

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АНАЛІЗУ ОПЕРАТОРСЬКОЇ ДІЯЛЬНОСТІ
RESEARCH AND SOFTWARE IMPLEMENTATION OF THE AI METHODS
FOR ANALYZING THE OPERATOR ACTIVITY

Виконав: студент(ка) 2 курсу, групи КНТ-110м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Шнейдер М.А.

(прізвище та ініціали)

Керівник Сердюк С.М.

(прізвище та ініціали)

Рецензент Касьян М.М.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
« » 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Шнейдера Максима Андрійовича

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Дослідження та програмна реалізація методів штучного інтелекту для аналізу операторської діяльності. Research and Software Implementation of the AI Methods for Analyzing the Operator Activity

керівник проекту (роботи) Сердюк Сергій Микитович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «05» листопада 2021 року № 435

2. Строк подання студентом проекту (роботи) 13 грудня 2021 року

3. Вихідні дані до проекту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Основні рішення з реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Сердюк С.М., доцент		
Нормоконтроль	Дейнега Л.Ю., ст.викладач		

7. Дата видачі завдання “ 28 ” лютого 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1-2 тижні	Завдання, ТЗ
2	Аналіз предметної області	3-4 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	5-6 тижні	Розділ 2
4	Розробка архітектури програми.	7 тиждень	Розділи 2, 3
5	Розробка програми.	8-9 тижні	Розділ 3
6	Розробка керівництва програміста.	10 тиждень	Розділ 4
7	Розробка керівництва оператора.	11 тиждень	Розділ 5
8	Оформлення пояснювальної записки та документів до неї.	12-13 тижні	Додатки
9	Нормоконтроль та рецензування.	14-15 тижні	
10	Захист роботи.	16 тиждень	

Студент(ка)

(підпис) Шнейдер М.А.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Сердюк С.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
121 с., 4 табл., 30 рис., 3 дод., 22 джерела.

ДІЯЛЬНІСТЬ ОПЕРАТОРА, ЛЮДИНО-МАШИННИЙ ІНТЕРФЕЙС,
ТЕХНОЛОГІЧНИЙ ПРОЦЕС, МОДЕЛЮВАННЯ, ГРАФ ПОДІЙ, ГРУПА
ВЗАЄМОПОВ'ЯЗАНИХ ПОДІЙ, ПЕРСЕПТРОН.

Об'єкт дослідження – процес проектування людино-машинних інтерфейсів (ЛМІ) автоматизованих систем управління технологічним процесом (АСУТП).

Предмет дослідження – діяльність проектувальника ЛМІ автоматизованої системи управління технологічним процесом (АСУТП).

Мета роботи – підвищення ефективності процесу проектування ЛМІ АСУТП шляхом розроблення і застосування програмного забезпечення на основі методів і моделей штучного інтелекту для аналізу діяльності операторів АСУТП.

Матеріали, методи та технічні засоби: методи інженерної психології; методи штучного інтелекту; мова програмування C#; програмна платформа .NET; фреймворк Windows Presentation Foundation (WPF); пакет прикладних програм MatLab; операційна система Windows 7 або вище.

Результати. Дістав подальшого розвитку відомий метод аналізу операторської діяльності Галактіонова. Метод, що розроблено, на відміну від попереднього, має засоби для формалізації процесу аналізу діяльності оператора АСУТП проектувальником ЛМІ АСУТП, шляхом розроблення і застосування програмного забезпечення на основі моделей штучного інтелекту.

Висновки. Використання розробленого програмного забезпечення дозволяє підвищити ефективність процесу проектування ЛМІ АСУТП.

Галузь використання – проектування людино-машинних систем (ЛМС).

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 121 pages, 4 tables, 30 figures, 3 appendixes, 22 sources.

OPERATOR ACTIVITIES, HUMAN-MACHINE INTERFACE, TECHNOLOGICAL PROCESS, MODELING, EVENT SCHEDULE, GROUP OF INTERCONNECTED EVENTS, PERSEPTRON.

Object of study is processes of designing human-machine interfaces (HMI) of automated process control systems (ACS).

Subject of study are the designer's activity of ACS HMI.

The purpose of the work is to increase the efficiency of the process of HMI ACS designing by developing and applying software based on methods and models of artificial intelligence for the analysis of the activities of operators ACS.

Materials, methods, and tools: methods of engineering psychology; methods of artificial intelligence; C# programming language; .NET software platform; Windows Presentation Foundation (WPF) framework; MatLab application package; operating system Windows 7 or higher.

Results. The well-known Galaktionov's method of analysis of operator activity has been further developed. The developed method, in contrast to the previous one, has the means for formalizing the process of analyzing the operator's activity of ACS by the designer of the HMI of the ACS, through the development and application of software based on artificial intelligence models.

Conclusions. The implementation of the developed software makes it possible to increase the design efficiency of the HMI of the ACS.

The field of use is a design of human-machine systems (HMS).

ЗМІСТ

	С.
Перелік скорочень та умовних познач 8	8
Вступ 9	9
1 Аналіз предметної області 11	11
1.1 Аналіз методів моделювання діяльності оператора..... 11	11
1.2 Метод розрахунку часу виявлення причин порушення оператором 18	18
1.3 Розробка технічного завдання на комплексний проєкт 27	27
1.3.1 Підстава для розробки 27	27
1.3.2 Мета розробки та її призначення 27	27
1.3.3 Основні вимоги до програмного продукту.....27	27
1.3.4 Вимоги до видів забезпечення і компонентів.....29	29
1.3.5 Вимоги до надійності.....30	30
1.3.6 Обмеження на установку і виконання програми.....30	30
1.3.7 Вимоги до програмної документації.....30	30
1.3.8 Порядок контролю та приймання 30	30
1.4 Постановка задач дослідження 31	31
1.5 Висновки до першого розділу 32	32
2 Матеріали і методи 34	34
2.1 Навчання нейронної мережі 34	34
2.2 Розробка архітектури системи 41	41
2.3 Функціональна структура програми 44	44
2.4 Висновки до другого розділу 46	46
3 Основні рішення з реалізації компонентів системи..... 48	48
3.1 Розробка класів Desktop-застосунку..... 48	48
3.2 Розробка графічного інтерфейсу користувача 52	52
3.3 Тестування програми..... 60	60
3.4 Висновки до третього розділу..... 63	63
4 Керівництво програміста..... 64	64
4.1 Призначення та умови застосування програми..... 64	64

4.2 Характеристики програми.....	64
4.3 Початкові та вихідні дані	64
4.4 Функціонування програми	65
4.5 Повідомлення програмісту.....	68
5 Керівництво оператора.....	69
5.1 Призначення та умови застосування програми.....	69
5.2 Виконання програми.....	69
5.3 Повідомлення оператору	75
Висновки	76
Перелік джерел посилання.....	77
Додаток А Приклад навчальної вибірки для III-го рівня тренуваності оператора.....	80
Додаток Б Текст програми	84
Додаток В Копії слайдів презентації	113

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

UML	– уніфікована мова моделювання;
WPF	– Windows Presentation Foundation;
АСУТП	– автоматизована системи управління технологічним процесом;
ГВСС	– група взаємопов'язаних подій;
ЛМІ	– людино-машинний інтерфейс;
ЛМС	– людино-машинна система;
СФО	– стандартні функціональні одиниці;
ЛМІ	– людино-машинний інтерфейс.

ВСТУП

З підвищенням рівня автоматизації та комп'ютеризації системи диспетчерського управління стають все більш поширеними у різних сферах бізнесу. Система диспетчерського управління складається з одного або кількох осіб-операторів, що взаємодіють з автоматизованими компонентами, такими як компоненти супутникового наземного управління, гнучкі виробничі системи або атомні електростанції. Оператори зазвичай виконують завдання, які потребують інтенсивного мислення, такі як моніторинг, планування, контроль у реальному часі та усунення несправностей, і зрештою несуть відповідальність за безпечну та ефективну роботу всієї системи.

Хоча розробки в галузі диспетчерського управління призвели до створення корисних застосунків у галузі проектування людино-машинних інтерфейсів (ЛМІ) та автоматизації, існує досить мало досліджень, присвячених питанням кількісної оцінки прийняття рішень людиною в диспетчерському управлінні.

Багато реальних АСУТП складні, динамічні і можуть мати низку невизначеностей. Люди-оператори працюють з комп'ютеризованими процесами при моніторингу стану системи та часто беруть участь у переплануванні, усуненні несправностей та завданнях диспетчерського управління. Під час виконання завдання як люди, так і комп'ютеризовані процеси мають свої сильні сторони та обмеження.

Одним із найважливіших завдань сучасних АСУТП є прогнозування розвитку аварійних ситуацій з урахуванням можливостей людини-оператора. У контексті динамічного управління, актуальним завданням є розрахунок часу виявлення оператором причин несправностей, що виникли, за допомогою засобів ЛМІ та усунення цих причин.

Дослідження в галузі автоматизації, орієнтованої на людину, в авіації, наземному супутниковому управлінні та управлінні атомними електростанціями призвели до розробки загальних рекомендацій щодо

проектування систем, що включають людські та комп'ютеризовані процеси у диспетчерському управлінні. Однак такі проблеми, як збільшення кількості людських помилок, нестача ситуаційної поінформованості та непрозорість через погано автоматизовані системи, залишаються, особливо у сценаріях, коли оператори повинні приймати рішення в умовах обмеженого часу [1].

Таким чином, актуальним є дослідження та програмна реалізація методів моделювання діяльності оператора АСУТП пов'язаною з ліквідацією аварійних ситуацій.

Предметом дослідження даної дипломної кваліфікаційної роботи магістра є діяльність проектувальника ЛМІ АСУТП.

Мета роботи – підвищення ефективності процесу проектування ЛМІ АСУТП шляхом розроблення і застосування програмного забезпечення на основі методів і моделей штучного інтелекту для аналізу діяльності операторів АСУТП.

Для досягнення поставленої мети у даній роботі необхідно вирішити такі задачі:

- проаналізувати методи кількісної оцінки діяльності операторів АСУТП та обрати метод, що відповідає цілям дослідження;
- розробити технічне завдання на програмний застосунок, що автоматизує діяльність проектувальника ЛМІ;
- розробити датасет для нейронної мережі;
- проектування та навчання нейронної мережі;
- розробити архітектуру системи;
- розробити моделі програмного забезпечення застосунку;
- розробити програмне та інформаційне забезпечення компонентів системи, що автоматизує діяльність проектувальника ЛМІ;
- перевірити адекватність моделі на якій базується розроблене програмне забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз методів моделювання діяльності оператора

Моделювання та допомога операторам у середовищах диспетчерського управління є необхідними передумовами для ефективного використання автоматизації у складних динамічних системах. Тому спостерігається тенденція до суттєвого збільшення використання засобів автоматизації в управлінні великими складними динамічними системами [2]. Впровадження автоматизації призводить до зміни ролі людини-оператора у таких системах. Він може виконувати різні функції та ролі, такі як: управління відповідно до нормативних інструкцій; бути включеним як елемент у петлю управління; здійснювати аналіз інформації, що надходить від контролера-супервізора, який відповідає за управління одним або декількома контролерами, що виконують рутинний контроль функцій технологічного процесу, що часто зустрічаються.

У системах диспетчерського управління роль людини-оператора насамперед пасивна і полягає у спостереженні за змінами у станах системи. Однак пасивна поведінка оператора змінюється на активну участь у випадках виявлення несправностей у контрольованих технологічних процесах. Вікенс [3] зазначає, що хоча ця дихотомія дещо перебільшена, чітко розрізняються дві основні функції оператора: нормальний моніторинг та точне налаштування системи; своєчасне виявлення, діагностика та прийняття рішень у випадках нечастих несправностей. Таким чином, моделі поведінки людини в таких системах разом з операторськими інтерфейсами повинні відображати зміни в ролі та функції оператора.

У разі зміни алгоритмів роботи оператора (зміни його ролей) необхідно враховувати, що моделі прийняття рішень операторами повинні відображати обидві ролі оператора – як активну, так і пасивну. Такі моделі представляють оператора, його рішення та дії в контексті його роботи у складі складної динамічної системи, стан якої зазвичай модифікуються автоматизованою системою управління. Ці моделі часто є більш складними, оскільки вони

повинні представляти формальні засоби для опису як цілей операторської діяльності, так і пов'язаних з ними функцій, процедур, завдань та дій [4].

Крім того, дизайн операторських інтерфейсів для систем з високим ступенем автоматизації також має бути відповідним. ЛМІ для високоавтоматизованих систем мають забезпечувати інформацію високого рівня для точного моніторингу та швидкого виявлення несправностей. За потреби екранні форми ЛМІ повинні надавати оператору більш докладну інформацію про аварійну ситуацію, щоб швидко і точно визначити причину та усунути несправність.

Деякі дані [5], [6] свідчать, що інформаційні моделі у таких інтерфейсах повинні мати ієрархічну структуру. У цьому випадку інформація про поточну ситуацію надається оператору в системі на кількох рівнях абстракції. Такі ЛМІ також мають і розширені можливості, що дозволяють оператору маніпулювати інтерфейсом, щоб "збільшити" (конкретизувати) дані для більш детального аналізу та перевірки на певних рівнях.

Наголошується [3], що в цій галузі переважає концептуальний етап моделювання, але є гостра необхідність у доопрацюваннях та розвитку як у теоретичному, так і в емпіричному напрямках.

Існує ряд вже відомих та нових технологій, які проектувальники можуть використовувати для створення робочих місць оператора, наприклад, кольори, інформаційну графіку, віконні системи, піктограми, голосове введення та висновки, когнітивне подання інформації [7]. Однак досить мало інформації про те, як реалізувати технології людино-комп'ютерної взаємодії під час роботи оператора в умовах обмеження часу.

Одним із відомих методів формального опису алгоритмів діяльності операторів у людино-машинних системах є узагальнений структурний метод А.І. Губінського [8]. Цей метод використовується для опису, проектування та оцінки якості дискретних процесів функціонування ЛМС.

Суть методу полягає у тому, що будь-яка дискретна діяльність оператора описується набором функціональних елементів, що зветься стандартними

функціональними одиницями (СФО). Ці елементи представляють робочі операції (дії) та логічні умови, що їх пов'язують. Кожна СФО характеризується набором показників. Основними показниками є вірогідність безпомилкового виконання завдання (надійність) і час виконання завдання (математичне очікування та середньоквадратичне відхилення).

Набір таких СФО утворює блоки, з яких будуються моделі процесу функціонування ЛМС. Функціональна модель ЛМС – це функціонально-семантична мережа, що представляє собою стохастичну модель, схожу на модель PERT, але на відміну від неї враховує також ймовірність можливих випадкових результатів - помилок людини та відмов машин. Найбільш поширені блоки називають стандартними функціональними структурами (СФС). Для СФС існують аналітичні вирази, що пов'язують показники надійності та часу виконання СФС з аналогічними показниками СФО.

Шляхом послідовної «згортки» СФО в СВС, заміна СВС еквівалентною СФО та подальша згортка еквівалентних СФО в нову СВС, можна остаточно скоротити функціонально-семантичну мережу до одного елемента, чий кількісні характеристики (час і надійність) будуть еквівалентні вихідній функціонально-семантичній мережі. Таким чином буде отримана повна оцінка всього алгоритму діяльності оператора (сценарію).

Основні типи елементів, представлених у узагальненому структурному методі, представлені на рис. 1.1.

Елементи типу «Функціонери» – описують будь-які типи дій, що фактично виконуються оператором і у виконання яких потрібні ресурси: час, вартість, запаси тощо.

Елементи типу «Композиціонери» - описують будь-які види умов (n -мірні відносини, логічні функції і т. д.), які визначають логіко-функціональні відносини між операціями та ресурсами і не вимагають ресурсів для їх реалізації. Іншими словами, функціонери відповідають реальним операціям або діям людини, робочим операціям технологічного обладнання, обладнання та програмного забезпечення у досліджуваному процесі функціонування, а

композиціонери відповідають певним відносинам між операціями та логічними функціями.

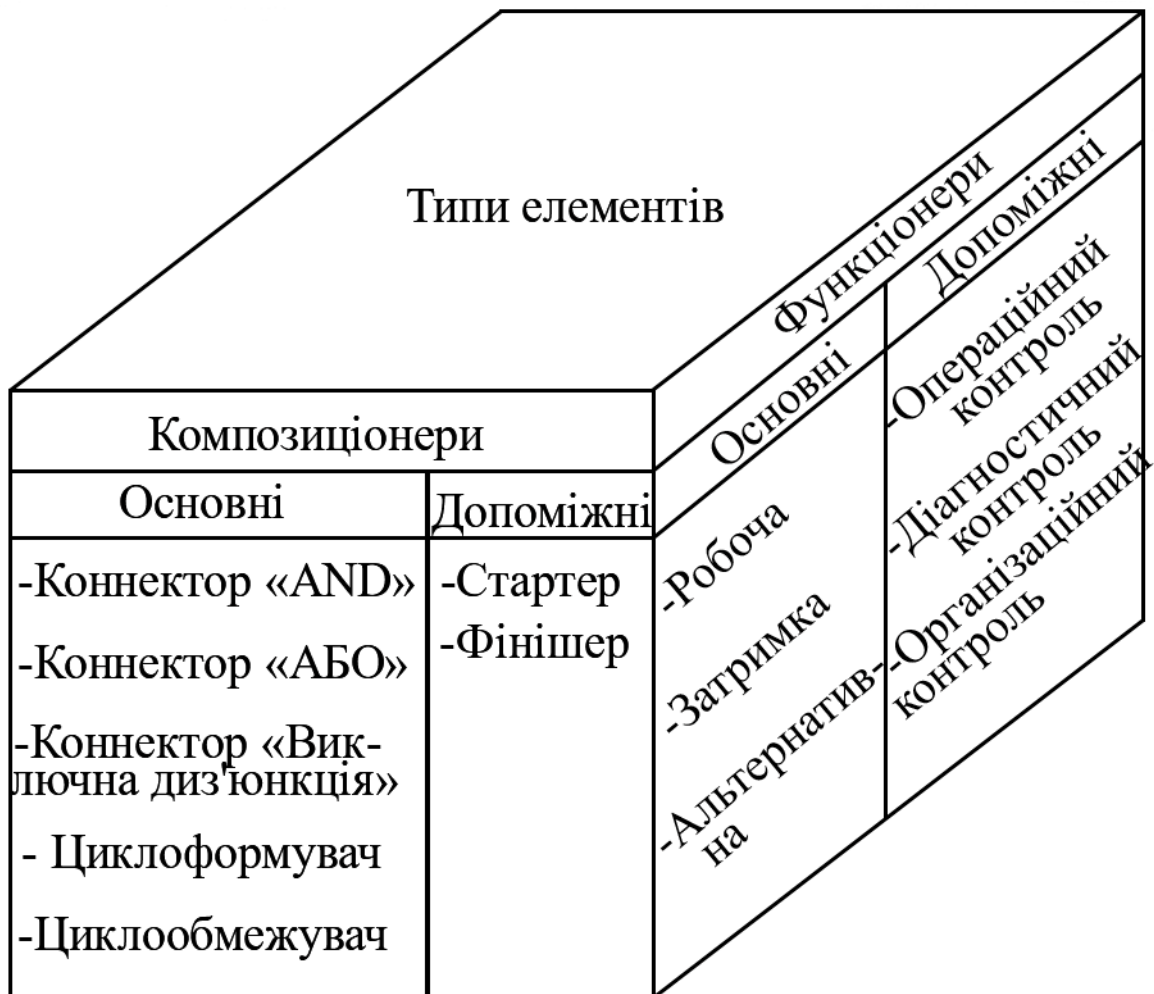


Рисунок 1.1 – Основні типи елементів

Для побудови реальної моделі функціонування процесу ЛМС з метою подальшої кількісної оцінки його показників, необхідно:

а) для кожного продукту праці, визначається певний набір показників продуктивності, якості та надійності, які, як очікується, будуть отримані в результаті оцінки, на основі наступної номенклатури показників: B^1 – ймовірність показників бездефектного продукту праці; B^0 – ймовірність виконання алгоритму діяльності з помилками; T – середній час, необхідний для

отримання продукту праці; $D(T)$ – дисперсія часу виконання алгоритму діяльності;

б) формується перелік робіт, що складають алгоритм діяльності оператора. Кожна робота має бути представлена як відповідна СФО з бібліотеки узагальненого структурного методу;

в) визначаються початкові параметри кожної СФО. Ці параметри можуть бути визначені на підставі довідкових, статистичних даних або розрахунковим шляхом;

г) будується модель процесів функціонування ЛМС у вигляді функціонально-семантичної мережі, з використанням початкових даних;

д) використовуючи відповідні дані про СВС, згорнути функціонально-семантичну мережу.

Недоліком узагальненого структурного методу є те, що його можна застосовувати лише до процесів у яких відсутня післядія та залежні операції.

У стандарті ISO/IEC DTR 9126-4 «Software Engineering – Software Product Quality. Part 4: Quality in use metrics» запроваджено метрики продуктивності, за допомогою яких рекомендується оцінювати ресурси, споживані користувачами, залежно від ефективності, досягнутої в конкретному контексті використання. Найпоширенішим ресурсом є час виконання задачі (Task time), хоча інші відповідні ресурси можуть включати зусилля, що витрачаються користувачем, матеріали або фінансові витрати на використання.

Відома багатоцільова модель COQUAMO [9], яка може прогнозувати характеристики якості в процесі розробки програмного забезпечення та надійність систем із різними версіями програмного забезпечення. Останні варіанти COQUAMO представлені трьома моделями:

- COQUAMO-1 для прогнозування якості на ранніх стадіях життєвого циклу;
- COQUAMO-2 для контролю якості у процесі розробки;
- COQUAMO-3 для виміру якості готового продукту.

COQUAMO – це статистична модель, в якій використовуються три основні показники: метрики фактора якості; драйвери якості та показники якості.

Для прогнозування якості COQUAMO-1 використовує п'ять типів драйверів якості:

- атрибути продукту (наприклад, початкові вимоги до якості);
- атрибути процесу (наприклад, використання інструменту);
- атрибути персоналу (наприклад, навички розробника);
- атрибути проекту (наприклад, початкові обмеження);
- організаційні атрибути (наприклад, допоміжне середовище).

Через труднощі з визначенням функціональних зв'язків із показниками якості та показників факторів якості, а також зв'язку між різними типами вимірювань, COQUAMO мав труднощі із забезпеченням прогнозування якості. Зазначається [9], що конструювання, калібрування та обслуговування моделей COQUAMO є трудомісткими. Виходячи з цього, вважаємо що модель COQUAMO використовувати недоцільно.

Відомий також ряд одноцільових (одноякісних) моделей, до яких включають: один з найбільш відомих методів прогнозування працездатності людини GOMS [10] і сімейство GOMS (модель рівня натискання клавіш (KLM)) [11]. GOMS використовується для прогнозування часу виконання реального завдання на основі аналізу діяльності компонентів. Дослідженню та розробці цих методів присвячено понад сто наукових праць (наприклад, [12]).

Однак, незважаючи на існування безлічі підручників, довідників та навчальних нотаток з використання методів GOMS (наприклад, [12]-[13]), вони «незвичайні ... не набули широкого поширення як інструмент для проектування в співтоваристві розробників інтерфейсів користувача» [14]. Причина цього в тому, що автори вважають трудомістким процес навчанням проектувальників цьому методу та побудови правильних моделей. Це стало приводом для розробки різних інструментів GOMS, що прискорюють процес побудови моделей. Порівняльний аналіз трьох інструментів для побудови моделей

GOMS: QGOMS, CATHCI, GLEAN3 показав [15], що вони ще не можуть діяти як повний набір інструментів для дизайнерів, і існує складність підключення до мокапу (повнорозмірної моделі будь-якого дизайну) системи [14]. Наприклад, у GLEAN3 для підключення до системи або макету розробник повинен знати мову програмування C++.

Порівняльний аналіз моделей сімейства GOMS з узагальненим структурним методом, показав наступне.

Загальні переваги: методи використовуються на ранніх етапах проектування, коли кінцевий продукт ще недоступний; дозволяє дати кількісну оцінку передбачуваного часу виконання завдання, що дає змогу порівняти альтернативи; використовують ієрархію цілей.

Загальні недоліки: не враховують кваліфікацію та особливості окремих користувачів; не враховують контекст використання (обсяг представленої інформації, її актуальність тощо).

Недоліки сімейства GOMS: обмежений набір модельних операторів, що може значно знизити точність передбачення; залежність моделі від суб'єктивної оцінки фахівця; рекомендовано для прогнозування часу виконання лише для невеликої кількості контрольних завдань, що підлягають аналізу; складність опису паралельних процесів.

Обмежений набір операторів моделі GOMS, фіксоване значення часових характеристик призвели до значних труднощів у застосуванні методу для мобільних телефонів і сенсорних пристроїв. Тому в сімействі GOMS відбувся ряд розширень, оновлень та перегляду специфікацій часу. Це дозволило досягти необхідної точності для опису взаємодії з мобільними телефонами (наприклад, [16]).

Огляд вітчизняних та зарубіжних джерел показав, що метод розрахунку часу виявлення причин порушення технологічного процесу людиною-оператором розглянуто та доведено до практичної інженерної методики в науковій праці А. І. Галактіонова [17]. Розглянемо його сутність та проведемо

порівняльний аналіз з GOMS та узагальненим структурним методом, що були розглянуті вище.

1.2 Метод розрахунку часу виявлення причин порушення оператором

Розробка методів кількісної оцінки різноманітних видів діяльності людини-оператора необхідна для того, щоб у будь-яких одиницях оцінити ступінь складності рішень окремих проблем оператором. Такі кількісні оцінки можуть бути вирішальними для прийняття проектних рішень розробниками ЛМІ.

В роботі [17] наведено результати експериментальних досліджень алгоритмів діяльності оператора під час виявлення причин порушень з урахуванням наступних чинників:

- кваліфікація оператора (рівень тренуваності оператора);
- способи пред'явлення інформації;
- інформаційна складність події (порушення);
- тип органів запиту інформації оператором;
- структура ЛМІ.

На підставі експериментів, проведених в Інституті автоматики та телемеханіки, отримано моделі процесу виявлення оператором складних подій (причин даного порушення). Кожна окрема подія може визначатися кількома параметрами, причому один і той же параметр, що контролюється може бути причиною одразу декількох подій.

Кожне порушення, несправність об'єкту технологічного процесу (основний параметр) формалізується у вигляді дерева подій (направленого графу без циклів), кожна гілка якого описує окрему складну подію. Група подій, що об'єднана у одну групу названа автором групою взаємопов'язаних подій (ГВСС).

Недоліком методу є те, що для кількісних розрахунків необхідне виконання наступних умов [17]:

- має бути визначено закон розподілу суб'єктивних ймовірностей подій чи їх ознак;
- має бути визначено якими значимістями подій чи ознак користується оператор під час вирішення завдань;
- мають бути дотримані умови закону великих чисел, стаціонарності та інші умови;
- має бути відомим закон розподілу суб'єктивних значимостей подій та їх ознак.

Згідно з [17] час виявлення причин порушення не є самостійною величиною, а складається з ряду інших складових значень, а саме:

- початковий час – величина, яка прямо пропорційно залежить від інформаційної складності події (відповідно до теорії інформації представляє собою кількість «інформації», яка міститься в певній події, що являє собою порушення і ланцюг його ознак) і від рівня тренуваності оператора;
- вплив методу контролю параметрів (автоматичний чи запит деяких параметрів оператором). Визначається в секундах і представляє собою час, який витратить оператор на з'ясування значення параметрів залежно від методу контролю цих параметрів;
- вплив кількості ірелевантної (неактуальної, непотрібної у даній ситуації) інформації, що пред'являється оператору на індивідуальних засобах відображення інформації;
- вплив типу засобів представлення інформації. Ця величина пов'язана з приростом часу виявлення події за рахунок необхідності зчитування показів приладу і за рахунок знаходження приладів, що позначають відхилення параметру, що контролюється від норми;
- вплив часу знаходження серед загальної кількості однотипних вимірювальних приладів в ЛМІ потрібного приладу.

Далі будуть розглянуті формули визначення цих величин і розрахунки (у п. 2.1) на прикладі ГВСС (рис. 1.2) з 5 подіями (вершини #49, #48, #11, #12, #16 відповідно). Розглянемо подію №4 (Досягнуто верхню припустиму межу матеріалу в бункері), що складається з чотирьох вершин (#1, #2, #9, #11).

Основним параметром, що є ознакою події №4 є $H_3 \nabla$ – «Вихід за нижню припустиму межу току навантаження електродвигуна дробарки», що відображається електромеханічним пристроєм з сильним демпфуванням. Як випливає з рис. 1.2, оператор має уточнити причину цього шляхом уточнення другорядних ознак (вершини #9, #10).

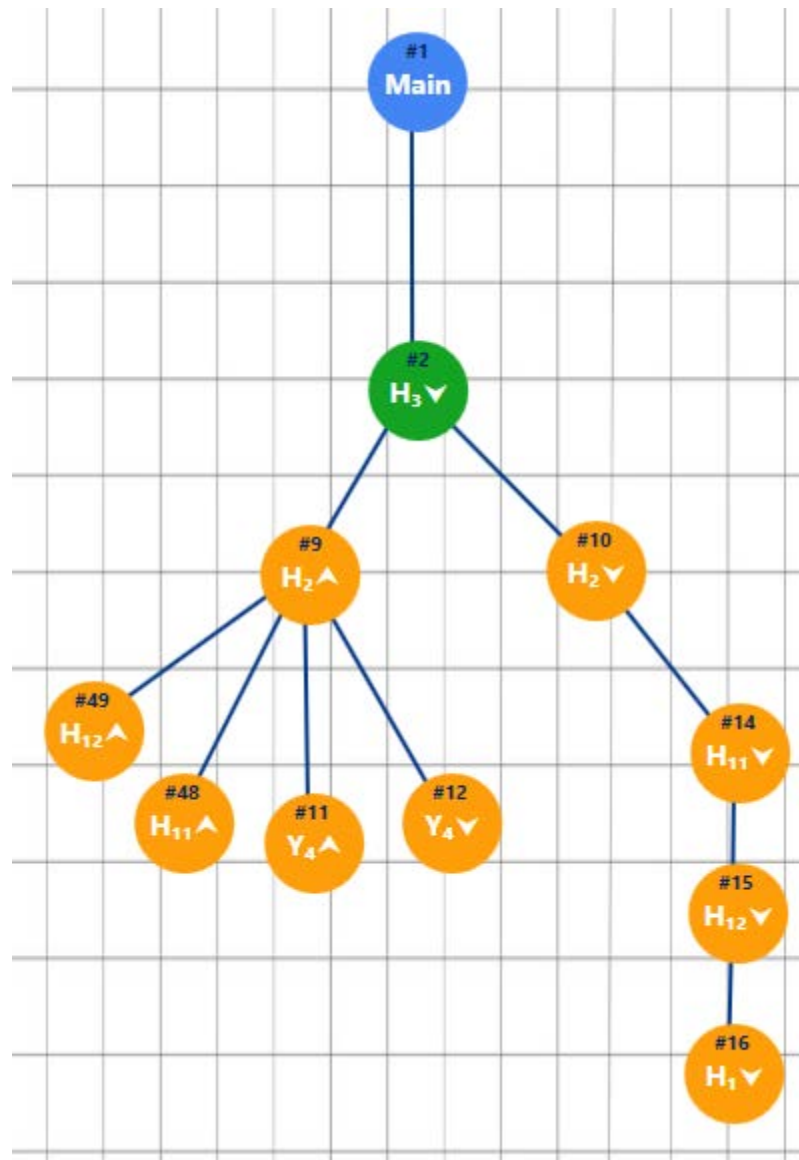


Рисунок 1.2 – Приклад ГВСС

У нашому випадку (для події №4) ознака #10 далі не розглядається. Другорядна ознака  («Верхня припустима межа току навантаження електродвигуна транспортеру 3), у свою чергу, потребує від оператора перевірки чотирьох інших ознак (вершини #49, #48, #11, #12), де вершина #11 і є подією №4 («Досягнута нижня припустима межа матеріалу в приймальному бункері»).

Розглянемо розрахунок складових часу виявлення причин порушення.

Визначення вихідного часу виявлення причин порушення здійснюється за формулою [17]:

$$T_{\text{вп}} = a + b I_{\text{под}_i}, \quad (1.1)$$

де i – номер події;

a , b – коефіцієнти, що визначаються в залежності від рівня тренованості оператора;

$I_{\text{под}_i}$ – частинна інформація, пов'язана з подією i .

Розрізняють наступні рівні тренованості оператора:

– I-й рівень відповідає випадку, коли оператор не визначив алгоритму, співпадаючого з структурною схемою ГВСС, але він здатен вирішувати задачі по виявленню складних події. На цьому рівні працюють оператори тих об'єктів, де ситуація постійно змінюється (у застосунку, що буде розроблятися даний рівень не підтримується);

– II-й рівень відповідає випадку, при якому у ході навчання оператор визначає для себе алгоритм, який співпадає зі структурною схемою подій, але оператор ще не знає ймовірностей параметрів, які контролюються. Цей випадок є характерним для тих об'єктів, де частоти виходу за норму параметрів не постійні у часі, чи коли є велика кількість параметрів, пов'язаних з можливими порушеннями, а також їх причинами і оператор не здатен їх запам'ятати і впорядкувати;

– III-й рівень, це такий рівень тренуваності оператора, при якому він визначає для себе алгоритм і познає закон розподілу ймовірностей основних параметрів, але він ще не знає ймовірностей виникнення допоміжних параметрів;

– IV-й рівень, це такий рівень, при якому оператор визначив для себе алгоритм, співпадаючий зі структурною схемою. Крім того, він знає ймовірності виникнення порушень і ймовірності появи частини подій (причин порушень) або ймовірностей виходу з норми частини допоміжних параметрів. На такому рівні працюють оператори транспортних систем, невеликих виробництв, або об'єктів з невеликою кількістю подій середньої складності;

– V-й рівень, це рівень роботи добре навчених операторів, що визначили алгоритм вирішення задач, а також пізнали всі ймовірності виходу з норми основних і допоміжних параметрів. Такі оператори використовують у якості допоміжного інструменту «образи» деяких подій («фігури», що складаються з індикаторів ЛМІ, що відображають подію);

– VI-й рівень відповідає найвищому рівню навченості, коли оператор у більшості випадків використовує образи подій. Цей рівень є характерним для невеликих об'єктів з невеликим числом подій (у застосунку, що буде розроблятися даний рівень не підтримується).

Тоді, формули для визначення вихідного часу, з урахуванням рівня тренуваності оператора наступні [17]:

$$T_{\text{вп}}^{\text{II}} = (0,69 \div 0,82) + (0,18 \div 0,33) I_{\text{под}_i} \quad (1.2)$$

$$T_{\text{вп}}^{\text{III}} = (0,58 \div 0,69) + (0,09 \div 0,18) I_{\text{под}_i} \quad (1.3)$$

$$T_{\text{вп}}^{\text{IV}} = (0,46 \div 0,58) + (0,03 \div 0,09) I_{\text{под}_i} \quad (1.4)$$

$$T_{\text{вп}}^{\text{V}} = (0,42 \div 0,46) + (0,025 \div 0,03) I_{\text{под}_i} \quad (1.5)$$

Параметр $I_{\text{под}_i}$, або іншими словами інформаційна складність події визначається за формулою визначеною у математичній теорії інформації [17].

$$I_{\text{под}_i} = -\log_2 p(a_k)_i^{(j)} + \sum_{j=2}^n \log_2 \{m\}_i^{(j)} \quad (1.6)$$

де a_k – ознака з номером k ;

$\{m\}_i^{(j)}$ – число ознак в умовній множині j -го ступеню, до якої відноситься відповідна ознака i -ї події;

p – відносна ймовірність появи ознаки a_k в умовній множині j -го ступеню $\{m\}_i^{(j)}$.

Таким чином, інформаційна складність залежить від кількості технологічних ліній, що використовуються у технологічному процесі, від кількості допоміжних ознак порушень, а також від загальної кількості ознак на певному рівні піддерева ознак.

Другою складовою при визначенні часу виявлення причин порушення є спосіб контролю параметрів. Тобто, час який витратить оператор на визначення значень параметрів залежить від способу контролю цих параметрів. Ця величини не дорівнює 0, якщо в події присутні параметри, для отримання значень яких оператор повинен зробити запит, наприклад за допомогою телефону. Обчислюється цей час за наступною формулою [17]:

$$\Delta T_{\text{вп.з}} = (t_1 + t_2) n, \quad (1.7)$$

де t_1 – час запиту одного параметру, в залежності від способу розміщення органу запиту і його типу (таблична величина);

t_2 – час зчитування значення параметру з приладу (таблична величина). Для сигналізаторів цей параметр має значення «0»;

n – кількість однакових приладів, для отримання значення яких необхідно зробити запит, у події.

Третьою складовою при визначенні часу виявлення причин порушення є вплив кількості іррелевантної інформації. Приріст часу виявлення подій при наявності $k_{\text{и}}$ іррелевантних сигналів визначається за формулою [17]:

$$T_{\text{вп}}^{\text{іpp}} = T_{\text{вп}} + 0,2k_{\text{іpp}} \quad (1.8)$$

Кількість іррелевантних сигналів $k_{\text{іpp}}$ можна приблизно розрахувати за такою формулою [17]:

$$k_{\text{іpp}} = 0,075 l \quad (1.9)$$

де l – позначає число ознак на даній ділянці об'єкту, яка просторово або логічно визначена на засобах відображення інформації ЛМІ.

Крім того, час виявлення причин порушення залежить від типу засобів відображення інформації, тому що у оператора може бути необхідність у зчитуванні показів різних приладів, що позначають відхилення від норми, а також знаходження цих приладів, серед інших, що розташовуються на робочому місці. Визначення приросту часу виявлення причин порушення у залежності від типу засобів представлення інформації розраховується за формулою [17]:

$$\Delta T'_{\text{вп.зв}} = nt \quad (1.10)$$

де t – час зчитування значення з певного приладу (засобу представлення інформації). Всі значення цього параметру є табличними довідковими даними;

n – позначає кількість приладів одного типу, що відображають ознаки даної події. Цей параметр підраховується по структурній схемі.

У складі ЛМІ оператора можуть бути присутні декілька однотипних приладів, які позначають різні параметри, що належать до різних подій. Вплив

часу знаходження серед загальної кількості однотипних вимірювальних приладів потрібного $\Delta T'_{\text{вп.п.}}$, визначається відповідно до графіку (рис. 1.3).

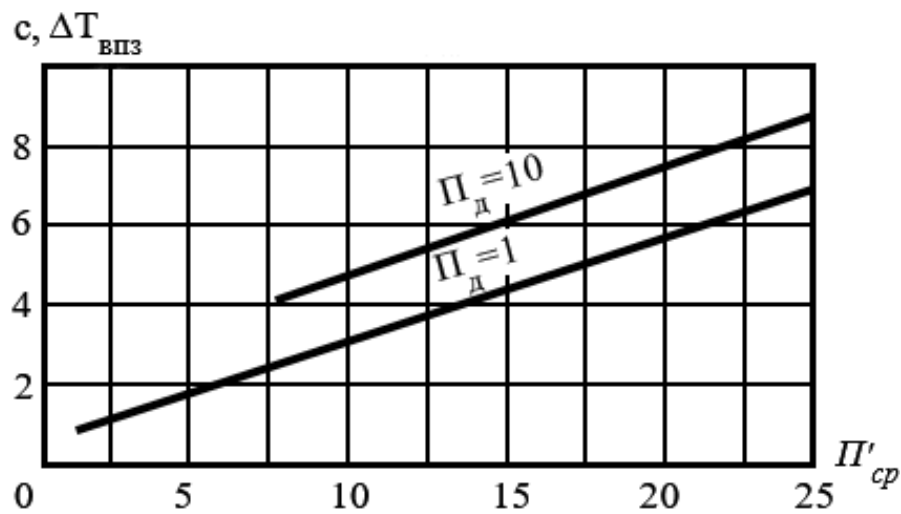


Рисунок 1.3 – Залежність вихідного часу виявлення причин на цифровому пристрої (пристроях) від числа пристроїв в умовній підмножині ознак [17]

Кінцевий час виявлення порушення $T_{\text{впк}}$ є сумою усіх перелічених вище складових, а саме: початковий час, час впливу методів контролю параметрів, час впливу кількості іррелевантної інформації, час впливу засобів представлення інформації, час, необхідний для знаходження потрібних приладів і розраховується за наступною формулою:

$$T_{\text{впк}} = T_{\text{вп}} + \Delta T_{\text{вп.з}} + T_{\text{вп}}^{\text{іпр}} + \Delta T'_{\text{вп.зв}} + \Delta T'_{\text{вп.п.}} \quad (1.11)$$

Розглянувши метод, запропонований [17] проведемо порівняльний аналіз цього методу, з іншими методами розглянутими у п. 1.1. Порівняння проведемо відповідно до наступних критеріїв:

- поширеність методу серед проектувальників ЛМІ;
- набір базових операторів, що використовуються в методі/моделі;
- складність опису паралельних процесів;

- можливість кількісного прогнозування часу виконання завдання;
- можливість використання на ранніх етапах проектування ЛМІ;
- врахування контексту використання (обсягу представленої оператору інформації);
- врахування кваліфікації оператора.

Результати порівняння методів наведено у табл. 1.1.

Таблиця 1.1 – Порівняння методів кількісної оцінки часу пошуку рішення оператором

Критерій \ Метод	Метод Губінського	Сімейство GOMS	Метод Галактіонова
Поширеність методу серед проектувальників ЛМІ	Не дуже поширений	Дуже поширений	Не дуже поширений
Набір базових операторів, що використовуються в методі/моделі	Обмежений	Дуже обмежений	Не обмежений
Складність опису паралельних процесів	Не має труднощів	Опис складний	Не має труднощів
Можливість кількісного прогнозування часу виконання завдання	Так	Так	Так
Можливість використання на ранніх етапах проектування ЛМІ	Так	Так	Так
Врахування контексту використання	Не враховує	Не враховує	Враховує
Врахування кваліфікації оператора	Не враховує	Не враховує	Враховує

Виходячи з даних табл. 1.1, найбільшими можливостями, для кількісної оцінки часу, потрібного оператору для виявлення несправностей, має метод

Галактіонова, оскільки він враховує як кваліфікацію оператора так і способи пред'явлення інформації оператору та її обсяг.

Для підвищення ефективності праці проектувальників ЛМІ, доцільно розглянути можливість навчання нейронної мережі для аналізу структур ГВСС із метою автоматизації рутинних операцій з розрахунку часу виявлення несправностей технологічного процесу.

1.3 Розробка технічного завдання

1.3.1 Підстава для розробки

Підставою для розробки служить завдання на дипломну кваліфікаційну роботу магістра на тему "Дослідження та програмна реалізація методів штучного інтелекту для аналізу операторської діяльності", затверджене наказом № 435 від 05 листопада 2021 р. по НУ «Запорізька політехніка».

1.3.2 Мета розробки та її призначення

Метою розробки є створення програмного продукту, який автоматизує діяльність проектувальників людино-машинних інтерфейсів АСУТП. Програмний застосунок призначений для аналізу операторської діяльності в умовах ліквідації аварійних ситуацій технологічного процесу.

1.3.3 Основні вимоги до програмного продукту

Розроблюване програмне забезпечення має надавати проектувальнику ЛМІ наступні функціональні можливості:

- графічне представлення структури ГВСС;
- створення нової вершини структурної схеми ГВСС у заданій точці робочої області;
- відображення зв'язків між вершинами;

- переміщення вершин ГВСС в межах робочої області;
- видалення вершини;
- відображення вершин двох типів (вершини-порушення і вершини-ознаки порушення);
- реєстрація ознак;
- редагування ознак;
- видалення ознак;
- підтримка двох режимів роботи. Перший - порушення є унікальними і реєструються окремо. Другий - порушення можуть бути ознаками інших порушень, та реєструються як ознаки;
- зчитування та відображення результатів моделювання діяльності оператора за допомогою нейронної мережі (отриманих за допомогою MatLab у форматі .asv);
- автоматичний розрахунок часу виявлення порушення;
- збереження поточної структури ГВСС;
- завантаження ГВСС з файлу;
- можливість експортувати структуру ГВСС у .png файл.

Вхідними даними, необхідними для функціонування програмного продукту, є структура ГВСС, яка представляє собою граф, де вершинами є порушення і їх ознаки.

Кожна ознака має наступні властивості: унікальна назва, спосіб контролю, засіб контролю параметрів/засіб запиту, орган запиту, метод розміщення запиту на мнемосхемі, стислий опис. Порушення мають бути визначені за допомогою таких самих параметрів, а також для них повинна бути відома частота виходу з ладу об'єктів технологічного процесу, стан яких вони описують.

Додатково повинні бути визначені наступні параметри для ГВСС: кількість технологічних ліній, рівень тренуваності оператора, спосіб

представлення інформації оператору (панель приладів, цифровий індикатор, операторна панель тощо).

Вихідними даними є орієнтовний час виявлення порушення оператором, а також: кількість інформації зв'язаної з подією (відповідно до теорії інформації вимірюється у двійкових одиницях), початковий час, час витрачений на запити, затримка (у залежності від кількості іррелевантної інформації, що надається оператору), вплив типу засобів представлення (у секундах), час витрачений на пошук засобів відображення інформації.

1.3.4 Вимоги до видів забезпечення і компонентів

Програмний застосунок, що розроблюється, має працювати під управлінням операційної системи Windows 7 або вище. Виходячи з цього, вимоги до апаратно-технічних засобів обумовлені вимогами до надійної роботи даної ОС.

Для забезпечення роботи програми та ОС необхідні наступні ресурси:

- процесор із тактовою частотою не менше 1 ГГц; для встановлення 64-бітного варіанта системи процесор повинен підтримувати 64-бітові інструкції;
- 1 Гбайт оперативної пам'яті для 32-бітових систем або 2 Гбайт – для 64-бітових систем;
- відеокарта з підтримкою DirectX 9 або новішої версії з драйвером WDDM 1.0;
- монітор (мінімум 800x600 пікселів);
- 16 Гбайт вільного місця на жорсткому диску для 32-бітних систем або 20 Гбайт для 64-бітних систем, а також 20 Мб вільного простору для програмного застосунку;
- маніпулятори вводу – клавіатура та миша.

Для реалізації програмного застосунку повинна використовуватися мова програмування C# та інтегроване середовище розробки Microsoft Visual Studio 2019.

1.3.5 Вимоги до надійності

Розроблюване програмне забезпечення повинно проводити обробку виключних ситуацій, повідомляючи оператора про некоректність введених їм вхідних даних або проведених операцій, обмежувати оператора у можливості введення вхідних даних (явно вказувати тип даних, який очікується, проводити їх валідацію).

1.3.6 Обмеження на установку і виконання програми

Обмеження на встановлення і використання програми обумовлені вимогами до апаратно-технічних засобів, описаними у п. 1.3.4.

1.3.7 Вимоги до програмної документації

Програмна документація повинна включати: технічне завдання; керівництво програміста; керівництво оператора; вихідний код програми.

1.3.8 Порядок контролю та приймання

Для верифікації розробленого програмного застосунку, програмний продукт та супровідна документація повинні пройти наступні стадії:

- тестування програмного продукту шляхом порівняння результатів розрахунків зроблених програмним застосунком з результатами відомих методів та технологій;

- затвердження працездатності продукту та достовірності супровідної документації науковим керівником;
- нормоконтроль документації до програмного продукту.

1.4 Постановка задач дослідження

Існує проблема інформаційної підтримки оператора АСУТП у складних виробничих ситуаціях, у тому числі при визначенні причин несправностей. Для вирішення цієї проблеми необхідно розпочати з відповідного аналізу причин та наслідків людської помилки. На етапі прийняття рішення в аварійній ситуації існує ряд факторів, які можуть призвести до людської помилки, якщо вони не будуть належним чином ідентифіковані. У дослідженні Діллона і Лю [18] представлено низку факторів, що є джерелом помилки людини-оператора, таких як: відсутність навичок; недостатня теоретична підготовка; погано написані інструкції з управління технічним процесом; погано спроектоване робоче місце та ЛМІ; недостатнє освітлення у робочому просторі та високий рівень шуму.

В іншому дослідженні [19] у якості основних причинних факторів названо: канали комунікації (впливають на час отримання відповідей на запити оператора); проектування та виготовлення робочого місця оператора; робоче середовище; складність роботи; документація (посібники та процедури); втома; особисті проблеми та відволікання під час роботи.

У дослідженні Латорелла та Прабху [20] було виявлено низку людських факторів, відповідальних за виникнення людської помилки, таких як: ліміти часу на ліквідацію аварійної ситуації; неадекватний зворотний зв'язок; складні умови навколишнього середовища (фактор навколишнього середовища) та загальний фактор людської помилки. Багато з цих факторів враховуються у підході Галактіонова, обраному для подальшого використання у даній роботі.

Метою даної роботи є підвищення ефективності процесів управління складними об'єктами шляхом розроблення і застосування програмного забезпечення на основі методів і моделей штучного інтелекту для аналізу діяльності операторів АСУТП.

Для досягнення поставленої мети роботи необхідно:

- провести аналіз методів кількісної оцінки діяльності операторів АСУТП та обрати метод, що відповідає цілям дослідження;
- розробити технічне завдання на програмний застосунок, що автоматизує діяльність проектувальника ЛМІ;
- навчити нейронну мережу розраховувати інформаційну складність події технологічного процесу;
- розробити архітектуру системи;
- розробити моделі програмного забезпечення застосунку (діаграми класів, компонентів та діаграму прецедентів);
- розробити програмне та інформаційне забезпечення компонентів системи, що автоматизує діяльність проектувальника ЛМІ;
- перевірити адекватність моделі на якій базується розроблене програмне забезпечення шляхом порівняння з ручними розрахунками за методом [17].

1.5 Висновки до першого розділу

Існує доволі багато методів моделювання діяльності оператора. Однак досить мало інформації про те, як реалізувати технології людино-комп'ютерної взаємодії під час роботи оператора в умовах обмеження часу. Це питання стає ще актуальнішим у випадку роботи оператора в аварійних ситуаціях. Тому дуже важливо мати можливість отримати кількісні показники діяльності оператора в аварійних ситуаціях.

Згідно з стандартом ISO/IEC DTR 9126-4 одним із важливих показників продуктивності автоматизованої системи є метрика «час виконання задачі» (Task time). Цей показник дуже важливий для оцінки можливостей оператора в умовах обмеженого часу на вирішення проблеми, що виникла у технологічному процесі.

Існує декілька методів, що дозволяють розрахувати час виявлення оператором причин порушення технологічного процесу. Їх порівняльний аналіз показав, що найбільшу точність (завдяки врахуванню кваліфікації оператора, особливостей засобів відображення інформації та врахуванню контексту використання) надає метод кількісної оцінки Галактіонова [13]. Недоліком даного методу є те, що зростання кількості можливих порушень технологічного процесу та складності виявлення причин їх виникнення призводить до різкого збільшення рутинних обчислень для проектувальника ЛМД. Тому поставлено завдання автоматизації розрахунків на основі навчання нейромережевої моделі.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Навчання нейронної мережі

Як було зазначено у п. 1.2 час, необхідний оператору для виявлення порушення, $T_{\text{впк}}$ розраховується за формулою (1.11) і є сумою п'яти складових: вихідного часу; час впливу методів контролю параметрів; час впливу кількості іррелевантної інформації; час впливу засобів представлення інформації та час, необхідний для знаходження потрібних приладів.

Найбільшу трудомісткість складає розрахунок вихідного часу $T_{\text{вп}}$, тому що в формулу його розрахунку (1.1) входить така складова як частинна інформація $I_{\text{под}_i}$, розрахунок якої і займає багато часу.

Розглянемо приклад розрахунку $I_{\text{под}_i}$ для події №4 (рис. 1.2). Згідно з формулою (1.6) маємо для випадку чотирьох однакових виробничих ліній у цеху:

$$I_{\text{под}_4} = -\log_2 1/4 + \log_2 1 + \log_2 2 + \log_2 4 = 5 \text{ (дв. одиниць)}$$

Тоді, наприклад для 5-го рівня тренованості оператора згідно з (1.5), отримаємо значення вихідного часу:

$$T_{\text{вп}} = 0,42 + 0,025 \times 5 = 0,545 \text{ с}$$

Для підвищення ефективності діяльності проектувальників ЛМІ доцільно провести навчання нейронної мережі розрахунку показника $I_{\text{под}_i}$. У якості вхідного вектору нейронна мережа отримує опис загальної структурної схеми подій (ГВСС) технологічної лінії та вірогідності ознак подій, появи порушень та вірогідності подій відповідно до свого ступеню тренованості.

Введемо наступні обмеження для структурної схеми:

- максимальна кількість вершин на першому рівні має дорівнювати 7 (тобто не більше 7 основних ознак (параметрів) порушень технологічного процесу);
- максимально припустиме розгалуження кожної вершини кожного рівня має бути не більш 7 (тобто не більше 7 варіантів допоміжних параметрів по комбінації яких визначається причина порушення);
- максимальне число рівнів структури має бути не більше 5.

Навіть з такими обмеженнями загальна структурна схема подій може описувати понад 117 000 подій.

Навчання нейронної мережі проводилося з учителем. У цьому випадку мережі пред'являються значення як вхідних, так і бажаних вихідних сигналів, і вона за деяким внутрішнім алгоритмом підлаштовує ваги синаптичних зв'язків.

Для створення навчальної вибірки були використані дані та розрахунки часу, необхідного оператору для виявлення порушення в технологічному процесі цеху цементного заводу, в якому оператор обслуговує чотири однакові лінії «дробарка-змішувач» [17]. Вхідний вектор – це опис ГВСС для 24 подій, а вихідний вектор - результати розрахунку $I_{\text{под}_i}$ (табл.13 [17]).

Експерименти по кодуванню інформації щодо опису ГВСС для формування навчальної вибірки для нейромережі, показали, що найбільш оптимальний варіант є той, що представлений у вигляді табл. 2.1.

У прикладі, що представлений у табл. 2.1 усі події рівноймовірні. Розглянемо перший рядок, що описує ГВСС події 4. Згідно даних табл. 2.1 на першому рівні, на якому розташована вершина  (#2 на рис. 1.2), знаходиться 7 основних параметрів (вершин). Параметр  є першою вершиною на цьому рівні (про це свідчить цифра «1» у 5-му стовпці). Для його уточнення згідно рис. 1.2 треба перевірити два параметра (про це свідчить цифра «2» у 6-му стовпці). Другорядна ознака  потребує від оператора перевірки чотирьох інших ознак (про це свідчить цифра «4» у 9-му стовпці). Для ідентифікації події №4 необхідно перевірити ознаку , тобто третю з

Таблиця 2.1 – Фрагмент навчальної вибірки для нейромережі

№ по- дії	№ вер- шини на рівні 1	Кіль- кість розга- лужень на рівні 1 від верши- ни рівня 0	Відно- сна ймові- рність ознаки на рівні 1	№ вер- шини на рівні 2	Кіль- кість роз- галу- жень на рівні 2 від вер- шини рівня 1	Відно- сна ймові- рність ознаки на рівні 2	№ вер- шини на рівні 3	Кіль- кість роз- галу- жень на рівні 3 від вер- шини рівня 2	Відно- сна ймові- рність ознаки на рівні 3	№ вер- шини на рівні 4	Кіль- кість роз- галу- жень на рівні 4 від вер- шини рівня 3	Відно- сна ймові- рність ознаки на рівні 4	№ вер- шини на рівні 5	Кіль- кість роз- галу- жень на рівні 5 від вер- шини рівня 4	Відно- сна ймові- рність ознаки на рівні 5	Резу- льтат $I_{\text{под}_i}$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
4	1	7	0.143	1	2	0,5	3	4	0,25	0	0	0	0	0	0	5,000
5	1	7	0.143	2	2	0,5	5	1	1	1	1	1	1	1	1	3,000

чотирьох вершин на рис. 1.2 (про це свідчить цифра «3» у 8-му стовпці). Вершини 4-го та 5-го рівнів для події №4 відсутні.

У другому рядку табл. 2.1 надано опис події №5 (вершина #16 на рис.1.2).

Приклад навчальної вибірки для випадку коли ймовірності ознак подій нерівні між собою та відомі оператору (тобто для III-го рівня тренованості оператора) наведено у табл. А.1 Додатку А. Для підвищення точності моделі для деяких подій було введено додаткові дані та розраховано значення вихідного вектору $I_{\text{под}i}$.

Наступний етап, після формування навчальної вибірки – пошук оптимальної архітектури нейромережі. Експериментальним шляхом виявлено, що оптимальним рішенням є використання двошарового персептрону з двома нейронами на першому шарі та одному на другому.

Навчання відбувалося у середовищі пакету прикладних програм Matlab з використанням мови Matlab. Код програми навчання для V-го рівня тренованості оператора наведено у Додатку Б. Розглянемо реалізацію процесу навчання персептрону.

Перш за все треба пронормувати до 1 значення всіх стовпців де вони перевищують одиницю:

```
minx=min(x);
maxx=max(x);
for i=1:1:size(x,1)
    for j=1:1:size(x,2)
        xN(i,j)=(x(i,j)-minx(j))/(maxx(j)-minx(j));
    end;
end;
```

Для створення персептрону використовуємо функцію newff(). З її допомогою створюють «класичні» багатошарові мережі з навчанням методом зворотного поширення помилки, тобто зміна ваг синапсів відбувається з урахуванням функції помилки, різниця між реальними та правильними

відповідями нейронної мережі, що визначаються на вихідному шарі, поширюється у зворотному напрямку – назустріч потоку сигналів. У нашому випадку персептрон має два приховані шари. У першому шарі 2 нейрона, у другому – 1, використовуються логістичні функції активації:

```
net = newff(repmat([0 1],size(x,2),1),[2,1],{'logsig','logsig'},'trainlm');
```

Як видно з наведеного коду, у якості навчальної функції зворотного поширення використовується (за замовчуванням) функція trainlm.

Нейромережа навчається за допомогою команди:

```
net = train(net,xN,yN)
```

Далі проведено моделювання нейромережи. Подаємо на вхід мережі вхідні значення xN, та отримуємо вихідні значення a:

```
a=sim(net,xN)
```

Для виводу значень ваги першого та другого шарів, а також для зміщення w0 нейронів першого та другого шарів використовуються наступні команди:

```
net.IW{1} %вага першого шару
```

```
net.LW{2,1} % вага другого шару
```

```
net.b{1} % зміщення w0 нейронів першого шару
```

```
net.b{2} % зміщення w0 нейронів другого шару
```

Середньоквадратична помилка розраховується як різниця між розрахованими значеннями $I_{\text{поді}}(y_N)$, та вихідними значеннями нейромережі a.

```
E=mse(yN-a)
```

Потім створюється масив перевірочних вхідних значень функції вхідні значення x1N, що відрізняється від xN. За допомогою отриманої нейромережі передбачаються значення вихідної функції y1mod:

```
y1mod=sim(net,x1N)
```

Результати навчання для III-го рівня тренуваності оператора наведено на рис. 2.1.

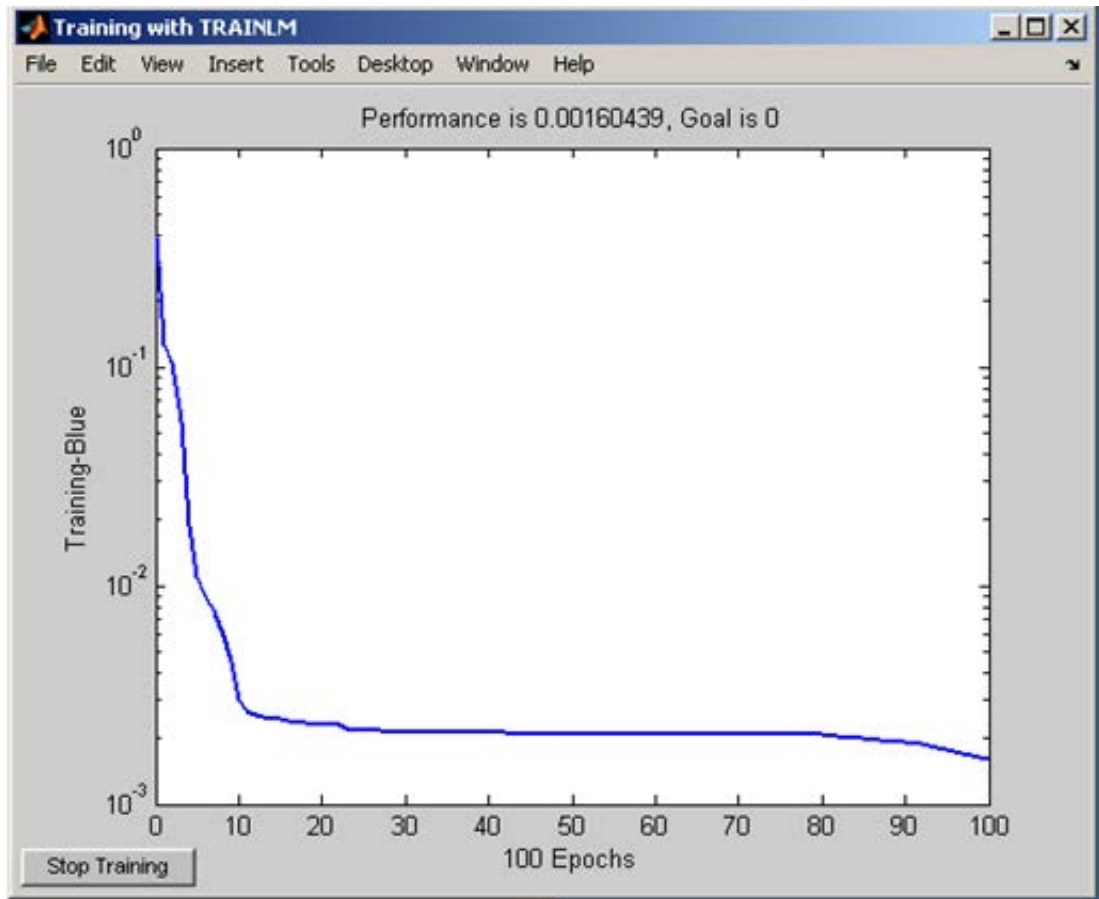


Рисунок 2.1 – Результати навчання для III-го рівня тренованості оператора

Результат навчання (див. рис. 2.1): графік показує час навчання нейронної мережі та помилку навчання. У цьому прикладі нейронна мережа пройшла 100 епох, поступово навчаючись і зменшуючи помилки, дійшла до 0,00160439.

Таким чином, вважаємо нейронну мережу навченою. Результати навчання зберігаються у файлі результатів Matlab з розширенням .asv. Для розрахунку інших компонент кінцевого часу виявлення порушення $T_{\text{впк}}$ за формулою (1.11), більша частина яких залежить від способу представлення інформації оператору на ЛМІ конкретного робочого місця, та організаційних моментів, доцільно розробити автоматизовану систему, яка б могла отримувати початкові дані про $I_{\text{под}_i}$ з файлу результатів Matlab та в інтерактивному режимі від проектувальника ЛМІ.

2.2 Розробка архітектури системи

Як вже було зазначено раніше, програмне забезпечення, що розроблюється має являти собою Desktop застосунок, функціонал якого б повністю задовольняв вимогам, зазначеним у технічному завданні.

Для проектування архітектури системи було обрано шаблон EDA (Event driven architecture) [21]. Він передбачає, що виконання програми визначається подіями – діями користувача (клавіатура, миша, сенсорний екран), повідомленнями інших програм і потоків.

Даний шаблон є дуже простим в реалізації – програмісту необхідно тільки визначити обробники для необхідних подій. Зазвичай обробниками подій є методи і функції. Через таку нескладну і гнучку структуру, шаблон EDA може бути втілений у життя за допомогою більшості мов програмування, які підтримують об'єктно-орієнтовану парадигму. Але цей шаблон має і достатньо вагомий недолік. У порівнянні з такими відомими підходами до проектування як MVVM, MVP, або MVC – використання підходу EDA дуже ускладнює майбутнє доповнення програми, її тестування і підтримку. Але саме простота реалізації нівелює цей недолік і для відносно невеликих програм EDA є найкращим вибором.

Графічне представлення архітектури шаблону EDA наведено на рис. 2.2.

Подієво-орієнтована система як правило складається з емітерів подій (Event Producer) (або агентів) і споживачів подій (Event Consumer) (або стоків). Стоки несуть відповідальність за здійснення реагування на появу події. Реакція не завжди може бути повністю забезпечена самим стоком. Наприклад, стік, може бути відповідальним лише за фільтрацію, трансформацію і відправку події до іншого компонента або він може забезпечити повністю самостійну реакцію на таку подію. Перша категорія стоків може бути заснована на традиційних компонентах, таких як проміжне програмне забезпечення, орієнтоване на обробку повідомлень в той час, як друга категорія стоків (самостійна реакція в

режимі онлайн) може вимагати більш придатної платформи (фреймворку) для виконання транзакцій[23].

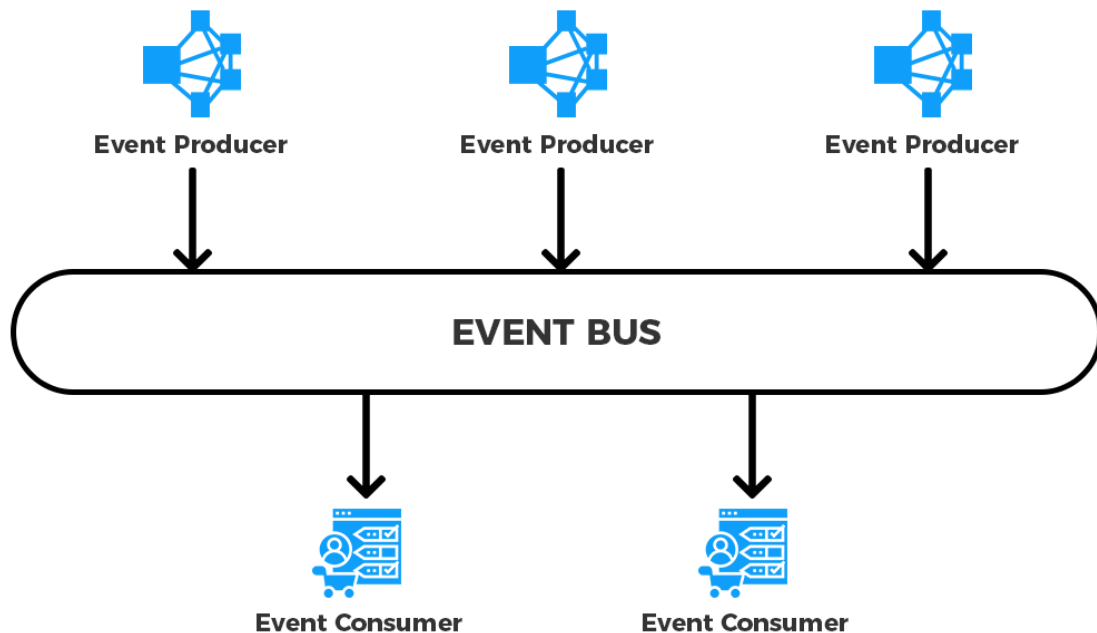


Рисунок 2.2 – Архітектура шаблону EDA [21]

Перелік модулів та компонентів, що проектуються наступний:

- рушій програми, що об'єднує всі модулі в єдину систему;
- ГВСС рушій, представлений класом GVSS (більш детально його буде розглянуто пізніше). Його основою є структура даних «граф», тобто він складається з вершин і взаємозв'язків між ними. Всі дії, які користувач може робити за допомогою цього класу – реалізовані у виді його методів;
- програма захоплення зображення структури ГВСС реалізує функціональність для захоплення зображення з робочої області програми і дає можливість користувачеві зберегти це зображення у форматі .png. Коли користувач вже повністю визначив структуру ГВСС у нього може виникнути необхідність збереження структури ГВСС у графічному вигляді для своїх особистих потреб (наприклад, щоб зробити звіт). Саме для цього даний модуль

вбудований у програму. Зображення повністю відповідає структурі ГВСС визначеній користувачем у програмі;

- програма збереження класів у бінарному форматі з можливістю подальшого відкриття збереженого раніше файлу у програмі;
- система графічного моделювання, яка дає можливість користувачеві створювати ГВСС у явному вигляді;
- модуль який дозволяє відкривати у програмі файл з результатами обчислень $I_{\text{поді}}$, отриманих за допомогою нейромережі, створеної на мові програмування MatLab.

Для візуалізації організації компонентів системи та зв'язків між ними використовуються діаграми компонентів. Вони дозволяють отримати високорівневе представлення про компоненти системи та надають допомогу в інформуванні та роз'ясненні функцій створюваної системи зацікавленим сторонам.

Модель архітектури системи, що проектується представлена у вигляді діаграми компонентів на рис. 2.3.

Desktop-застосунок

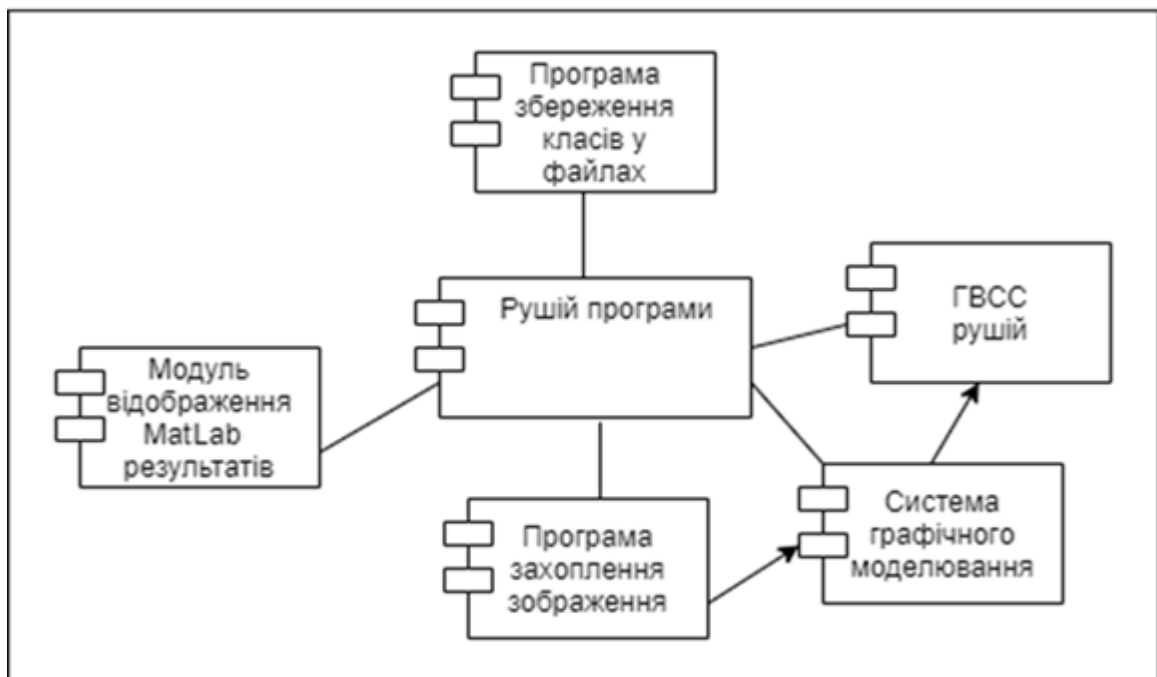


Рисунок 2.3 – Діаграма компонентів

Розглянемо більш детально взаємозв'язки між компонентами системи. Як витікає з рис. 2.3, рушій програми пов'язаний з усіма модулями за допомогою асоціації. Це означає що він пов'язує всі виконувані модулі між собою. Крім того, компонента «Програма захоплення зображення» залежить від «Системи графічного моделювання», тобто для отримання зображення ГВСС, вона повинна бути графічно визначена. З іншого боку «Система графічного моделювання» є самостійним модулем відповідно до «Програми захоплення зображення», але залежить від «ГВСС рушія», бо саме визначені у класі GVSS дані під час виконання програми відображаються на робочу область у графічному вигляді.

Як результат, отримуємо програму, яка відповідає всім вимогам технічного завдання.

2.3 Функціональна структура програми

Визначення послідовності дій користувача має велике значення під час проектування графічного інтерфейсу програмного забезпечення. Це дає змогу визначити моменти інтересу користувача до тієї чи іншої активності та наділити його функціональністю, потрібною у даний момент часу, не перевантажуючи інтерфейс. З цією метою було розроблено функціональну схему системи, зображену на рис. 2.4.

З функціональної схеми (рис. 2.4) можна побачити, що робота користувача з програмним забезпеченням поділена на п'ять етапів.

Для більш детального аналізу активностей користувача була розроблена модель у вигляді діаграми прецедентів, представленої на рис. 2.5, що описує взаємодію користувача з програмним забезпеченням без прив'язки до часу.

З цієї діаграми можна побачити взаємозв'язок окремих функцій системи, що також становить великий інтерес при проектуванні програмного забезпечення, наприклад, для вдалого вибору і розміщення елементів керування на формах.

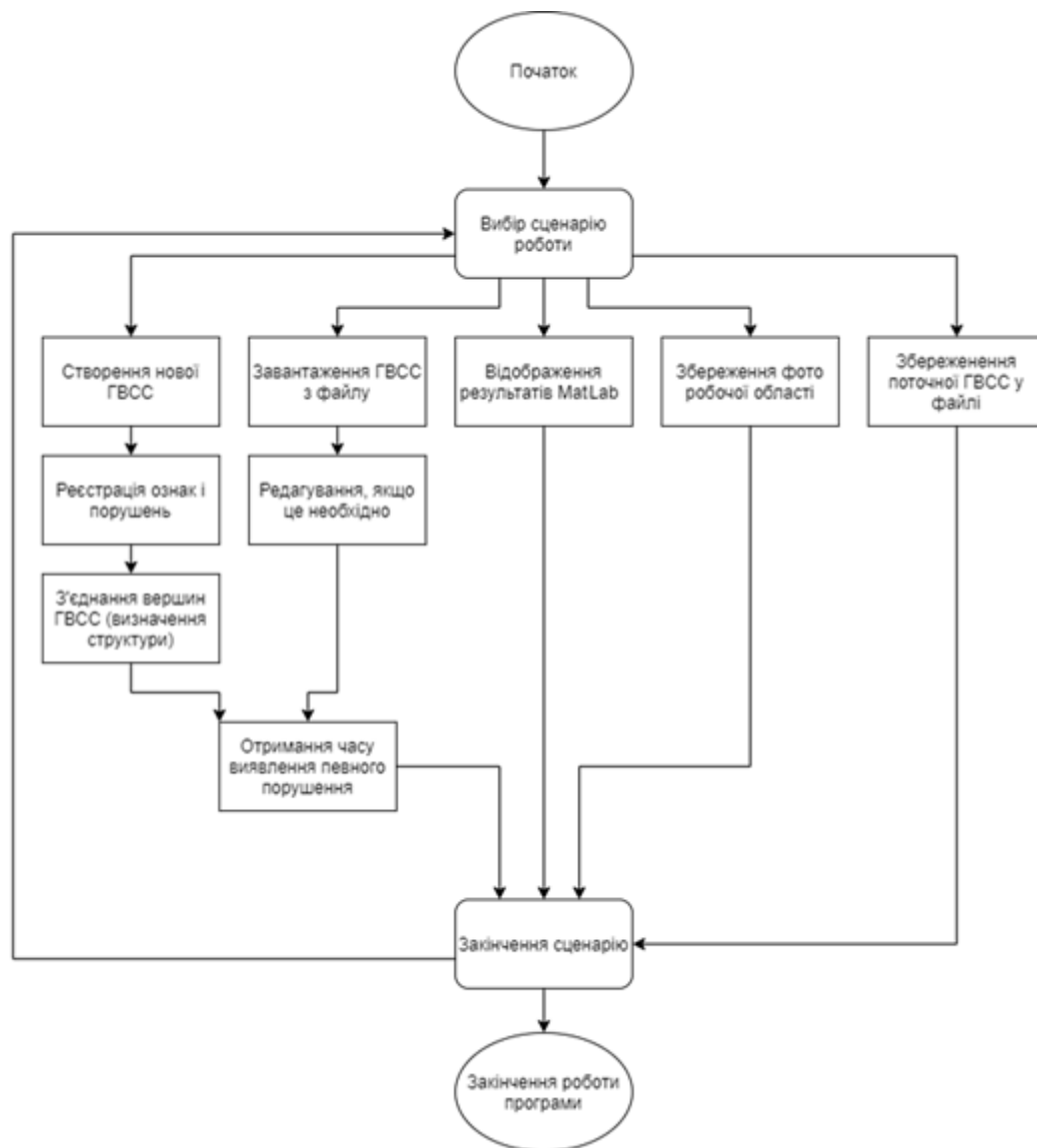


Рисунок 2.4 – Функціональна схема системи

З діаграми прецедентів також випливає, що робота зі структурою ГВСС і отримання часу виявлення порушення є двома взаємопов'язаними процесами, бо неможливо отримати час виявлення порушення не визначивши структуру порушення.



Рисунок 2.5 – Діаграма прецедентів

А ось «відображення результатів MatLab» є повністю незалежним прецедентом і щоб побачити результати користувачеві необхідно просто завантажити файл зі спеціальним розширенням.

2.4 Висновки до другого розділу

У даному розділі для підвищення ефективності діяльності проектувальників ЛМІ проведено навчання нейронної мережі розрахунку показника інформаційної складності події $I_{\text{под}_i}$, який є складовою частиною для розрахунку часу, необхідного оператору для виявлення порушення в технологічному процесі.

Навчання нейронної мережі у вигляді двошарового перцептрону проводилося з учителем з використанням навчання методом зворотного поширення помилки. Навчання відбувалося у середовищі пакету прикладних програм Matlab з використанням мови Matlab.

Було сформовано чотири навчальні виборки (по одній для кожного з чотирьох рівнів тренуваності операторів – II, III, IV та V.

Результати розрахунку навченої нейромережі є вихідними для подальшого використання автоматизованою системою для розрахунку часу $T_{\text{впк}}$.

Для проектування архітектури системи було обрано шаблон EDA який передбачає, що виконання програми визначається подіями – тобто діями користувача, повідомленнями інших програм і потоків.

Для формалізації програмного забезпечення застосунку, в даному розділі було розроблено моделі у вигляді UML-діаграм – діаграма компонентів та діаграма прецедентів.

Також було розроблено функціональну схему системи.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Розробка класів Desktop-застосунку

Оскільки зазначена в технічному завданні мова програмування C# є виключно об'єктно-орієнтованою, то все функціонування програмного забезпечення виражається у взаємодії окремих класів, сутностей предметної області.

Згідно розробленої архітектури програмного забезпечення, застосунок що розробляється містить класи, які відповідають за бізнес-логіку програми і за його зовнішній вид (інтерфейс користувача).

Розглянемо розроблені класи/структури, які відповідають за бізнес логіку:

- class GVSS є основним класом програми, що відповідає за формування структури ГВСС і включає методи взаємодії з нею;
- class Vertex є класом, який представляє вершину у ГВСС. Є включеним у клас GVSS і не доступний ззовні;
- class Edge є класом, який представляє зв'язок (ребро) між двома вершинами у ГВСС. Є включеним у клас Vertex і не доступний ззовні;
- struct Correspondence – структура, який представляє собою відповідність між певною вершиною ГВСС і ознакою, яку представляє дана вершина. Є включеною у клас GVSS і не доступна ззовні;
- class Sensor є класом, який представляє ознаку і містить всю необхідну інформацію про неї. У більшості випадків використовується класом GVSS, але є окремою сутністю.
- class Event є класом, який представляє подію (подія = порушення + ланцюг ознак). Містить повну інформацію про подію і за допомогою методів цього класу можна визначити час виявлення порушення і інші результуючі величини. Є окремою сутністю, але у більшості випадків використовується саме класом GVSS;

- class `VertexAndNumOfSubvertices` є класом, який представляє собою структуру даних, яка визначає певну вершину і кількість вершин у піддереві. Є включеним у клас `Event` і використовується ним для розрахунків, але є доступним ззовні;

- class `DefaultSettings` – статичний клас, який представляє собою збірку всіх налаштувань системи за замовчуванням: тут містяться всі розрахункові дані, а також переклади всіх написів, які представлені у графічному інтерфейсі користувача;

- class `BinarySerilization` – статичний клас, який представляє собою сервіс для збереження будь-яких класів у мові програмування C# у бінарному форматі;

- класи виключень (є спадкоємцями класу `Exception`): `InvalidArgument`, `InvalidSensor`, `UnexpectedError`, `NotFoundException`, `SensorRepeatsException` – використовуються для обробки помилок.

Для визначених класів було створено діаграму класів, зображену на рис. 3.1.

Розроблено також класи/структури, які відповідають за графічний інтерфейс:

- class `ParameterValueHelper` є статичним класом, який допомагає ініціалізувати поля з випадючими списками (комбобокси) у графічному інтерфейсі;

- class `InputCorrector` є статичним класом, який відповідає за динамічне реагування інтерфейсу на користувацьке введення даних;

- struct `ErrorWhileInput` є допоміжною структурою, яка є включеною у клас `InputCorrector` і не доступна ззовні. Використовується для специфікації помилок при введенні даних користувачем;

- class `RenderVisualService` – статичний клас, який представляє собою сервіс, що використовується для отримання зображення робочої області з ГВСС структурою у програмі;

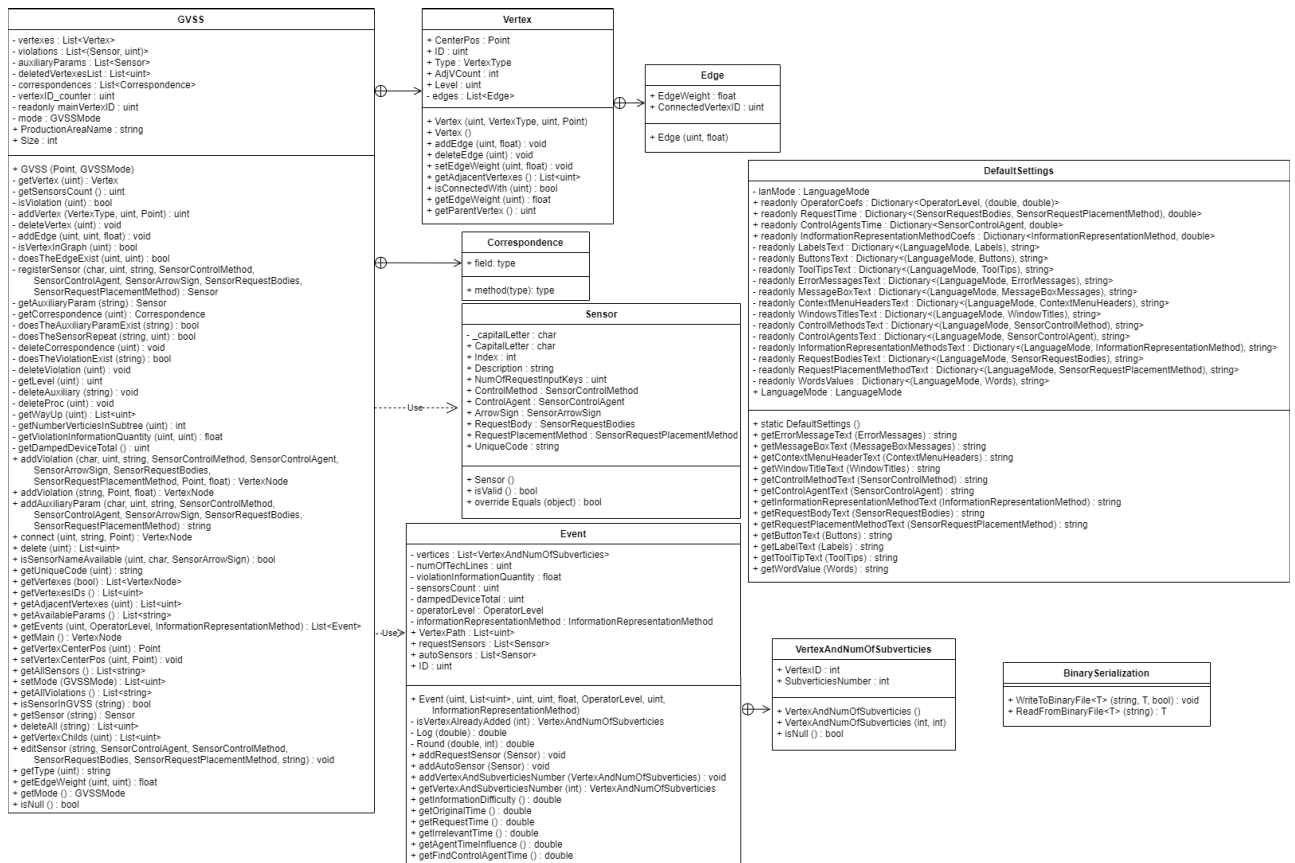


Рисунок 3.1 – Діаграма класів, що забезпечують бізнес-логіку

- class MainWindow є класом, який представляє собою головне вікно програми. Насичений функціоналом і об'єднує всі інші вікна. Відповідає також за графічне моделювання структури ГВСС;
- class EditDialog є класом, який представляє собою вікно програми, у якому користувач може змінювати певну ознаку;
- class MatLABResultWindow є класом, який представляє собою вікно з результатами, отриманими за допомогою MatLAB;
- class ShowTime є класом, який представляє собою вікно, що містить інформацію щодо отриманих результатів обчислень;
- class VertexDialog є класом, який представляє собою вікно і дозволяє користувачеві додавати вершини до структури ГВСС;
- class VertexNode є класом, який використовується як посередник між класом GVSS і іншою логікою програми (наприклад, графічним інтерфейсом). Містить в собі стислу інформацію про певну вершину;

– class Computations є допоміжним статичним класом, який використовується для генерації текстового представлення назви ознаки відповідно до переданих параметрів;

– class EventDifficulty є допоміжним класом, який використовується класом MatLABResultWindow для позначення події і часу її виявлення;

– struct Connection – допоміжна структура, яка використовується в класі MainWindow і позначає з'єднання двох вершин (ребро) ГВСС структури.

Розроблена діаграма вищезазначених класів зображена на рис. 3.2.

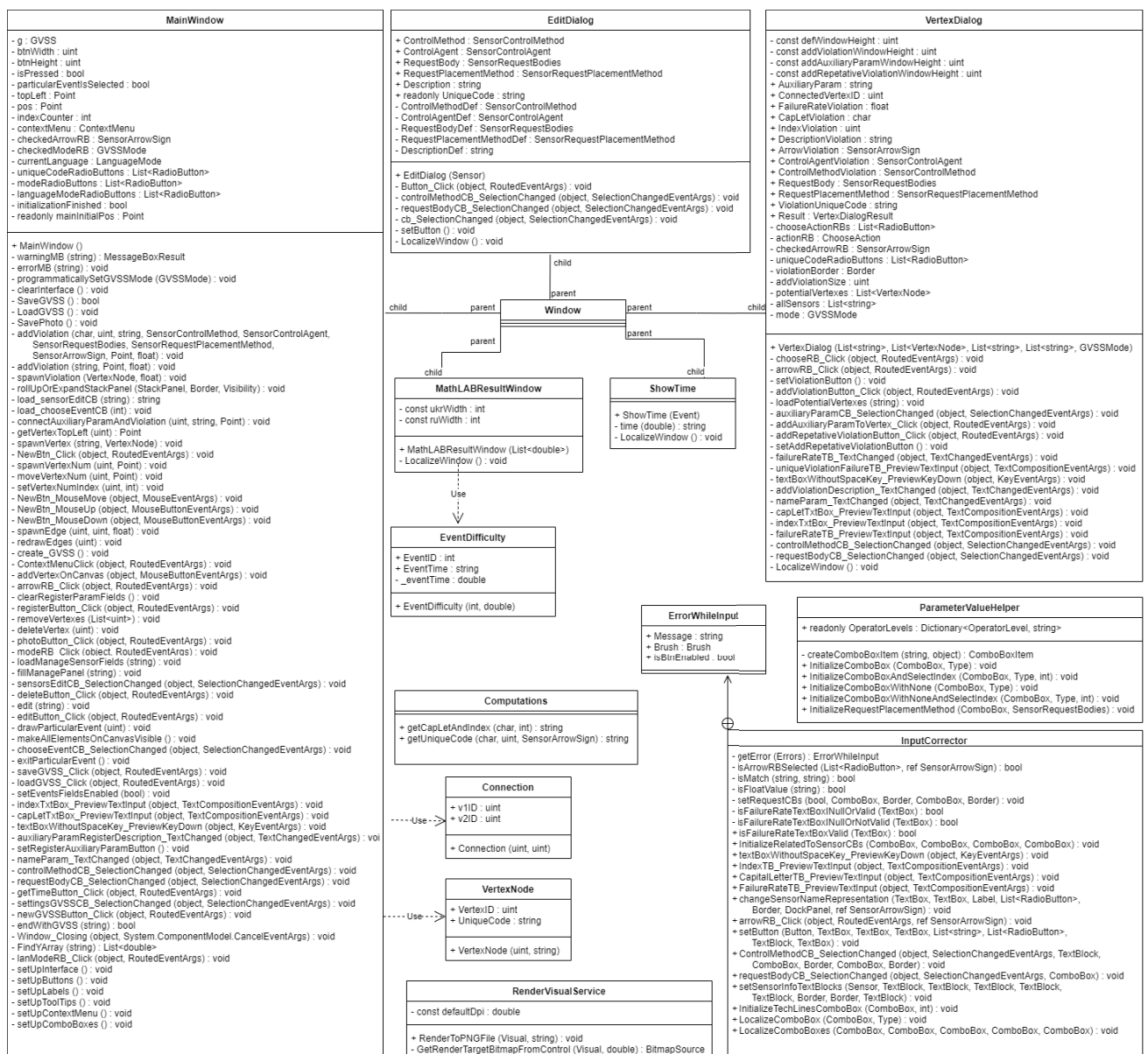


Рисунок 3.2 – Діаграма класів інтерфейсу

3.2 Розробка графічного інтерфейсу користувача

Будь який програмний інтерфейс має містити окремний контролер або групу контролерів, що є абстракцією відносно до виконуваної програмою функції. Відповідно до функцій системи, специфікованих у програмі були розроблені:

- початкова форма, що надає доступ до інших компонентів системи;
- форма додавання нової вершини на робочу область ГВСС (мається на увазі графічна зміна структури ГВСС);
- форми редагування ознаки ГВСС;
- форма з результатами програмних обчислень;
- форма з результатами обчислень MatLab.

При проектуванні форм були використані наступні елементи керування:

- кнопки;
- поля для вводу;
- поля з випадаючими списками;
- кнопки-перемикачі.

Всі форми побудовані відповідно згідно до загальних правил побудови ергономічних людино-орієнтованих інтерфейсів зазначених в стандарті ISO 9241-210:2010 «Ергономіка взаємодії людина—система. Частина 210. Людино-орієнтоване проектування інтерактивних систем».

Усі форми згідно з відомим шаблоном компоновання Visual Framework (Візуальна схема) [22], виконані з використанням одного набору кольорів, подібних шрифтів та геометричних форм елементів керування, що повторюються. Дизайн виконано у теплих синіх тонах, що створюють відчуття комфорту та ненапруженості інтерфейсу.

Розглянемо всі ці форми більш детально.

Головна форма застосунку (рис. 3.3) представляє собою вікно, розподілене на 8 блоків.

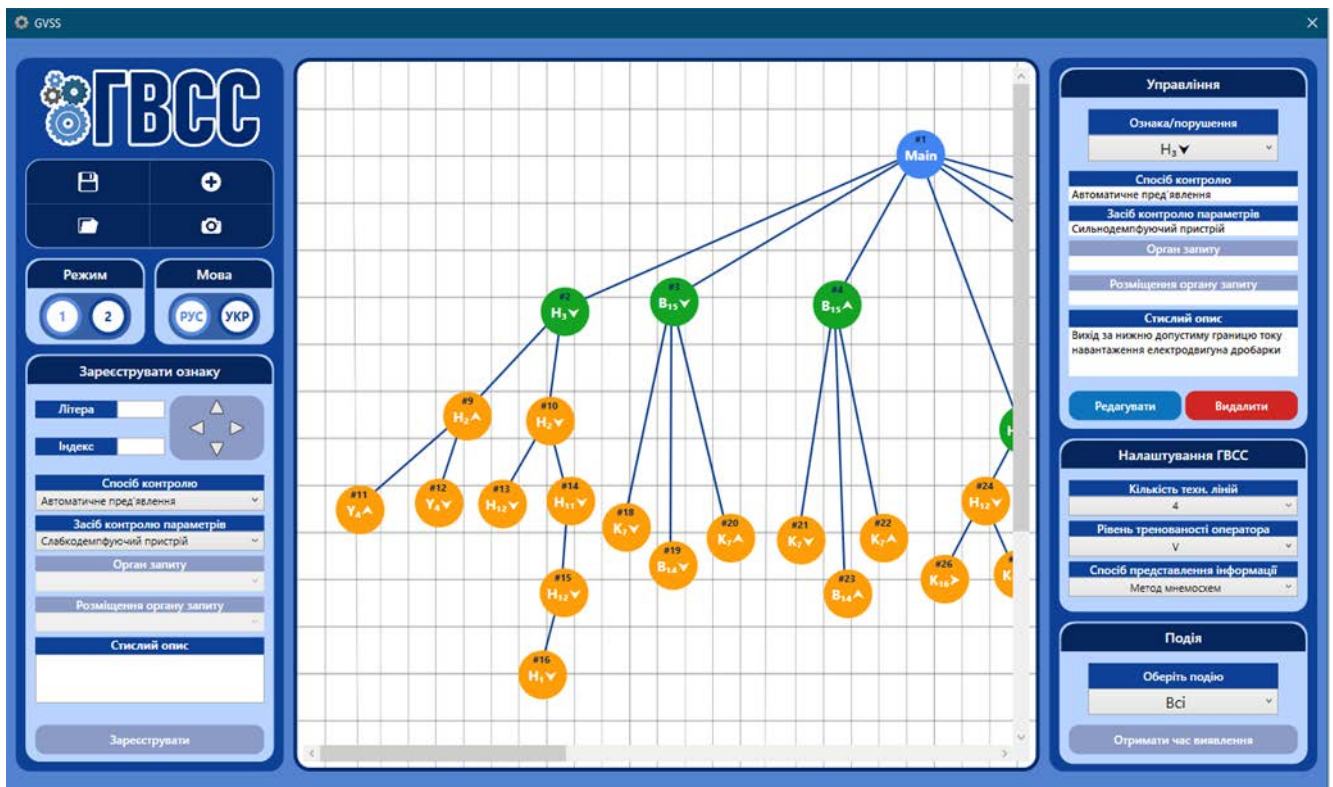


Рисунок 3.3 – Головна форма Desktop-застосунку

Перший блок (рис.3.4) – блок з кнопками управління містить чотири кнопки: «Зберегти», «Завантажити», «Створити нову ГВСС», «Зберегти фото структури».

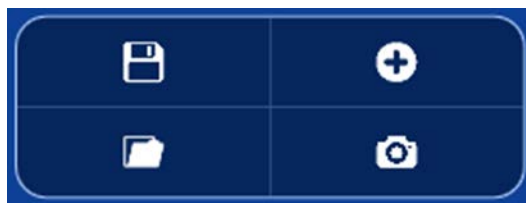


Рисунок 3.4 – Блок з кнопками управління

Другий блок (рис.3.5) – блок вибору режиму містить дві кнопки перемикавання режимів. Далі у розділі 5 буде описаний кожний з цих режимів.

Третій блок (рис.3.6) – блок вибору мови інтерфейсу містить дві кнопки перемикавання мови інтерфейсу (українська і російська).

Четвертий блок (рис.3.7 а) – блок реєстрації ознаки містить поля, які необхідно заповнити для реєстрації ознаки. У заповненому вигляді має вигляд наведений на рис. 3.7 б.

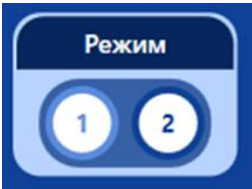


Рисунок 3.5 – Блок вибору режиму



Рисунок 3.6 – Блок вибору мови

Зареєструвати ознаку

Літера

Індекс

▲

◀

▶

▼

Спосіб контролю

Автоматичне пред'явлення

▼

Засіб контролю параметрів

Слабкодемпфуючий пристрій

▼

Орган запиту

▼

Розміщення органу запиту

▼

Стислий опис

Зареєструвати

Зареєструвати ознаку

Літера

Y

Індекс

19

▲

◀

Y₁₉

▶

▼

Спосіб контролю

Запит значень

▼

Засіб запиту (пристрій)

Слабкодемпфуючий пристрій

▼

Орган запиту

Тумблер

▼

Розміщення органу запиту

Розміщення на мнемосхемі

▼

Стислий опис

Деякий опис

Зареєструвати

а

б

а – елементи форми; б – заповнений блок реєстрації

Рисунок 3.7 – Блок реєстрації ознаки

П'ятий блок (рис.3.8) – блок графічного моделювання ГВСС, що представляє собою робочу область, на якій користувач може у явному вигляді створювати структуру ГВСС. Переміщення по ній відбувається за допомогою «скрол-барів». Кожна вершина ГВСС, що розташована у даній області підтримує контекстне меню.

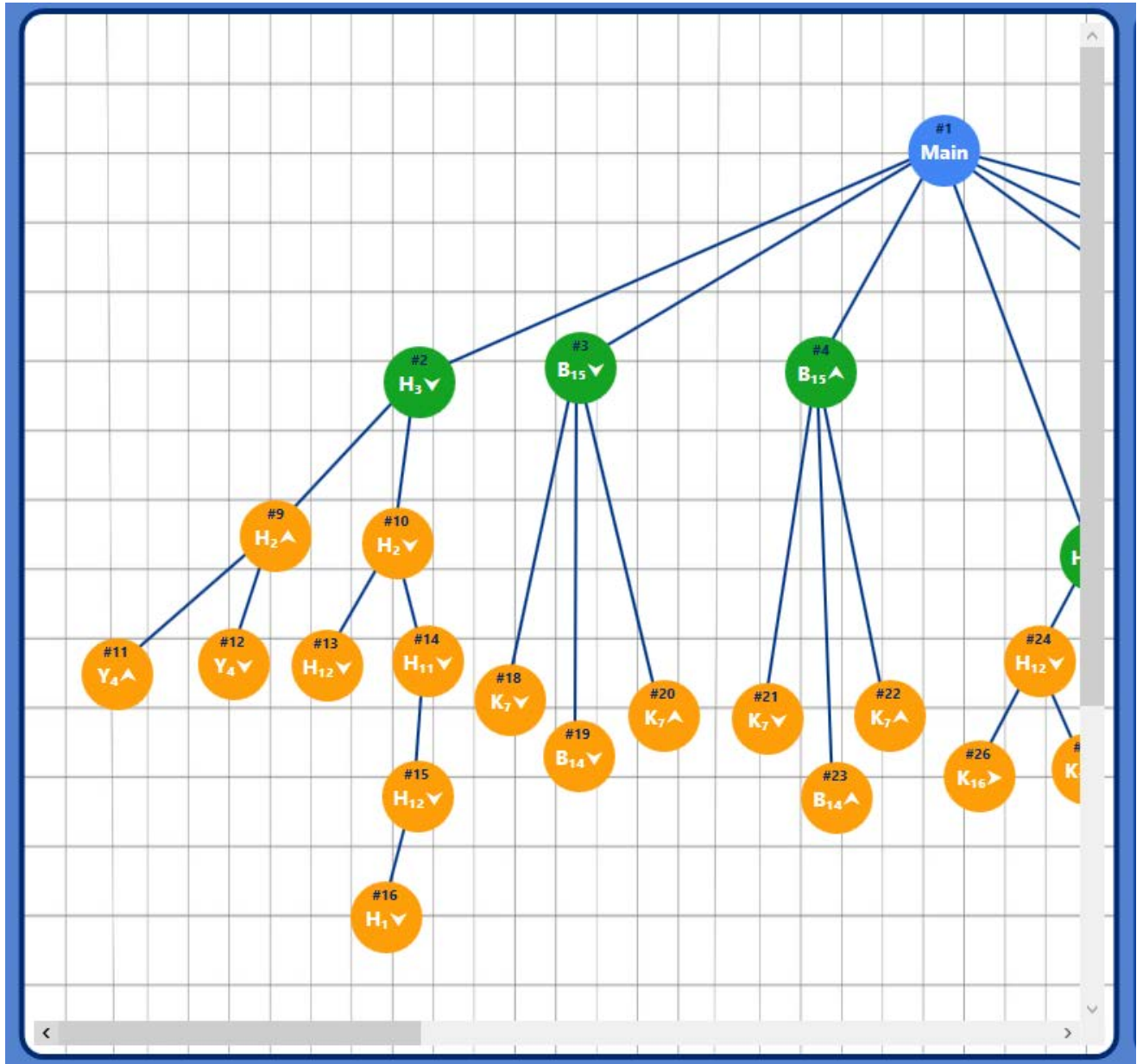


Рисунок 3.8 – Блок графічного моделювання

Шостий блок (рис 3.9) – блок управління ознаками дозволяє переглянути інформацію стосовно певної ознаки, а також викликати форму редагування або видалити ознаку.

Рисунок 3.9 – Блок управління ознаками

Сьомий блок (рис 3.10) – блок налаштування ГВСС, що містить загальну інформацію щодо ГВСС і дозволяє її редагувати.

Рисунок 3.10 – Блок налаштування ГВСС

Восьмий блок (рис 3.11) – блок подій. У цьому блоці користувач може обрати подію і отримати час виявлення порушення, яке позначає дана подія.

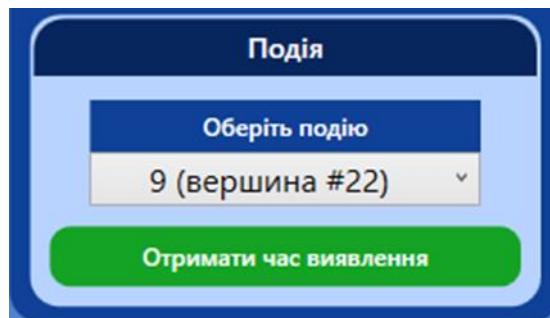


Рисунок 3.11 – Блок подій

Другою формою застосунку є форма додавання нової вершини на робочу область ГВСС (рис. 3.12). Ця форма викликається при натисканні користувачем лівою кнопкою миші на робочу область (блок графічного моделювання головного вікна).

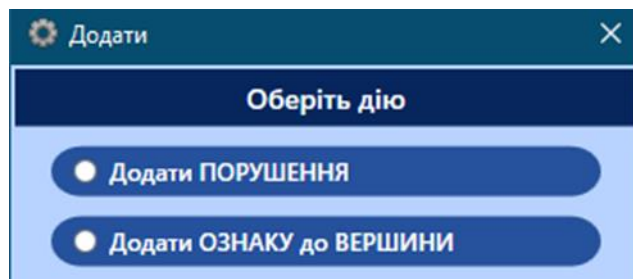


Рисунок 3.12 – Форма додавання нової вершини

Вона містить дві кнопки-перемикачі: «Додати порушення» і «Додати ознаку до порушення». Відповідно до обраної опції форма буде пропонувати певний функціонал користувачеві.

Якщо користувач обере опцію «Додати порушення» і буде активовано перший режим ГВСС, то форма буде виглядати відповідно до рис. 3.13

Якщо користувач обере опцію «Додати порушення» і буде активовано другий режим ГВСС, то форма буде виглядати відповідно до рис. 3.14

Рисунок 3.13 –Режим ГВСС №1 «Додати порушення»

Рисунок 3.14 – Режим ГВСС №2 «Додати порушення»

Якщо користувач обере опцію «Додати ознаку до вершини», то форма буде виглядати відповідно до рис. 3.15

Рисунок 3.15 – «Додати ознаку до вершини»

Третя форма – форма редагування ознаки ГВСС (рис. 3.16).

Рисунок 3.16 – Форма редагування ознаки ГВСС

Викликається при натисканні на кнопку «Редагувати» у блоці управління на головній формі, або при виборі опції «Редагувати» у контекстному меню блоку графічного моделювання. Дозволяє користувачеві змінювати властивості певної ознаки ГВСС.

Четверта форма – форма з результатами обчислень (рис. 3.17).

Подія №4	
Кількість інформації	Початковий час
5 д.е.	0.54 сек.
Час, витрачений на запити	Затримка (ір. інформ.)
5.75 сек.	0.15 сек.
Вплив засобів предст.	Пошук засобів предст.
0.4 сек.	1.8 сек.
Загальний час виявлення події	
8.64 сек.	

Рисунок 3.17 – Форма з результатами обчислень

Програмний код основних класів наведено у Додатку Б.

3.3 Тестування програми

Тестування програми проводилося шляхом порівняння результатів розрахунків показників ГВСС у автоматичному та ручному режимі. У якості прикладу, наведемо результати для події №4 (рис.1.2).

Значення $I_{\text{под}_i}$ та $T_{\text{вп}}$ було розраховано у ручному режимі у п. 2.1 і дорівнюють відповідно 5 дв. одиниць та 0,54 с.

Введемо опис події №4 в автоматизовану систему (рис. 3.18)

Ознака/порушення	Ознака/порушення	Ознака/порушення	Ознака/порушення	Ознака/порушення
H ₂ ▼	Y ₄ ▲	Y ₄ ▼	H ₁₁ ▼	H ₁₂ ▼
Спосіб контролю	Спосіб контролю	Спосіб контролю	Спосіб контролю	Спосіб контролю
Запит значень	Автоматичне пред'явлення	Автоматичне пред'явлення	Автоматичне пред'явлення	Автоматичне пред'явлення
Засіб запиту (пристрій)	Засіб контролю параметрів	Засіб контролю параметрів	Засіб контролю параметрів	Засіб контролю параметрів
Цифровий трьохрозрядний індикатор	Сигналізатор	Сигналізатор	Сильнодемпфуючий пристрій	Сильнодемпфуючий пристрій
Орган запиту	Орган запиту	Орган запиту	Орган запиту	Орган запиту
Номеронабирач				
Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту
В агрегатній групі				
Стислий опис	Стислий опис	Стислий опис	Стислий опис	Стислий опис
Нижня допустима границя току навантаження е. дв. транспортера 3	Верхній допустимий рівень матеріалу в приймальному бункері	Нижній допустимий рівень матеріалу в приймальному бункері	Вихід за нижню допустиму границю току навантаження лівого електродвигуна	Вихід за нижню допустиму границю току навантаження е. дв. правого змішувача

Ознака/порушення	Ознака/порушення	Ознака/порушення	Ознака/порушення	Ознака/порушення
H ₁ ▼	H ₃ ▼	H ₁₁ ▲	H ₁₂ ▲	H ₂ ▲
Спосіб контролю	Спосіб контролю	Спосіб контролю	Спосіб контролю	Спосіб контролю
Запит значень	Автоматичне пред'явлення	Автоматичне пред'явлення	Автоматичне пред'явлення	Запит значень
Засіб запиту (пристрій)	Засіб контролю параметрів	Засіб контролю параметрів	Засіб контролю параметрів	Засіб запиту (пристрій)
Цифровий трьохрозрядний індикатор	Сильнодемпфуючий пристрій	Сильнодемпфуючий пристрій	Сильнодемпфуючий пристрій	Цифровий трьохрозрядний індикатор
Орган запиту	Орган запиту	Орган запиту	Орган запиту	Орган запиту
Номеронабирач				Номеронабирач
Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту
В агрегатній групі				В агрегатній групі
Стислий опис	Стислий опис	Стислий опис	Стислий опис	Стислий опис
Вихід за нижню допустиму границю току навантаження електродвигуна дробарки	Вихід за нижню допустиму границю току навантаження електродвигуна дробарки	Вихід за нижню допустиму границю току навантаження лівого е. дв. змішувача	Вихід за верхню допустиму границю току навантаження е. дв. правого змішувача	Верхня допустима границя току навантаження електродвигуна транспортера 3

Налаштування ГВСС
Кількість тех. ліній
4
Рівень тренуваності оператора
V
Спосіб представлення інформації
Метод мнемосхем

Рисунок 3.18 – Приклад ГВСС, на основі якої будуть проводитися розрахунки

Розрахуємо час, який витратить оператор на визначення значення параметрів залежно від методу контролю цих параметрів. Цей час не дорівнює 0, якщо в події присутні параметри, для отримання значень яких оператор повинен зробити запит. Згідно з даних рис. 3.18 у події №4 тільки один параметр **H₂▲** з'ясовується шляхом запиту по телефону (використовується номеронабирач).

Для випадку 3-х значного номеру, розраховуємо першу складову t_1 формули (1.7) з урахуванням того, що час набору однієї цифри є 1,8 с [17]:

$$t_1 = 1.8 \times 3 = 5.4 \text{ с},$$

Дані зчитуються з цифрового приладу, тому друга складова t_2 формули (1.7) є 0,35 с. Тоді час, який витратить оператор на визначення значення параметру **H₂▲**:

$$\Delta T_{\text{вп.3}} = (5,4 + 0,35) \times 1 = 5,75 \text{ с}$$

Розрахуємо по формулі (1.8) вплив кількості іррелевантних сигналів з урахуванням того, що на ЛМІ логічно та просторово виділено 10 ознак, тобто $l = 10$. Тоді згідно з формулами (1.8) та (1.9) маємо:

$$T_{\text{ВП}}^{\text{іpp}} = 0,2 \times 0,075 \times 10 = 0,15 \text{ с}$$

Розрахуємо по формулі (1.10) вплив типу засобів відображення інформації. Будемо враховувати, що на технологічній ділянці є два засоби автоматичного пред'явлення інформації для вершин #2 та #11. Час зчитування з сильнодемпфуючого пристрою (для вершини #2) – 0,4 с [17], а для сигналізатора, що сповіщає про подію (для вершини #11) цей час дорівнює 0.

Тоді:

$$\Delta T'_{\text{ВП.ЗВ}} = 0,4 \text{ с}$$

Далі за допомогою графіку (рис. 1.3) розрахуємо час знаходження потрібного приладу серед загальної кількості однотипних вимірювальних приладів. У події №4 тільки одна ознака, яка автоматично пред'являється оператору – це ознака , тому $P_d=1$. Всього таких ознак у ГВСС п'ять, тобто $P_{\text{ср}}=5$. З даних графіка випливає, що:

$$\Delta T'_{\text{ВП.П.}} = 1,8 \text{ с}$$

Тоді, кінцевий час виявлення порушення згідно з формулою (1.11):

$$T_{\text{ВПК}} = 0,54 + 5,75 + 0,15 + 0,4 + 1,8 = 8,64 \text{ с}$$

Показники, що були розраховані системою наведені на рис. 3.17. Порівняння розрахунків показує їх ідентичність.

3.4 Висновки до третього розділу

У третьому розділі було розглянуто основні рішення щодо реалізації автоматизованої системи – класи, які відповідають за бізнес-логіку програми та за інтерфейс з користувачем. Було розглянуто моделі програмного забезпечення застосунку у вигляді діаграм класів (відповідно для бізнес логіки та для інтерфейсу).

Також розглянуто розробку інтерфейсу користувача для роботи з ГВСС, що складається з чотирьох форм.

Тестування системи шляхом порівняння результатів отриманих у автоматичному режимі, з результатами ручного розрахунку, підтвердило їх ідентичність.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

Дане програмне забезпечення призначене для моделювання структури ГВСС і автоматичного обчислення часу виявлення порушення оператором.

Для коректного функціонування Desktop-застосунку необхідна версія ОС Windows 7 або вище.

4.2 Характеристики програми

Програмне забезпечення відловлює можливі помилки в роботі оператора, запобігаючи від можливих помилок в їх роботі, наприклад, при спробі користувача зареєструвати ознаку, яка має не унікальне ім'я, кнопка зареєструвати змінить свій колір на сіро-синій і буде вимкненою, а також користувач побачить відповідне повідомлення червоним кольором у блоці реєстрації ознак на головній формі. Усі повідомлення, отримувані оператором під час виключних ситуацій описані у пункті 4.4.

Для запуску програмного забезпечення необхідно виконати файл GVSS.exe. Програма не потребує та не підтримує виклик з додатковими аргументами з командного рядка.

4.3 Початкові та вихідні дані

Початковими даними, необхідними для функціонування програмного продукту, є структура ГВСС, яка представляє собою граф, де вершинами є порушення і їх ознаки. Кожна ознака має такі властивості: унікальна назва, спосіб контролю, засіб контролю параметрів/засіб запиту, орган запиту, метод розміщення запиту на мнемосхемі, стислий опис. Порушення мають бути визначені за допомогою таких самих параметрів, а також для них повинна бути відома частота виходу з ладу. Додатково повинні бути визначені такі параметри

для ГВСС: кількість технологічних ліній, рівень тренуваності оператора, спосіб представлення інформації.

Вихідними даними є орієнтовний час виявлення порушення оператором, а також: кількість інформації в події (вимірюється у двійкових одиницях), початковий час, час витрачений на запити, затримка (залежність від кількості іррелевантної інформації), вплив типу засобів представлення, час витрачений на пошук засобів відображення інформації.

4.4 Функціонування програми

В програмі доволі багато класів, які визначають її роботу, структуру і зовнішній вигляд. Але рушієм програми є по більшій мірі всього два класи: GVSS і Event. GVSS – представляє так би мовити структуру даних ГВСС і визначає всі дії, які можна з нею робити. Event – клас, який уособлює певну подію у структурі ГВСС і відповідає за розрахунок часу виявлення порушення (результат роботи програми і мета, з якою вона створювалась). Тобто, кажучи простими словами, якщо ми візьмемо і дістанемо ці два класи з програми і під’єднаємо їх до консольного застосунку, то програма буде працювати так само добре.

Це означає, що взаємодія безпосередньо з програмою відбувається за допомогою інтерфейсу саме цих двох класів, а інші класи програми визначають її графічну реалізацію. Детальна структура інтерфейсу (приватні методи) цих класів наведено у табл. 4.1.

Таблиця 4.1 – Перелік атрибутів та методів класів GVSS і Event

Клас	Атрибут / Метод	Тип даних	Короткий опис
1	2	3	4
GVSS	Size	int	Розмір структури ГВСС
	addViolation	VertexNode	Додати порушення
	addAuxiliaryParam	string	Зареєструвати ознаку
	connect	VertexNode	З'єднати дві вершини
	delete	List<uint>	Видалити вершину
	isSensorNameAvailable	bool	Чи доступне ім'я ознаки для реєстрації
	getUniqueCode	string	Отримати унікальне ім'я певної вершини
GVSS	getVertexes	List<VertexNode>	Отримати список вершин
	getVertexesIDs	List<uint>	Отримати список унікальних номерів вершин
	getAdjacentVertexes	List<uint>	Отримати список суміжних вершин певній вершині
	getEvents	List<Event>	Отримати події поточної ГВСС
	getMain	VertexNode	Отримати інформацію щодо головної вершини ГВСС
	getVertexCenterPos	Point	Отримати координати вершини ГВСС

Продовження табл. 4.1

1	2	3	4
GVSS	setVertexCenterPos	Void	Визначити координати певної вершини ГВСС
	getAllSensors	List<string>	Отримати список всіх зареєстрованих ознак
	setMode	List<uint>	Змінити режим ГВСС
	getAllViolations	List<string>	Отримати список всіх порушень
	isSensorInGVSS	bool	Чи знаходиться певна ознака в структурі ГВСС
	getSensor	Sensor	Отримати ознаку за її унікальним ім'ям
	deleteAll	List<uint>	Видалити ознаку (видалити всі вершини, які представляють дану ознаку)
	editSensor	void	Редагувати ознаку
GVSS	getType	string	Отримати тип вершини
	getEdgeWeight	float	Отримати вагу ребра (в нашому випадку це частота порушення)
	getMode	GVSSMode	Отримати поточний режим ГВСС
	isNull	bool	Чи є має ГВСС порожню структуру?

Продовження табл. 4.1

1	2	3	4
Event	getInformationDifficulty	double	Отримати інформаційну складність події
	getOriginalTime	double	Отримати початковий час події
	getRequestTime	double	Отримати час витрачений на запити
	getIrrelevantTime	double	Отримати час затримки, що залежить від кількості іррелевантної інформації в системі
	getAgentTimeInfluence	double	Отримати час, який залежить від типу засобу відображення
	getFindControlAgentTime	double	Отримати час, витрачений на пошук засобу відображення

4.5 Повідомлення програмісту

Повідомлення програмісту у програмі не передбачені.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення та умови виконання програми

Дане програмне забезпечення призначене моделювання структури ГВСС і автоматичного обчислення часу виявлення порушення оператором.

Для коректного функціонування Desktop-застосунку необхідна версія ОС Windows 7 або вище.

5.2 Виконання програми

Для запуску програмного застосунку необхідно виконати файл GVSS.exe. Програма не потребує та не підтримує виклик з додатковими аргументами з командного рядка. Після запуску програми оператор потрапляє на її головну форму (рис. 3.3).

Програма реалізує певний функціонал взаємодії з ГВСС і тому у користувача можуть виникнути такі потреби:

- а) робота з системою:
 - 1) зберегти ГВСС;
 - 2) завантажити ГВСС з файлу;
 - 3) створити нову ГВСС;
 - 4) зберегти фото ГВСС;
 - 5) переглянути таблицю результатів, отриманих за допомогою MatLAB;
 - 6) змінити мову інтерфейсу
- б) робота зі структурою ГВСС:
 - 1) змінити режим ГВСС;
 - 2) змінити налаштування ГВСС;
 - 3) зареєструвати ознаку;
 - 4) додати вершину на робочу область;
 - 5) переміщення вершини;

- 6) редагувати ознаку/вершину;
- 7) переглянути інформацію про вершину/ознаку;
- 8) видалити ознаку;
- 9) видалити вершину;
- 10) отримати час виявлення порушення.

Розглянемо способи виконання даних функцій.

Для того, щоб зберегти ГВСС необхідно натиснути на кнопку «Зберегти» у блоці управління програмою на головній формі і визначити шлях для збереження файлу.

Для того, щоб завантажити ГВСС необхідно натиснути на кнопку «Завантажити» (позиція 1 на рис. 5.1) у блоці управління програмою на головній формі і визначити шлях до файлу (позиція 2 на рис. 5.1) та натиснути кнопку «Open» (позиція 3 на рис. 5.1)/

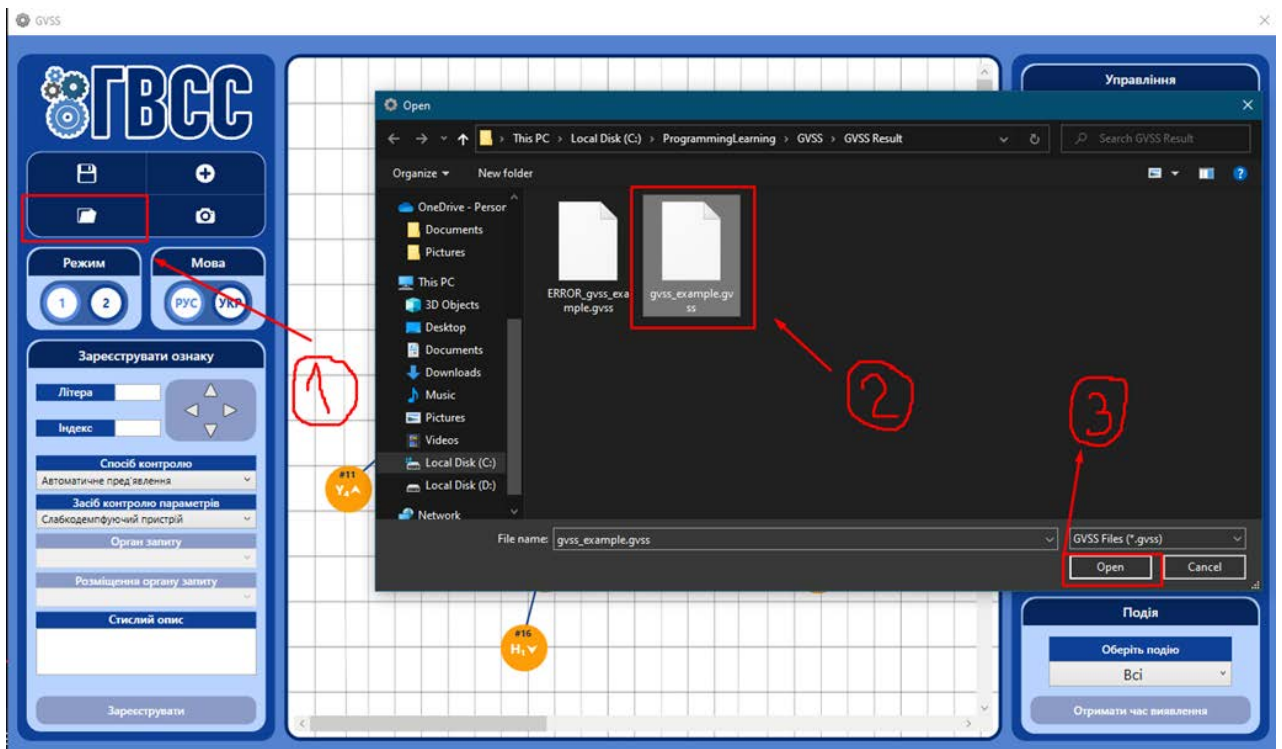


Рисунок 5.1 – Процес завантаження ГВСС

Для того, щоб створити нову ГВСС необхідно натиснути на кнопку «Створити нову ГВСС»  у блоці управління програмою на головній формі.

Для того, щоб зберегти фото ГВСС необхідно натиснути на кнопку «Зберегти фото ГВСС»  у блоці управління програмою на головній формі і визначити шлях до файлу.

Приклад збереженого фото ГВСС наведений на рис. 5.2.

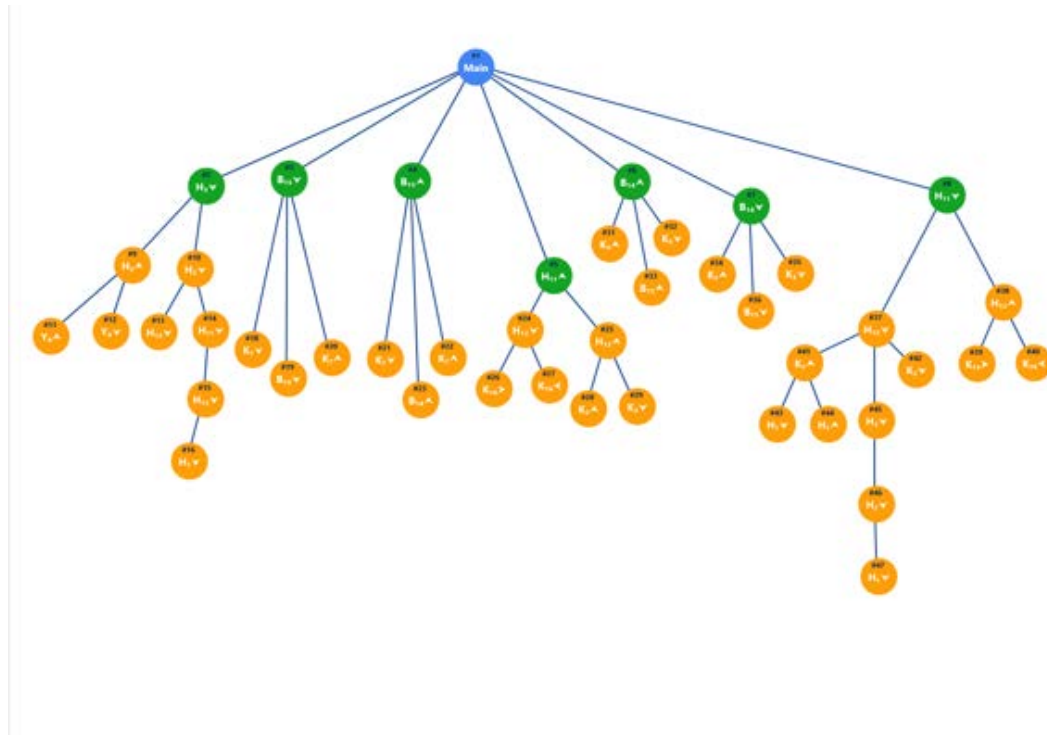


Рисунок 5.2 – Збережене фото структури ГВСС

Для того, щоб переглянути результати, отримані за допомогою MatLAB необхідно натиснути на кнопку «Завантажити» (позиція 1 на рис. 5.3) у блоці управління програмою на головній формі і визначити шлях до файлу з розширенням .asv (позиції 2 та 3 на рис. 5.3) і натиснути кнопку «Open» (позиція 4 на рис. 5.3).

Для того, щоб змінити мову інтерфейсу необхідно обрати необхідну мову у блоці вибору мови на головній формі (рис. 3.6).

Треба зазначити окремо, що програма має два режими роботи. Перший режим: «Порушення – унікальні і реєструються окремо». Це означає, що для реєстрації порушень призначений спеціальний блок на формі додавання вершини і те, що користувач не може використовувати порушення як ознаки до інших порушень.

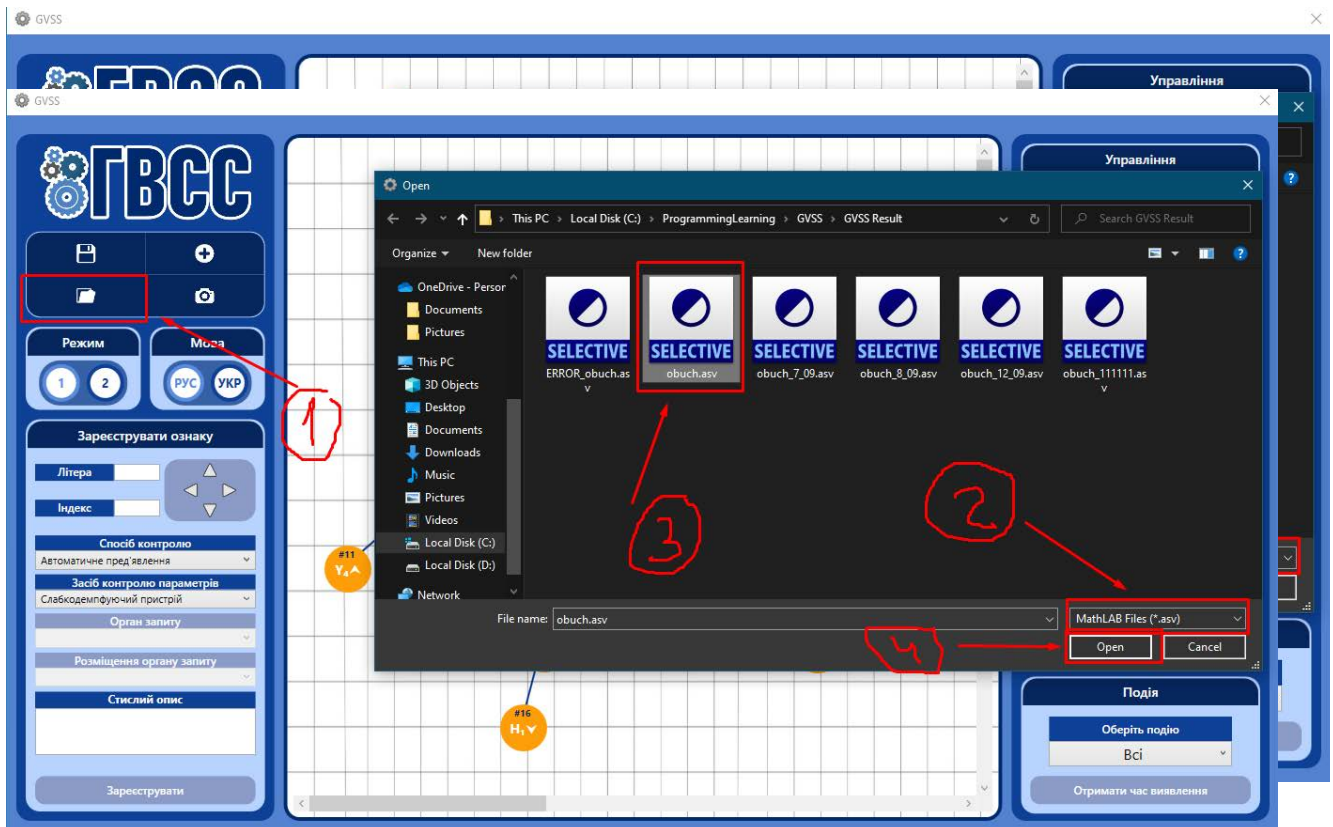


Рисунок 5.3 – Перегляд результатів, отриманих за допомогою MatLAB

Другий режим: «Порушення можуть бути ознаками інших порушень». Це означає, що порушення можуть бути ознаками так само як і ознаки порушеннями, тому обидві ці сутності реєструються в одному місці, а саме у блоці реєстрації ознак на головній формі. Зміна режиму з другого на перший може призвести до втрати даних: деякі вершини, які не відповідають критеріям нового режиму – будуть видалені.

Щоб змінити режим треба обрати необхідний режим у блоці вибору режиму на головній формі програми (рис. 3.5).

У програмі є три опції ГВСС доступні до зміни: «Кількість технологічних ліній», «Рівень тренованості оператора», «Спосіб представлення інформації». Змінити ці параметри можна за допомогою вибору необхідних параметрів у полях з випадаючими списками блоку налаштувань ГВСС на головній формі програми.

Щоб зареєструвати ознаку, необхідно заповнити всі поля, які відповідають за властивості ознаки, у блоці реєстрації на головній формі (рис. 3.7) та натиснути кнопку «Зареєструвати».

Щоб додати вершину необхідно натиснути на необхідну точку на робочій області і далі у діалоговому вікні «Додати вершину» вибрати всі необхідні параметри нової вершини і натиснути кнопку «Додати».

Щоб переміщувати вершину по робочій області необхідно натиснути на потрібній вершині лівою кнопкою миші і утримуючи її, перемістити у потрібне місце.

Щоб редагувати ознаку необхідно обрати ознаку у блоці управління ознаками на головній формі або обрати таку функцію у контекстному меню робочої області, а потім у діалоговому вікні «Змінити ознаку» по-новому визначити параметри ознаки і натиснути кнопку «Змінити».

Щоб переглянути інформацію про вершину або ознаку необхідно відповідно натиснути на вершину, що знаходиться на робочій області, правою кнопкою миші, і обрати відповідну функцію у контекстному меню або обрати цю ознаку у полі з випадаючим списком у блоці управління ознаками на головній формі. Інформацію можна побачити у блоці управління ознаками на головній формі.

Для видалення ознаки необхідно обрати цю ознаку у полі з випадаючим списком у блоці управління ознаками на головній формі і натиснути кнопку «Видалити». Будуть видалені всі вершини, які містять цю ознаку і всі їх дочірні вершини.

Для видалення вершини необхідно натиснути на вершину, що знаходиться на робочій області, правою кнопкою миші, і натиснути кнопку «Видалити». Буде видалена відповідна вершина і всі її дочірні вершини.

Для розрахунку часу виявлення порушення, необхідно спочатку визначити або завантажити структуру ГВСС, потім обрати необхідну подію у блоці «Події» і натиснути на кнопку «Отримати час виявлення». В результаті

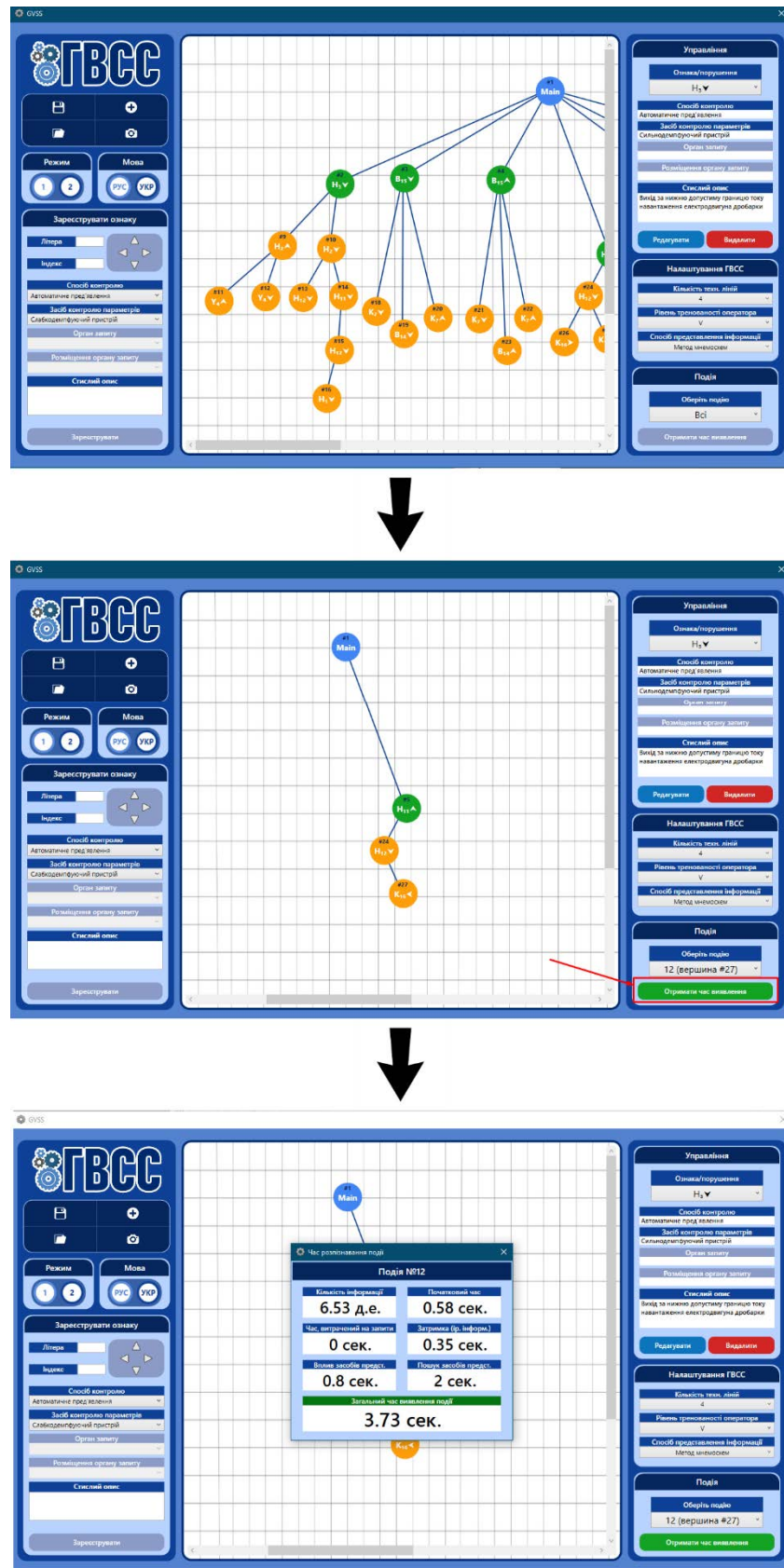


Рисунок 5.4 – Процес отримання часу виявлення певної події

5.3 Повідомлення оператору

Всі повідомлення, що може отримати оператор під час роботи з програмним забезпеченням описані у таблиці 5.1.

Таблиця 5.1 – Повідомлення оператору

Текст	Причина появи	Подальші дії
Ім'я недоступне	При реєстрації події користувач визначає ім'я для ознаки і ознака з таким ім'ям вже є у системі.	Визначити нове ім'я
Незаповнені поля	При реєстрації ознаки ви не заповнюєте всі необхідні поля і вам не дозволено зареєструвати таку ознаку	Заповнити всі поля
Неможливо відкрити файл	При спробі відкрити файл, внутрішня структура якого є пошкодженою.	Відкрити інший файл
Неможливо додати вершину у піддерево	Виникає при спробі додати вершину у піддерево структури ГВСС, якщо у піддереві вже є така ж ознака	Додати іншу ознаку

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи магістра було виконано наступні задачі:

- проведено аналіз методів кількісної оцінки діяльності операторів АСУТП та обрано метод Галактіонова, що відповідає цілям дослідження;
- розроблено технічне завдання на програмний застосунок, що автоматизує діяльність проектувальника ЛМІ;
- навчено нейронну мережу (у вигляді двошарового персептрону) розрахунку інформаційної складності події технологічного процесу;
- розроблено архітектуру системи;
- розроблено моделі програмного забезпечення застосунку (діаграми класів, компонентів та діаграму прецедентів);
- розроблено програмне та інформаційне забезпечення компонентів системи, що автоматизує діяльність проектувальника ЛМІ;
- перевірено адекватність моделі на якій базується розроблене програмне забезпечення шляхом порівняння з ручними розрахунками за методом [17].

Наукова новизна результатів дослідження полягає у тому, що дістав подальшого розвитку відомий метод аналізу операторської діяльності Галактіонова. Метод, що розроблено, на відміну від попереднього, має засоби для формалізації процесу аналізу діяльності оператора АСУТП проектувальником ЛМІ АСУТП, шляхом розроблення і застосування програмного забезпечення на основі моделей штучного інтелекту.

Практична цінність роботи полягає в тому, що на підставі проведених досліджень розроблено програмне та інформаційне забезпечення, що забезпечують підвищення ефективності діяльності проектувальника людино-машинних інтерфейсів АСУТП.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Subhashini, G. Human-centered time-pressured decision making in dynamic complex systems: thesis ... doctor of philosophy / Subhashini Ganapathy. – Hamilton: Wright State University, 2006. – 119 p.
2. Mitchell, C. M. GT-MSOCC: A Domain for Research on Human-Computer Interaction and Decision Aiding in Supervisory Control Systems / C. M. Mitchell // IEEE Transactions on Systems, Man, and Cybernetic. – 1987. – Vol. 17(4). – P. 553 – 572.
3. Wickens, C. D. Engineering Psychology and Human Performance. – Columbus: Merrill, 1994. – 360 p.
4. Rasmussen, J. The role of hierarchical knowledge representation in decision making and system management / J. Rasmussen // IEEE Transactions on Systems, Man, and Cybernetic. – 1997. – Vol. 115(2). – P. 234 – 245.
5. Мунипов, В. М. Эргономика: человекоориентированное проектирование техники, программных средств и среды: Учебник / В. М. Мунипов, В. П. Зинченко. – М.: Логос, 2001. – 356 с.
6. Ашеров, А. Т. Ергономіка інформаційних технологій: оцінка, проектування, експертиза: Навч. посібник / А. Т. Ашеров, Г. І. Сажко. – Харків: УІПА, 2005. – 255 с.
7. Зенкин, А.А. Когнитивная компьютерная графика / Под ред. Д.А. Поспелова. – М.: Наука, 1991. – 256 с.
8. Информационно-управляющие человеко-машинные системы: исследование, проектирование, испытание. Справочник / А.Н. Адаменко, А.Т. Ашеров, И.Л. Бердников и др.; под общ. ред. А. И. Губинского. – М.: Машиностроение, 1993. – 528 с.
9. Wagner, S. Software Product Quality Control / S. Wagner. – Berlin: Springer, 2013. – 210 p.
10. Card, S. K. The Psychology of Human-Computer Interaction / S. K. Card, A. Newell, T. P. Moran. – NY: CRC Press, 2008. – 473 p.

11. Kieras, D. E. A Guide to GOMS model usability evaluation using NGOMSL / D. E. Kieras // *The Handbook of Human-Computer Interaction*. 2nd ed. – North Holland: Elsevier Science B.V., 1997. – Ch. 7, – Ch. 31. – P. 135-157, 733-767.

12. John, B.E., Kieras, D. E. The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast / B.E. John, D. E. Kieras // *ACM Transactions on Computer-Human Interaction*. – 1996. – Vol. 3(4). – P. 320 – 351.

13. Raskin, J. *The Humane Interface* / J. Raskin. – Boston: Addison Wesley, 2000. – 233 p.

14. John, B.E. Predictive Human Performance Modeling Made Easy / B.E. John, K. Prevas, D.D. Salvucci // *Human factors in computing systems: CHI '04*, New York, 24-29 April 2004: proceedings. – NY: ACM, 2004. – P. 455 – 462.

15. Baumeister, L. A Comparison of Tools for Building GOMS Models / L. Baumeister, B. E. John, M. Byrne // *Human factors in computing systems: CHI 2000*, Hague, The Netherlands, 1-6 April 2000: proceedings. – Hague: ACM, 2000. – P. 502 – 509.

16. Holleis, P. Keystroke-level model for advanced mobile phone interaction / P. Holleis, F. Otto, H. Hussmann // *Human factors in computing systems: CHI '07*, New York, 28 April - 03 May 2000: proceedings. – NY: ACM, 2000. – P. 1505 – 1514.

17. Галактионов, А. И. Представление информации оператору (исследование деятельности человека – оператора производственных процессов / А. И. Галактионов. – М.: Энергия, 1969. – 136 с.

18. Dhillon, B.S. Human error in maintenance: a review / B.S. Dhillon, Y. Liu // *Journal of Quality in Maintenance Engineering*. – 2006. – Vol. 12(1). – P. 21 – 36.

19. Virovac, D. The influence of human factor in aircraft maintenance / D. Virovac, A. Domitrovic, E. Bazijanac // *Promet-Traffic & Transportation*. – 2017. – Vol. 29(3). – P. 257 – 266.

20. Latorella, K.A. A review of human error in aviation maintenance and inspection / K.A. Latorella, P.V. Prabhu // *International Journal of Industrial*

Ergonomics. – 2000. – Vol. 26(2). – P. 133 – 161.

21. Lakhwinder, K. Understanding the Event-driven Architecture [Electronic resource] / K. Lakhwinder. – Access mode: <https://softobiz.com/understanding-the-event-driven-architecture/>.

22. Тидвелл, Дж. Разработка пользовательских интерфейсов / Дж. Тидвелл. – СПб.: Питер, 2008 – 416 с.

ДОДАТОК А
Приклад навчальної вибірки
для III-го рівня тренованості оператора

Таблиця А.1 – Навчальна вибірка для III-го рівня тренованості оператора

№ по- дії	№ вер- шини на рівні 1	Кіль- кість розга- лужень на рівні 1 від верши- ни рівня 0	Відно- сна ймові- рність ознаки на рівні 1	№ вер- шини на рівні 2	Кіль- кість розга- лужень на рівні 2 від верши- ни рівня 1	Відно- сна ймові- рність ознаки на рівні 2	№ вер- шини на рівні 3	Кіль- кість розга- лужень на рівні 3 від верши- ни рівня 2	Відно- сна ймові- рність ознак на рівні 3	№ вер- шини на рівні 4	Кіль- кість розга- лужень на рівні 4 від верши- ни рівня 3	Відно- сна ймові- рність ознак на рівні 4	№ вер- шини на рівні 5	Кіль- кість розга- лужень на рівні 5 від верши- ни рівня 4	Відно- сна ймові- рність ознаки на рівні 5	Резу- льтат $I_{\text{поді}}$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
4	1	7	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48
4	2	7	0.173	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	6.984
4	3	7	0.193	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	6.988
4	5	7	0.117	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.56
4	7	7	0.140	2	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.288
4																
4	1	6	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48
4	2	6	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48
4	3	6	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48
4	5	6	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48
4	6	6	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48
4																
4	1	5	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48
4	2	5	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48
4	3	5	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48

Продовження табл. А.1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
4	4	5	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48
4	5	5	0.123	1	2	0.369	1	1	0.682	1	1	0.375	1	1	0.483	7.48
1	1	7	0.123	2	2	0.631	1	3	0.256	0	0	0	0	0	0	5.65
2	1	7	0.123	2	2	0.631	2	3	0.594	0	0	0	0	0	0	4.442
3	1	7	0.123	2	2	0.631	3	3	0.15	0	0	0	0	0	0	6.425
3	1	7	0.143	1	3	0.333	4	5	0.2	5	5	0.2	2	4	0.25	11.02 5
3	4	7	0.143	1	3	0.333	4	5	0.2	5	5	0.2	2	4	0.25	11.02 5
3	7	7	0.143	1	3	0.333	4	5	0.2	5	5	0.2	2	4	0.25	11.02 5
3	1	2	0.143	1	3	0.333	4	5	0.2	5	5	0.2	2	4	0.25	11.02 5
5	4	7	0.137	2	2	0.471	2	2	0.456	0	0	0	0	0	0	5.088
7	4	7	0.137	2	2	0.471	1	2	0.544	0	0	0	0	0	0	4.832
9	4	7	0.137	1	2	0.529	2	2	0.387	0	0	0	0	0	0	5.158
10	4	7	0.137	1	2	0.529	1	2	0.613	0	0	0	0	0	0	4.493
10	4	7	0.137	1	5	0.529	5	5	0.613	0	0	0	0	0	0	4.493
10	4	7	0.137	5	5	0.529	1	5	0.613	0	0	0	0	0	0	4.493
10	1	7	0.137	1	5	0.529	5	5	0.613	0	0	0	0	0	0	4.493
10	7	7	0.137	1	5	0.529	5	5	0.613	0	0	0	0	0	0	4.493
10	3	7	0.137	3	5	0.529	3	5	0.613	0	0	0	0	0	0	4.493
6	7	7	0.140	1	2	0.286	3	3	0.231	0	0	0	0	0	0	6.764
6	1	7	0.140	1	2	0.286	3	3	0.231	0	0	0	0	0	0	6.764
6	3	7	0.140	1	2	0.286	3	3	0.231	0	0	0	0	0	0	6.764

Продовження табл. А.1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
8	7	7	0.140	2	2	0.714	1	2	0.398	0	0	0	0	0	0	4.651
11	7	7	0.140	2	2	0.714	2	2	0.602	0	0	0	0	0	0	4.054
9	4	7	0.137	1	2	0.529	2	2	0.387	0	0	0	0	0	0	5.16
10	4	7	0.137	1	2	0.529	1	2	0.613	0	0	0	0	0	0	4.493
10	1	3	0.333	1	5	0.2	5	5	0.2	0	0	0	0	0	0	6.232
10	1	1	1	3	5	0.2	4	5	0.2	0	0	0	0	0	0	4.644
12	7	7	0.14	1	2	0.286	1	3	0.473	1	2	0.497	0	0	0	6.733
13	7	7	0.14	1	2	0.286	1	3	0.473	2	2	0.503	0	0	0	6.718
13	1	7	0.14	1	2	0.286	1	3	0.473	1	2	0.497	0	0	0	6.733
13	3	7	0.14	1	2	0.286	1	3	0.473	1	2	0.497	0	0	0	6.733
13	5	7	0.14	1	2	0.286	1	3	0.473	1	2	0.497	0	0	0	6.733
13	5	7	0.14	2	2	0.286	3	3	0.473	2	2	0.497	0	0	0	6.733
14	7	7	0.14	1	2	0.286	2	3	0.296	1	1	0.434	1	1	0.518	8.533
15	5	7	0.117	3	3	0.408	0	0	0	0	0	0	0	0	0	4.39
16	5	7	0.117	1	3	0.271	0	0	0	0	0	0	0	0	0	4.979
17	5	7	0.117	2	3	0.321	0	0	0	0	0	0	0	0	0	4.733
18	6	7	0.117	3	3	0.533	0	0	0	0	0	0	0	0	0	4.002
19	6	7	0.117	1	3	0.352	0	0	0	0	0	0	0	0	0	4.601
20	6	7	0.117	2	3	0.115	0	0	0	0	0	0	0	0	0	6.211
21	3	7	0.193	3	3	0.475	0	0	0	0	0	0	0	0	0	3.447
22	3	7	0.193	1	3	0.114	0	0	0	0	0	0	0	0	0	5.506
23	3	7	0.193	2	3	0.411	0	0	0	0	0	0	0	0	0	3.657
24	2	7	0.173	2	3	0.342	0	0	0	0	0	0	0	0	0	4.078
25	2	7	0.173	1	3	0.289	0	0	0	0	0	0	0	0	0	4.322
26	2	7	0.173	3	3	0.369	0	0	0	0	0	0	0	0	0	3.97

ДОДАТОК Б
Текст програми

A.1 Текст розробленого застосунку

A.1.1 Код програми навчання персептрону

```
clear all
```

```
x=[
0.102      0.451      0.285 0      0
0.128      0.475      0.304 0      0
0.102      0.451      0.189 0      0
0.126      0.477      0.211 0      0
0.102      0.451      0.489 0.627 0
0.126      0.471      0.519 0.648 0
0.103      0.451      0.491 0.248 0
0.128      0.471      0.516 0.272 0
0.102      0.453      0.491 0.093 0.401
0.122      0.471      0.518 0.116 0.424
0.102      0.450      0.491 0.093 0.578
0.122      0.473      0.518 0.116 0.602
0.101      0.202      0      0      0
0.125      0.232      0      0      0
0.102      0.304      0.276 0      0
0.128      0.328      0.298 0      0
0.102      0.304      0.699 0      0
0.128      0.328      0.725 0      0
0.102      0.614      0.553 0.21  0.71
0.128      0.634      0.579 0.415 0.914
0.112      0.619      0.559 0.219 0.795
0.121      0.630      0.571 0.319 0.832
0.102      0.614      0.553 0.679 0
0.129      0.636      0.579 0.711 0
0.182      0.220      0.618 0      0
0.208      0.242      0.639 0      0
0.182      0.274      0.413 0      0
0.208      0.293      0.437 0      0
0.182      0.274      0.299 0      0
0.208      0.293      0.321 0      0
0.182      0.274      0.253 0.266 0
0.208      0.293      0.281 0.288 0
0.182      0.274      0.253 0.311 0
0.209      0.294      0.281 0.334 0
0.182      0.274      0.253 0.233 0
0.205      0.297      0.283 0.258 0
0.183      0.171      0      0      0
0.208      0.198      0      0      0
0.188      0.291      0      0      0
0.209      0.315      0      0      0
0.181      0.294      0      0      0
0.208      0.317      0      0      0
0.181      0.271      0.253 0.142 0.608
0.201      0.296      0.283 0.168 0.638
0.184      0.273      0.256 0.144 0.361
0.205      0.293      0.282 0.167 0.389
0.141      0.470      0.601 0      0
0.166      0.492      0.628 0      0
0.139      0.473      0.372 0.371 0.313
0.167      0.495      0.398 0.411 0.345
0.141      0.475      0.372 0.380 0.143
0.168      0.497      0.399 0.415 0.167
0.143      0.471      0.375 0.381 0.501
0.161      0.491      0.395 0.405 0.529
```

0.143	0.471	0.375	0.591	0
0.163	0.491	0.395	0.619	0
0.089	0.701	0	0	0
0.121	0.732	0	0	0
0.171	0.478	0	0	0
0.186	0.496	0	0	0
0.169	0.501	0.545	0	0
0.186	0.521	0.562	0	0
0.169	0.501	0.435	0	0
0.186	0.521	0.451	0	0
0.135	0.377	0	0	0
0.152	0.394	0	0	0
0.135	0.328	0	0	0
0.151	0.349	0	0	0
0.138	0.271	0.195	0	0
0.271	0.195	0.138	0	0
0.159	0.291	0.217	0	0
0.138	0.271	0.356	0	0
0.159	0.291	0.372	0	0
0.372	0.159	0.291	0	0
0.138	0.271	0.123	0	0
0.271	0.123	0.138	0	0
0.159	0.291	0.142	0	0
0.133	0.268	0.291	0.619	0.143
0.133	0.268	0.291	0.702	0.149
0.268	0.133	0.143	0.291	0.619
0.157	0.288	0.314	0.725	0.169
0.725	0.169	0.288	0.157	0.314
0.137	0.261	0.291	0.699	0.202
0.151	0.287	0.317	0.729	0.223
0.139	0.265	0.297	0.690	0.302
0.154	0.281	0.312	0.724	0.333
0.141	0.261	0.289	0.699	0.265
0.159	0.289	0.316	0.728	0.289
0.14	0.266	0.291	0.695	0.021
0.154	0.281	0.312	0.724	0.042
0.141	0.269	0.296	0.703	0.024
0.08	0.281	0	0	0
0.08	0.269	0	0	0
0.132	0.288	0	0	0
0.115	0.288	0	0	0
0.08	0.711	0	0	0
0.115	0.734	0	0	0
0.109	0.455	0.289	0	0
0.121	0.470	0.302	0	0
0.109	0.455	0.192	0	0
0.121	0.470	0.205	0	0
0.110	0.458	0.289	0	0
0.119	0.470	0.301	0	0
0.110	0.458	0.291	0	0
0.119	0.468	0.301	0	0
0.11	0.458	0.194	0	0
0.119	0.467	0.203	0	0
0.11	0.455	0.499	0.629	0
0.121	0.471	0.512	0.648	0
0.11	0.456	0.497	0.254	0
0.121	0.470	0.513	0.271	0
0.11	0.456	0.497	0.095	0.406
0.121	0.470	0.512	0.110	0.421
0.111	0.456	0.497	0.095	0.580
0.121	0.471	0.512	0.112	0.598
0.109	0.212	0	0	0
0.122	0.231	0	0	0

0.109	0.308	0.278	0	0
0.123	0.324	0.293	0	0
0.109	0.309	0.703	0	0
0.118	0.324	0.721	0	0
0.109	0.618	0.559	0.216	0.804
0.121	0.631	0.572	0.315	0.818
0.109	0.617	0.559	0.682	0
0.123	0.633	0.574	0.699	0
0.186	0.225	0.619	0	0
0.201	0.241	0.633	0	0
0.187	0.277	0.416	0	0
0.201	0.292	0.433	0	0
0.188	0.279	0.301	0	0
0.202	0.293	0.318	0	0
0.188	0.279	0.259	0.268	0
0.202	0.293	0.276	0.284	0
0.189	0.279	0.258	0.316	0
0.202	0.293	0.275	0.330	0
0.187	0.277	0.259	0.239	0
0.201	0.291	0.273	0.251	0
0.188	0.173	0	0	0
0.203	0.191	0	0	0
0.188	0.297	0	0	0
0.201	0.312	0	0	0
0.188	0.276	0.263	0.148	0.618
0.201	0.293	0.276	0.165	0.633
0.188	0.277	0.259	0.149	0.367
0.201	0.293	0.281	0.163	0.382
0.148	0.477	0.611	0	0
0.161	0.491	0.623	0	0
0.145	0.477	0.376	0.382	0.328
0.163	0.491	0.391	0.405	0.341
0.148	0.478	0.377	0.386	0.147
0.161	0.491	0.390	0.401	0.161
0.149	0.478	0.376	0.389	0.511
0.161	0.491	0.391	0.401	0.523
0.148	0.477	0.376	0.597	0
0.161	0.491	0.391	0.613	0
0.147	0.508	0	0	0
0.161	0.522	0	0	0
0.173	0.482	0	0	0
0.184	0.493	0	0	0
0.173	0.507	0.549	0	0
0.184	0.519	0.561	0	0
0.173	0.506	0.438	0	0
0.184	0.520	0.450	0	0
0.138	0.379	0	0	0
0.150	0.391	0	0	0
0.138	0.329	0	0	0
0.150	0.343	0	0	0
0.139	0.273	0.197	0	0
0.152	0.284	0.212	0	0
0.139	0.274	0.358	0	0
0.153	0.284	0.370	0	0
0.139	0.273	0.126	0	0
0.151	0.284	0.140	0	0
0.138	0.270	0.296	0.707	0.148
0.151	0.284	0.311	0.723	0.164
0.138	0.270	0.296	0.707	0.208
0.151	0.284	0.311	0.723	0.218
0.139	0.273	0.297	0.709	0.318
0.150	0.284	0.310	0.723	0.329
0.139	0.273	0.297	0.709	0.269

```

0.150      0.284      0.310 0.723 0.282
0.139      0.273      0.297 0.709 0.028
0.150      0.284      0.310 0.723 0.040
0.089      0.271      0      0      0
0.105      0.285      0      0      0
0.089      0.717      0      0      0
0.105      0.734      0      0      0
]
minx=min(x);
maxx=max(x);
%нормирование X
for i=1:1:size(x,1)
    for j=1:1:size(x,2)
        xN(i,j)=(x(i,j)-minx(j))/(maxx(j)-minx(j));
    end;
end;
xN %нормированные X

y=[
6.254
5.756
6.845
6.299
6.148
5.643
7.455
6.878
10.2
9.381
9.703
8.895
5.615
5.108
6.861
6.322
5.526
5.04
7.587
5.796
7.222
6.429
5.411
4.887
5.339
4.957
5.601
5.227
6.069
5.673
8.2
7.673
8.002
7.429
8.43
7.828
4.998
4.601
4.192
3.926
4.232
3.923
9.828
9.118

```

10.543
9.828
4.651
4.285
8.48
7.733
9.48
8.764
7.703
7.222
6.069
5.673
4.002
3.497
3.614
3.438
4.439
4.197
4.764
4.516
4.296
4.061
4.496
4.246
7.098
7.098
6.644
6.232
5.878
5.878
7.764
7.764
7.243
10.08
9.828
10.08
9.2
9.2
9.38
8.828
8.764
8.243
8.965
8.381
12.6
11.252
12.287
5.475
5.539
4.717
4.917
4.135
3.567
6.128
5.861
6.718
6.417
6.102
5.895
6.088
5.895
6.673
6.467

5.993
5.725
7.31
6.984
10.02
9.587
9.48
9.04
5.436
5.148
6.749
6.417
5.399
5.179
7.265
6.467
5.288
5.002
5.271
5.026
5.54
5.299
5.984
5.733
8.118
7.764
7.861
7.533
8.288
7.966
4.943
4.688
4.164
3.995
9.703
9.2
10.48
9.966
4.536
4.345
8.243
7.861
9.381
8.966
7.56
7.265
5.984
5.725
3.743
3.573
3.584
3.463
4.375
4.222
4.707
4.536
4.257
4.09
4.461
4.279
7.059
6.764
6.2

```

5.957
7.703
7.381
9.703
9.288
9.288
8.895
8.644
8.333
8.895
8.533
12.15
11.362
5.375
5.064
3.97
3.698
]
minY=min(y);
maxY=max(y);
%нормирование Y
for i=1:1:size(y,1)
    yN(i)=(y(i)-minY)/(maxY-minY);
end;
yN %нормированный Y

xN=xN';
net = newff(repmat([0 1],size(x,2),1),[2,1],{'logsig','logsig'},'trainlm');

net = train(net,xN,yN)
a=sim(net,xN)

[yN' a'] %вывод значений Y реального и модельного (пронормированных)
E=mse(yN-a)

net.IW{1} %веса первого слоя
net.LW{2,1} %веса второго слоя
net.b{1} %смещения w0 нейронов первого слоя
net.b{2} %смещения w0 нейрона второго слоя

x1=[
0.113 0.461 0.295 0 0
0.113 0.461 0.199 0 0
0.113 0.461 0.506 0.638 0
0.113 0.461 0.506 0.261 0
0.113 0.461 0.506 0.101 0.411
0.113 0.461 0.506 0.101 0.589
0.113 0.222 0 0 0
0.113 0.317 0.287 0 0
0.113 0.317 0.713 0 0
0.116 0.624 0.567 0.31 0.811
0.116 0.624 0.567 0.69 0
0.193 0.231 0.628 0 0
0.193 0.284 0.421 0 0
0.193 0.284 0.31 0 0
0.193 0.284 0.269 0.277 0
0.193 0.284 0.269 0.321 0
0.193 0.284 0.269 0.246 0
0.193 0.18 0 0 0
0.193 0.305 0 0 0
0.193 0.284 0.269 0.156 0.627

```

```

0.193      0.284      0.269 0.156 0.373
0.154      0.483      0.619 0      0
0.154      0.483      0.381 0.392 0.333
0.154      0.483      0.381 0.392 0.151
0.154      0.483      0.381 0.392 0.516
0.154      0.483      0.381 0.608 0
0.154      0.517      0      0      0
0.179      0.487      0      0      0
0.179      0.513      0.555 0      0
0.179      0.513      0.445 0      0
0.145      0.384      0      0      0
0.145      0.337      0      0      0
0.145      0.279      0.205 0      0
0.145      0.279      0.363 0      0
0.145      0.279      0.131 0      0
0.145      0.279      0.301 0.717 0.156
0.145      0.279      0.301 0.717 0.212
0.145      0.279      0.301 0.717 0.323
0.145      0.279      0.301 0.717 0.278
0.145      0.279      0.301 0.717 0.031
0.1        0.279      0      0      0
0.1        0.721      0      0      0
]
minx1=min(x1);
maxx1=max(x1);
%нормирование X
for i=1:1:size(x1,1)
    for j=1:1:size(x1,2)
        x1N(i,j)=(x1(i,j)-minx1(j))/(maxx1(j)-minx1(j));
    end;
end;

y1=[
6.059
6.644
5.878
7.179
9.966
9.315
5.322
6.644
5.293
6.601
5.143
5.158
5.442
5.878
7.93
7.733
8.118
4.849
4.083
9.966
10.183
4.442
8.078
9.2
4.133
5.878
3.651
3.523
4.293
4.608

```

```

4.166
4.354
6.913
6.088
7.56
9.48
9.118
8.48
8.703
11.855
5.163
3.794
]
x1N=x1N';
minY1=min(y1);
maxY1=max(y1);
%нормирование Y
for i=1:1:size(y1,1)
    y1N(i)=(y1(i)-minY1)/(maxY1-minY1);
end;

y1mod=sim(net,x1N)

[y1N' y1mod']
E=mse(y1N-y1mod)

```

A.1.2 Файл Event.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace gvss_project
{
    public class Event
    {
        public class VertexAndNumOfSubverticies
        {
            public int VertexID { get; private set; }
            public int SubverticiesNumber { get; private set; }
            public VertexAndNumOfSubverticies(int vertexID, int
subvertexesNumber)
            {
                if (vertexID <= 0 || subvertexesNumber < 0)
                {
                    throw new InvalidArgument();
                }
                VertexID = vertexID;
                SubverticiesNumber = subvertexesNumber;
            }
            public VertexAndNumOfSubverticies()
            {
                VertexID = -1;
                SubverticiesNumber = -1;
            }
            public bool isNull()
            {

```

```

        if (VertexID == -1 && SubverticiesNumber == -1)
            return true;
        return false;
    }
}

private List<VertexAndNumOfSubverticies> vertices;
private uint numOfTechLines;
private float violationInformationQuantity;
private uint sensorsCount;
private uint dampedDeviceTotal;
private OperatorLevel operatorLevel;
private InformationRepresentationMethod
informationRepresentationMethod;
public List<uint> VertexPath { get; private set; }
public List<Sensor> requestSensors;
public List<Sensor> autoSensors;
public uint ID { get; private set; }

private VertexAndNumOfSubverticies isVertexAlreadyAdded(int vID)
{
    foreach (VertexAndNumOfSubverticies vertex in vertices)
    {
        if (vertex.VertexID == vID)
        {
            return vertex;
        }
    }
    throw new NotFoundException();
}

private double Log(double value)
{
    return -Math.Log(value, 2);
}

private double Round(double value, int decimalNumbers = 2) {
    return Math.Round(value, decimalNumbers);
}

public Event(uint id, List<uint> vertexPath, uint numOfTechLines, uint
sensorsCount, float violationInformationQuantity, OperatorLevel operatorLevel,
uint dampedDeviceTotal, InformationRepresentationMethod
informationRepresentationMethod)
{
    vertices = new List<VertexAndNumOfSubverticies>();

    requestSensors = new List<Sensor>();
    autoSensors = new List<Sensor>();

    this.numOfTechLines = numOfTechLines;
    this.operatorLevel = operatorLevel;
    this.sensorsCount = sensorsCount;
    this.violationInformationQuantity = violationInformationQuantity;
    this.dampedDeviceTotal = dampedDeviceTotal;
    this.informationRepresentationMethod =
informationRepresentationMethod;

    ID = id;
    VertexPath = vertexPath;
}

public void addRequestSensor(Sensor s)

```

```

    {
        if (s.ControlMethod == SensorControlMethod.Request)
            requestSensors.Add(s);
    }
    public void addAutoSensor(Sensor s) {
        if (s.ControlMethod == SensorControlMethod.Auto)
            autoSensors.Add(s);
    }

    public void addVertexAndSubverticiesNumber(VertexAndNumOfSubverticies
node)
    {
        VertexAndNumOfSubverticies v = new VertexAndNumOfSubverticies();
        try
        {
            v = isVertexAlreadyAdded(node.VertexID);
            throw new InvalidArgument();
        }
        catch (NotFoundException)
        {
            vertices.Add(node);
        }
    }
    public VertexAndNumOfSubverticies getVertexAndSubverticiesNumber(int
vID)
    {
        return isVertexAlreadyAdded(vID);
    }

    public double getInformationDifficulty()
    {
        double techLineInfo = Log(1.0 / numOfTechLines);
        double sum = techLineInfo + Log(violationInformationQuantity);

        foreach (VertexAndNumOfSubverticies v in vertices)
        {
            sum -= Log((float)v.SubverticiesNumber);
        }
        return Round(sum);
    }
    public double getOriginalTime()
    {
        double informationDifficulty = getInformationDifficulty();
        (double, double) ab = DefaultSettings.OperatorCoefs[operatorLevel];
        double time = ab.Item1 + ab.Item2 * informationDifficulty;
        return
Round(DefaultSettings.IndformationRepresentationMethodCoefs[informationRepresen
tationMethod] * time);
    }
    public double getRequestTime() {
        double time = 0;
        double t1, t2;
        foreach (Sensor s in requestSensors) {
            t1 = DefaultSettings.RequestTime[(s.RequestBody,
s.RequestPlacementMethod)];
            t2 = DefaultSettings.ControlAgentsTime[s.ControlAgent];
            if (s.RequestBody == SensorRequestBodies.Dialer ||
s.RequestBody == SensorRequestBodies.KeyboardDevice)
            {
                t1 *= s.NumOfRequestInputKeys;
            }
            time += t1 + t2;
        }
    }

```

```

    }
    return Round(time);
}
public double getIrrelevantTime() {
    return Math.Round(0.2*(sensorsCount *
0.075),2,MidpointRounding.AwayFromZero);
}
public double getAgentTimeInfluence() {
    double time = 0;
    foreach (Sensor s in autoSensors) {
        time += DefaultSettings.ControlAgentsTime[s.ControlAgent];
    }
    return Round(time);
}
public double getFindControlAgentTime() {
    double time = 0;

    uint dampedDevicesInEvent = 0;

    foreach (Sensor s in requestSensors) {
        if (s.ControlAgent == SensorControlAgent.HighlyDampedDevice ||
s.ControlAgent == SensorControlAgent.WeakDampedDevice)
            dampedDevicesInEvent++;
    }
    foreach (Sensor s in autoSensors)
    {
        if (s.ControlAgent == SensorControlAgent.HighlyDampedDevice ||
s.ControlAgent == SensorControlAgent.WeakDampedDevice)
            dampedDevicesInEvent++;
    }

    if (dampedDevicesInEvent != 0) {
        time = 0.02 * dampedDeviceTotal + 1.7 + (0.1 *
(dampedDevicesInEvent - 1));
    }

    return Round(time, 1);
}
}
}
}

```

A.1.3 Файл GVSS.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace gvss_project
{
    [Serializable]
    public enum GVSSMode { UniqueViolations, RepetativeViolations }

    [Serializable]
    public class VertexNode
    {
        public uint VertexID { get; private set; }
        public string UniqueCode { get; private set; }
    }
}

```

```

    public VertexNode(uint vID, string uniqueCode) {
        VertexID = vID;
        UniqueCode = uniqueCode;
    }
}

[Serializable]
public class GVSS
{
    [Serializable]
    private enum VertexType { Main, Violation, AuxiliaryParam };

    [Serializable]
    private class Vertex
    {
        [Serializable]
        private class Edge
        {
            public float EdgeWeight { get; set; }
            public uint ConnectedVertexID { get; set; }
            public Edge(uint conVerID, float edgeWeight)
            {
                EdgeWeight = edgeWeight;
                ConnectedVertexID = conVerID;
            }
        }

        //properties and fields
        public Point CenterPos { get; set; }
        public uint ID { get; private set; }
        public VertexType Type { get; private set; }
        public int AdjVCount {
            get {
                return edges.Count;
            }
        }
        public uint Level { get; private set; }
        private List<Edge> edges;
        //additional information corresponding to the task

        public Vertex() { }
        public Vertex(uint vID, VertexType type, uint level, Point
centerPos)
        {
            ID = vID;
            edges = new List<Edge>();
            Type = type;
            Level = level;
            CenterPos = centerPos;
        }
        public void addEdge(uint anotherVertexID, float edgeWeight)
        {
            if (anotherVertexID != this.ID)
                edges.Add(new Edge(anotherVertexID, edgeWeight));
        }
        public void deleteEdge(uint anotherVertexID)

```

```

{
    if (anotherVertexID != this.ID)
    {
        foreach (Edge edge in edges)
        {
            if (edge.ConnectedVertexID == anotherVertexID)
            {
                edges.Remove(edge);
                break;
            }
        }
    }
}

public void setEdgeWeight(uint anotherVertexID, float edgeWeight)
{
    foreach (Edge edge in edges)
    {
        if (edge.ConnectedVertexID == anotherVertexID)
        {
            edges[edges.IndexOf(edge)].EdgeWeight = edgeWeight;
            break;
        }
    }
}

public List<uint> getAdjacentVertexes()
{
    List<uint> adjVs = new List<uint>();
    foreach (Edge edge in edges)
    {
        adjVs.Add(edge.ConnectedVertexID);
    }
    return adjVs;
}

public bool isConnectedWith(uint anotherVertexID)
{
    if (anotherVertexID != this.ID)
    {
        foreach (Edge edge in edges)
        {
            if (anotherVertexID == edge.ConnectedVertexID)
            {
                return true;
            }
        }
    }
    return false;
}

public float getEdgeWeight(uint anotherVertex)
{
    foreach (Edge edge in edges)
    {
        if (edge.ConnectedVertexID == anotherVertex)
        {
            return edge.EdgeWeight;
        }
    }
}

```

```

        }
    }
    return -1;
}
public uint getParentVertex() {
    if (ID != 1)
    {
        return edges[0].ConnectedVertexID;
    }
    else
        throw new UnexpectedError();
}
}

[Serializable]
private struct Correspondence
{
    public string SensorCode { get; private set; }
    public uint VertexID { get; private set; }
    public uint VertexIndex { get; private set; }

    public Correspondence(string sensorCode, uint vertexID, uint
vertexIndex)
    {
        SensorCode = sensorCode;
        VertexID = vertexID;
        VertexIndex = vertexIndex;
    }
}

//fields and properties

private List<Vertex> vertexes;
private List<(Sensor, uint)> violations;
private List<Sensor> auxiliaryParams;
private List<uint> deletedVertexesList;
private List<Correspondence> correspondences;
private uint vertexID_counter;
private readonly uint mainVertexID;
private GVSSMode mode;

public string ProductionAreaName { get; private set; }
public int Size { get { return vertexes.Count; } }

//private methods
private Vertex getVertex(uint vID) {
    foreach (Vertex v in vertexes)
    {
        if (v.ID == vID)
            return v;
    }
    throw new NotFoundException();
}
private uint getSensorsCount() {
    if (mode == GVSSMode.RepetativeViolations)
    {
        return (uint)auxiliaryParams.Count;
    }
    else if (mode == GVSSMode.UniqueViolations)
    {
        return (uint)(auxiliaryParams.Count + violations.Count);
    }
}

```

```

    }
    else throw new UnexpectedError();
}
private bool isViolation(uint vID) {
    if (getVertex(vID).Type == VertexType.Violation)
    {
        return true;
    }
    else {
        return false;
    }
}
private uint addVertex(VertexType type, uint level, Point centerPos) {
    uint id = ++vertexID_counter;
    vertexes.Add(new Vertex(id, type, level, centerPos));
    return id;
}
private void deleteVertex(uint vertexID) {
    if (isVertexInGraph(vertexID))
    {
        List<uint> connectedVertexes = new List<uint>();
        foreach (Vertex v in vertexes)
        {
            if (v.ID == vertexID)
            {
                connectedVertexes = v.getAdjacentVertexes();
                deletedVertexesList.Add(v.ID);
                vertexes.Remove(v);
                break;
            }
        }
        if (connectedVertexes.Count > 0) {
            foreach (Vertex v1 in vertexes)
            {
                foreach (uint v2 in connectedVertexes)
                {
                    if (v1.ID == v2)
                    {
                        v1.deleteEdge(vertexID);
                    }
                }
            }
        }
    }
}
private void addEdge(uint firstV, uint secondV, float edgeWeight) {
    if (isVertexInGraph(firstV) && isVertexInGraph(secondV) &&
!doesTheEdgeExist(firstV, secondV))
    {
        uint counter = 0;
        foreach (Vertex v in vertexes)
        {
            if (v.ID == firstV)
            {
                v.addEdge(secondV, edgeWeight);
                counter++;
            }
            else if (v.ID == secondV)
            {
                v.addEdge(firstV, edgeWeight);
                counter++;
            }
        }
        if (counter == 2)
    }
}

```

```

        break;
    }
}
}
private bool isVertexInGraph(uint vID) {
    foreach (Vertex vertex in vertexes)
    {
        if (vertex.ID == vID)
            return true;
    }
    return false;
}
private bool doesTheEdgeExist(uint firstV, uint secondV) {
    foreach (Vertex v in vertexes)
    {
        if ((v.ID == firstV && v.isConnectedWith(secondV)) ||
            (v.ID == secondV && v.isConnectedWith(firstV)))
        {
            return true;
        }
    }
    return false;
}
private Sensor registerSensor(char captionLetter, uint index, string
description, SensorControlMethod controlMethod, SensorControlAgent controlAgent,
SensorArrowSign arrowSign, SensorRequestBodies requestBody,
SensorRequestPlacementMethod requestPlacementMethod) {
    Sensor s = new Sensor();
    s.CapitalLetter = captionLetter;
    s.Index = (int)index;
    s.Description = description;
    s.ControlMethod = controlMethod;
    s.ArrowSign = arrowSign;
    s.ControlAgent = controlAgent;
    s.RequestBody = requestBody;
    s.RequestPlacementMethod = requestPlacementMethod;

    if (s.isValid() && isSensorNameAvailable(index, s.CapitalLetter,
s.ArrowSign))
        return s;
    else throw new InvalidSensor();
}
private Sensor getAuxiliaryParam(string code) {
    foreach (Sensor s in auxiliaryParams) {
        if (s.UniqueCode == code)
            return s;
    }
    throw new NotFoundException();
}
private Correspondence getCorrespondence(uint vID) {
    foreach (Correspondence c in correspondences) {
        if (c.VertexID == vID)
            return c;
    }
    throw new NotFoundException();
}
private bool doesTheAuxiliaryParamExist(string code) {
    foreach (Sensor s in auxiliaryParams)
    {
        if (s.UniqueCode == code)
            return true;
    }
    return false;
}

```

```

    }
    private bool doesTheSensorRepeat(string sensorCode, uint
connectedVertexID) {

        List<uint> path = getWayUp(connectedVertexID);
        foreach (uint v in path) {
            if (getUniqueCode(v) == sensorCode)
                return true;
        }

        List<uint> vertexesAround = getAdjacentVertexes(connectedVertexID);
        uint level = getLevel(connectedVertexID);
        foreach (uint v in vertexesAround) {
            if (getLevel(v) > level && sensorCode == getUniqueCode(v))
                return true;
        }

        return false;
    }
    private void deleteCorrespondence(uint vID) {
        foreach (Correspondence c in correspondences) {
            if (c.VertexID == vID) {
                correspondences.Remove(c);
                return;
            }
        }
        throw new NotFoundException();
    }
    private bool doesTheViolationExist(string uniqueCode) {
        foreach ((Sensor, uint) v in violations)
        {
            if (v.Item1.UniqueCode == uniqueCode) {
                return true;
            }
        }
        return false;
    }
    private void deleteViolation(uint vID) {
        foreach ((Sensor, uint) node in violations) {
            if (node.Item2 == vID) {
                violations.Remove(node);
                return;
            }
        }
        throw new NotFoundException();
    }
    private uint getLevel(uint vID) {
        foreach (Vertex v in vertexes) {
            if (v.ID == vID) {
                return v.Level;
            }
        }
        throw new NotFoundException();
    }
    private void deleteAuxiliary(string sensorCode) {
        if (doesTheAuxiliaryParamExist(sensorCode)) {
            foreach (Sensor s in auxiliaryParams) {
                if (s.UniqueCode == sensorCode) {
                    auxiliaryParams.Remove(s);
                    break;
                }
            }
        }
    }

```

```

    }
}
private void deleteProc(uint curVertexID) {

    List<uint> adjV = getAdjacentVertexes(curVertexID);
    foreach (uint v in adjV) {
        if (getLevel(v) > getLevel(curVertexID)) {
            deleteProc(v);
        }
    }
    if (getVertex(curVertexID).Type == VertexType.Violation) {
        string aP = getCorrespondence(curVertexID).SensorCode;
        if (mode == GVSSMode.RepetativeViolations &&
!doesTheAuxiliaryParamExist(aP)) {
            deleteAuxiliary(aP);
        }
        deleteViolation(curVertexID);
    }

    deleteCorrespondence(curVertexID);
    deleteVertex(curVertexID);
    return;
}

private List<uint> getWayUp(uint startVertexID) {

    uint curVertex = startVertexID;
    List<uint> path = new List<uint>();
    bool completed = false;

    while (!completed) {

        path.Add(curVertex);
        List<uint> adjVertexes = getAdjacentVertexes(curVertex);
        uint level = getLevel(curVertex);
        foreach (uint v in adjVertexes) {
            if (level > getLevel(v)) {
                if (v == mainVertexID)
                    completed = true;
                else {
                    curVertex = v;
                }
                break;
            }
        }

    }

    return path;
}

private int getNumberVerticesInSubtree(uint vID) {
    if (isVertexInGraph(vID) && vID != mainVertexID)
    {
        uint parentID = getVertex(vID).getParentVertex();
        uint parentLevel = getLevel(parentID);
        List<uint> verticies = getAdjacentVertexes(parentID);

        int counter = 0;

        foreach (uint v in verticies) {
            if (getLevel(v) > parentLevel) {

```

```

        ++counter;
    }
    }
    return counter;
}
else {
    throw new InvalidArgument();
}
}
private float getViolationInformationQuantity(uint vID, uint
numOfTechLines) {
    if (isViolation(vID))
    {
        float sum = 0;
        foreach ((Sensor, uint) violation in violations) {
            sum += getEdgeWeight(mainVertexID, violation.Item2);
        }

        return getEdgeWeight(mainVertexID, vID) / sum;
    }
    else
        throw new InvalidArgument();
}
private uint getDampedDeviceTotal() {
    uint count = 0;

    foreach (Sensor s in auxiliaryParams) {
        if (s.ControlAgent == SensorControlAgent.HighlyDampedDevice ||
s.ControlAgent == SensorControlAgent.WeakDampedDevice) {
            count++;
        }
    }
    if (mode == GVSSMode.UniqueViolations) {
        foreach ((Sensor, uint) v in violations) {
            if (v.Item1.ControlAgent ==
SensorControlAgent.HighlyDampedDevice || v.Item1.ControlAgent ==
SensorControlAgent.WeakDampedDevice) {
                count++;
            }
        }
    }

    return count;
}

//public methods

public GVSS(Point mainCenterPos, GVSSMode mode =
GVSSMode.RepetativeViolations)
{
    this.mode = mode;
    this.vertexes = new List<Vertex>();
    this.violations = new List<(Sensor, uint)>();
    this.auxiliaryParams = new List<Sensor>();
    this.correspondences = new List<Correspondence>();
    this.vertexID_counter = 0;
    this.ProductionAreaName = "Производственная система";

    //creating main vertex
    mainVertexID = this.addVertex(VertexType.Main, 0, mainCenterPos);
    correspondences.Add(new Correspondence("Main", mainVertexID, 0));
}
public VertexNode addViolation(char captionLetter, uint index, string
description, SensorControlMethod controlMethod, SensorControlAgent

```

```

controlAgent, SensorArrowSign arrowSign, SensorRequestBodies requestBody,
SensorRequestPlacementMethod requestPlacementMethod, Point centerPos, float
probability) {

    Sensor s = registerSensor(captionLetter, index, description,
controlMethod, controlAgent, arrowSign, requestBody, requestPlacementMethod);
    uint level = 1;
    uint vID = addVertex(VertexType.Violation, level, centerPos);
    violations.Add((s, vID));
    if (mode == GVSSMode.RepetativeViolations)
        auxiliaryParams.Add(s);
    correspondences.Add(new Correspondence(s.UniqueCode, vID,
(uint)(vertexes.Count - 1)));

    addEdge(mainVertexID, vID, probability);
    return new VertexNode(vID, s.UniqueCode);

}

public VertexNode addViolation(string auxiliaryCode, Point centerPos,
float probability) {

    if (mode == GVSSMode.RepetativeViolations)
    {
        if (doesTheAuxiliaryParamExist(auxiliaryCode) &&
!doesTheViolationExist(auxiliaryCode))
        {

            uint level = 1;
            uint vID = addVertex(VertexType.Violation, level,
centerPos);

            Sensor s = getAuxiliaryParam(auxiliaryCode);
            violations.Add((s, vID));
            correspondences.Add(new Correspondence(s.UniqueCode, vID,
(uint)(vertexes.Count - 1)));

            addEdge(mainVertexID, vID, probability);
            return new VertexNode(vID, s.UniqueCode);
        }
        else
        {
            throw new InvalidArgument();
        }
    }
    else {
        throw new UnexpectedError();
    }

}

public string addAuxiliaryParam(char captionLetter, uint index, string
description, SensorControlMethod controlMethod, SensorControlAgent
controlAgent, SensorArrowSign arrowSign, SensorRequestBodies requestBody,
SensorRequestPlacementMethod requestPlacementMethod) {
    Sensor s = registerSensor(captionLetter, index, description,
controlMethod, controlAgent, arrowSign, requestBody, requestPlacementMethod);
    auxiliaryParams.Add(s);
    return s.UniqueCode;
}

public VertexNode connect(uint upperVertex, string auxiliaryVertexCode,
Point centerPos) {
    if (upperVertex == mainVertexID
|| !isVertexInGraph(upperVertex)
|| !doesTheAuxiliaryParamExist(auxiliaryVertexCode))

```

```

        throw new InvalidArgument();
    else if (doesTheSensorRepeat(auxiliaryVertexCode, upperVertex))
        throw new SensorRepeatsException();

    Sensor s = getAuxiliaryParam(auxiliaryVertexCode);
    uint level =
vertexes[(int)getCorrespondence(upperVertex).VertexIndex].Level+1;
    uint vID = addVertex(VertexType.AuxiliaryParam, level, centerPos);
    correspondences.Add(new Correspondence(s.UniqueCode, vID,
(uint)(vertexes.Count - 1)));

    float probability = 1.0f; //count probability
    addEdge(upperVertex, vID, probability);
    return new VertexNode(vID,s.UniqueCode);

}
public List<uint> delete(uint vID) {
    if (vID != mainVertexID && isVertexInGraph(vID)) {
        deletedVertexesList = new List<uint>();
        deleteProc(vID);
        return deletedVertexesList;
    }
    else
        throw new InvalidArgument();
}
public bool isSensorNameAvailable(uint index, char capitalLetter,
SensorArrowSign arrowSign) {
    Sensor s = new Sensor();
    s.Index = (int)index;
    s.CapitalLetter = capitalLetter;
    s.ArrowSign = arrowSign;

    foreach ((Sensor, uint) value in violations) {
        if (value.Item1 == s)
            return false;
    }

    foreach (Sensor value in auxiliaryParams) {
        if (value == s)
            return false;
    }

    return true;
}
public string getUniqueCode(uint vID) {
    if (isVertexInGraph(vID)) {
        foreach (Correspondence c in correspondences)
        {
            if (c.VertexID == vID)
                return c.SensorCode;
        }
        throw new UnexpectedError();
    }
    else
        throw new InvalidArgument();
}
public List<VertexNode> getVertexes(bool withoutMain = true)
{
    List<VertexNode> vList = new List<VertexNode>();
    foreach (Correspondence c in correspondences)
    {
        if (withoutMain && c.SensorCode == "Main") {

```

```

        continue;
    }
    vList.Add(new VertexNode(c.VertexID, c.SensorCode));
}
return vList;
}
public List<uint> getVertexesIDs() {
    List<uint> vList = new List<uint>();
    foreach (Correspondence c in correspondences)
    {
        vList.Add(c.VertexID);
    }
    return vList;
}
public List<uint> getAdjacentVertexes(uint vertexID)
{
    foreach (Vertex v in vertexes)
    {
        if (v.ID == vertexID)
        {
            return v.getAdjacentVertexes();
        }
    }
    return new List<uint>();
}
public List<string> getAvailableParams() {
    List<string> list = new List<string>();
    foreach (Sensor s in auxiliaryParams) {
        list.Add(s.UniqueCode);
    }
    return list;
}

public List<Event> getEvents(uint numOfTechLines = 1, OperatorLevel
operatorLevel = OperatorLevel.Third, InformationRepresentationMethod
informationRepresentationMethod =
InformationRepresentationMethod.MnemonicDiagram) {

    if (numOfTechLines < 1)
        throw new InvalidArgument();

    List<Event> events = new List<Event>();
    List<uint> path = new List<uint>();
    uint eventCounter = 1;

    foreach (Vertex v in vertexes) {
        if (v.AdjVCount == 1 && v.Type != VertexType.Main && v.Type !=
VertexType.Violation) {
            path = getWayUp(v.getAdjacentVertexes()[0]);
            path.Insert(0, v.ID);
            path.Add(mainVertexID);

            Event e = new Event(eventCounter++, path, numOfTechLines,
getSensorsCount(), getViolationInformationQuantity(path[path.Count -
2],numOfTechLines), operatorLevel, getDampedDeviceTotal(),
informationRepresentationMethod);

            foreach (uint tempV in path) {
                if (tempV != mainVertexID) {

                    Sensor s =
getSensor(getCorrespondence(tempV).SensorCode);
                    if (s.ControlMethod == SensorControlMethod.Request)

```

```

        {
            e.addRequestSensor(s);
        }
        else if (s.ControlMethod ==
SensorControlMethod.Auto) {
            e.addAutoSensor(s);
        }

        if (tempV != path[path.Count - 2]) {
            e.addVertexAndSubverticiesNumber(new
Event.VertexAndNumOfSubverticies((int)tempV,
getNumberVerticiesInSubtree(tempV)));
        }
    }

    events.Add(e);
}

return events;
}
public VertexNode getMain() {
    foreach (Correspondence c in correspondences) {
        if (c.VertexID == mainVertexID) {
            return new VertexNode(c.VertexID, c.SensorCode);
        }
    }
    throw new UnexpectedError();
}
public Point getVertexCenterPos(uint vID) {
    foreach (Vertex v in vertexes) {
        if (v.ID == vID)
            return v.CenterPos;
    }
    throw new NotFoundException();
}
public void setVertexCenterPos(uint vID, Point pos) {
    foreach (Vertex v in vertexes)
    {
        if (v.ID == vID) {
            v.CenterPos = pos;
            return;
        }
    }
    throw new NotFoundException();
}
public List<string> getAllSensors() {
    List<string> allSensors = new List<string>();

    foreach ((Sensor, uint) value in violations)
    {
        allSensors.Add(value.Item1.UniqueCode);
    }

    foreach (Sensor value in auxiliaryParams)
    {
        allSensors.Add(value.UniqueCode);
    }
}

```

```

        if (allSensors.Count == 0)
            return allSensors;
        return allSensors.Distinct().ToList();
    }
    public List<uint> setMode(GVSSMode newMode) {

        if (newMode == this.mode) {
            return new List<uint>();
        }

        List<uint> vertexesToRemove = new List<uint>();

        if (newMode == GVSSMode.RepetativeViolations)
        {
            //были уникальные - стали повторяющиеся
            foreach ((Sensor, uint) violation in violations) {
                auxiliaryParams.Add(violation.Item1);
            }

        }
        else if (newMode == GVSSMode.UniqueViolations)
        {
            //были повторяющиеся - стали уникальными
            // remove from auxiliary params, delete vertexes

            foreach ((Sensor, uint) violation in violations) {
                foreach (Sensor auxiliary in auxiliaryParams) {
                    if (violation.Item1.UniqueCode == auxiliary.UniqueCode)
                    {
                        auxiliaryParams.Remove(auxiliary);
                        break;
                    }
                }
            }

            List<uint> toRemoveVertexes = new List<uint>();

            foreach((Sensor, uint) violation in violations) {
                foreach (Correspondence c in correspondences)
                {
                    if (c.SensorCode == violation.Item1.UniqueCode &&
getVertex(c.VertexID).Type == VertexType.AuxiliaryParam) {
                        toRemoveVertexes.Add(c.VertexID);
                    }
                }
            }

            foreach (uint v in toRemoveVertexes) {
                vertexesToRemove.AddRange(delete(v));
            }
        }

        this.mode = newMode;
        if (vertexesToRemove.Count != 0)
        {
            return vertexesToRemove.Distinct().ToList();
        }
        else return new List<uint>();
    }

```

```

    }
    public List<string> getAllViolations() {

        List<string> v = new List<string>();
        foreach ((Sensor, uint) viol in violations) {
            v.Add(viol.Item1.UniqueCode);
        }
        return v;
    }

    public bool isSensorInGVSS(string uniqueCode) {
        bool isIn = false;

        foreach (Sensor s in auxiliaryParams)
        {
            if (s.UniqueCode == uniqueCode) {
                isIn = true;
                break;
            }
        }

        if (!isIn && mode == GVSSMode.UniqueViolations) {
            foreach ((Sensor, uint) v in violations) {
                if (v.Item1.UniqueCode == uniqueCode) {
                    isIn = true;
                    break;
                }
            }
        }

        return isIn;
    }

    public Sensor getSensor(string uniqueCode) {

        if (isSensorInGVSS(uniqueCode))
        {
            foreach (Sensor sens in auxiliaryParams) {
                if (sens.UniqueCode == uniqueCode)
                    return sens;
            }
            if (mode == GVSSMode.UniqueViolations) {
                foreach ((Sensor, uint) v in violations) {
                    if (v.Item1.UniqueCode == uniqueCode) {
                        return v.Item1;
                    }
                }
            }
        }
        throw new InvalidArgument();
    }

    public List<uint> deleteAll(string uniqueCode) {
        if (isSensorInGVSS(uniqueCode))
        {
            List<uint> toRemove = new List<uint>();
            foreach (Correspondence c in correspondences) {
                if (c.SensorCode == uniqueCode) {
                    toRemove.Add(c.VertexID);
                }
            }

            List<uint> deletedVertexesList = new List<uint>();

            foreach (uint v in toRemove) {

```

```

        deletedVertexesList.AddRange(delete(v));
    }

    deleteAuxiliary(uniqueCode);
    if (deletedVertexesList.Count != 0)
        return deletedVertexesList.Distinct().ToList();
    return deletedVertexesList;
}
else {
    throw new InvalidArgument();
}
}

public List<uint> getVertexChilds(uint vID) {
    if (isVertexInGraph(vID))
    {
        Vertex v = getVertex(vID);
        uint vLevel = v.Level;
        List<uint> adjVertexes = v.getAdjacentVertexes();
        foreach (uint adjV in adjVertexes) {
            if (getLevel(adjV) < vLevel) {
                adjVertexes.Remove(adjV);
                break;
            }
        }
        return adjVertexes;
    }
    else {
        throw new InvalidArgument();
    }
}

public void editSensor(string uniqueCode, SensorControlAgent
controlAgent, SensorControlMethod controlMethod, SensorRequestBodies
requestBody, SensorRequestPlacementMethod requestPlacementMethod, string
description) {
    foreach ((Sensor, uint) v in violations) {
        if (v.Item1.UniqueCode == uniqueCode) {
            v.Item1.ControlMethod = controlMethod;
            v.Item1.ControlAgent = controlAgent;
            v.Item1.Description = description;
            v.Item1.RequestBody = requestBody;
            v.Item1.RequestPlacementMethod = requestPlacementMethod;
            break;
        }
    }
    foreach (Sensor a in auxiliaryParams)
    {
        if (a.UniqueCode == uniqueCode)
        {
            a.ControlMethod = controlMethod;
            a.ControlAgent = controlAgent;
            a.Description = description;
            a.RequestBody = requestBody;
            a.RequestPlacementMethod = requestPlacementMethod;
        }
    }
}

public string getType(uint vID) {
    if (isVertexInGraph(vID))
    {
        if (vID == mainVertexID)
            return "Main";
    }
}

```

```

        foreach ((Sensor, uint) v in violations)
        {
            if (v.Item2 == vID)
                return "Violation";
        }

        return "Auxiliary";
    }
    else throw new InvalidArgument();
}

}

public float getEdgeWeight(uint firstV, uint secondV)
{
    if (doesTheEdgeExist(firstV, secondV))
    {
        foreach (Vertex v in vertexes)
        {
            if (v.ID == firstV)
            {
                return v.getEdgeWeight(secondV);
            }
            else if (v.ID == secondV)
            {
                return v.getEdgeWeight(firstV);
            }
        }
    }
    return -1;
}

public GVSSMode getMode() {
    return mode;
}

public bool isNull() {
    return vertexes.Count == 1 && correspondences.Count == 1 &&
auxiliaryParams.Count == 0 && violations.Count == 0;
}

}

}

```

ДОДАТОК Б**Копії слайдів презентації**

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АНАЛІЗУ ОПЕРАТОРСЬКОЇ ДІЯЛЬНОСТІ

РОЗРОБИВ: ст. групи КНТ-110м

Шнейдер М.А.

КЕРІВНИК: доцент, к.т.н.

Сердюк С.М.

Рисунок Б.1 – Слайд №1

- **Об’єкт дослідження** – процес проектування людино-машинних інтерфейсів (ЛМІ) АСУТП.
- **Предмет дослідження** – діяльність проектувальника ЛМІ АСУТП.
- **Мета роботи** – підвищення ефективності процесу проектування ЛМІ АСУТП шляхом розроблення і застосування програмного забезпечення на основі методів і моделей штучного інтелекту для аналізу діяльності операторів АСУТП.



Завдання дослідження

- проаналізувати методи кількісної оцінки діяльності операторів АСУТП та обрати метод, що відповідає цілям дослідження;
- розробити технічне завдання на програмний застосунок, що автоматизує діяльність проектувальника ЛМІ;
- розробити датасет для нейронної мережі;
- проектування та навчання нейронної мережі;
- розробити архітектуру системи;
- розробити моделі програмного забезпечення застосунку;
- розробити програмне та інформаційне забезпечення компонентів системи, що автоматизує діяльність проектувальника ЛМІ;
- перевірити адекватність моделі на якій базується розроблене програмне забезпечення.



НУ «Запорізька політехніка»

3

Рисунок Б.3 – Слайд №3

Порівняльний аналіз методів кількісної оцінки часу пошуку рішення оператором

Критерій \ Метод	Метод Губінського	Сімейство GOMS	Метод Галактіонова
Поширеність методу серед проектувальників ЛМІ	Не дуже поширений	Дуже поширений	Не дуже поширений
Набір базових операторів, що використовуються в методі/моделі	Обмежений	Дуже обмежений	Не обмежений
Складність опису паралельних процесів	Не має труднощів	Опис складний	Не має труднощів
Можливість кількісного прогнозування часу виконання завдання	Так	Так	Так
Можливість використання на ранніх етапах проектування ЛМІ	Так	Так	Так
Врахування контексту використання	Не враховує	Не враховує	Враховує
Врахування кваліфікації оператора	Не враховує	Не враховує	Враховує

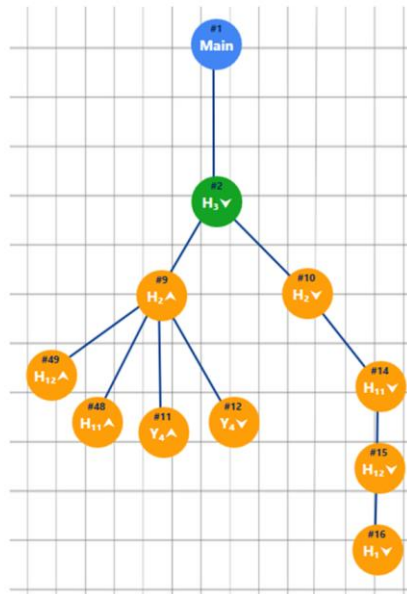


НУ «Запорізька політехніка»

4

Рисунок Б.4 – Слайд №4

Дослідження операторської діяльності (метод Галактіонова)



Час виявлення порушення: $T_{вп} = T_{вп} + \Delta T_{вп.з} + T_{іпрвп} + \Delta T'_{вп.зв} + \Delta T'_{вп.п.}$



НУ «Запорізька політехніка»

5

Рисунок Б.5 – Слайд №5

Фрагмент навчальної вибірки для нейромережі

№ події	№ вершини на рівні 1	Кількість розгалужень на рівні 1 від вершини рівня 0	Відносна ймовірність ознаки на рівні 1	№ вершини на рівні 2	Кількість розгалужень на рівні 2 від вершини рівня 1	Відносна ймовірність ознаки на рівні 2	№ вершини на рівні 3	Кількість розгалужень на рівні 3 від вершини рівня 2	Відносна ймовірність ознаки на рівні 3	№ вершини на рівні 4	Кількість розгалужень на рівні 4 від вершини рівня 3	Відносна ймовірність ознаки на рівні 4	№ вершини на рівні 5	Кількість розгалужень на рівні 5 від вершини рівня 4	Відносна ймовірність ознаки на рівні 5	Результат $I_{поді}$
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
4	1	7	0.143	1	2	0.5	3	4	0.25	0	0	0	0	0	0	5,000
5	1	7	0.143	2	2	0.5	5	1	1	1	1	1	1	1	1	3,000

Навчання для загальної структурної схеми подій, що може описувати понад 117 000 подій

Вихідний час виявлення причин несправності: $T_{вп} = a + b I_{поді}$,

$$I_{поді} = -\log_2 p(a_k)_i^{(j)} + \sum_{j=2}^{j=n} \log_2 \{m\}_i^{(j)}$$



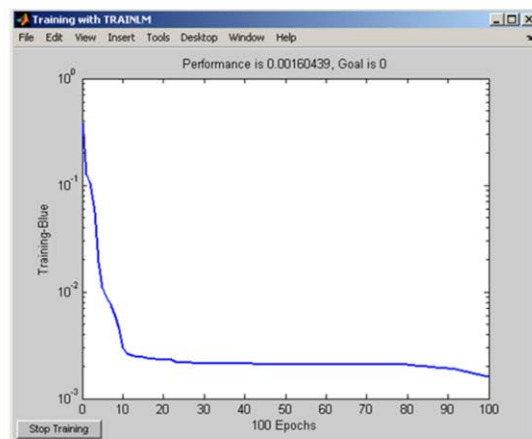
НУ «Запорізька політехніка»

6

Рисунок Б.6 – Слайд №6

Результати навчання для III-го рівня тренованості оператора

```
%нормування X
for i=1:1:size(x,1)
    for j=1:1:size(x,2)
        xN(i,j)=(x(i,j)-minx(j))/(maxx(j)-minx(j));
    end;
%нормування Y
for i=1:1:size(y,1)
    yN(i)=(y(i)-minY)/(maxY-minY);
end;
%завдання мережі та її тренування
net = newff(repmat([01],size(x,2),1),[2,1],{'logsig','logsig'},'trainlm');
net = train(net,xN,yN)
a=sim(net,xN)
```

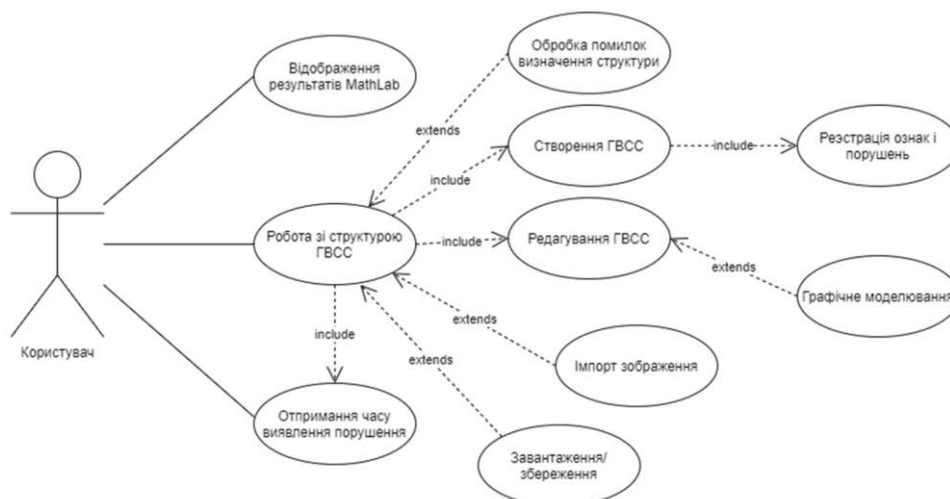


НУ «Запорізька політехніка»

7

Рисунок Б.7 – Слайд №7

Діаграма прецедентів



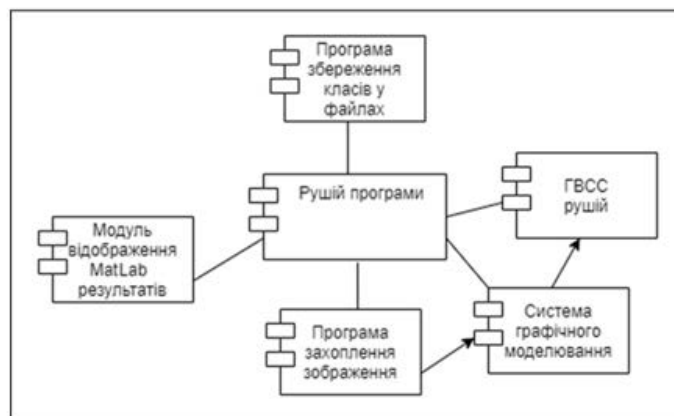
НУ «Запорізька політехніка»

8

Рисунок Б.8 – Слайд №8

Діаграма компонентів

Desktop-застосунок

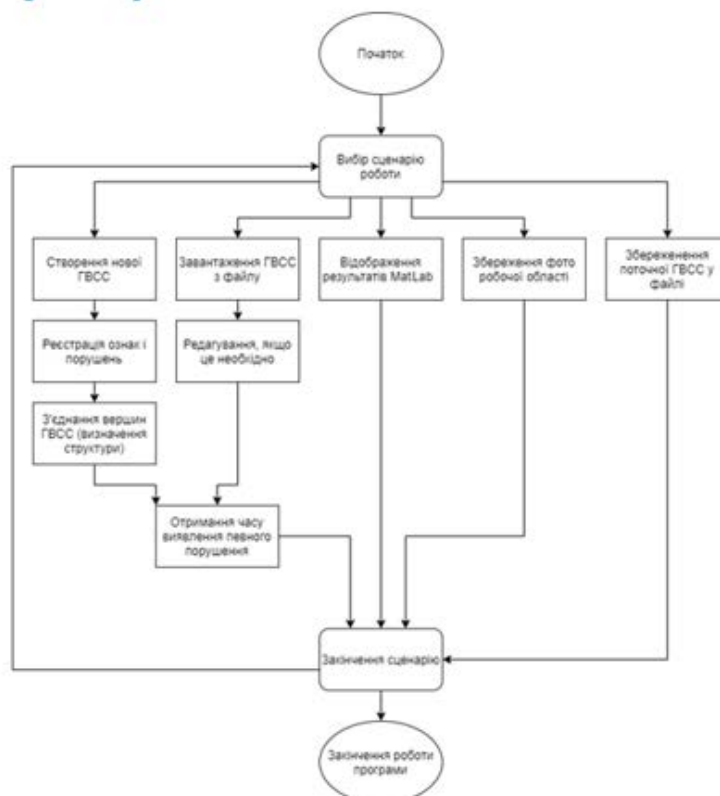


НУ «Запорізька політехніка»

9

Рисунок Б.9 – Слайд №9

Функціональна схема системи

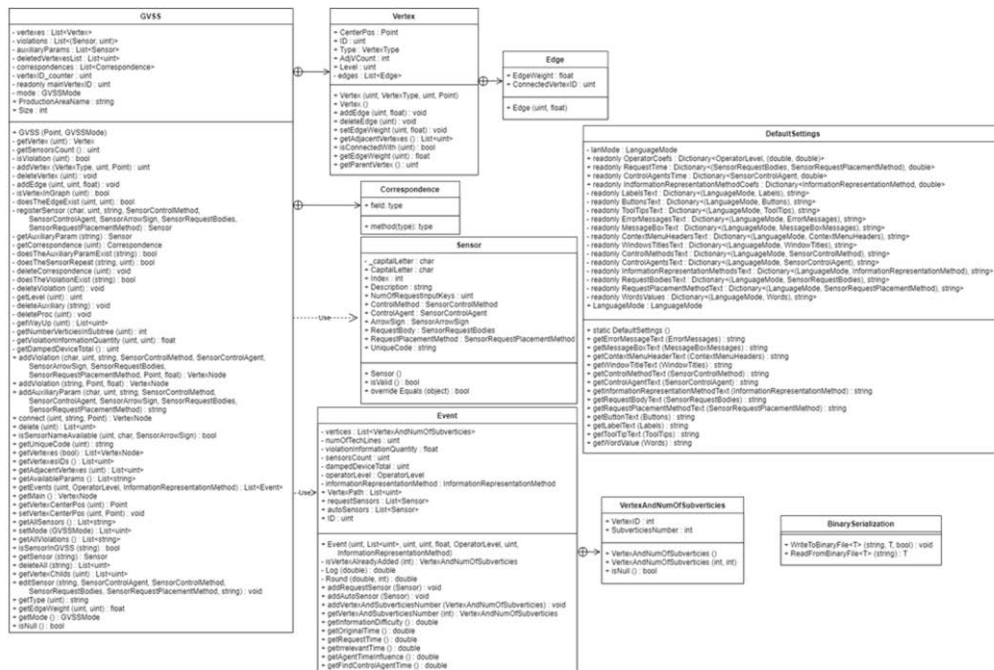


НУ «Запорізька політехніка»

10

Рисунок Б.10 – Слайд №10

Діаграма класів

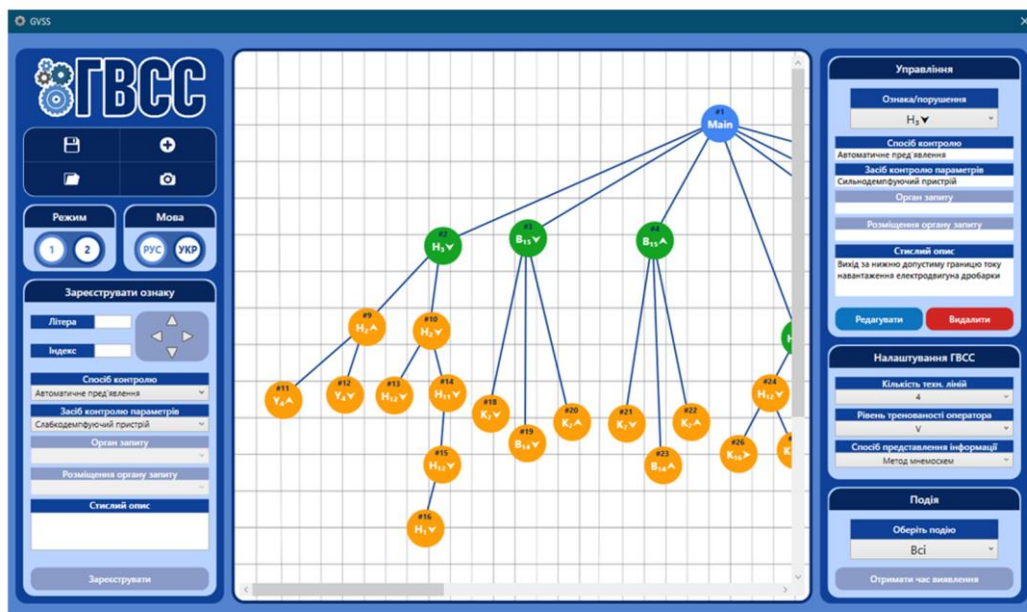


НУ «Запорізька політехніка»

11

Рисунок Б.11 – Слайд №11

Головна форма застосунку



НУ «Запорізька політехніка»

12

Рисунок Б.12 – Слайд №12

Приклади форм

Додати порушення

Додати ознаку до вершини

Редагування ознаки ГВСС



НУ «Запорізька політехніка»

13

Рисунок Б.13 – Слайд №13

Тестування застосунку

Ознака/порушення	Ознака/порушення	Ознака/порушення	Ознака/порушення	Ознака/порушення
H ₂ ▼	Y ₄ ▲	Y ₄ ▼	H ₁₁ ▼	H ₁₂ ▼
Спосіб контролю	Спосіб контролю	Спосіб контролю	Спосіб контролю	Спосіб контролю
Запит значень	Автоматичне пред'явлення	Автоматичне пред'явлення	Автоматичне пред'явлення	Автоматичне пред'явлення
Засіб запиту (пристрій)	Засіб контролю параметрів	Засіб контролю параметрів	Засіб контролю параметрів	Засіб контролю параметрів
Цифровий трикоординатний індикатор	Сильнодемпфуючий пристрій	Сильнодемпфуючий пристрій	Сильнодемпфуючий пристрій	Сильнодемпфуючий пристрій
Орган запиту	Орган запиту	Орган запиту	Орган запиту	Орган запиту
Номероналічач	Номероналічач	Номероналічач	Номероналічач	Номероналічач
Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту
В агрегатній групі	В агрегатній групі	В агрегатній групі	В агрегатній групі	В агрегатній групі
Стислий опис	Стислий опис	Стислий опис	Стислий опис	Стислий опис
Нижня допустима границя току навантаження є дв. транспортера 3	Верхня допустима границя току навантаження є дв. транспортера 3	Нижня допустима границя току навантаження є дв. транспортера 3	Вихід за нижню допустиму границю току навантаження лівого електродвигуна	Вихід за нижню допустиму границю току навантаження є дв. правого змішувача
Ознака/порушення	Ознака/порушення	Ознака/порушення	Ознака/порушення	Ознака/порушення
H ₁ ▼	H ₂ ▼	H ₁₁ ▲	H ₁₂ ▲	H ₂ ▲
Спосіб контролю	Спосіб контролю	Спосіб контролю	Спосіб контролю	Спосіб контролю
Запит значень	Автоматичне пред'явлення	Автоматичне пред'явлення	Автоматичне пред'явлення	Запит значень
Засіб запиту (пристрій)	Засіб контролю параметрів	Засіб контролю параметрів	Засіб контролю параметрів	Засіб запиту (пристрій)
Цифровий трикоординатний індикатор	Сильнодемпфуючий пристрій	Сильнодемпфуючий пристрій	Сильнодемпфуючий пристрій	Цифровий трикоординатний індикатор
Орган запиту	Орган запиту	Орган запиту	Орган запиту	Орган запиту
Номероналічач	Номероналічач	Номероналічач	Номероналічач	Номероналічач
Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту	Розміщення органу запиту
В агрегатній групі	В агрегатній групі	В агрегатній групі	В агрегатній групі	В агрегатній групі
Стислий опис	Стислий опис	Стислий опис	Стислий опис	Стислий опис
Вихід за нижню допустиму границю току навантаження електродвигуна дробарки	Вихід за нижню допустиму границю току навантаження електродвигуна дробарки	Вихід за нижню допустиму границю току навантаження лівого є дв. змішувача	Вихід за верхню допустиму границю току навантаження є дв. правого змішувача	Верхня допустима границя току навантаження електродвигуна транспортера 3
Налаштування ГВСС				
Кількість тек. ліній				
4				
Рівень тривожності оператора				
V				
Спосіб представлення інформації				
Метод множення				

Приклад опису ГВСС, на основі якої проводились розрахунки



НУ «Запорізька політехніка»

14

Рисунок Б.14 – Слайд №14

Результати розрахунків

$$I_{\text{под}4} = -\log_2 1/4 + \log_2 1 + \log_2 2 + \log_2 4 = 5 \text{ (дв. одиниць)}$$

$$T_{\text{ВП}} = 0,42 + 0,025 \times 5 = 0,545 \text{ с}$$

$$\Delta T_{\text{ВП.З}} = (5,4 + 0,35) \times 1 = 5,75 \text{ с}$$

$$T_{\text{ВП}}^{\text{іпр}} = 0,2 \times 0,075 \times 10 = 0,15 \text{ с}$$

$$\Delta T'_{\text{ВП.ЗВ}} = 0,4 \text{ с}$$

$$\Delta T'_{\text{ВП.П.}} = 1,8 \text{ с}$$

$$T_{\text{ВПК}} = 0,54 + 5,75 + 0,15 + 0,4 + 1,8 = 8,64 \text{ с}$$

Час розпізнавання події	
Подія №4	
Кількість інформації	Початковий час
5 д.е.	0.54 сек.
Час, витрачений на запити	Затримка (іпр. інформ.)
5.75 сек.	0.15 сек.
Вплив засобів предст.	Пошук засобів предст.
0.4 сек.	1.8 сек.
Загальний час виявлення події	
8.64 сек.	



НУ «Запорізька політехніка»

15

Рисунок Б.15 – Слайд №15

Висновки

- **Наукова новизна** результатів дослідження полягає у тому, що дістав подальшого розвитку відомий метод аналізу операторської діяльності Галактіонова. Метод, що розроблено, на відміну від попереднього, має засоби для формалізації процесу аналізу діяльності оператора АСУТП проектувальником ЛМІ АСУТП, шляхом розроблення і застосування програмного забезпечення на основі моделей штучного інтелекту.
- **Практична цінність** комплексної роботи полягає в тому, що на підставі проведених досліджень розроблено програмне та інформаційне забезпечення, що забезпечують підвищення ефективності діяльності проектувальника людино-машинних інтерфейсів АСУТП.



НУ «Запорізька політехніка»

16

Рисунок Б.16 – Слайд №16

**Доклад закінчено,
дякую за увагу!**



НУ «Запорізька політехніка»

17

Рисунок Б.17 – Слайд №17