпретатор Java 8, яка підтримує більшість таких активно застосованих платформ. Також він розповсюджується під вільною ліцензією Oracle, яка допускає застосовувати його без обмежень і також без будь-яких додатках, включаючи пропрієтарні. Для таких як JVM MIL (з можливістю компіляції), і JVM (з можливістю компіляції), LLVM та інших є реалізації інтерпретаторів. Проект JDK пропонує для реалізацію Java з застосуванням JIT-компіляції, яка збільшує швидкість значно при виконання Java-програм.

«Java –мова програмування, яка дуже швидко та активно прогресує, також виходять нові версії (зміною мовних властивостей / додаванням) десь приблизно раза в два роки» [13].

«Оскільки Java — це компільована мова, та математичні алгоритми, які часто працюють набагато швидше ніж в інтерпретованих мовах, таких як python чи навіть php» [13].

У свій магістерської роботі ми будемо використовувати мову програмування Java. На даний час це вважається найбільш зручній та розвинута мова програмування. Котра дуже багато де використовується, але також ще можна розглянути

NumPy можна також аналізувати як гарну та незайняту альтернативу MATLAB, оскільки ця мова є програмування MATLAB зовні вона відображає NumPy: та обидві вони є інтерпретовані, завдяки цьому обидві дозволяють писати дуже швидкі програми користувачам поки інших будуть більшість операцій проводяться над матрицями або масивами, але не над скалярами. В MATLABy є багато переваг великій кількості може бути доступних додаткових тулбоксів, які можна включаючи як пакет imulink.Також є основні пакети, які доповнюють NumPy, це: ciPy - це бібліотека, яка додає більше подібної функціональності MATLAB; Matplotlib - пакет який допомагає створення у графіки в стилі MATLAB. «Внутрішньо

як MATLAB, так і NumPy базується на бібліотеці LAPACK, призначеної для вирішення основних задач лінійної алгебри» [13].

«TensorFlow - для машинного навчання цілій задач відкрита програмна бібліотека, яка була розроблена компанією Google для задоволення її потреб у системах, здатних тренувати і будувати нейронні мережі для розшифровування та виявлення кореляцій і образів , аналогічно до розуміння та навчання, які використовують люди. Її наразі застосовують як для розробки продуктів Google так і для досліджень, часто замінюючи на закритого попередника його ролі, DitBelief. TensorFlow це є початкова розробка команди GoogleBrain для використання внутрішнього в Google, до того часу поки її не випущено було під відкритою ліцензією Apache 20 9 листопада 2015 року»[13]. TensorFlow забезпечує ППІ для Python, а також для C++, Hakell, Java та Go».

Серед використань, для яких TensorFlow вона є основою, також є програмне забезпечення зображень автоматизованого опису, таке як DeepDream. 26 жовтня 2015 року Google офіційно реалізувала RankBrain, який підтримує TenorFlow. RankBrain після цього обробляє суттєве число пошукових записів, замінюючи та на основі результатів пошуку доповнюючи традиційні статичні алгоритми.

2.4.1 Вибір інструментів розробки центрального серверу бази даних

Інформаційна система в’їзного та виїзного контролю автотранспорту складається зокрема із центру обробки, мережі передачі даних, робочих місць системи.

Центр обробки включає в себе центральний сервер бази даних, куди стікаються всі дані, сформовані робочими місцями системи, сервер збору даних і WEB-сервер робочих місць системи. Все це називається процесинговим центром системи.

Для ІСПТВРКП буде розроблений лише центральний сервер бази даних.

Для зберігання і забезпечення цілісності призначених для користувача і інших даних необхідна управління системою базами даних (СУБД). В якості системи управління базами даних необхідно вибрати продукт, який являє собою сукупність програмних і лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням і використанням баз даних.

Найбільш простий підхід буде при використані СУБД який заснований на оцінці того що, існуючі системи якою мірою задовольняють основним вимогам комп'ютерної системи створюваного проекту. Більш складним і дорогим будуть варіантом створення випробувального проекту на основі декількох СУБД та наступний вибір підходящого з кандидатів найбільш. Але також і в цій умові необхідно обмежувати можливих систем коло, критерії відбору спираючись на якісь. Перелік потреб до СУБД, що застосовуються при аналізі тієї або іншої інформаційної системи, від поставлених цілей може змінюватися в залежності. Проте, також можна виділити кілька груп критеріїв:

«Моделювання даних:

- модель даних;
- процедури і тригери;
- кошти пошуку;
- передбачені типи даних;
- реалізація мови запитів» [14].

«Особливості архітектури та функціональні можливості:

- мобільність
- масштабованість;
- розподіленість;
- операційні системи;

- мережеві можливості» [14].

«Вимоги до робочого середовища:

- підтримувані апаратні платформи;
- мінімальні вимоги до обладнання;
- операційні системи;
- підтримувані апаратні платформи» [14].

Таким чином, ми для розроблюваної системи необхідно вибрати СУБД, яка найбільш би підходила висунутим вимогам і була безкоштовною. Нижче наведені деякі з найбільш поширених.

«Microsoft SQL Server – система в управління базами даних (СУБД), розроблена корпорацією Microsoft. Основна використовувана мова запитів – Transact-SQL, створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO по структурованій мові запитів (SQL) з розширеннями. Використовується для невеликих і середніх за розміром баз даних, а за останні 5 років – для великих баз даних масштабу підприємства, конкурує з іншими СУБД в цьому сегменті ринку» [14].

«MySQL – вільна система (СУБД). MySQL є власністю компанії Sun Microsystems, що здійснює розробку і підтримку програми. Поширюється під GNU General Public License та під власною комерційною ліцензією на вибір. Крім цього компанія MySQL AB розробляє функціональність за замовленням ліцензійних користувачів, саме завдяки такому замовленню майже в найранніших версіях з'явився механізм реплікації» [14].

«PostgreSQL – вільна об'єктно-реляційна система управління (СУБД), що працює як клієнт-серверна система. Ґрунтуючись на базових поняттях реляційних БД, PostgreSQL підтримує і ряд "об'єктних" операцій. PostgreSQL відповідає базовій специфікації SQL99 і підтримує велику кількість можливостей, описаних стандартом SQL92» [14].

Для реалізації центрального сервера ІСПТВРКП була обрана СУБД PostgreSQL [14].

2.5 Визначення перспективних нейромережевих архітектур для розпізнавання

«Проведений аналіз у даному стану найромережевих що є доцільність застосування у конкретного типу НМ тому слід визначати що і з умовами прикладної задачі та основі співставлення характеристик мереж. До умов та вказаних характеристик:

- сфера застосування;
- обмеження навчання процесу;
- вимоги обчислення потужностей;
- параметри навчальних даних,;
- реалізації НМ тавимоги до обмеження технічно та вихідної інформації.

Розглянемо таки вказані характеристики які є в ракурсі комп'ютерної системи та розпізнавання голосових сигналів. До навчальних даних відносяться такі основні параметри:

- Навчальний приклад декілька параметрів, які характеризують;
- Вид параметрів, безперервний (числовий) або дискретний (символьний).
- Загальна кількість навчальних прикладів;
- В навчальних прикладах Наявність помилок (шуму);
- Можливість нормалізації та видалення шуму та необхідність попередньої обробки вхідних даних.
- Повнота вибірки, тобто що моделюється на всіх аспектів процесу можливість відображення. Пропорційність навчальних прикладів, які різним аспектам процесу відповідають, що моделюється.

Загальні навчання обумовлюються обмеження процесу:

- Можливістю до навчання в процесі експлуатації.
- Максимальним терміном навчання.

- Необхідністю представлення в навчальних даних очікуваного вихідного сигналу НМ. Цим визначається тип навчання − з вчителем або без вчителя.

- Можливістю автоматизації процесу навчання, яка визначається кількістю та важливістю емпіричних параметрів. Вказана можливість багато в чому визначає умови застосування НМ. Мережі в яких процес навчання не автоматизовано можуть використовуватись тільки в лабораторних умовах.

- Вимогами до якості навчання, яке звичайно оцінюють по величині максимальної та середньої помилки розпізнавання навчальних та тестових даних. При цьому тестові дані повинні не значно відрізнятись від навчальних.

- Можливістю навчання НМ в лабораторних умовах. Доцільність навчання в лабораторних умовах пояснюється потребами оптимального механізму створення та оновлення бази знань НМ» [15].

2.6Алгоритм перевірки розмітки програмного коду

У загальному випадку є системою перевірки та виправлення розмітки коду програм також може бути і програмний комплекс, що виконує алгоритми та перевірки структур коду і також подальшого їх використання до проекту.

Перевірка та виправлення розмітки коду програм базується на наступних процедурах:

- Синтаксичний аналіз
- Локалізація
- Розпізнавання
- Нормалізація
- Сегментація

Такі процедури ми будемо виконувати у дипломному проекті, для досягнення мити.

На рис 2.4 наведена схема роботи системи перевірки та виправлення .

Рисунок 2.4 - Алгоритм перевірки та виправлення розмітки коду програм

Перша процедура призначена для виявлення і локалізації кожної структури з файлу зі структурами даних. Далі знайдені структури вирізаються і зберігаються окремо до БД, для подальшого їх використання. Нормалізація полягає в приведенні розмірів до зменшених та видалення зайвих коментарів. Тут виконується чистка від зайвого шуму. Процедура сегментації забезпечує поділ структур на символи. Процедура перевірки структур проекту зі зразком являє собою порівняння патернів у зразку з проектом. Остання процедура (процедура синтаксичного аналізу)

виконується для визначення елементів рядка, що містить певний патерн. Дані елементи можуть відрізнятися відповідно до стандартів країн реєстрації автотранспортних засобів. В якості вхідних даних будемо використовувати приклади відомих патернів, а саме зразковий файл з цими структурами [16].

2.7 Інтелектуальні системи виправлення розмітки коду програм

Розглянемо такий загальні принципи що до побудови систем і перевірки та також виправлення розмітки коду програм.

Сучасні IT-компанії зазвичай визначають необхідність та побудови централізованої цієї системи, а ступінь розвитку в науки і техніки визначає актуальність та використання при цьому і сучасних і апаратних засобів устаткування, передового та програмного забезпечення, у сфери інформаційних технологій останніх досягнень.

Розглянемо тільки лише деякі з завдань, та вирішення яких на високотехнологічному рівні, тому можливо при цьому використанні інтелектуальних перевірки і виправлення.

Крім функцій і інтелектуального аналізу файлів та проекту система вносити нові структури та паттерни до вже існуючої БД та потім конфігурувати систему для вибору використання відповідних патернів.

Інтелектуальні системи перевірки та виправлення розмітки коду програм - це нове слово на ринку систем чистоти коду, такі системи дозволять приймаючи на роботу новачків не боятись порушення чистоти коду програм, так як будуть автоматично виправлені.

Головна відмінність інтелектуального виправлення розмітки коду програм від традиційних beutifier у тому, що завдання така система є централізованою і дозволяє без окремих налаштувань на ПК програміста реформатувати його код, і продемонструвати йому як все має виглядати.

Основні можливості систем інтелектуального виправлення:

- Аналіз коду. Можна виявити рівень програміста, та як він ставиться до чистоти коду.
- Легування операцій зберігання коду, та просте ведення статистики.
- Розпізнавання нових структур коду. Дозволяє адміну або тім-ліду вносити нові зразкові файли у систему. У процесі нові структури будуть записані до БД.
- Виправлення розмітки коду програм.

Такі системи інтелектуального виправлення ми також використовуємо.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПЕРЕВІРКИ ТА ВИПРАВЛЕННЯ РОЗМІТКИ КОДУ ПРОГРАМ

3.1 Контекстна побудова діаграми

Графічна будова моделі наступає з контекстної діаграми, що відображає системи як єдиного цілого та контекст функціонування модельованої. У прямокутнику в якому записується основне призначення (праці) та модельованої системи. У даному разі опрацьовується у системі розпізнавання і автовиправлення файлів програмного коду.

Взаємодія з робіт із навколишнім світом і між собою також буде записувати у вигляді стрілок. Стрілки виявляють собою якусь інформацію яка також іменниками іменуються. В IDEF0 розрізняють п'ять видів стрілок, а в роботі було використано чотири:

«Вхід (input) – це є інформація і матеріал, яка може використовуються і перетворюються у роботі для отримання результату (виходу).Тому на вхід ми розроблюваної системи використовуємо такі дані як «Множина файлів проектів» та «Зразковий набір даних». Дійсно, «Множина файлів проектів» та «Зразковий набір даних зображено на рис. 1.1 – це щось, таке що переробляється в процесі «Розпізнавання і автовиправлення файлів програмного коду» для одержання результату».

«Керування (Control) – це є правила, та стратегії і стандарти процедури або якими і буде виконується робота. У даному разі ми застосовані такі керування: Завдання, Критерії, Алгоритми та Обробка».

«Вихід (Output) – це є матеріал чи інформація, які у нас виконуються роботою. На виході проектувальної цієї системи маємо бачити «Множина виправлених файлів проекту», яка отримана за допомогою моделі, що може бути у будь-якому зображенню з X поставити у відповідність клас з Y».

«Механізм (Mechanism) – це є ресурси, котрі виконують цю роботу, наприклад це може бути персонал підприємства та верстати і пристрої й т. ін. На рис. 3.1 стрілки програмісти, Обладнання і згорткова нейронна мережа це є сам процес для роботи розпізнавання та автовиправлення програмного коду»[17].

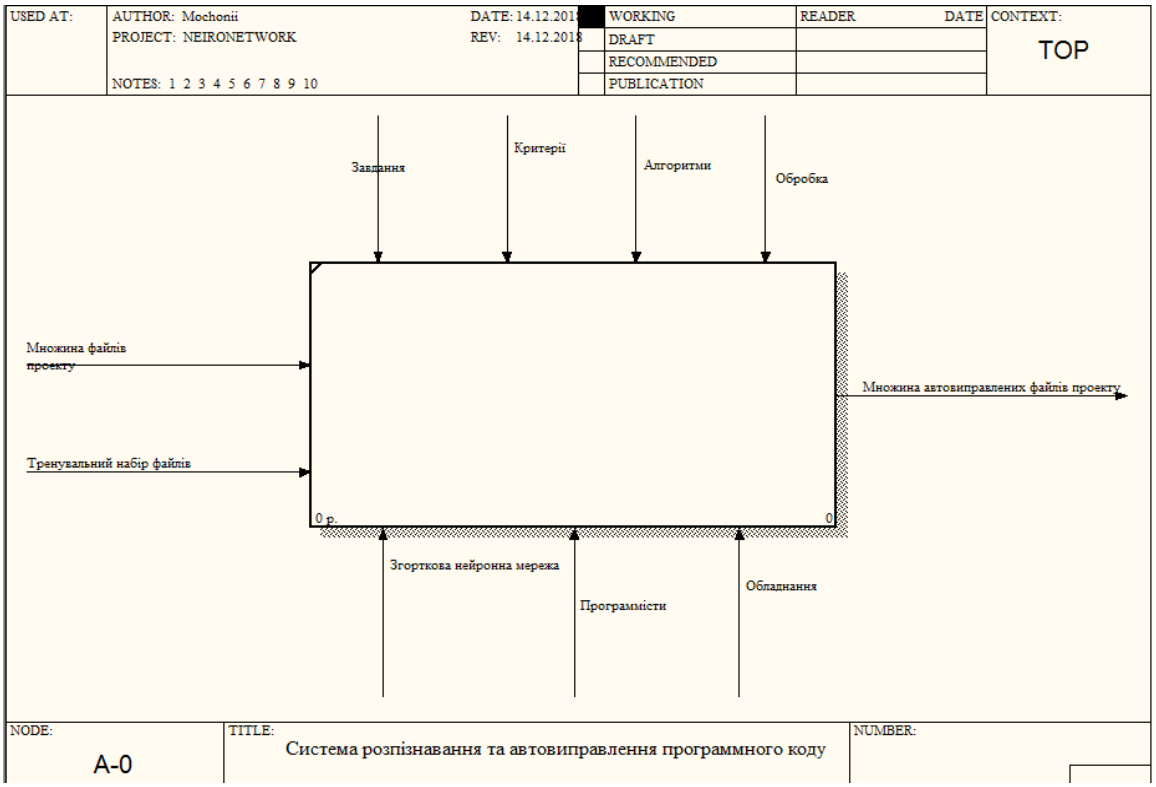


Рисунок 3.1 – Контекстна діаграма

3.2 Побудова діаграми декомпозиції

Системи головної функції є деталізація виконується за допомогою якою декомпозиції діаграм, які будуються так саме як й контекстна, але також включають в себе ще кількість більшу робіт. Кожна також робота, так саме, може бути декомпозирована.

Роботи на даних діаграмах і декомпозиції розташовуються спершу верхнього кута вікна та по діагоналі від лівого ідучи до правого нижнього кута. Такий порядок важаться ладом домінування. У верхньому куті відображаються більш важлива робота і вона зазвичай перша виконується

часом. Після цього розташовуються менш важливі вже вправо та вниз або роботи, що пізніше виконуються. Відповідно на рисунку 3.2 розміщено зліва направо чотири види робіт: «Передобробка файлів», «Обробка файлів», і «Розпізнавання файлів», «Збір ». Поетапний перехід відбувається з умовами, які підписані біля стрілок переходів. Наприклад, при переході від передобробки до обробки файлів відбувається «Нормалізація». Крім того, ми на цю діаграму переносяться входи та виходи, керування та механізми попередньої контекстної, які тісно взаємодіють зі всіма видами робіт. Для зручності стрілки та відповідності керування та механізму позначено різними кольорами, щоб була можливість чітко розпізнавати всі розгалуження [18].

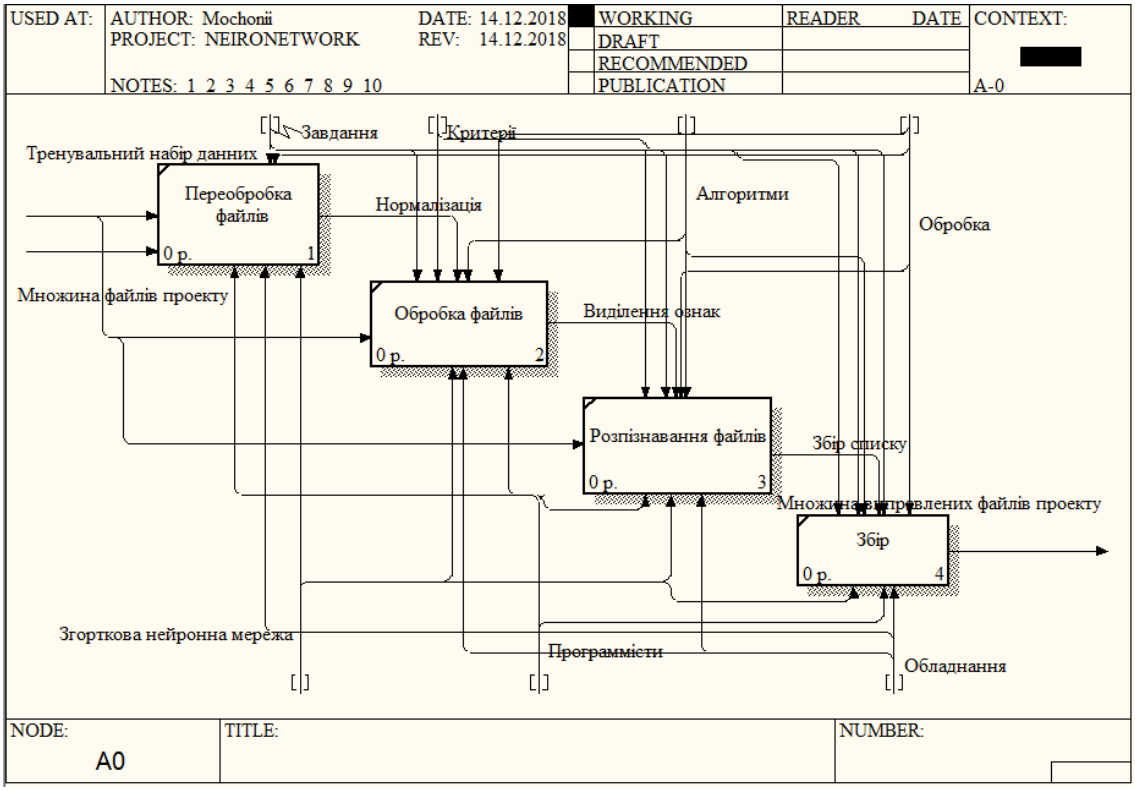


Рисунок 3.2 – Діаграма декомпозиції

3.3Аналіз і документування діаграм в BPwin

Діаграма вузлів дерева демонструє ієрархію в моделі робіт та також дозволяє розібрати всю модель разом, але вона не демонструє цей взаємозалежність між роботами (стрілки).

«У діалозі Node Tree Definition ми вказуємо глибину дерева - Number of Levels (за замовчуванням - 3) а так само корінь дерева (батьківська робота поточної діаграми -за замовчуванням). За замовчуванням нижній рівень декомпозиції демонструється у вигляді переліку, а, також інші роботи у вигляді прямокутників. Для відображення всього цього дерева в вигляді прямокутників було виключено опцію Bullet Last Level» [19].

Розроблювана діаграма містить корінь під назвою «Розпізнавання і автовиправлення файлів», на наступному рівні розміщені такі роботи як «Передобробка файлів», «Обробка файлів», «Розпізнавання файлів», «Збір». Ще на нижчому рівні розташовані роботи, що ієрархічно успадковуються від тогочасного рівня. Так, «Нормалізація файлів» має батьківський вузол дерева «Передобробка файлів» - «Обробка файлів» та інше.

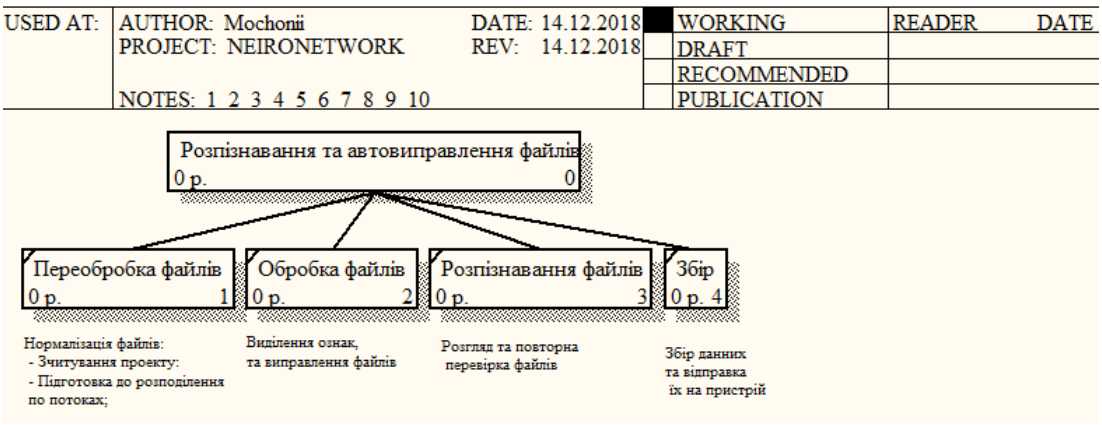


Рисунок 3.3 – Діаграма дерева вузлів

3.4 Побудова FEO діаграми

Діаграми «тільки для експозиції» (FEO) часто вживаються в моделі для демонстрацій інших точок зору, також окремих деталей та відображення, котрий не утримає явно синтаксисом IDEF0. В даному випадку ця діаграма

успадкована від контекстної діаграми та має також подібний до неї вигляд [19].

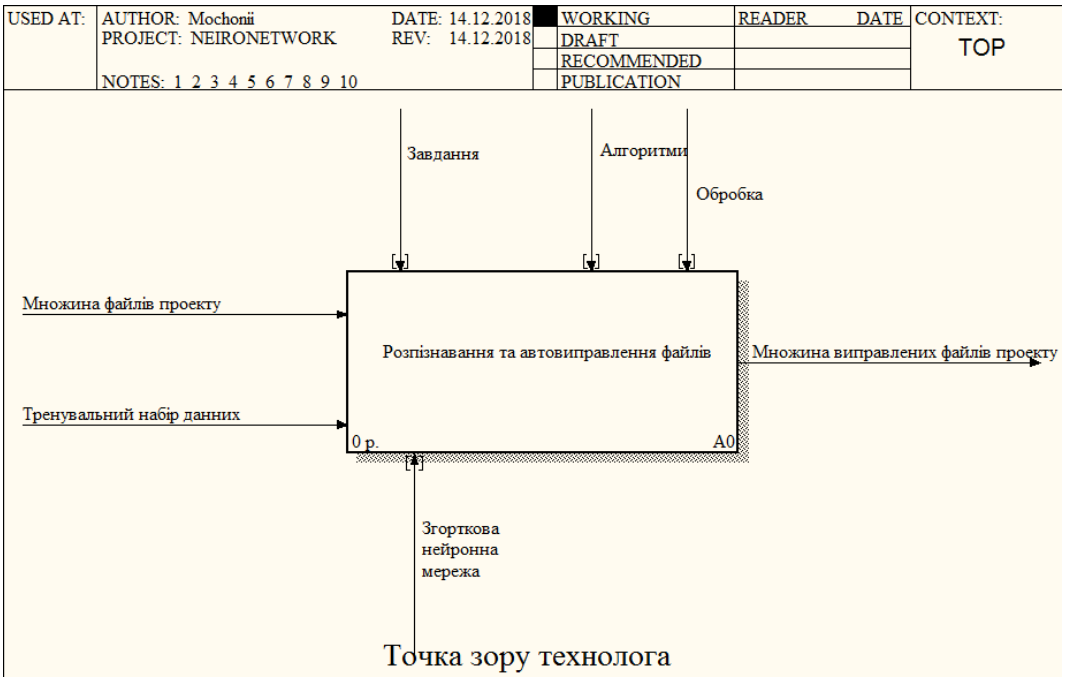


Рисунок 3.4– Діаграма FEO

3.5 Діаграми потоків даних (Data Flow Diagramming)

Діаграми даних потоків (Data flow diagramming - DFD) може застосовуються для відображування документообігу та обробки інформації. Схожий IDEF0, DFD відображає собою як систему модельну мережу взаємозв'язаних між собою робіт. Їх можна також застосовувати як додаток до моделі IDEF0 для більш точного відображення поточних обмін документообігу у корпоративних для інформації системах обробки.

DFD описує:

- обробки інформації та функції;

- документи (стрілки, arrow), співробітників або відділи, які теж так саме беруть участь в обробці та також об'єкти;

- інформації;

- посилання зовнішні (external references), які забезпечують інтерфейс та також відношення із зовнішніми об'єктами, які розташовуються за границями цієї модельованої системи;

- таблиці для збереження документів (сховище даних, data store).

На відміну від IDEF0, де розглядається ця система як взаємозалежні роботи, DFD ще і розглядає сукупність предметів як систему.

Для розгляду та створення такої діаграми було взято саме цю тематику як участі нейронної мережі та в процесах системи розпізнавання, а ще і автовиправлення файлів

Роботи. У DFD які входи на виходи перетворять роботи також являють функції системи собою. На зробленій діаграмі можна також побачити такі роботи: «Згорткова нейронна мережа (опис)» і «Навчання нейронної мережі», «Використання нейронної мережі».

Зовнішні сутності. Зовнішні сенс відображають в входи в таку систему і/або так саме і виходи із системи.

«Сховище даних. Відображують об'єкти в спокої що описують об'єкти в русі на відміну від стрілок. На розробленій діаграмі ми бачимо лише тільки одне сховище даних: «Параметри нейронної мережі».

Стрілки (Потоки даних). Вони описують рух об'єктів з однієї частини системи в іншу. Кожна сторона роботи оскільки в DFD не має чіткого призначення, також як IDEF0, стрілки можуть підходити й виходити з будь-якої грані прямокутника роботи. На даній діаграмі позначені такі внутрішні потоки даних як «Алгоритми», «Критерії», «Завдання», «Обробка» і «Збір файлів», а також вихідні результуючі стрілки «Виправлений файл» та «Зібраний проект».

Як видно із рисунку 3.5 останнім етапом системи збір файлів є забезпечення збору та файлів проекту який піддаєтеся обробці, що і є результатом поставленої задачі [20].

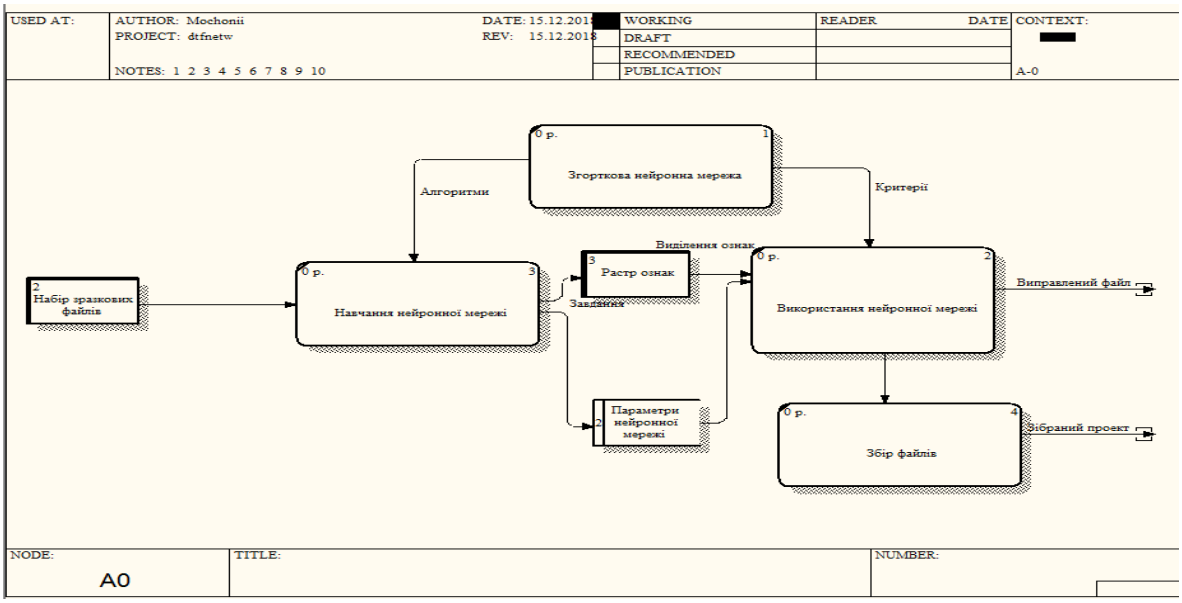


Рисунок 3.5 – Діаграма DFD

3.6 Побудова діаграми IDEF3

«IDEF3 – цей метод коли виконуються в процесі тільки у певній послідовності має таку саму можливість яка буде аналітикам відображати ситуацію, а так саме також описати об'єкти, які беруть участь разом в одному процесі».

Частиною структурного аналізу є техніка опису таких набору даних як IDEF3. На відміну від інших та деяких методик описів та також процесів IDEF3 не обмежує надмірно твердими рамками синтаксису та її тільки аналітика, що може також привести до створення та і неповних або суперечливих моделей. Метод створення процесів може бути IDEF3 та також використаний. Він доповнює IDEF0 та містить все необхідне для структури моделей, які у подальшому можуть бути застосовані для імітаційного аналізу.

Розглянемо ще ми такий процес як для виконання побудови діаграм IDEF3, яка буде включати в себе робота автора (аналітика) та одного чи декількох спеціалістів предметної області.

На рис. 3.6 ми бачимо такий опис процесу "Розпізнавання і автовиправлення файлів проекту" у методології IDEF3 [21].

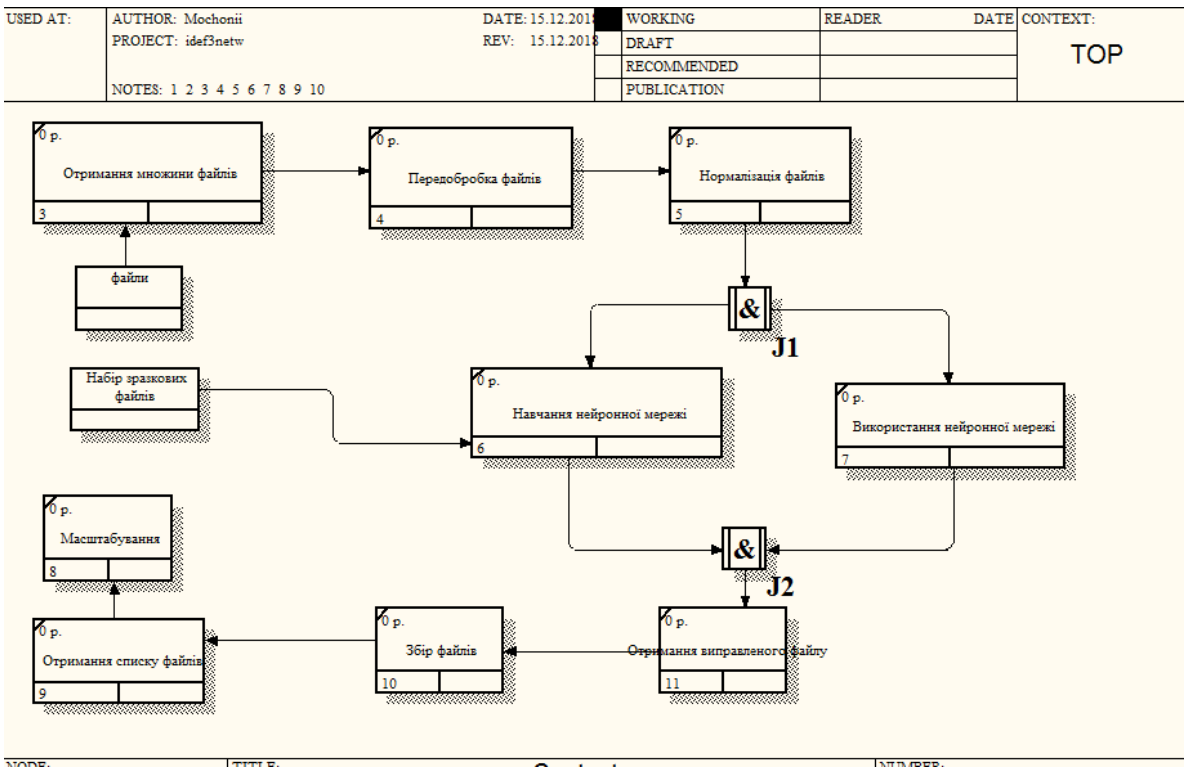


Рисунок 3.6 - Опис процесу "Розпізнавання і автовиправлення файлів" у методології IDEF3

На даній діаграмі розміщені роботи, які описують процес розпізнавання і автовиправлення файлів, а саме «Отримання множини файлів», «Передобробка файлів», «Нормалізація файлів», «Генерування нових даних», «Обробка зображень», «Виділення ознак знаків», «Використання нейронної мережі», «Навчання нейронної мережі», Отримання виправленого файлу», «Отримання списку файлів», «Масштабування (файлів)».

Для логіки взаємодії стрілок відображення злитті й розгалуженні а ще так само або для відображення безлічі подій, які повинні бути або можуть у завершені перед початком наступної роботи, використаються перехрестя (Junction). Розрізняють також перехрестя для злиття (Fan-in Junction) і також розгалуження стрілок (Fan-out Junction). В даній діаграмі використовувалися такі типи перехресть як, Synchronous AND, Asynchronous OR [22].

3.7 Архітектура, структура системи ІСПТВРКП та принцип роботи

Для впровадження проекту ІСПТВРКП було обрано побудування системи та перевірки виправлення розмітки коду програм на базі програмного забезпечння розробленого за данною темою, ПЗ apache та хостинг і звичайний компютер(його потужності вистачить з головою).

На рисунку 3.7 продемонстрована запропонована архітектура розробленої системи ІСПТВРКП. Вона включає зокрема ті складові, що подані в попередніх пунктах. На рисунку зображена архітектура з ухилом на процес перевірки та виправлення розмітки коду програм.

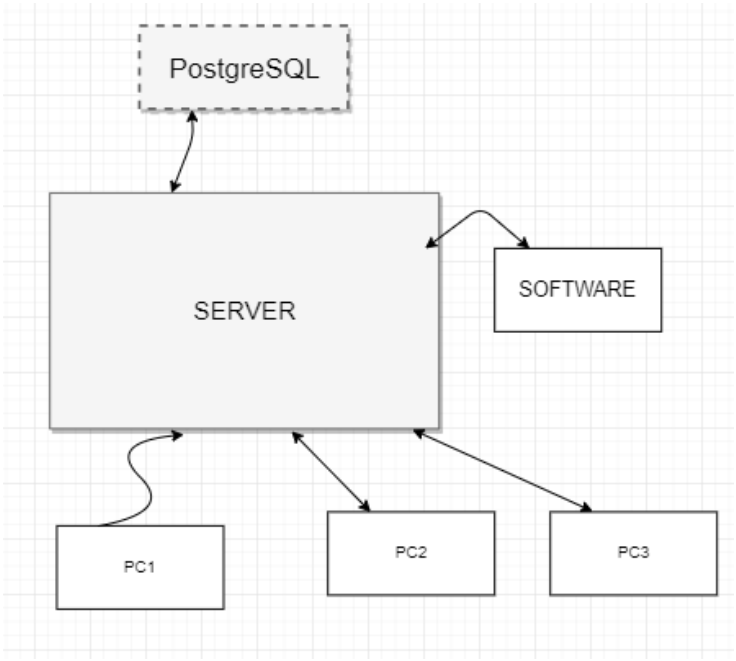


Рисунок 3.7 - Архітектура системи ІСПТВРКП

На даному рисунку показано архітектуру централізованої системи перевірки та виправлення розмітки коду програм. В даному випадку для 3х програмістів. Роботу цієї системи глобально можна пояснити. PosygreSQL використовується для зберігання великої кількості патернів, яке обробляє ПЗ

яке крутиться на сервері. ПЗ - це наша система перевірки та виправлення розмітки коду програм. Сервер приймає дані від комп'ютерів програмістів і відповідає правильно відформатованим кодом програм. Слід зауважити що обмін даними двосторонній. [23].

3.8 Особливості реалізації навчання нейромережової системи перевірки та виправлення розмітки коду програм

Дана підсистема була реалізована у вигляді плагіна для ICE, що може бути застосований за допомогою команди зберігання коду. Інтерфейсу система немає, так як являє собою звичайний бек-енд.

Тренування системи відбувається наступним чином. Після запуску відповідної команди програма автоматично перевіряє у поточній директорії наявність директорії data/ що має містити тестові файли з тренувальним набором даних. Якщо вони є, то почнеться навчання системи. Після старту навчання програма починає виводити дані про поточний крок навчання з інформацію про похибку і затрачений час на певний крок, тощо(рис. 3.8).

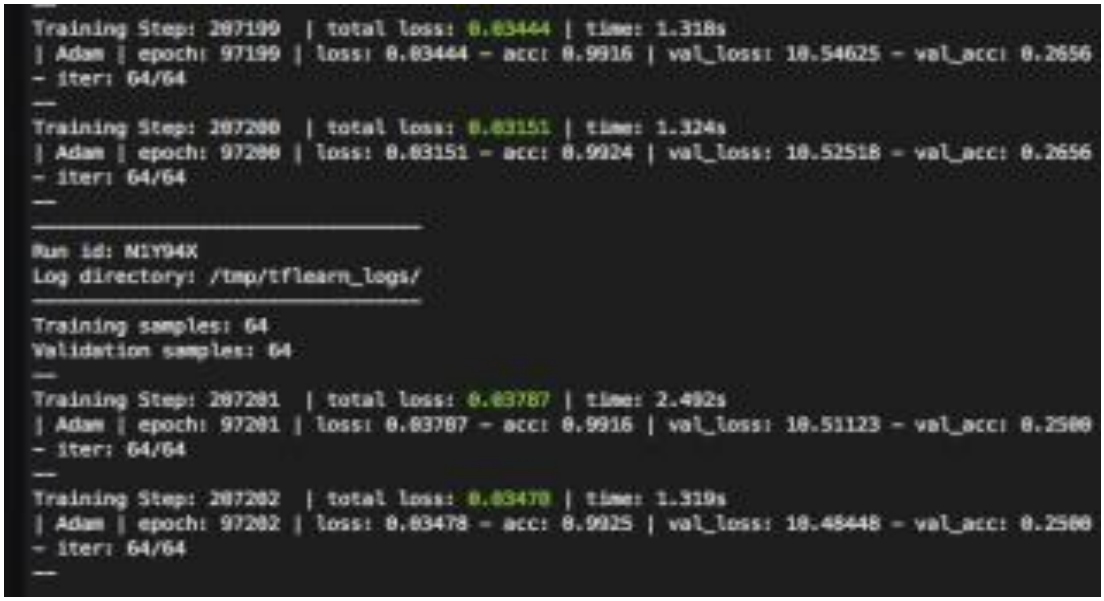


Рисунок 3.8 – Вивід програми про поточний стан навчання нейромережевої системи

Усі дані про навчання автоматично записуються в теку/tmp/rflearn_log. Використовуючи ці дані, за допомогою CLI додатку Tensorboard можна проаналізувати, як працює система (рис. 3.11.).

Комп'ютерну підсистему розроблено на мові java з використанням технології Tensorf.low.

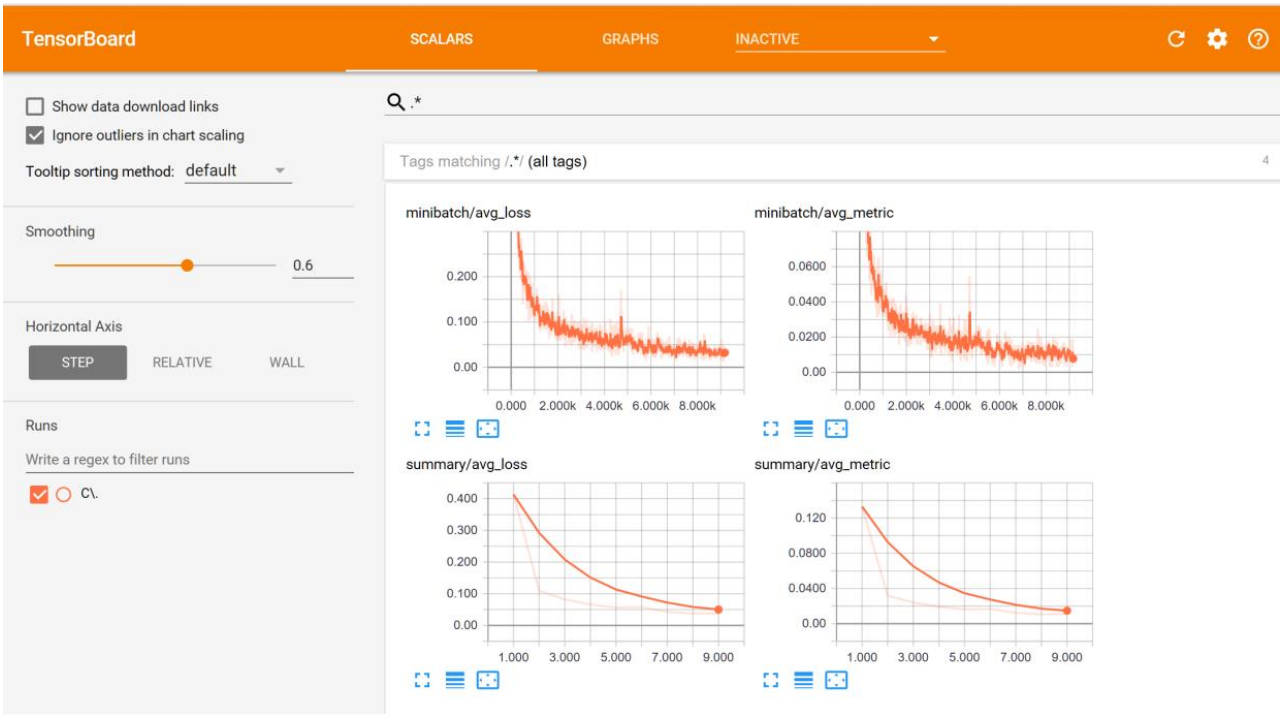


Рисунок 3.9 – Інформація про роботу нейромережевої системи

Тестування підсистеми проходило за допомогою вибраного файлу з тестового набору (рис. 3.10).

```
interface Product { }

class ConcreteProductA implements Product { }

class ConcreteProductB implements Product { }

abstract class Creator {
    public abstract Product factoryMethod();
}

class ConcreteCreatorA extends Creator {
    @Override
    public Product factoryMethod() { return new ConcreteProductA(); }
}

class ConcreteCreatorB extends Creator {
    @Override
    public Product factoryMethod() { return new ConcreteProductB(); }
}

public class FactoryMethodExample {
    public static void main(String[] args) {
        // an array of creators
        Creator[] creators = {new ConcreteCreatorA(), new ConcreteCreatorB()};
        // iterate over creators and create products
        for (Creator creator: creators) {
            Product product = creator.factoryMethod();
            System.out.printf("Created {%s}\n", product.getClass());
        }
    }
}
```

Рисунок 3.10 – Файл з тестовго набору

Після обробки проекту системою отримано наступнийрезультат. Файл проекту був перевірений та виправлений (рис. 3.11).

```
interface Product { }

class ConcreteProductA implements Product { }

class ConcreteProductB implements Product { }

public class FactoryMethodExample {
   public static void main(String[] args) {
      ConcreteProductA a = new ConcreteProductA();
      ConcreteProductB b = new ConcreteProductB();
      System.out.printf("Created {%s}\n", a.getClass());
      System.out.printf("Created {%s}\n", b.getClass());

   }
}
```

```
interface Product { }

class ConcreteProductA implements Product { }

class ConcreteProductB implements Product { }

abstract class Creator {
   public abstract Product factoryMethod();
}

class ConcreteCreatorA extends Creator {
   @Override
```

```
class ConcreteCreatorA extends Creator {
   @Override
   public Product factoryMethod() { return new ConcreteProductA(); }
}

class ConcreteCreatorB extends Creator {
   @Override
   public Product factoryMethod() { return new ConcreteProductB(); }
}

public class FactoryMethodExample {
   public static void main(String[] args) {
      Creator[] creators = {new ConcreteCreatorA(), new ConcreteCreatorB()};
         Product product1 = creators[0].factoryMethod();
         System.out.printf("Created {%s}\n", product1.getClass());
         Product product2 = creators[1].factoryMethod();
         System.out.printf("Created {%s}\n", product2.getClass());

      }
   }
}
```

Рисунок 3.11 – Результат обробки файлу проекту

Особливості реалізації системи ІСПТВРКП.

Комп'ютерна система кодоаналізу (випробувань) ІСПТВРКП призначена для зчитування структур данних та патернів з прикладу і їх автоматичного пошуку по базах даних. Система дозволяє контролювати патерни та структури данних які застосовуються у ІТ-компанії.

Проведення випробувань формує базу даних патернів які мають бути використаними. У базі фіксуються:

«патерн»;

«проект який правится»;

«дата і час перевірки та виправлення»;

«рівень відповідності патернам».

Адмін панель сервера має вигляд як на рисунку 3.12.

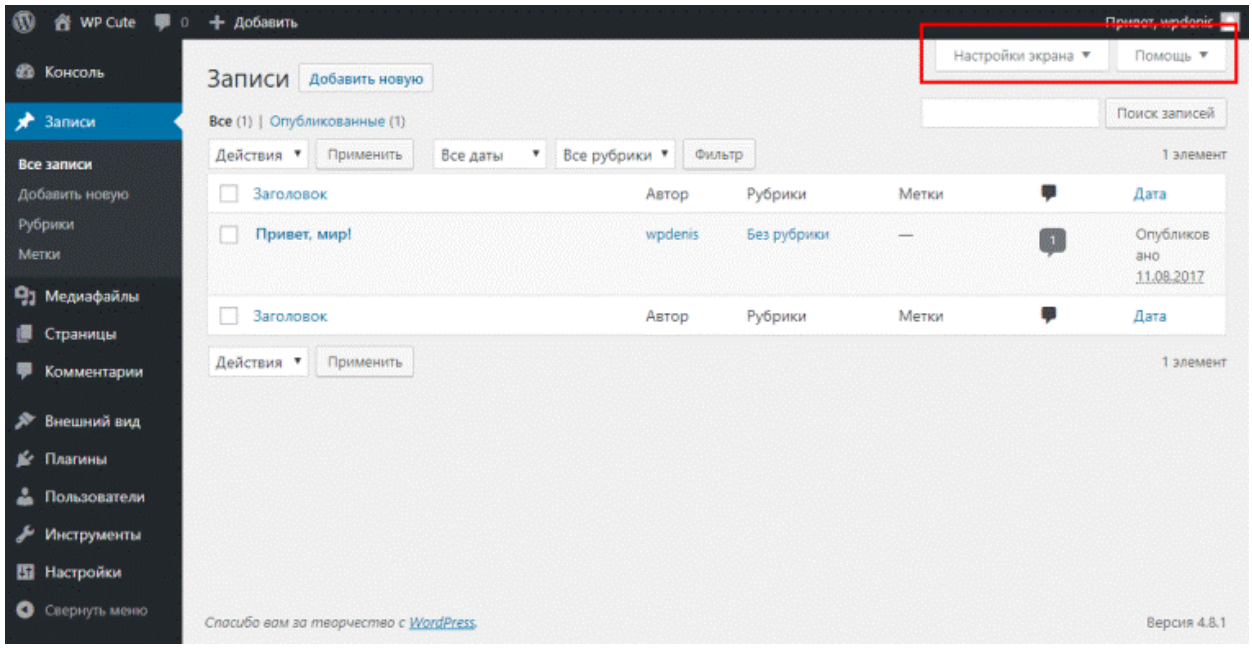


Рисунок 3.12 – Адмін панель серверу

Далі показано журнал перевірок та виправлень, колір показує рівень відповідності (рис. 3.12).

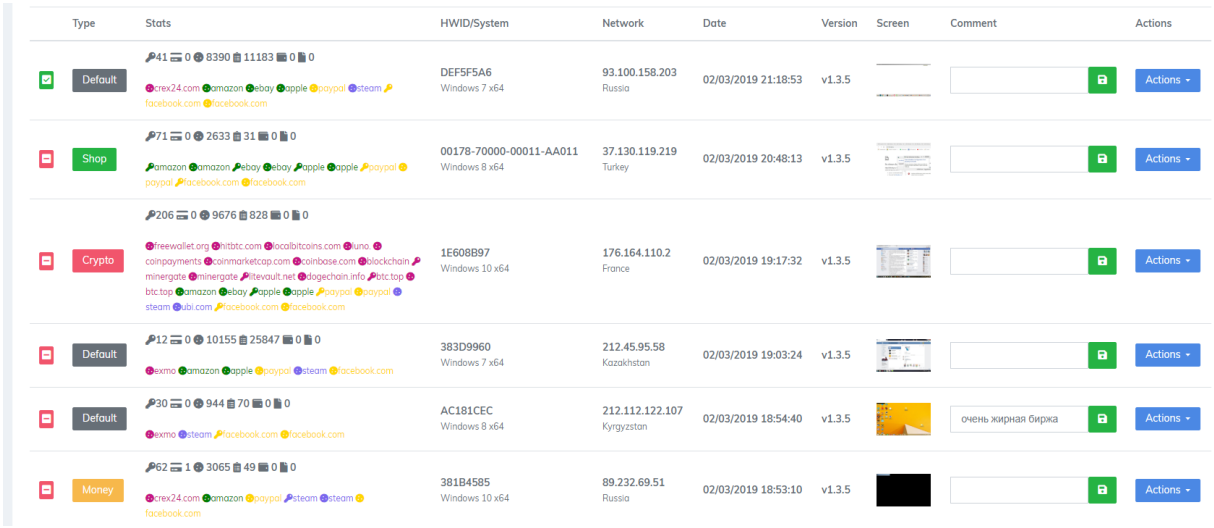


Рисунок 3.13 – Журнал виправлень

Також наша система дозволяє сформувати звіт в текстовому файлі (рис. 3.14).

.

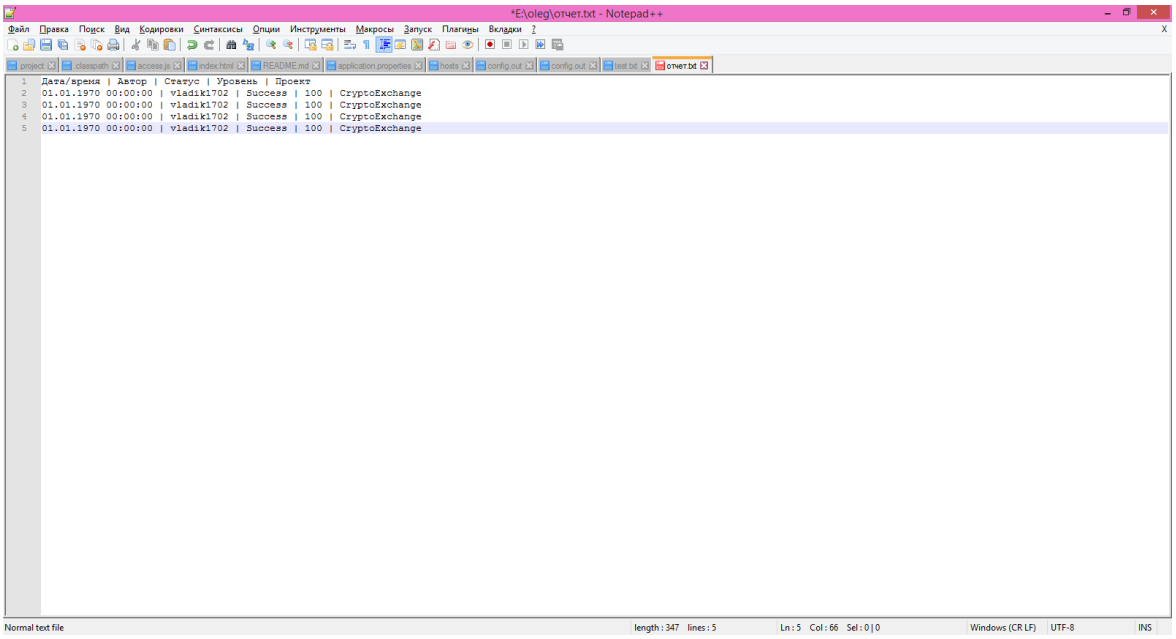


Рисунок 3.14 – Текстовий файл зі звітом

Тестування підсистеми проходило за допомогою вибраного файлу з тестового набору.

4 СПЕЦІАЛЬНА ЧАСТИНА

4.1 Перевірка на валідність коду

Валідність кода – це один з критеріїв, який показує професійність веб-програміста. Всім нам відомо, що HTML досить гнучка мова, ви можете запороти десятки стрічок коду, але при цьому сайт беде нормально працювати [24].

Під час роботи над сайтом розробник має вважати на багато чинників, які впливають на вид документа. Відвідувачі сайту мають не лише різні операційні системи та браузери, але і такі параметри, як кількість кольорів на моніторі, його роздільність, тощо.

Після закінчення верстання слід провести ряд перевірок і у разі виявлення явних помилок, внести в код відповідні зміни. Зрозуміло, це зручніше робити за допомогою спеціалізованих програм.

Відлагодження – це процес знаходження помилок в коді і виправлення небажаного поведінки елементів в браузері. Як правило відхилення макета від первісного дизайну відстежується в процесі верстання, але бувають ситуації, коли помилки необхідно виправити вже на робочому сайті.

Наприклад, помилка може бути виявлена після додавання нового блоку контенту, тестування сайту в різних версіях браузерів, при різних роздільностях монітора і інших умов. Також розробник повинен вміти швидко розуміти чужий код, відстежити причину появи помилки і усунути її. Розуміння логіки чужого коду потрібне при роботі в команді, або при поверненні до власної роботи через якийсь час, коли вона вже сприймається як чужа.

На об'ємних сайтах з десятками тисяч рядків вихідного коду HTML вичленувати проблемне місце досить складно, тому потрібно мати

інструмент, який дозволяє показати код HTML і CSS вибраного фрагмента і провести над ним експерименти.

Більшість існуючих Інтернеті сторінок, на жаль, не є валідними. Багато розробників вважають, абсолютно суворе дотримання стандартів HTML – зовсім не обов'язкова умова того, щоб сайт вийшов якісним і ефективним.

Дійсно, існує багато успішних сайтів, код яких не проходить прискіпливої перевірки валідатором. Немає сумнівів, що можна зробити хороший сайт, не домагаючись повної відповідності коду до формальних правил мови. Однак, розробка валідного коду має ряд переваг як для самого розробника, так і для кінцевого користувача. Тому, варто привчити себе писати грамотний код і перевіряти його валідатором.

1. Перевірка документів валідатором дозволяє уникнути дрібних прикрих помилок – неправильно вкладених тегів, пропущених дужок і лапок тощо.

2. Сучасні браузери підтримують стандарти W3C значно краще, ніж їх попередні версії. Ця тенденція зберігається, тому відповідність DTD набуває все більшої важливості для правильного відображення сторінок в браузерах.

3. Оскільки валідний код відповідає певним формальним правилам, його легше інтерпретувати й обробляти. Він швидше аналізується і відображається в браузері, з ним легше працювати пошуковикам і системам індексації.

4. Валідність коду гарантує сумісність сторінок не лише з існуючими, але і з майбутніми версіями браузерів. Тому, не доведеться переписувати ваші сторінки після виходу нової версії Internet Explorer або Opera [25].

4.2 Основні типи природних і штучних мереж

4.2.1 Рівні тестування

Модульне тестування.

Відноситься до тестів, тестів, які перевіряють функціональність певного розділу коду, зазвичай на функціональному рівні. В об'єктно-орієнованому середовищі, це, як правило, тестування на рівні класу, а мінімальні модульні тести містять у собі конструктори та деструктори.

Такі типи тестів зазвичай пишуться розробниками під час роботи над кодом(стиль «білої скриньки»), щоб впевнитись, що дана функція працює так, як очікувалося. Одна функція може мати кілька тестів, щоб переглянути всі випадки використання коду. Модульне тестування саме по собі не може перевірити функціонування частини ПЗ, а використовується щоб гарантувати, що основні блоки ПЗ працюють незалежно один від одного.

Модульне тестування – це процес розробки ПЗ, що включає в себе синхронізовані застосування широкого спектру для запобігання дефектів та для виявлення стратегій з метою зниження ризиків розробки ПЗ, часу та витрат. Воно виконується розробником ПЗ або інженером, під час будівельної фази життєвого циклу розробки ПЗ. Модульне тестування спрямоване на усунення помилок проектування. Ця стратегія спрямована на підвищення якості одержуваного ПЗ, до такого рівня, як вимагає процес контролю якості.

Залежно від очікуваної організації розробки ПЗ, модульне тестування може включати статичний аналіз коду, аналіз потоку даних аналізу метрик,

експертні оцінки коду, аналізу покриття коду та інші методи перевірки ПЗ.

Інтеграційне тестування.

Інтеграційне тестування є типом тестування ПЗ, яке прагне перевірити інтерфейси між компонентами від програмного дизайну. Програмні компоненти можуть бути інтегровані як в рамках ітеративного підходу, так і всі разом.

Інтеграційне тестування працює над виявленням дефектів у інтерфейса та взаємодії інтегрованих компонентів(модулів). Воно

проводиться до тих пір, поки великі групи тестованих компонентів ПЗ, які відповідають потрібній архітектурі, починають працювати як система.

Рівні інтеграційного тестування:

– компонентний інтеграційний рівень (Component Integration testing);

– перевіряється взаємодія між компонентами системи після проведення компонентного тестування;

– системний інтеграційний рівень (System Integration Testing). Перевіряється взаємодія між різними системами після проведення системного тестування.

Підходи до інтеграційного тестування:

– знизу вгору (Bottom Up Integration). Усі низькорівневі модулі, процедури або функції збираються воєдино і потім тестуються. Після чого збирається наступний рівень модулів для проведення інтеграційного тестування. Даний підхід вважається корисним, якщо всі або практично всі модулі розроблюваного рівня готові. Також даний підхід допомагає визначити за результатами тестування рівень готовності додатків;

– зверху вниз (Top Down Integration). У першу чергу тестуються компоненти верхнього рівня ієрархії об'єктів з використанням заглушок замість компонентів більш низького рівня;

– великий вибух («Big Bang» Integration). Усі або практично усі розроблені модулі збираються разом у вигляді закінченої системи або її основної частини й потім проводиться інтеграційне тестування. Такий підхід дуже хороший для збереження часу. Проте, якщо тест кейси та їхні результати записані неправильно, то сам процес інтеграції дуже ускладниться, що стане перепоною для команди тестування при досягненні основної мети інтеграційного тестування.

4.2.2 Системне тестування

Тестує інтегровану систему для перевірки відповідності всім вимогам. Крім того, системне тестування ПЗ повинно гарантувати, що програма працює так, як очікувалося, а також, що її не можно знищити або пошкодити її робоче середовище, яке викличе процеси в цьому середовищі, що переведуть систему в неробочий стан. Системне інтеграційне тестування перевіряє, чи система інтегрується в будь-яку зовнішню систему (або системи) відповідно до системних вимог.

Альфа-тестування – імітація реальної роботи з системою штатними розробниками або реальна робота з системою потенційними користувачами/замовником. Найчастіше альфа-тестування проводиться на ранній стадії розробки продукту, але у деяких випадках може застосовуватися для закінченого продукту як внутрішнього приймального тестування. Іноді альфа-тестування виконується під відлагоджувачем або з використанням середовища, яке допомагає швидко виявляти знайдені помилки. Виявлені помилки можуть бути передані тестувальникам для додаткового дослідження у середовищі, подібному тому, в якому буде використовуватися програма.

Бета-тестування – у деяких випадках виконується поширення версії обмеженнями (за функціональністю або часом роботи) для певної групи осіб, з тим щоб переконатися, що продукт містить достатньо мало помилок. Іноді бета-тестування виконується для того, щоб отримати зворотній зв'язок про продукт від його майбутніх користувачів.

Часто для вільного-відкритого ПЗ стадія альфа-тестування характеризує функціональне наповнення коду, а бета-тестування – стадію виправлення помилок. При цьому, як правило, на кожному етапі розробки проміжні результати роботи доступні кінцевим користувачам.

4.2.3 Покриття коду

Тестувальники використовують тестові скрипти на різних рівнях: як у модульному, так і в інтеграційному та системному тестуванні. Тестові скрипти, як правило, пишуться для перевірки компонентів, у яких найбільш висока ймовірність появи відмов або вчасно не знайдена помилка може бути дорогою.

Покриття коду, за своєю суттю, є тестуванням методом білого ящика. Тестоване ПЗ збирається зі спеціальними налаштуваннями або бібліотеками або запускається в особливому середовищі, в результаті чого для кожної використовуваної(виконуваної) функції програми визначається місце-знаходження цієї функції у вихідному коді. Цей процес дозволяє розробникам та фахівцям із забезпечення якості визначити частини системи, які, при нормальній роботі, використовуються дуже рідко або ніколи не використовуються (такі як код обробки помилок тощо). Це дозволяє зорієнтувати тестувальників на тестування найбільш важливих режимів.

Як правило, інструменти та бібліотеки, які використовуються для отримання покриття коду, вимагають значних витрат продуктивності та/або пам'яті, неприпустимих при нормальному функціонуванні ПЗ. Тому вони можуть використовуватися тільки в лабораторних умовах.

4.2.4 Приймальне тестування

Формальний процес тестування, який перевіряє відповідність системи вимогам і проводиться з метою: визначення чи задовольняє система приймальним критеріям; винесення рішення замовником або іншою уповноваженою особою приймається додаток чи ні.

Приймальне тестування виконується на основі набору типових тестових випадків та сценаріїв, розроблених на основі вимог до даного додатку. Рішення про проведення приймального тестування приймається тоді, коли: продукт досяг необхідного рівня якості; замовник ознайомлений з Планом приймальних Робіт (Product Acceptance Plan) або іншим документом,

де описаний набір дій, пов'язаних з проведенням приймального тестування, дата проведення, відповідальні тощо.

Фаза приймального тестування триває до тих пір, доки замовник не виносить рішення про відправлення програми на доопрацювання або видачі додатка [26]

5 ОБҐРУНТУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

Метою дипломної роботи є дослідження методів та засобів для ідентифікації користувачів в системах створення моделі на основі штучної нейронної мережі для розпізнавання та автовиправлення файлів.

Головною метою розділу є обґрунтування економічної ефективності впровадженої даної розробки.

Щоб виконати оцінку економічної ефективності необхідно розрахувати трудомісткість реалізації проекту, витрати на оплату праці найманим працівникам, витрати апаратного і програмного забезпечення, амортизаційні відрахування, витрати енергоресурсів та інші витрати, які є основними пунктами виконання обчислень, а також показники економічної ефективності розробки проекту.

5.1 Розрахунок норм часу на виконання науково-дослідної роботи

Реалізація проекту системи розпізнавання і автовиправлення файлів складається із підготовчого етапу, етапу технічної пропозиції, створення технічного завдання, проектування системи, практичної реалізації, тестування, верифікації та заключного етапу.

Норми часу на виконання науково-дослідницької роботи розраховуватимуться на основі часу виконання стадії в годинах, що наведені в таблиці 5.1 разом із інформацією про виконавців і сумарною кількості затраченого часу.

Таблиця 5.1 – Операції технологічного процесу та їх час виконання

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1	Підготовча стадія	Проектний менеджер Інженер-програміст	10
2	Технічна пропозиція	Проектний менеджер Інженер-програміст	10
3	Створення технічного завдання	Проектний менеджер Інженер-програміст	10
4	Проектування системи	Інженер-програміст	10
5	Практична реалізація	Інженер-програміст	45
6	Тестування системи	Тестувальник	10
7	Верифікація системи	Тестувальник Інженер-програміст Проектний менеджер	15
8	Створення документації	Інженер-програміст	5
9	Заключна стадія	Проектний менеджер	5
Разом			120

В підсумку на реалізацію проекту інформаційної системи управління доступом з використанням інформаційних технологій розпізнавання образів необхідно 120 людино-годин, залучення трьох спеціалістів та виконання дев’яти різноманітних стадій реалізації проекту.

5.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

Визначення витрат на оплату праці та відрахувань на соціальні заходи прямо залежить від кількості витраченого працівниками часу на роботу, ставки в годину чи місяць, кількість відрахувань на соціальні заходи встановлених в законному порядку на час розрахунку.

Додаткова заробітна плата – це складова заробітної плати працівників, до якої включають витрати на оплату праці, не пов’язані з виплатами за фактично відпрацьований час.

При розрахунку заробітної плати кількість робочих днів у місяці слід в середньому приймати – 24,5 дні/міс., або ж 196 год./міс. (тривалість робочого дня – 8 год.).

Наймані працівники для розробки системи розпізнавання і автовиправлення файлів працюють згідно контракту, який в якому вказано їхню погодинну ставку.

У штаті найманих працівників для розробки інформаційної системи залучено проектного менеджера, інженера-програміста і тестувальника.

Тарифні ставки учасників процесу розробки системи перевірки та виправлення коду программ:

- Проектний менеджер – 160 грн./год.
- Інженер-програміст – 150 грн./год.
- Тестувальник – 130 грн./год.

Основна заробітна плата розраховується за формулою 5.1:

$$З_{осн.} = Т_с \cdot К_г \, , \qquad (5.1)$$

де $Т_с$ – тарифна ставка, грн.; $К_г$ – кількість відпрацьованих годин.

Оскільки всі види робіт в виконує три спеціаліста, то основна заробітна плата буде розраховуватись за даною формулою 5.1;

$$З_{осн.} = 160 \cdot 25 + 150 \cdot 80 + 130 \cdot 15 = 17950 \text{ грн.}$$

Додаткова заробітна плата становить 10–15 %% від суми основної заробітної плати й визначається за формулою 5.2.

Коефіцієнт додаткових виплат працівникам становить 0,1.

$$З_{дод.} = З_{осн.} \cdot К_{допл.} \qquad (5.2)$$

де $К_{допл}$ – коефіцієнт додаткових виплат працівникам

$З_{дод} = 17950 \cdot 0,1 = 1795$ грн.

Звідси загальні витрати на оплату праці (фонд заробітної плати) визначаються за формулою 5.3:

$$В_{о.п.} = З_{осн.} + З_{дод.} \quad (5.3)$$

$$В_{о.п.}=17950 + 1795 = 19745 \text{ грн.}$$

З цієї суми утримуються обов'язкові відрахування на заробітну плату:

– Єдиний соціальний внесок (ЄСВ), що становить 22%;

– Військовий збір (ВЗ), що становить 1,5%;

Сума відрахувань становить 23,5% від фонду оплати праці та визначається за формулою 5.4:

$$B_{с.з.} = Ф_{оп} \cdot 0,235 \quad (5.4)$$

де $Ф_{оп}$ – фонд оплати праці, грн.

$B_{с.з.} = 19745 \cdot 0,235 = 4640,075$

Усі витрати обчислюються детально наведені в таблиці 5.2 та обчислюються за формулою 5.5:

$$В_{зп} = ФЗП + ФОП \quad (5.5)$$

$$В_{зп} = 17950 + 4640,075= 22590,075 \text{ грн.}$$

Таблиця 5.2 – Розрахунки витрат на оплату праці

№ з/п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	Нарахув. на ФОП, грн.	Всього витрати на плату праці, грн. (6=3+4+5)
		Тарифна ставка, грн.	Кількість відпрацьованих год.	Фактично нарах. з/пл., грн.			
А	Б	1	2	3	4	5	6
1.	Проектний менеджер	160	25	4000	400	-	-
2.	Інженер-програміст	150	80	12000	1200	-	-
3.	Тестувальник	130	15	1950	195	-	-
Разом		-	120	17950	1795	4640,075	22590,075

Згідно проведених розрахунків витрата на оплату праці становлять 22590,075грн

5.3 Розрахунок матеріальних витрат

Матеріальні витрати є невід'ємною частиною перевірки та виправлення розмітки коду програм та визначаються як добуток кількості витрачених матеріалів та їх ціни за формулою 5.6:

$$M_{ві} = q_i \cdot p_i \,, \tag{5.6}$$

де: q_i – кількість витраченого матеріалу і–го виду; p_i – ціна матеріалу і–го виду.

Звідси, загальні матеріальні витрати можна визначити за формулою

$$З_{м.в.} = \sum M_{ві} \,. \tag{5.7}$$

Результати проведених розрахунків наведено у таблиці 5.3.

Таблиця 5.3 – Результати розрахунків матеріальних витрат.

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Фактично витрачено матеріалів	Ціна одиниці, грн.	Загальна сума витрат, грн.
1	INTERNET	шт.	1	180	180
2	Папір для друку	листів	500	0,15	75,00
3	Чорнила для принтера	шт.	1	80,00	80,00
Всього					335

Згідно проведених розрахунків, матеріальні витрати становлять 335 грн.

5.4 Розрахунок витрат на електроенергію

Однією із статей витрат є витрати на електроенергію під час проходження усіх етапів реалізації кінцевого продукту.

Затрати на електроенергію одиниці обладнання визначаються за формулою 5.8:

$$З_в = W \cdot T \cdot S, \tag{5.8}$$

де W – необхідна потужність, кВт; T – кількість годин на реалізацію розробки; S – вартість кіловат-години електроенергії.

Вартість кіловат-години електроенергії слід приймати згідно існуючих на даний час тарифів. Отже, 1 кВт з ПДВ коштує 2,42 грн.

Потужність комп’ютерів для реалізації кінцевого продукту – 400 Вт, кількість годин роботи обладнання згідно таблиці 5.1 – 265 годин.

Визначимо витрати на електроенергію згідно формули 5.11:

$$З_в = 0{,}4 \cdot 120 \cdot 2{,}42 = 116.16 \text{грн.}$$

Згідно формули затрати на електроенергію становлять 116.16 грн.

5.5 Розрахунок суми амортизаційних відрахувань

Для будь якої діяльності характерною є властивість зношування на зниження якості властивостей інструментарію та фондів за допомогою яких ведеться діяльність.

Для вирішення проблеми із відновленням даних фондів використовується амортизація, що являє собою процес трансформації вартості основних фондів на вартість продукції, яка щойно була створена, задля повного відновлення основних фондів.

Для визначення амортизаційних відрахувань використовується формула 5.9:

$$А = \frac{Б_в \cdot Н_а}{100\ \%} \qquad (5.9)$$

де – балансова вартість обладнання, грн;

– норма амортизаційних відрахувань в рік, %;

– річний робочий фонд часу, год;

– фактичний час роботи обладнання по написанню програми, год.

Комп’ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Річний робочий фонд становитиме 2352 годин, так як робочий день становить 8 годин, а кількість робочих днів в місяці становить 24,5 годин.

Для даної розробки засобом розробки є комп’ютер. Його сума становить 23500 грн. Отже, амортизаційні відрахування будуть рівні:

$$A = 23500 \cdot 5\% / 100\% = 1175 \text{ грн.}$$

Згідно проведених обчислень амортизаційні відрахування становлять 1175 грн.

5.6 Обчислення накладних витрат

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління спілкою та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$H_в = B_{о.п.} \cdot 0{,}2 \ldots 0{,}6 \,, \qquad (5.10)$$

де $H_в$ – накладні витрати.

Отже, накладні витрати становлять згідно формули 5.10:

$$H_в = 17950 \cdot 0{,}2 = 3590 \text{ грн.}$$

Накладні витрати згідно розрахунку формули, становить 3590 грн.

Таблиця 5.4 – Кошторис витрат на НДР

Зміст витрат	**Сума, грн.**	**В%дозагальної суми**
Витрати на оплату праці	22590	70,8
Відрахуваннянасоціальні заходи	4640	14,6
Матеріальні витрати	335	0,3
Витрати на електроенергію	256,52	0,4
Амортизаційні відрахування	1175	1,5
Накладні витрати	3590	12,4
Собівартість	31246,42	100

Собівартість ($С_в$) програмного продукту розрахуємо за формулою:

$$С_в = В_{о.п.} + В_{с.з.} + З_{м.в.} + З_в + А + Н_в \, . \quad (5.11)$$

Отже, собівартість програмного продукту дорівнює:

Св 22590 + 4640 + 335 + 256,52 + 1175 + 3590 = 31411,52 грн.

Загальний кошторис витрат та визначення собівартості тема становить 31411,52 грн.

5.7 Розрахунок ціни програмного продукту

Ціну науково-дослідної роботи можна визначити можна визначити за формулою:

$$Ц = \frac{C_в \cdot (1 + Р_{рен}) + К \cdot В_{н.і}}{К} \cdot (1 + ПДВ) \quad (5.12)$$

де $Р_{рен.}$ – рівень рентабельності, 30 %; $К$ – кількість замовлень, од. (встановлюється лише при розробці програмного продукту та мікропроцесорних систем); $В_{н.і.}$ – вартість носія інформації, грн. (встановлюється лише при розробці програмного продукту); *ПДВ* – ставка податку на додану вартість, (20 %%).

Оскільки розробка є прикладною, і використовуватиметься тільки для одного підприємства, то для розрахунку ціни не потрібно вказувати коефіцієнти К та Ві.н , оскільки їх в даному випадку не потрібно.

Тоді, формула для обчислення ціни розробки буде мати вигляд:

$$Ц = C_в \cdot (1 + Р_{рен}) (1 + ПДВ) \quad (5.13)$$

Звідси ціна на роботу складе:

$Ц$ = 31411,52 · (1 + 0,3) · (1 + 0,2) = 49001,16 грн.

Загальний розрахунок ціни програмного продукту становить 49001,16 грн.

5.8 Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність ($E_р$) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_р = \frac{П}{C_в} \tag{5.14}$$

де $П$ – прибуток; $C_в$ – собівартість.

Плановий прибуток ($П_{пл}$) знаходимо за формулою:

$$П_{пл} = Ц - C_в . \tag{5.15}$$

Розраховуємо плановий прибуток:

$$П_{пл} = 49001{,}16 - 31411{,}52 = 17589{,}64 \text{грн.}$$

Отже, формула для визначення економічної ефективності набуде вигляду:

$$E_р = \frac{П}{C_в} \tag{5.16}$$

Тоді

$$E_р = 17332{,}89 / 31411{,}52\ 31411{,}52 = 0{,}55.$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($T_р$):

$$T_р = \frac{1}{E_р} \tag{5.17}$$

Термін окупності дорівнює:

$$T_p = 1 / 0,5 = 1,78 \text{ р.}$$

Згідно формул плановий прибуток від розробки становить 17497,99 грн., економічна ефективність дорівнює 0,55, а термін окупності становить 1,78 роки що вважається доцільним та економічно вигідним.

5.9 Висновки до п’ятого розділу

В розділі дипломної роботи освітнього рівня «магістр» було розраховано основі техніко-економічні показники побудови геоінформаційної карти (див. таблиця 5.5).

Орієнтоване значення економічної ефективності становить 0,55 що є достатньо високим значенням.

Період окупності повинен варіюватися від 1 до 3 років, тоді розвиток вважається доцільним та економічно вигідним. Термін окупності створення моделі на основі штучної нейронної мережі для розпізнавання та автовиправлення файлів даної роботи становить 1,78 років

Таблиця 5.5 – Техніко-економічні показники науково-дослідної роботи

№ п/п	Показник	Значення
1.	Собівартість, грн.	31411,52
2.	Плановий прибуток, грн.	17589,64
3.	Ціна, грн.	49001,16
4.	Економічна ефективність	0,55
5.	Термін окупності, рік	1,78

На основі проведених розрахунків можна зробити висновок, що створення моделі на основі штучної нейронної мережі для розпізнавання та автовиправлення файлів є доцільним у зв’язку з невеликим терміном окупності та великим обсягом планового прибутку.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

6.1.Охорона праці

Охорона праці – це система правових,соціально-економічних, організаційно-технічних,санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, спрямованих на збереження здоров'я та працездатності людини в процесі трудової діяльності

6.1.1 Охорона праці жінок

Охорона праці жінок – важливий інститут трудового права. Рівні права і можливості в одержанні освіти загальної і соціальної підготовки призвели дотого, що жіноча праця за своєю кваліфікацією зрівнялася з чоловічою.

Поширена теорія гармонійного поєднання виробничої праці з материнством є оманливою, оскільки нею приховується експлуатація жінки. Громадська думка не схильна вважати, що жінка взагалі не повинна працювати в суспільному виробництві. Але на сьогоднішній день серед виробничих спеціальностей є багато професій, які традиційно вважаються жіночими. Склались галузі народного господарства, де жінки займають чільні місця. До таких можна віднести торгівлю і громадське харчування, охорона здоров'я і соціальний захист населення, народна освіта та багато ішого. В останні роки створена законодавча база для забезпечення гендерної рівності, в тому числі в трудових відносинах.Але, враховуючи фізіологічні особливості жіночого організму, законодавство про працю передбачає низку обмежень у виконанні певних робіт, а при деяких роботах встановлює жінкам переваги.

При виборі трудової діяльності робота жінки повинна бути достойною, добре оплачуваною – це одні з основних вимог якими повинна керуватись жінка при виборі роботи. Вимушене суміщення сімейних обов'язків з роботаю на виробництві не проходять безслідно. Найвідчутнішим негативним наслідком фактичного подвійного навантаження жінки є хронічна перевтома, що призводить до зниження продуктивності її праці на виробництві.

Рівність прав жінки і чоловіка забезпечується: -наданням жінкам рівних з чоловіками можливостей у громадсько політичній і культурній діяльності, у здобутті освіти і професійній підготовці, у праці та винагороди до неї;

- спеціальними заходами щодо охорони праці і здоров'я встановлення пенсійних пільг;

-створення умов, які дають жінкам можливість поєднувати працю з материнством;

-правовим захистом, матеріальною та моральною підтримкою материнства і дитинства ,включаючи надання оплачуваних відпусток та інших пільг вагітним жінкам і матерям.

Нормативною базою для охорони праці жінок виступає КЗпП України, яких містить правові приписи , що диференційовано регулюють трудові відносини з цього приводу. В узагальненому вигляді ці норми можна умовно поділити на 2 групи, що виражають законодавче закріплення охорони праці жінок.Перша група норм регламентує питання з приводу робіт, на яких забороняється праця жінок,а друга група – з приводу забезпечення права жінкам на повноцінне виховання дітей.

Держава намагається не допустити застосування праці жінок на важких роботах і на роботах із шкідливими і небезпечними умовами праці. Не дозволяється використання праці жінок на підземних роботах і гірничо-добувній промисловості та на будівництві підземних споруд, за винятком жінок, які займають керівні посади і не виконують фізичної роботи [27].

6.1.2 Пільги та гарантії для вагітних жінок та жінок, які мають дітей

На підставі медичного висновку жінкам надається оплачувана відпустка у зв'язку з вагітністю та пологами тривалістю 70 календарних днів до пологів і 56 (у разі народження двох і більше дітей та у разі ускладнення пологів - 70) календарних днів після пологів, починаючи з дня пологів. Тривалість відпустки у зв'язку з вагітністю та пологами обчислюється сумарно і становить 126 календарних днів (140 календарних днів - у разі народження двох і більше дітей та у разі ускладнення пологів). Вона надається жінкам повністю незалежно від кількості днів, фактично використаних до пологів. За бажанням жінки їй надається відпустка для догляду за дитиною до досягнення нею трирічного віку з виплатою за ці періоди допомоги відповідно до законодавства. Підприємства, установи та організації за рахунок власних коштів можуть надавати жінкам частково оплачувану відпустку та відпустку без збереження заробітної плати для догляду за дитиною більшої тривалості. Відпустка для догляду за дитиною до досягнення нею віку трьох років не надається, якщо дитина перебуває на державному утриманні, крім прийомних дітей у прийомних сім’ях та дітей-вихованців у дитячих будинках сімейного типу. У разі, якщо дитина потребує домашнього догляду, жінці в обов'язковому порядку надається відпустка без збереження заробітної плати тривалістю, визначеною у медичному висновку, але не більш як до досягнення дитиною шестирічного віку. Відпустки для догляду за дитиною, передбачені частинами третьою, четвертою та шостою цієї статті, можуть бути використані повністю або частинами також батьком дитини, бабою, дідом чи іншими родичами, які фактично доглядають за дитиною. За бажанням жінки або осіб, зазначених у частині сьомій цієї статті, у період перебування їх у відпустці для догляду за

дитиною вони можуть працювати на умовах неповного робочого часу або вдома.

За бажанням жінки або інших осіб, які фактично доглядають за дитиною, у період перебування їх у відпустці по догляду за дитиною вони можуть працювати на умовах неповного робочого часу або вдома. При цьому за ними зберігається право на одержання допомоги в 7 період відпустки для догляду за дитиною до досягнення нею трирічного віку.

У разі надання жінкам відпустки у зв'язку з вагітністю та пологами власник або уповноважений ним орган зобов'язаний за заявою жінки приєднати до неї щорічні основну і додаткову відпустки незалежно від тривалості її роботи на даному підприємстві, в установі, організації в поточному робочому році.

Надання додаткової оплачуваної відпустки жінці, яка працює і має двох або більше дітей віком до 15 років, або дитину з інвалідністю, або яка усиновила дитину, одинокій матері, батьку, який виховує дитину без матері (у тому числі й у разі тривалого перебування матері в лікувальному закладі), а також особі, яка взяла дитину під опіку, чи одному із прийомних батьків[28].

Таким чином, охорона праці жінок являється важливим інститутом трудового права, котрий диференціює цілий комплекс правових норм у регулюванні взаємовідносин працівників – жінок з роботодавцем з додержанням специфіки жіночої праці. Слід зауважити, що охорона праці жінок вимагає у сучасних умовах вдосконалення як нормативного, так і організаційного. Мова йде про доречність прийняття окремого закону щодо охорони праці жінок чи, принаймні, виділення норми щодо такої охорони в окремий розділ. Наявність такого механізму захисту законних прав та інтересів жінок дозволяє у повній мірі реалізувати державні програми у сфері розвитку праці жінок, яка тісно пов’язується з веденням домашнього господарства, вихованням дітей. Рівень захисту жінок у тому чи іншому

суспільстві свідчить про зрілість держави, розвиненість її державно-правових інститутів та рівень суспільно-правової свідомості.

6.2 Методи підвищення продуктивності праці працівників галузі ІТ

6.2.1 Значення охорони праці у роботі фахівців з інформаційних технологій

Комп'ютери вже давно увійшли в повсякденне життя, а норми роботи з ними вже давно вкоренилися в зарубіжних ІТ-компаніях, проте на території України цей процес все ще триває, хоч і підходить до завершальної стадії. Саме тому розгляд даного питання є актуальним для такого міста, як Маріуполь, так як у столиці та деяких великих обласних центрах це питання практично вирішене.

На перший погляд, робота за комп'ютером здається безпечною, але саме легковажність до неї може призвести до певних проблем у здоров'ї людини. Професія програміста та інших фахівців ІТ-технологій пов'язана з колосальним розумовим напруженням. Розробники – настільки захоплені люди, що навіть відволікаючись від роботи над проектом, продовжують думати про роботу. Нерідко відпочинком вони вважають паралельну заміну основної діяльності, наприклад, читання профільної літератури, верстку сайтів, вивчення нових мов програмування. Однак мозок не може до безкінечності приймати виключно корисну інформацію, яку розробник прагне направляти в русло особистісного та професійного зростання.

Адже мозок людини не машина: він не може нескінченно зберігати і переробляти дані практично не втрачаючи продуктивності [29].

Тому зараз багато ІТ-компаній обладнують свої офіси кімнатами відпочинку та лаунж-зонами, які забезпечують психофізіологічне

розвантаження працівників. Адже окремим робочим столом з ноутбуком вже давно нікого не здивуєш. Тому, бажаючи підвищити продуктивність працівників, міжнародні компанії змагаються, перетворюючи нудні одноманітні офіси в креативні простори, де нові ідеї народжуються без титанічних зусиль. Наприклад, винахідники компанії Inventionland трудяться в казкових декораціях. Тут і гігантський гоночний трек, і пряниковий будиночок, і дуже реалістичний піратський корабель, на палубі якого розташувалися комп'ютерні столи, ніжками яких служать винні бочки. Робочі місця співробітників компанії Google в Цюріху нагадують гігантські вулики, а офіс шведського інтернет-провайдера Bahnhof розташувався в бомбосховищі часів холодної війни і походить на підземний притулок землян після глобальної катастрофи. А щоб співробітників не тягнуло додому, роботодавці створюють і можливість релаксувати, не відходячи від робочого місця, обладнавши басейни, ігрові кімнати та спортзали [30].

Адже можна цілий день просидіти біля монітора, а у вечорі відчути страшну втому, як від важкої фізичної праці. Однак це почуття помилкове і боротися з ним допоможе спорт. Активний відпочинок відмінно бадьорить, розганяє кров, додає сил. Чимало програмістів відчувають потребу в активному відпочинку на підсвідомому рівні, вибираючи спорт як хобі для проведення вільного часу.

При цьому не варто забувати, що умови праці програмістів також характеризуються можливістю впливу на них наступних небезпечних і шкідливих виробничих факторів: шуму; тепловиділень, причому шкоди організму можуть завдати не тільки високі, але і низькі температури; іонізуючих і неіонізуючих випромінювань: рентгенівське, інфрачервоне, електромагнітне випромінювання ВЧ і СВЧ діапазону; статичної електрики; недостатнє штучне та природнє освітлення; візуальні фактори: яскравість, контрастність, мерехтіння зображення, відблиски тощо.

За таких умов зростає роль та значення охорони праці, як системи правових, соціально-економічних, організаційно-технічних, санітарно-

гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці. Адже в кінцевому рахунку плоди науково-технічного прогресу можуть бути ефективними лише в тій мірі, в якій вони забезпечують людині безпеку, комфортність і зручність трудової діяльності [31].

6.2.2Аналіз умов праці ІТ-фахівців.

Аналіз умов праці ІТ-фахівців свідчить про наявність та можливий вплив наступних шкідливих та небезпечних чинників: шуму; несприятливому мікроклімату (тепловиділення, високі температури при нагріві центральних процесорів та блоків живлення та низькі – при проведенні ремонтних робіт та обслуговування криогенного обладнання квантових комп’ютерів; іонізуючі і неіонізуючі випромінювання (рентгенівське, інфрачервоне, електромагнітне ВЧ і СВЧ діапазону, статичної електрики); недостатнє штучне та природнє освітлення; візуальні фактори: надмірна яскравість, контрастність, мерехтіння зображення, відблиски тощо.

6.2.3 Підвищення ефективності системи управління

Для підвищення ефективності системи управління охорони праці (СУОП) дуже важлива роль належить формуванню і розвитку інформаційної культури фахівців ІТ-технологій, яка впливає на удосконалення інформаційного контуру сучасних підприємств, дозволяє створювати надійні прогнози щодо стану умов праці, показників здоров’я та працездатності, виробничого травматизму і професійної захворюваності, визначати політику розвитку підприємств, установ та організацій на основі різноманітних стратегій охорони праці (інноваційні, маркетингові, інвестиційні, фінансові, технологічні, диверсифікаційні).

З метою вдосконалення охорони праці фахівців з сучасних інформаційних технології розроблені системні заходи з правової оптимізації, соціального захисту працівників, діагностики їх професійної придатності, необхідності обґрунтування диференційованих норм праці при роботі з

новою комп’ютерною технікою в залежності від віку, статі, категорії зорової роботи, режимів праці та відпочинку (протягом робочого дня, тижня, щорічного режиму відпустки). Для підвищення ефективності розумової працездатності та зорової роботи повинне здійснюватися ергономічна оптимізація в рамках системи «оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров’я фахівця ІТ-індустрії. В цьому напрямі заслуговує на увагу створення при великих центрах інформаційних технологій кімнат (кабінетів) психофізіологічного розвантаження (на 5 місць). Заслуговує на увагу зарубіжний досвід створення у приміщеннях та в зоні їх розміщення на територіях підприємств спеціальних візуальних комфортних умов та забезпечення вимог виробничої естетики, дотримання норм рівнів виробничого шуму та акустичної тиші за межами офісу. Також дуже важливим є використання в офісних приміщеннях та кабінетах психофізіологічного розвантаження функціональної музики, яка сприяє попередженню перевтоми і підтриманню необхідного рівня розумової працездатності фахівців комп’ютерної галузі. Все це сприяє підвищенню ефективності роботи з новою комп’ютерною технікою приблизно на 10-15% у порівнянні з традиційною моделлю охорони праці.

6.3 Застосування основних способів знезаражування. Норми витрат дезактивуючих, дегазуючих і дезінфікуючих речовин. Визначення повноти знезаражування.

6.3.1 Знезаражування

Проводячи знезаражування, потрібно пам'ятати: територія для проведення знезаражування має бути достатньою, щоб забезпечити необхідні дії людей і техніки, розміщення тварин і всього, що підлягає знезараженню; людей, техніку, тварин необхідно розміщувати з підвітряної сторони від місця аварії; знезаражування необхідно починати за принципом від простого

до складного; спочатку виділити велику забруднену масу для попередження небажаних контактів із зоною високої концентрації; суворо контролювати перебування в індивідуальних засобах захисту; в холодну пору дії людей скуті, є труднощі в їх обслуговуванні, у разі замерзання заражених ділянок з'являються додаткові труднощі в ліквідації наслідків; готуючи й застосовуючи розчини для знезаражування, слід пам'ятати, що не всі розчини сумісні один з одним; на результати знезаражування суттєво впливають кількість води та її тиск; для знезаражування техніки, апаратури, приладів та ін. можна застосувати пар під низьким і високим тиском, але потрібно пам'ятати, що при високому тиску може утворюватися заражений аерозоль, здатний поширюватися за межі осередку зараження.

У разі необхідності потрібно організувати знезаражування території, будівель, складів, овочесховищ і продукції.

Дезактивація - це видалення РР з поверхнірізних об'єктів, а також із продуктів харчування, фуражу, сировини і води. Для визначеннянеобхідност в дезактиваці проводять дозиметричний контроль радіоактивного забруднення. Дезактивацію можна проводити часткову або повну.

Профілактична дезінфекція проводиться з метою запобігання виникненню й поширенню збудника захворювання в навколишньому середовищі.

Осередкова дезінфекція здійснюється в оточенні інфікованого хворого (поточна) і після його ізоляції, виписки або переведення до іншого відділення (заключна) [32].

6.3.2 Якими методами проводять дезінфекцію

Виділяють п’ять основних методів дезінфекції: **хімічний, фізичний, механічний, біологічний** та **комбінований. Кожен з цих методів використовується в практиці як окремо, так і в комбінації з іншими.**

Хімічний – основний метод дезінфекції, який полягає в застосуванні різних хімічних речовин та їхніх сполук для знищення патогенних й умовно

патогенних мікроорганізмів на поверхнях, всередині об’єктів і предметів навколишнього середовища, а також в повітрі й різних субстратах.

Основні способи проведення дезінфекції із застосуванням хімічних дезпрепаратів:

- зрошення об’єктів обробки за допомогою спеціальної дезінфекційної техніки.

- нанесення аерозолю дезінфекційного засобу на об’єкти обробки за допомогою розпилювача.

- занурення в робочий розчин дезінфекційного засобу посуду, медичних виробів, манікюрного інструментарію, предметів догляду за хворими, інвентарю і т. д.

- протирання різних поверхонь серветкою, змоченою робочим розчином дезінфекційного засобу.

Препарати, які використовуються для дезінфекції повинні відповідати ряду вимог, серед яких: широкий спектр антимікробної активності, безпека для людини та навколишнього середовища, хороша розчинність в воді, ефективність при взаємодії з органічними забрудненнями, нейтральний запах тощо.

Фізичний - проводять за допомогою впливу на об’єкт знезараження різних фізичних факторів: кип’ятіння, випалювання, використання дії ультрафіолетового опромінення тощо.

Основа фізичного методу – термообробка. Більшість патогенних мікроорганізмів гинуть при температурі 60-70 ° С, проте їх спори здатні витримати й більш високі температури

Підбір конкретного методу залежить від багатьох факторів, включаючи мету знезараження, тип оброблюваного об’єкта, вид збудника, умови, в яких здійснюється дезінфекція та інших.

Механічна- проводиться з метою зменшення концентрації мікроорганізмів на об’єктах навколишнього середовища. До механічних

методів належить вологе прибирання, миття рук, видалення зараженого шару ґрунту, фільтрація води, прибирання приміщень пилососом тощо.

Варто зазначити, що механічна дезінфекція не знищує мікроби, а лише частково видаляє їх з об'єктів знезараження, виконуючи допоміжну функцію. Цей метод застосовується також для санітарної обробки людей, фільтрації повітря, води та інших рідин і т. д.

Усі механічні прийоми застосовуються для:

-очищення оброблюваних об'єктів від бруду, жиру та білкових частинок;

-видалення певної кількості мікроорганізмів, що знаходяться на поверхні рук людини, предметах, в повітрі і в воді.

Якість механічної дезінфекції залежить від устаткування,яке використовується для цієї мети. Наприклад, вологе прибирання з використанням ганчірок та щіток дає значно кращі результати, ніж сухе прибирання.

Біологічний - полягає у знищенні збудників інфекційних захворювань мікробами-антагоністами.

Антагонізм мікроорганізмів – тип взаємодії мікроорганізмів, при якому один штам повністю знищує або уповільнює ріст іншого.

У сучасній дезінфекції цей спосіб не застосовують через його трудомісткість.

Комбінований- ґрунтується на поєднанні декількох вищевказаних методів дезінфекції.

Профілактична дезінфекція проводиться з метою запобігання виникненню й поширенню збудника захворювання в навколишньому середовищі. **Осередкова дезінфекція** здійснюється в оточенні інфікованого хворого (поточна) і після його ізоляції, виписки або переведення до іншого відділення (заключна) [33].

7 ЕКОЛОГІЯ

7.1 Моніторинг атмосферного повітря

Зміни у навколишньому природному середовищі відбуваються під впливом природних і антропогенних (зумовлених діяльністю людини) біосферних факторів. Пізнання цих змін неможливе без виокремлення антропогенних процесів на фоні природних, для чого й організовують спеціальні спостереження за різноманітними параметрами біосфери, які змінюються внаслідок людської діяльності. Саме у спостереженні за довкіллям, оцінюванні його фактичного стану, прогнозуванні його розвитку полягає сутність моніторингу.

Моніторинг атмосферного повітря – це система спостережень за станом атмосфери, його забрудненням і природними явищами, які відбуваються в ньому, а також оцінка і прогноз стану атмосферного повітря [34].

В даний час в багатьох містах промислово розвинених країн створюється мережа пунктів спостереження (моніторингу) за забрудненням повітря. За останнє десятиліття дана система отримала значне розширення і розвиток. Збільшилося число міст, в яких ведеться контроль за забрудненням повітря, число пунктів спостережень в них і спостережуваних інгредієнтів. Розроблені нові методи і технічні засоби вимірів, у тому числі автоматичні прилади і системи контролю. Характерною особливістю розвитку моніторингу є і те, що організацією і вдосконаленням його у ряді країн активно зайнялися метеорологічні відомства. Це дозволило підвищити науково-технічний рівень спостережень, що проводилися, і одночасно з виміром концентрацій шкідливих речовин вивчити метеорологічні, топографічні і інші чинники, що визначають їх розподіл в атмосфері[35].

Моніторинг у галузі охорони атмосферного повітря проводиться з метою отримання, збирання, оброблення, збереження та аналізу інформації

про рівень забруднення атмосферного повітря, оцінки та прогнозування його змін і ступеня небезпечності та розроблення науково обґрунтованих рекомендацій для прийняття рішень у галузі охорони атмосферного повітря (ст. 32 Закону України “Про охорону атмосферного повітря”). Він є складовою частиною державної системи моніторингу довкілля України[36].

До об’єктів моніторингу атмосферного повітря належить: атмосферне повітря, у тому числі атмосферні опади; викиди забруднюючих речовин в атмосферне повітря.

Суб’єктами, які здійснюють моніторинг атмосферного повітря, є: Мінприроди України, МНС України, Державна санітарно-епідеміологічна служба МОЗ України, їх органи на місцях, підприємства, установи, організації, діяльність яких призводить або може призвести до погіршення стану атмосферного повітря.

Проведення моніторингу атмосферного повітря має на меті отримання: первинних даних контролю за викидами та спостережень за станом забруднення; узагальнених даних про рівень забруднення на певній території за певний проміжок часу; узагальнених даних про склад та обсяги викидів забруднюючих речовин; оцінки рівня та ступеня небезпечності забруднення для довкілля та життєдіяльності населення; оцінки складу та обсягів викидів забруднюючих речовин.

Порядок організації та проведення моніторингу в галузі охорони атмосферного повітря затверджено постановою Кабінету Міністрів України від 9 березня 1999 р. зі змінами, внесеними постановою Кабінету Міністрів від 24 вересня 1999 р [37].

Відповідно до ст. 27 Закону України “Про охорону атмосферного повітря” контроль у галузі охорони атмосферного повітря здійснюється з метою забезпечення дотримання вимог законодавства про охорону атмосферного повітря. Виділяються такі його види: державний, виробничий, громадський [3].

Мінприроди України здійснює свою діяльність у галузі охорони атмосферного повітря спільно з санітарно-епідеміологічною службою МОЗ України та його органами на місцях у частині додержання нормативів екологічної безпеки та інших правил і нормативів, спрямованих на запобігання негативному впливу на здоров'я людей; Державною автомобільною інспекцією МВС України та її органами на місцях у частині додержання нормативів вмісту забруднюючих речовин у відпрацьованих газах та шкідливого впливу фізичних факторів, встановлених для відповідного типу автомобільного транспорту та сільськогосподарської техніки; іншими державними органами, а також органами місцевого самоврядування відповідно до законодавства України.

Місцеві органи державної виконавчої влади контролюють, як виконуються і дотримуються правила по оздоровленню навколишнього середовища, як здійснюється санітарна охорона атмосферного повітря. Вони забезпечують проведення заходів щодо охорони навколишнього середовища, запобігання, зниження інтенсивності й усунення шуму у виробничих, жилих і громадських приміщеннях, у дворах, на вулицях і площах населених пунктів [38].

Згідно зі ст. 29 Закону України "Про охорону атмосферного повітря" виробничий контроль у галузі охорони атмосферного повітря здійснюють суб'єкти господарювання, які в своїй діяльності використовують джерела шкідливих хімічних, біологічних і фізичних впливів на атмосферне повітря і які призначають осіб, що відповідають за проведення виробничого контролю в галузі охорони атмосферного повітря [39].

7.2. Організаційні форми, види і способи статистичного спостереження в екології

7.2.1 Проведення статистичного спостереження

Статичне спостереження в екології - це планомірний, науково-організований збір масових даних про екологічні явища і процеси. Здійснюється шляхом реєстрації за заздалегідь розробленою програмою спостереження.

Об'єктом спостереження є стан забруднення навколишнього середовища (природних об'єктів) атмосферного повітря, природних водних об'єктів, земель та ґрунтів. Збір даних проводиться не стихійно, а регулярно, що дає змогу вивчити тенденції, напрями, закономірності розвитку екологічних явищ і процесів. План статистичного спостереження передбачає широке коло питань методики та організації збору статистичної інформації, контролю її якості та вірогідності. Для об'єкта статистичного спостереження характерне те, що його не можна вивчати безпосередньо в цілому, для цього потрібно виділити в його складі окремі одиниці.

Одиниця статистичного спостереження - це складовий елемент об'єкта дослідження, який є основою рахунку і носієм істотних ознак та властивостей, які підлягають реєстрації. Це первинний елемент об'єкта дослідження. Одиницю спостереження встановлюють, виходячи із завдань спостереження і складності об'єкта дослідження. Правильне визначення одиниці спостереження має істотне значення для організації і проведення статистичного дослідження. Цим значною мірою зумовлюється об'єктивність одержаних результатів.

Основне завдання статистичного спостереження - отримання вірогідних статистичних даних, які об'єктивно характеризують явища і процеси суспільного життя. Інформація статистичного спостереження

повинна бути об'єктивною і якісною, а отже, забезпечуватись правильною науковою організацією її' одержання, належним виконанням самого спостереження. Завдання статистичного спостереження зумовлюється завданнями, які ставляться перед дослідженням певних екологічних процесів і явищ і випливають з потреб управління ними.

Наукова організація статистичного спостереження зумовлює дотримання певних вимог щодо його здійснення:

- статистичне спостереження повинно здійснюватися на науковій основі заздалегідь розробленою програмою, яка забезпечувала б науковий підхід до вирішення методологічних та організаційних питань;

- статистичне спостереження повинне забезпечувати збір масових даних, у яких відбивається вся сукупність фактів. Неповнота зведень про досліджувані процеси призведе до помилкових висновків з результатів аналізу;

- орієнтація статистичного спостереження на збирання не тільки інформації, яка безпосередньо характеризує досліджуваний об'єкт, а й такої, що сприяє зміні його стану;

- інформація, одержана за результатами статистичного спостереження, повинна бути вірогідною;

Дані статистичного спостереження повинні бути порівнювані. Лише в такому разі забезпечується їх узагальнення і зіставлення у просторі й часі.

7.2.2 Організаційні форми, види і способи статистичного спостереження

Форми спостереження. У статистичний практиці застосовують дві організаційні форми спостереження: звітність і спеціально організовані статистичні спостереження.

Звітність — це форма статистичного спостереження, при якій статистичні дані надходять у статистичні органи від підприємств і установ у вигляді обов'язкових і таких, що мають юридичну силу звітів про їх роботу.

Звітність підприємств, установ та організацій є поки що основним джерелом статистичної інформації. У ній передбачається система твердо регламентованих показників, які характеризують діяльність підприємств, установ та організацій. Зміст звіту, форма і термін подання також встановлюється вищим статистичним органом. Звітність складають на основі документів первинного оперативно-технічного і бухгалтерського обліку. Вірогідність гарантується також юридичною відповідальністю керівників підзвітних підприємств та організацій.

Перелік усіх форм із зазначенням їх реквізитів називають табелем звітності. За різними ознаками статистичну звітність поділяють на окремі види. Насамперед розрізняють типову і спеціалізовану звітність:

- типова звітність має єдину форму і зміст для всіх підприємств окремої галузі або всього народного господарства;

- спеціалізована звітність властива тим підприємствам чи окремим виробництвам, що мають свої специфічні особливості.

За періодичністю подання звітність буває тижнева, двотижнева, місячна, квартальна, різна; за способом подання - термінова (телеграфна) і поштова. Вид звітності впливає на техніку збору і зведення статистичної інформації. Удосконалення статистичної звітності на сучасному етапі відбувається у напрямі скасування термінової звітності та скорочення кількості поштових звітів.

За порядком проходження звітність поділяють на централізовану і децентралізовану:

- централізована звітність проходить через систему державної статистики, де обробляється і передається відповідним органам управління;

- децентралізована опрацьовується у відповідних міністерствах чи відомствах, а зведення подають статистичним органам.

Другою за значенням організаційною формою спостереження є спеціально організоване статистичне спостереження. Застосовують його у випадках, коли не можна застосувати звітність або складати звітність

нераціонально. Коли необхідно детально вивчити явище поряд з вивченням його у формі звітності або потрібно перевірити вірогідність даних звітності.

Спеціально організоване статистичне спостереження поєднує в собі такі організаційні форми: а) перепис, б) суцільне і несуцільне обстеження.

Види і способи спостереження. Різноманітність соціально-економічних явищ потребує різних видів спостереження.

Різновид спостереження визначається ознакою групування: охватом одиниць сукупності, часом проведення, способом одержання статистичних даних.

За охватом одиниць сукупності спостереження поділяють на суцільне і несуцільне:

- при суцільному спостереженні обстеженню і реєстрації підлягають усі без винятку елементи сукупності; прикладами суцільного спостереження є статистична звітність, яку складають і подають державні і кооперативні підприємства чи установи, а також перепис населення;

При несуцільному спостереженні обліку підлягають не всі елементи сукупності, наприклад обстеження бюджетів населення.

Несуцільні спостереження поділяють на такі види: спостереження основного масиву, вибіркове, монографічне і анкетне:

- спостереження основного масиву охоплює переважну частину елементів сукупності, обсяг значень істотної ознаки у яких визначає розмір явища. Цей метод використовують при вивчені екологічного стану регіонів;

- при вибірковому спостереженні також обстежуються не всі елементи сукупності, а певна, випадково відібрана їх частина. Таке спостереження застосовують для вивчення якості природних сфер, екологічного стану НПС, забрудненості об'єктів середовища тощо;

- монографічне спостереження передбачає детальне обстеження лише окремих типових елементів сукупності. До цього вдаються з метою поглибленого вивчення тих сторін екологічних явищ, які не були висвітлені масовим обстеженням;

За часом проведення статистичне спостереження поділяють на поточне, періодичне і одноразове:

- поточне спостереження полягає в безперервній реєстрації фактів по мірі їх виникнення. Так здійснюється облік викидів шкідливих речовин в атмосферне повітря, природні водойми, ґрунти;

- періодичне спостереження проводиться регулярно, здебільшого через рівні проміжки часу;

Одноразове спостереження проводять епізодично з метою вирішення певних соціально-економічних завдань. Прикладом є обстеження при аварійних та інших надзвичайних ситуаціях.

За способом одержання статистичних даних виділяють: безпосередній облік фактів, документальний облік і опитування респондентів:

- безпосередній облік фактів передбачає безпосередній огляд, перелік, вимірювання, зважування тощо. Так проводять інвентаризацію викидів на підприємствах;

- документальний облік ґрунтується на даних різноманітних документів первинного обліку. Найбільш широкого вжитку він набув при складанні статистичної звітності, екологічних паспортів, паспортів забруднюючих речовин тощо;

Опитування респондентів - це таке спостереження, при якому відповіді на питання формуляра записують зі слів респондента.

Опитування буває експедиційне, само реєстрація, кореспондентське і анкетне:

- при експедиційному опитуванні спеціально підготовлені реєстратори заповнюють формуляри спостереження і одночасно перевіряють правдивість відповідей на питання;

Само реєстрація - це опитування, при якому респонденти самі заповнюють статистичні формуляри.

- працівники статистичних органів лише інструктують їх і перевіряють повноту та правильність одержаних відомостей;

- кореспондентське опитування здійснюють спеціальні дописувачі, які заповнюють формуляри згідно з інструкцією і передають відомості статистичним органам;

При анкетному опитуванні анкети респондентам вручають особисто або висилають поштою. Опитування може проводитись також у формі інтерв'ю. Це спосіб допускає довільність відповідей респондентів на поставлені питання, з'ясування їх думок. Різноманітність екологічних явищ, їх специфіка, особливості статистичного вимірювання потребують поєднання зазначених способів і видів спостереження [40].

ВИСНОВКИ

В ході виконання дипломної роботи була досягнута основна його мета – створено інформаційну систему для розпізнавання та автовиправлення коду на основі штучної нейронної мережі; досліджено залежність точності та самостійного виправлень від особливостей обраної архітектури

Розроблений алгоритм інформаційної системи призначений для перевірки та виправлення розмітки коду програм на різного роду IT-підприємствах.

Головним завданням було створення алгоритму, який дає високий відсоток точності та швидкості перевірок самостійного виправлення файлів проекту. На даний момент одним із найефективніших є метод, що використовує штучних нейронних мереж.

В даній роботі використано інтелектуальну систему перевірки та виправлення розмітки коду програм, яка дозволить значно спростити управління якістю коду та його чистотою. Таким чином можна привести у компанію стажера та не боятись довірити йому проект. Звичайно система буде йому давати підказки.

При розробці системи була спроектована база даних, яка дозволяє зберігати дані, що необхідні для роботи системи. Системою управління базою даних була обрана СУБД PostgreSQL (вимовляється «Пост-грес-К'ю-ель» об'єктно-реляційна система керування базами даних).

Під час роботи було спроектовано та побудовано контекстну діаграму та також діаграму декомпозиції стандарту IDEF0, діаграми дерева вузлів і FEO, діаграму потоків даних (Dataflowdiagramming - DFD) та діаграму потоку робіт стандарту опису процесів IDEF3.

В результаті досліджень було отримано наступні результати:

1. проаналізовано методи та моделі розпізнавання символьної інформації;

2. розроблено структуру інтелектуальної системи нейромережевої перевірки структур коду;

3. розроблено нейронну мережу для написання розмітки коду;

4. здійснено програмну реалізацію інтелектуальної системи нейромережевого розпізнавання структур коду;

5. проведено тестування та аналіз результатів.

Окремі результати роботи представленні на двох наукових конференціях:

1. Веселовська В.О. Інформаційна комп’ютерна система контролю та управління доступом XVII міжнародна наукова – практична конференція «Математичне та програмне забезпечення інтелектуальних систем» Дніпровський національний університет імені Олеся Гончара, (Дніпро ,Україна 20-22 листопада 2019 року).

2. Веселовська В.О. Статистичний багатомовний переклад запитів при інформаційному пошуку Матеріали VII науково-технічної конфції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 11 – 12 грудня 2019 р.). – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2019. – 196 с.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Goodfellow I. Deep Learning / Goodfellow I., Bengio Y. and Courville A.; Cambridge MA : MIT Press [2017] – 777 pages.

2.J.R. Ullmann, An Algorithm for Subgraph Isomorphism — National Physical Laboratory, England, 1976.

3. F. E. Allen. Control flow analysis - Proceedings of a symposium on Compiler optimization, 1970.

4. Springenberg, J. T. Striving for Simplicity: The All Convolutional Net / Springenberg, J. T. Dosovitskiy, A.; Brox, T. & Riedmiller, M. [Електронний ресурс]: – Режим доступу: https://arxiv.org/abs/1412.6806. Заголовок з екрану.

5. Ruder S. (2016) An overview of gradient descent optimisation algorithms. [Електронний ресурс]: – Режим доступу: https://arxiv.org/abs/1609.04747.Заголовок з екрану. – Дата доступу до ресурсу: 07.10.19

6. Nielsen M. Neural Networks And Deep Learning. [Електронний ресурс]:– Режим доступу: http://neuralnetworksanddeeplearning.com/.Заголовок з екрану.

7.Каллан Р. Основные концепции нейронных сетей = TheEssenceofNeuralNetworksFirstEdition. — 1-ше. — «Вильямс», 2001. — С. 288. — ISBN 5-8459-0210-X. (рос.)

8. Кудрявцев Л. Д. Математический анализ. — 2-е изд. — М.: Высшая школа, 1973. — Т. 1.

9. P. Sermanet and Y. LeCun, “Traffic sign recognition with multiscale convolutional networks,” in International Joint Conference on Neural Networks (IJCNN), Jul. 2011, pp. 2809–2813. [Online]. [Електронний ресурс]: – Режим доступу: http://ieeexplore.ieee.org/document/6033589/ Заголовок з екрану.

10. N. McLaughlin, J. M. D. Rincon, and P. Miller, “Dataaugmentation for reducing dataset bias in person re-identification,” in International Conference on Advanced Video and Signal Based Surveillance (AVSS), no. 12, Aug. 2015, pp. 1–6. [Online] [Електронний ресурс]: – Режим доступу: http://ieeexplore.ieee.org/abstract/document/ 7301739/ Заголовок з екрану.

11.Linux [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Linux Заголовок з екрану.

12.Python [Електронний ресурс]: – Режим доступу: https://tproger.ru/tag/python/ Заголовок з екрану.

13.TensorFlow [Електронний ресурс]: – Режим доступу: https://www.tensorflow.org/ Заголовок з екрану.

14. MySQL [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/MySQL Заголовок з екрану.

14. XVII Міжнародна науково-практична конференція «Математичне та програмне забезпечення інтелектуальних систем».

15.http://scs.kpi.ua/sites/default/files/files/2017/%D0%91%D0%B0%D0%BA%D0%B0%D0%BB%D0%B0%D0%B2%D1%80%D0%B8/%D0%9A%D0%B0%D0%BC%D0%BF%D0%BE%D0%B2.pdf- Дата доступу до ресурсу:07.10.19.

16.Рекурентна нейронна мережа Елмана [Електронний ресурс]: – Режимдоступу:https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D0%B5%D0%BD%D1%82%D0%BD%D0%B0_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0.

17.TripletLoss [Електронний ресурс]: – Режим доступу: https://towardsdatascience.com/losslesstripletloss7e932f990b24?gi=66af7d094995 Заголовок з екрану.

18.https://stud.wiki/programming/3c0a65635b2ac78b4d43b88521216c37_0.html.

19. Фуллер М., Скотт К. UML у короткому викладі. Застосування стандартної мови об'єктного моделювання. – М.: Мир, 1999. – 199 с.

20. Маклаков С.В. BPwin і Erwin. Кошт-засіб-CASE-засоби розробки інформаційних систем. - М.: ДИАЛОГ-МИФИ, 2001. – 314 с.

21.ПО MilestoneXProtectLPRCameraLicense [Електронний ресурс]: – Режим доступу: https://secur.ua/ip-videonablusenie/ programnoeobespechenie/ milestone/milestonexprotectlprcameralicense.html Заголовок з екрану.

22. Sistemniy-analz-Metodichn-vkazvki-do-lr (2).

23.Комп’ютерні системи, мережі та системне програмування. Методичні вказівки з виконання кваліфікаційної роботи магістра та самостійної роботи для студентів спеціальностей: 8.05010201 – “Комп’ютерні системи та мережі”, 8.05010202 – “Системне програмування”, 8.05010203 – “Спеціалізовані комп’ютерні системи” / Укл.: Казимир В.В., Вервейко О.І. – Чернігів: ЧДТУ, 2012. – 34 с.

24.http://wpfan.com.ua/scho-take-validnist-i-navischo-vona/ - Дата доступу до ресурсу:10.10.19.

25.https://www.victoria.lviv.ua/library/students/wd/work9.html - Дата доступу до ресурсу:04.10.19.

26.https://uk.wikipedia.org/w/index.php?title=%D0%A2%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F&action=edit&section=20 - [Електронний ресурс]: –Режим доступу ресурсу:07.10.19.

27.https://www.bibliofond.ru/view.aspx?id=797683 - [Електронний ресурс]: –Режим доступу ресурсу:07.10.19.

28.https://wiki.legalaid.gov.ua/index.php/%D0%9F%D1%80%D0%B0%D1%86%D1%8F_%D0%B6%D1%96%D0%BD%D0%BE%D0%BA:_%D0%BF%D1%96%D0%BB%D1%8C%D0%B3%D0%B8,_%D0%B3%D0%B0%D1%80%D0%B0%D0%BD%D1%82%D1%96%D1%97 - [Електронний ресурс]: –Режим доступу ресурсу:07.10.19.

29.Психология безопасности труда / Укладач Кальянов А.В. // Донецкий областной совет профсоюза, 2008. – 32 с.

30. Сьогодні UA [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.segodnya.ua/lifestyle/fun/pochti-kak-u-google-chemudivlyayut-ofisy-ukrainskih-it-kompaniy--764025.html – відкритий.

31.Конспект лекцій з курсу «Охорона праці в галузі» / Укладачі: Яскілка В.Я., Олійник М.З. – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016. – 56 с.

32.https://pidruchniki.com/16631116/bzhd/znezarazhuvannya.[Електронний ресурс]: – дата доступу ресурсу: 07.10.19.

33. https://uk.interdez.com.ua/press/metody-dezinfektsii.html.

34. Клименко М. О., Прищепа А. М., Вознюк Н. М. Моніторинг довкілля: Підручник. – К.: Видавничий центр “Академія”, 2006. – 360 с.

35. Безуглая Э. Ю. Мониторинг состояния загрязнения атмосферы в городах.– Л.: Гидрометеоиздат, 1986. – 200 с.

36.Закон України «Про охорону атмосферного повітря». – К., 1992

37.Постанова від 9 березня 1999 р. N 343 Київ Про затвердження порядку організації та проведення моніторингу в галузі охорони атмосферного повітря.

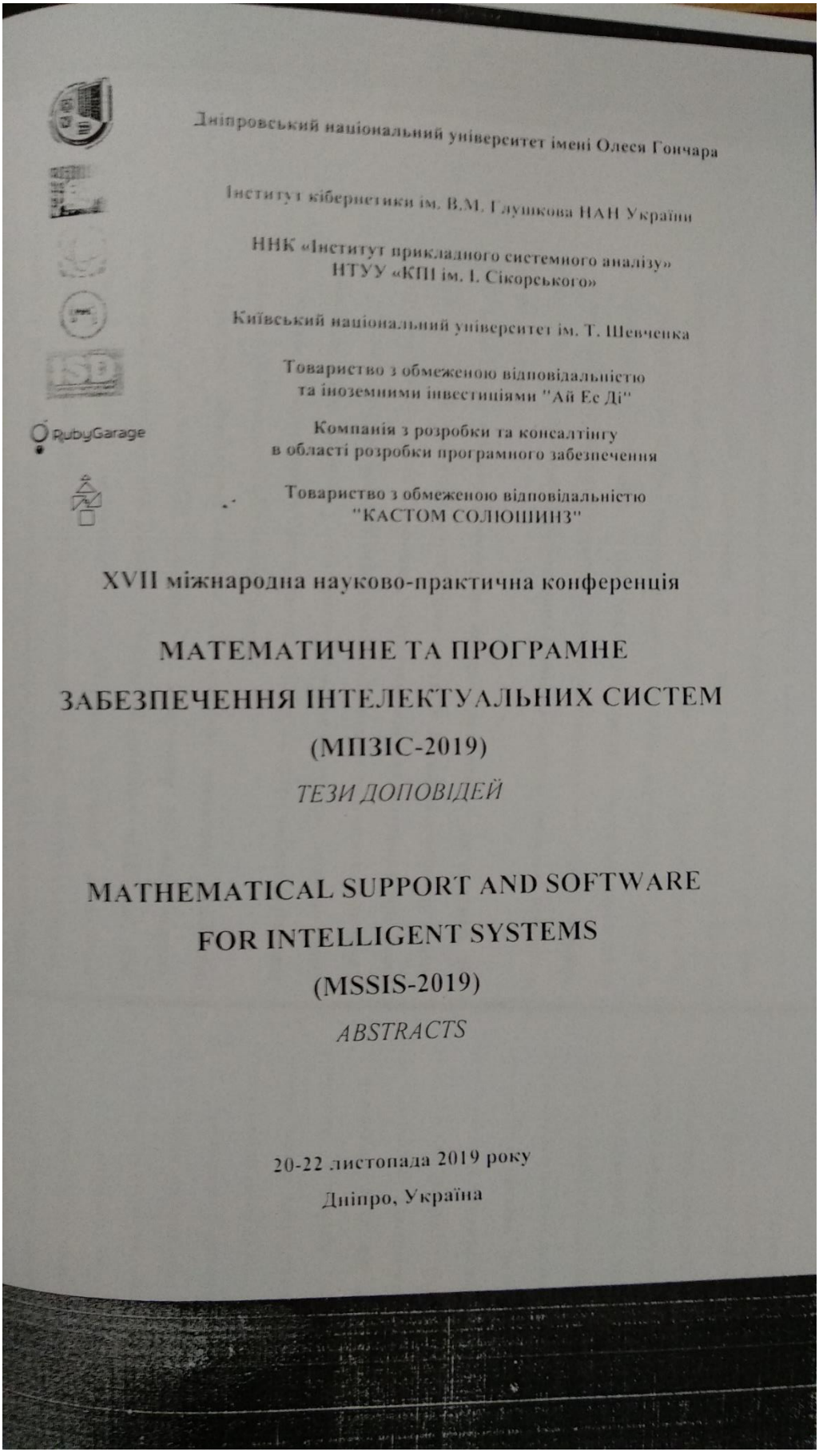
38. Бринчук М. М. Правовая охрана атмосферного воздуха. – М., 1986.

39.Положення про державну систему моніторингу довкілля, затвердженого постановою Кабінету Міністрів України від 30 березня 1998 р. N 391.

40.http://4exam.info/book_257_glava_10_FORMUVANNJA_BAZI_STATISTICHNIKH_DANIKH_V_EKOLOG%D0%86%D0%87.html [Електронний ресурс]: –Режим доступу ресурсу :07.10.19

ДОДАТКИ

Додаток А



Дніпровський національний університет імені Олеся Гончара

Інститут кібернетики ім. В.М. Глушкова НАН України

ННК «Інститут прикладного системного аналізу»
НТУУ «КПІ ім. І. Сікорського»

Київський національний університет ім. Т. Шевченка

Товариство з обмеженою відповідальністю
та іноземними інвестиціями "Ай Ес Ді"

RubyGarage

Компанія з розробки та консалтінгу
в області розробки програмного забезпечення

Товариство з обмеженою відповідальністю
"КАСТОМ СОЛЮШИНЗ"

XVII міжнародна науково-практична конференція

МАТЕМАТИЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ

(МПЗІС-2019)

ТЕЗИ ДОПОВІДЕЙ

MATHEMATICAL SUPPORT AND SOFTWARE FOR INTELLIGENT SYSTEMS

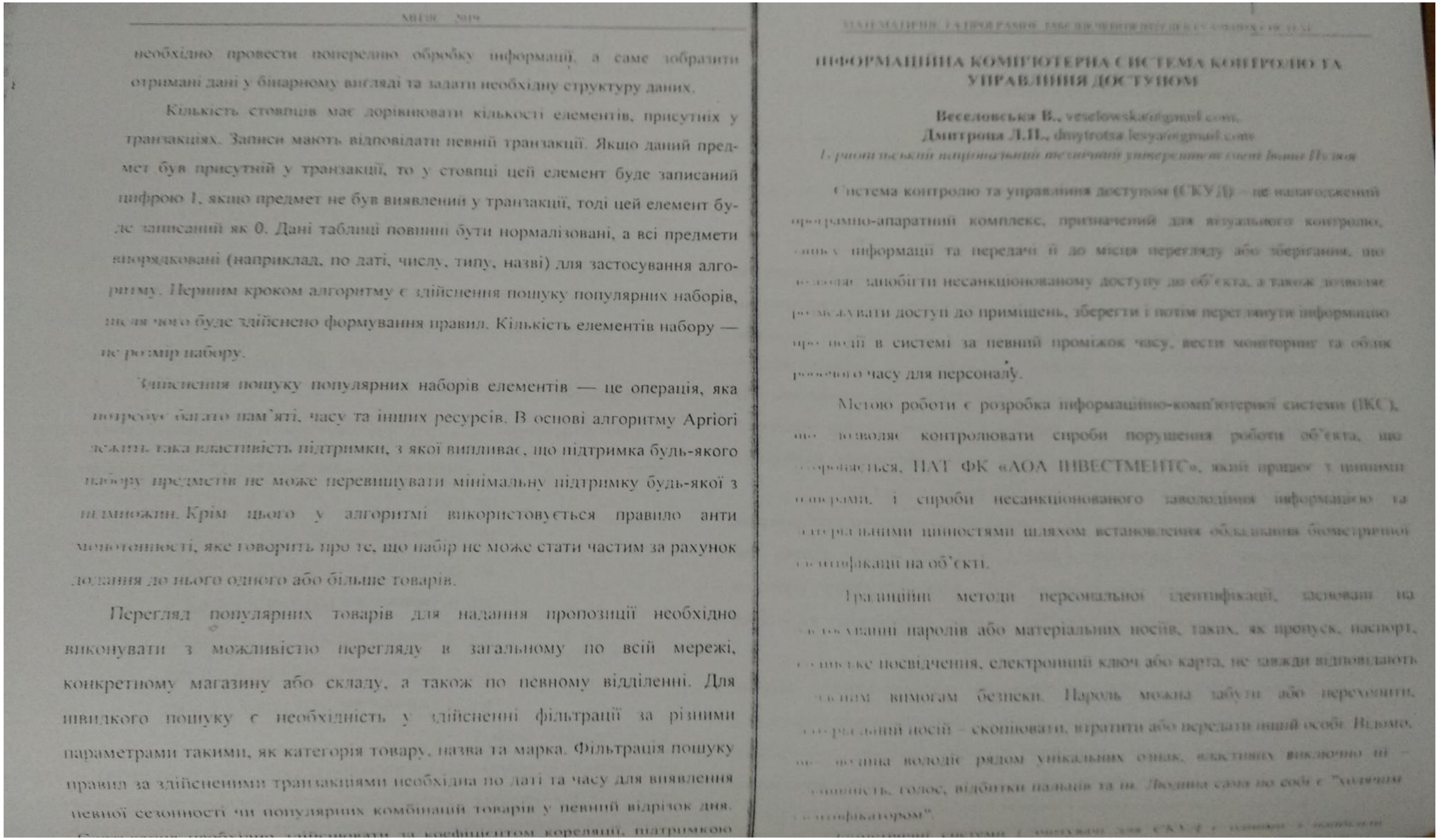
(MSSIS-2019)

ABSTRACTS

20-22 листопада 2019 року

Дніпро, Україна

Продовження додатку А



необхідно провести попередню обробку інформації, а саме зобразити отримані дані у бінарному вигляді та задати необхідну структуру даних.

Кількість стовпців має дорівнювати кількості елементів, присутніх у транзакціях. Записи мають відповідати певній транзакції. Якщо даний предмет був присутній у транзакції, то у стовпці цей елемент буде записаний цифрою 1, якщо предмет не був виявлений у транзакції, тоді цей елемент буде записаний як 0. Дані таблиці повинні бути нормалізовані, а всі предмети впорядковані (наприклад, по даті, числу, типу, назві) для застосування алгоритму. Першим кроком алгоритму є здійснення пошуку популярних наборів, після чого буде здійснено формування правил. Кількість елементів набору — це розмір набору.

Здійснення пошуку популярних наборів елементів — це операція, яка потребує багато пам'яті, часу та інших ресурсів. В основі алгоритму Apriori лежить така властивість підтримки, з якої випливає, що підтримка будь-якого набору предметів не може перевищувати мінімальну підтримку будь-якої з підмножин. Крім цього у алгоритмі використовується правило анти монотонності, яке говорить про те, що набір не може стати частим за рахунок додання до нього одного або більше товарів.

Перегляд популярних товарів для надання пропозиції необхідно виконувати з можливістю перегляду в загальному по всій мережі, конкретному магазину або складу, а також по певному відділенні. Для швидкого пошуку є необхідність у здійсненні фільтрації за різними параметрами такими, як категорія товару, назва та марка. Фільтрація пошуку правил за здійсненими транзакціями необхідна по даті та часу для виявлення певної сезонності чи популярних комбінацій товарів у певний відрізок дня.

ІНФОРМАЦІЙНА КОМП'ЮТЕРНА СИСТЕМА КОНТРОЛЮ ТА УПРАВЛІННЯ ДОСТУПОМ

Веселовська В., veselowska@gmail.com,
Дмитроца Л.П., dmytrotsa.lesya@gmail.com
Тернопільський національний технічний університет імені Івана Пулюя

Система контролю та управління доступом (СКУД) – це налагоджений програмно-апаратний комплекс, призначений для візуального контролю, [illegible] інформації та передачі її до місця перегляду або зберігання, що дозволяє запобігти несанкціонованому доступу до об'єкта, а також дозволяє розмежувати доступ до приміщень, зберегти і потім переглянути інформацію про події в системі за певний проміжок часу, вести моніторинг та облік робочого часу для персоналу.

Метою роботи є розробка інформаційно-комп'ютерної системи (ІКС), що дозволяє контролювати спроби порушення роботи об'єкта, що охороняється, ПАТ ФК «АОА ІНВЕСТМЕНТС», який працює з цінними паперами, і спроби несанкціонованого заволодіння інформацією та матеріальними цінностями шляхом встановлення обладнання біометричної ідентифікації на об'єкті.

Традиційні методи персональної ідентифікації, засновані на використанні паролів або матеріальних носіїв, таких, як пропуск, паспорт, службове посвідчення, електронний ключ або карта, не завжди відповідають сучасним вимогам безпеки. Пароль можна забути або перехопити, матеріальний носій – скопіювати, втратити або передати іншій особі. Відомо, що людина володіє рядом унікальних ознак, властивих виключно їй – [illegible], голос, відбитки пальців та ін. *Людина сама по собі є "ходячим ідентифікатором".*

Додаток Б

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

VII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ »

11–12 грудня 2019 року

ТЕРНОПІЛЬ
2019

УДК 004.6
В. Веселовська, Л. Дмитроца
(Тернопільський національний технічний університет імені Івана Пулюя)

СТАТИСТИЧНИЙ БАГАТОМОВНИЙ ПЕРЕКЛАД ЗАПИТІВ ПРИ ІНФОРМАЦІЙНОМУ ПОШУКУ

UDC 004.6
V. Veselovskaya, L. Dmytrotsa
(Ternopil Ivan Puluj National Technical University, Ukraine)

STATYSTYCHNYY BAHATOMOVNYY PEREKLAD ZAPYTIV PRY INFORMATSIYNOMU POSHUKU

Розглядається можливість покращення релевантної видачі результатів інформаційного пошуку на запити користувача, враховуючи багатомовність вхідних даних розроблюваної системи обробки високошвидкісних потоків текстових даних.

Основною проблемою при частковому перекладі (на рівні запитів) є складність виявлення тематичних зв'язків між змістом пошукових запитів та змістом текстових документів через різне представлення у схожих текстах на різних мовах одної конкретної ситуації чи події. Даючи запит, людина формулює його, керуючись лише своїми представленнями про зміст необхідного документа. Розповсюджені лексичні засоби серед інформаційно-пошукових систем будуються на списках ключових слів. Виражений таким чином семантичний зміст документа обмежується цими списками для різних предметних галузей. Порівнюючи документи, представлені на різних мовах, але з однаковим тематичним змістом, їх схожість виявиться лише у разі співпадіння понять ключових слів для списків на цих мовах.

Одним із варіантів вирішення проблеми є розширення лексичного складу запитів та списків ключових слів або пар ключових слів. Також можливим є залучення алгоритмів статистичного перекладу. Даний тип машинного перекладу широко використовується у великих комерційних організаціях, оскільки потребує великих потужностей для обробки та зберігання мовних пар документів для кожної мови окремо. Ця технологія передбачає існування відкритих онлайн-сервісів – це Google translate та Яндекс-перекладач. Для роботи подібних систем необхідна наявність великих баз паралельних текстів, де зберігаються словосполучення (N-грами) та їх переклади (рис.1).

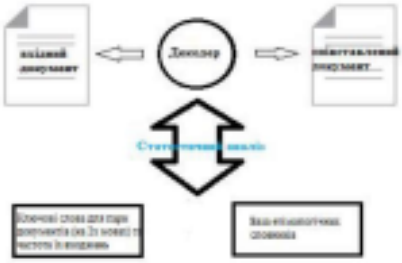


Рис. Блок-схема видачі запитів

Звузивши задачу з повнотекстового перекладу до перекладу лише запитів та видачі результатів на мові оригіналу, можливо суттєво знизити затрати розрахункових потужностей, реалізуючи алгоритми статистичного пошуку. Керуючись правилами спільного походження більшості слів для переліку мов, що входять до однієї мовної групи (слов'янські, романські та ін. мови), можливим є реалізація програмного алгоритму для більш детального виокремлення семантичного навантаження ключових слів запитів та документів на різних мовах. Планується реалізація даного підходу для розроблюваної системи обробки високошвидкісних потоків текстових даних, залучивши до роботи електронні бази етимологічних словників для різних мов (словники, що містять інформацію про фонетичні та семантичні зміни окремих слів та морфем конкретної мови).