Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-20М-1,4
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Тарасов О.В.
«____» _____ 2021 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Володимир ПЕТРЕНЮК
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань ._12 _____*"Інформаційні технології"*_____

Спеціальність *122 "*Комп'ютерні науки*"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2021 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Тарасову Олексію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 39-13 від 02.08.2021 року

3. Строк подання студентом роботи до захисту *10.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*7. Економічна ефективність розробленої*
2.Перегляд аналогічних існуючих систем.	*програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	
6. Наукова новизна	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2021	14.11.2021
Охорона праці	Оришака О.В.	06.10.2021	16.11.2021

7. Дата видачі завдання « 6 » вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	10.12.2021 р.	

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Тарасов О.В. Дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Метою розробки є дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Об'єктом дослідження є процес інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Предметом дослідження є методи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Методи дослідження базуються на методах хмарних технологій, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі Embarcadero RAD Studio Delphi 10.3.2.

Ключові слова: Комп'ютерні науки, хмарні технології, бізнес-процеси

ABSTRACT

Tarasov O.V. Research and software implementation of the system of intelligent implementation of cloud technologies in the business processes of the enterprise. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualification work on the second (master's) level of higher education the software which is intended for system of intellectual introduction of cloud technologies in business processes of the enterprise is developed.

The purpose of development is research and software implementation of the system of intelligent implementation of cloud technologies in the business processes of the enterprise.

The object of research is the process of intelligent implementation of cloud technologies in the business processes of the enterprise.

The subject of research is the methods of intelligent implementation of cloud technologies in the business processes of the enterprise.

Research methods are based on methods of cloud technologies, methods of mathematical statistics, methods of software development.

The result is the software implementation of the system of intelligent implementation of cloud technologies in the business processes of the enterprise.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in the environment Embarcadero RAD Studio Delphi 10.3.2.

Keywords: Computer science, cloud technologies, business processes

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ААУ	–	автономне адаптивне управління
БД	–	блок датчиків
БЗ	–	база знань
БОС	–	блок оцінки стану
БПР	–	блок прийняття рішень
ГІП	–	графічний інтерфейс користувача (GUI)
ВО	–	виконавчий орган
ЕОМ	–	електрона обчислювальна машина
КА	–	кінцевий автомат
НРС	–	недетермінований автомат Рабина-скотта
НМ	–	нейронна мережа
МНРС	–	модифікований недетермінований автомат Рабина-скотта
НАНУ	–	Національна академія наук України
ОУ	–	об'єкт управління
ПЧО	–	просторово-часовий образ
ВВ	–	випадкова величина
ПЗ	–	програмне забезпечення
СПДНМ	–	система побудови й дослідження нейронних мереж
УС	–	управляюча система
ФР	–	функція розподілу
ФРО	–	апарат формування й розпізнавання образів
Z^+	–	множина ненегативних цілих чисел
$G(V, N)$	–	граф із множиною вершин V і множиною ребер N
$i \to j$	–	ребро, спрямоване з вершини i у вершину j
$X \leftrightarrow Y$	–	взаємооднозначне відображення множини X на множину Y
$\pi(X)$	–	множина кінцевих підмножин множини X

$R[a,b]$	–	множина речовинних чисел на $[a,b]$, $R \equiv R[-\infty,+\infty]$
B^N	–	простір двійкових векторів розмірності N
Λ	–	порожнє слово із множини вхідних слів КА
0	–	*неправда* у вираженні тризначної логіки
1	–	*істина* у вираженні тризначної логіки
$\lozenge$	–	*невизначеність* у вираженні тризначної логіки
$\vec{X} \subset \vec{Y}$	–	$\vec{X}$ є підвектор (сукупність обраних компонентів) вектора $\vec{Y}$
$X \supset Y$	–	клас Y є нащадком класу X

ВСТУП

Актуальність теми. Традиційний проектний бізнес поступово міняється внаслідок поширення таких підходів як пропозиція інфраструктури у вигляді сервісу. У теж час конкуренція на ринку системної інтеграції посилюється в результаті прагнення великих вендорів працювати із замовниками прямо, пропозиції IT-послуг на базі власних ЦОД з боку телекомунікаційних операторів і інших факторів. На це накладаються нерадісні перспективи стагнації українського проектного ринку. Все це змушує системних інтеграторів мінятися самим і шукати нові джерела росту.

Всі замовники хочуть говорити про цифрову трансформацію, але мало хто представляє, що це в дійсності таке. Щоб задовольнити запит з боку замовників на зміни й самим відповідати вимогам ринку, у даній роботі покажемо, як системний інтегратор здійснив цифрову трансформацію власного бізнесу. За підсумками перетворень у компанії з'явилося два нових бізнес-підрозділи – консалтингу в області цифрової трансформації й хмарних сервісів.

Для здійснення трансформації з однієї сторони ми зберігаємо, підтримуємо, і плавно видозмінюємо поточний бізнес, з іншої сторони за допомогою інституту трансформації ми піддали ревізії практично всі аспекти нашого життя. Зміст транформації полягає в тому, що за допомогою простих методик удалося визволити інтелектуальний потенціал співробітників.

Одним з наслідків стала поява продуктового напрямку. Насамперед під продуктами в даному контексті розуміються різні види сервісів і готових рішень, які можуть бути тиражовані. Це дозволяє збільшити клієнтську базу за рахунок тих компаній, хто не готовий платити за повністю кастомізоване рішення.

Багато клієнтів зацікавлені в досвіді цифрової трансформації. Ми відразу надаємо їм цифрові інструменти, за допомогою яких вони могли б заробляти більше грошей.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

– Дослідження системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

– Програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Об'єктом дослідження є процес інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Предметом дослідження є методи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Методи дослідження базуються на методах хмарних технологій, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

– Розроблено вітчизняний продукт інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями

параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LV Науково-технічна конференція здобувачів вищої освіти «Наука – виробництву», 2021, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №12.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Інфраструктурні хмарні сервіси, керовані сервіси в області інфраструктури й експертиза по побудові приватних хмар були об'єднані в підрозділі «Хмарні Сервіси». Для продуктового бізнесу характерні інші правила гри й процеси, ніж ті, що властиві проектній діяльності. Якщо по завершенні проекту спілкування із замовником зводиться до мінімуму, то при наданні сервісів із замовником доводиться спілкуватися щодня.

Виділення в окремий бізнес дало крім того більшу свободу дій і дозволило прискорити прийняття рішень. Початий редизайн внутрішніх процесів дозволив оптимізувати роботу й, як затверджується, «позитивно позначився на ціновій політиці».

Хмарні сервіси дозволяють залучати нових замовників, з ким до того ніколи не працював – по внутрішній статистиці 70% це саме нові замовники. При цьому більше 80% замовників розміщають у хмарі бізнес-критичні сервіси. Для забезпечення необхідного ступеня надійності хмара розміщається у власному ЦОД, сертифікованому на відповідність вимогам і Tier 3.

Для пошуку перспективних тиражуємих рішень створена група продуктових інновацій. Чому ми цим зайнялися саме зараз? Ми усвідомили, що вміємо це робити. Прикладами можуть служити як підрозділ хмарних сервісів, так і підрозділ комунікацій, де успішно розвиваються керовані сервіси, однак, щоб процес виводу нових продуктових напрямків прискорився, його треба виділити.

Завдяки трансформації й новим напрямкам удасться збільшити виторг на 20%, однак основний дохід (90% оберту) принести – і ще видимо довго буде приносити – проектний бізнес.

1.2 Область застосування

Областю застосування є бізнес-процеси підприємства. Бізнес-процес (процес) – це сукупна послідовність дій по перетворенню ресурсів, отриманих на вході, у кінцевий продукт, що має цінність для споживача, на виході.

Завдяки такому визначенню, стає зрозуміло, що бізнес-процеси існують усередині кожної організації, незалежно від того, формалізовані вони чи ні. В організації може бути прийнятий функціональний підхід до керування, що розглядає компанію як набір підрозділів, кожен з яких виконує певні функції.

У цьому випадку окремі підрозділи орієнтовані на виконання своїх власних показників, але не завжди – на кінцевий результат компанії, що може викликати конфлікт інтересів між підрозділами й негативно позначатися на загальній результативності бізнесу. Приведемо типовий конфлікт («грозову хмару», у термінах Теорії обмежень) між відділами продажів і закупа торговельної компанії. Відділ продажів для збільшення оберту вимагає забезпечити максимально можливі асортименти й підтримувати постійна наявність товару на складі, а відділ поставок закуповує вузькі асортименти товару більшими партіями, тому що його головний показник роботи – одержання більше низької ціни від постачальника для зниження витрат – ніяк не пов'язаний зі збільшенням обсягу продажів компанії.

Переваги процесного підходу перед функціональним

Процесний підхід розглядає бізнес як набір процесів – основних бізнес-процесів, що управляють процесів (які ставлять цілі) і підтримуючих. Основні бізнес-процеси – це процеси, які безпосередньо заробляють гроші. Підтримуючі – процеси, без яких не можуть існувати основні бізнес-процеси, це процеси забезпечення різноманітними ресурсами.

Кожний бізнес-процес має:

– свою певну мету, підлеглу загальної цілі компанії;

– власника, що може управляти ресурсами й відповідає за виконання процесу;

– ресурси;

– систему контролю якості й виправлення помилок;

– систему показників процесу.

Сукупність всіх дій по перетворенню матеріалів і інформації в закінчений продукт для клієнта називається потоком створення цінності. Потік створення цінності зручно представляти графічно – у вигляді карти бізнес-процесів. На рисунку нижче зображена карта бізнес-процесів компанії. Карта дозволяє наочно побачити потік створення цінності в цілому, зрозуміти послідовність і взаємозв'язок процесів, а також можливості поліпшення.

Технологія опису бізнес-процесу робить всі операції компанії прозор і зрозумілими, дозволяє аналізувати операції й знаходити в них проблеми, що приводять до збоїв. Головне, що бізнес-процеси дозволяють розуміти взаємодія між розрізненими підрозділами: що, кому й для чого вони передають або приймають на кожному етапі. Як наслідок, процесний підхід значно спрощує адаптацію нових співробітників і знижує залежність роботи компанії від людського фактора. Важливо, що процесна система спрощує керування операційними витратами.

Наявність проробленої системи бізнес-процесів значно спрощує приведення діяльності компанії на відповідність вимогам стандартам якості ISO 9001:2015.

Впровадження СМК на підприємстві в обов'язковому порядку вимагає створення й опису бізнес-процесів.

Розробка бізнес-процесів

Розглянемо порядок розробки бізнес-процесів. Для початку необхідно створити робочу команду проекту зі співробітників компанії. Звичайно, однієї робочої команди буває недостатньо. Тоді до її діяльності залучають тимчасову

групу з підрозділів клієнтів і постачальників конкретний бізнес-процесу, які забезпечують входи, виходи й ресурси бізнес-процесу.

Щоб зрозуміти, як функціонує система й зберегти накопичений досвід, спочатку записують, як процес реально функціонує зараз. Потрібно пам'ятати, що метою опису є виявлення зв'язків між діями, що уживаються, а не фіксування дрібних подробиць. Тому опис бізнес-процесів рекомендується стандартизувати, використовуючи стандартні форми й карти процесу.

Описувати бізнес-процес рекомендується методом послідовних наближень. Після завершення опису бізнес-процесу рекомендується провести роботу з його поліпшення (повторити цикл дій до одержання прийнятного результату).

В описі бізнес-процесу можна виділити наступні розділи:

- Стандартні форми бізнес-процесу.
- Карта бізнес-процесу.
- Маршрути бізнес-процесу.
- Матриці бізнес-процесу.
- Блок-схеми бізнес-процесу.
- Опис стиків бізнес-процесу.
- Допоміжні описи бізнес-процесу.
- Розгорнутий опис бізнес-процесу.
- Документування бізнес-процесу.
- Визначення показників і індикаторів бізнес-процесу.
- Регламент виконання бізнес-процесу.

Розглянемо докладніше кожний етап.

1. Стандартні форми опису бізнес-процесу

Рекомендуємо використовувати типовий зразок стандартної форми опису бізнес-процесу. Це дозволить домогтися єдиного підходу до фіксування процесу різними людьми, що потім значно полегшить аналіз процесів.

2. Карта бізнес-процесу

Карта бізнес-процесу – графічне подання бізнес-процесу у вигляді блок-схеми. Зверніть увагу, що кожному учасникові бізнес-процесу відповідає окремий стовпець. Рядки – це тимчасові інтервали. Оформлена карта дозволяє синхронізувати операції й простежити шлях проходження інформації між підрозділами компанії.

На етапі складання карти бізнес-процесу, від співробітника, що виконує цю роботу, не потрібно компетенції в області описуваних процедур бізнес-процесу. Він тільки фіксує знання виконавців, що і як вони роблять. Необхідно одержати відповіді на питання:

– Яким документом завершується робочий цикл, щоб його можна було почати спочатку?

– Кому передається цей документ?

– Що цьому передує?

– Хто залучений у цей процес усередині й поза організацією?

– Хто видає завдання для запуску процесу?

При складанні карти бізнес-процесу варто скористатися популярною питальною формулою 5W1H. Коротко, це 5 питань W:

– Who?(Хто робить дану операцію?).

– Why? (Чому або навіщо виконується ця операція?).

– What? (Що являє собою ця операція?).

– When? (Коли потрібно проводити дану операцію?).

– Where? (Де виробляється операція?).

і одне питання H

– How? (Як відбувається ця операція? Чи можна зробити це інакше або внести поліпшення?).

Якщо карта виходить занадто складною – це сигнал про те, що в керуванні організацією немає належного порядку.

3. Маршрути бізнес-процесу

У реальних бізнес-процесах часто бере участь кілька підрозділів підприємства. Для них необхідно розподілити ролі в процесі. Крім того, зустрічаються розгалуження й паралельні дії. Тому подання у вигляді маршрутів дуже зручно. Маршрути дають нам логістичну схему процесу – рух матеріалів, людей, грошових і інформаційних потоків. Блок-схеми використовуються для розшифровки логіки дій команди.

4. Матриці бізнес-процесу

Матриця (таблиця) аналізу взаємодії процесів дозволяє виділити найважливіший бізнес-процеси, установити їхній взаємозв'язок і оцінити ступінь впливу процесів на функціонування СМК.

Аналіз ланцюжка процесів виявляє, що обмін інформацією йде між всіма підпроцесами. Ланцюжок процесів йде з лівого верхнього кута в правий нижній. Внутрішні відносини « постачальник-споживач» показані як прямокутники, у яких зазначені вимоги для дій, що виконуються раніше.

5. Складання блок-схеми бізнес-процесу

Блок-схема процесу являє собою наочну схему всього ланцюжка взаємин між всіма учасниками бізнес-процесу (споживачами, постачальниками й виконавцями). У процесі складання блок-схеми ставляться питання:

– Чи порівнянна цінність від даний бізнес-процесу з витратами на його проведення?

– Наскільки він інтегрований з іншими бізнес-процесами?

– Чи можуть бути відразу виявлені помилки цього бізнес-процесу?

– Що зроблено для поліпшення й забезпечення якості цього бізнес-процесу?

6. Опис стиків бізнес-процесів

Сутужніше всього описувати діяльність підприємства на стиках бізнес-процесів. Згода між власниками процесів іноді одержати дуже складно. Спочатку складіть опис виходів. Спочатку впишіть їх до реєстру, потім визначите

показники результативності й значення, до яких потрібно прагнути. Опишіть процес виміру цих показників. Продумайте можливість переходу від них до інших показників ефективності, що цікавить інших користувачів.

Потім складіть аналогічний опис входів.

7. Допоміжні описи бізнес-процесів

Як допоміжний опис використовуються компоновочні схеми, мнемосхеми, діаграми Ганта й сіткові графіки. Дві останні зручно використовувати для процесів керування проектами.

8. Розгорнутий опис бізнес-процесів

Розгорнутий опис бізнес-процесу може бути в будь -якій зручній для підприємства формі, але повинне містити основні положення:

– повне найменування бізнес-процесу;

– код бізнес-процесу;

– визначення бізнес-процесу, що розкриває його основний зміст;

– ціль бізнес-процесу;

– власник бізнес-процесу, відповідальний за перспективне планування процесу;

– керівник бізнес-процесу, відповідальний за поточне ведення процесу;

– нормативи бізнес-процесу;

– входи бізнес-процесу (потоки, що надходять ззовні й підлягають перетворенню);

– виходи бізнес-процесу (результати перетворення);

– ресурси, якими розташовує бізнес-процес;

– бізнес-процеси внутрішніх і зовнішніх постачальників – джерела входів;

– бізнес-процеси споживачів – користувачі результатів розглянутий бізнес-процесу;

– вимірювані параметри процесу;

– показники результативності процесу.

9. Документування бізнес-процесу

Бізнес-процеси, що входять у систему СМК, підлягають документуванню. Найбільш зручною формою опису є процедура. Бізнес-процес може бути описаний однієї або декількома процедурами, залежно від складності. Зручно зробити єдиний вид для опису всіх бізнес-процесів.

10. Визначення показників і індикаторів бізнес-процесу

Бізнес-процес повинен бути охарактеризований якимись показниками, щоб процес можна було виміряти й оцінити його ефективність. Всі показники входять в 4 основні групи:

– якість;

– час виконання;

– кількість;

– витрати.

Крім того, прийнято виділяти особливі групи – групу індикаторів бізнес-процесу, групу вимог, групу забезпечення бажаного протікання процесу, групу рекомендацій.

Група індикаторів бізнес-процесу показує ступінь досягнення цілі.

Група вимог містить у собі:

– людські ресурси;

– інфраструктура;

– умови виробничого середовища.

Група забезпечення бажаного протікання процесу:

– інформація;

– інструкції з виконання робіт;

– час.

Група рекомендацій:

– фінанси;

– логістика;

– постачальники;

– партнери й т.д.

11. Регламент виконання бізнес-процесу

Великі бізнес-процеси доцільно оформляти у вигляді окремого документа «Регламенту виконання бізнес-процесу». Інші бізнес-процеси можуть бути оформлені у вигляді положень про підрозділ і посадових інструкцій.

У регламент варто закласти вимоги, що забезпечують відповідність циклу Шухарта-Демінга:

– визначення планових показників бізнес-процесу на наступний період;

– аналіз власником бізнес-процесу відхилень від нормального ходу процесу і їхнє документування;

– аналіз результативності коригувальних заходів;

– формування звітності для вищестоящого керівництва.

Розробка й опис бізнес-процесів – перший крок на шляху впровадження СМК на підприємстві. Попереду – постійна й кропітка робота з їхнього доведення до всього персоналу, аналізу й, якщо буде потреба, впровадженню коригувальних дій.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Програмне забезпечення в поданні процесів

Подання процесів також може проводиться із застосуванням спеціального програмного забезпечення. Використання автоматизованих методів дозволяє знизити витрати, провести симуляцію й візуалізацію можливих рішень, а також застосовувати й модифікувати раніше розроблені рішення. Існують різні програмні продукти для рішення завдань аналізу й організації процесів. Їх можна розділити на 3 групи:

– Стандартні графічні пакети, для подання процесів в електронному виді (візуалізація). Наприклад: ABC-FlowCharter, CorelFlow, Visio.

– Програмне забезпечення для аналізу побудовано на базі графічних пакетів і дозволяє, поряд з візуалізацією, обробляти деякі дані процесів. Наприклад: Ablauf-Profi, Proplan, Vamos-BE.

– Процесно-орієнтоване програмне забезпечення. Ця група пропонує широкі функціональні можливості. Звичайно в даних продуктах реалізовані модулі для аналізу процесу, моделювання й візуалізації, а також підтримуються оцінка й документація. Деякі системи дозволяють будувати анімаційні моделі. Наприклад: SYCAT, ARISToolset, AENEIS, AIBAS.

Коли ми говоримо про створення бізнес-процесів, багато хто мають на увазі побудова блок-схем бізнес-процесів. У зв'язку з цим одне із частих питань – які інструменти краще використовувати? Адже без гарних інструментів моделювання бізнес-процесів стає непростим завданням.

Я підготував для вас добірку й короткий опис інструментів по керуванню й моделюванню бізнес-процесів.

Звичайно, існує велика кількість різних інструментів моделювання й керування процесами. Моїм завданням було розповісти про тих, які не вимагають величезних проектів інтеграції й можуть бути використані з мінімальними витратами. До речі, із цієї причини я не став розглядати платформи ARIS, IBM і т.д.

Інструменти керування бізнес-процесами

BizAgi Suite

Якщо ви хочете одержати не тільки моделі й описи бізнес-процесів, але й створити застосунки, які виконуються, по них, то це саме те, що потрібно. BizAgi Suite складається, по суті, із двох модулів – BizAgi Modeler, що використовується для моделювання й опису бізнес-процесів, і BizAgi Studio, що дозволяє перетворити моделі у застосунки, які виконуються. Класно те, що це не вимагає навичок програмування, тобто кожному під силу робити додатка.

Застосунок, що виконується – це застосунок на базі BizAgi Engine, що перетворює модель у програму. Наприклад, ви можете створити модель узгодження заявки на закупівлю й перетворити її в застосунок, що дозволить учасникам процесу виконувати в цьому додатку всі операції процесу – створення заявки, проходження заявки через різні стадії узгодження, коментування, доробки заявки й т.д.

Коротше, BizAgi Suite – це крутий засіб автоматизації й контролю процесів. Воно дозволяє гарантувати виконання процесів відповідно до опису. Переоцінити таку можливість із погляду керування неможливо.

Функціонал і особливості:

- Моделювання бізнес-процесів, їхня перевірка й аналіз.
- Створення опису бізнес-процесів.
- Створення додатків, що виконуються, на базі моделей.
- Виконання й відстеження процесів у реальному часі.

– Призначення процесів співробітникам.

– Призначення інших ресурсів бізнес-процесам.

– Програма українською мовою.

Вартість:

– Безкоштовно, до 20 співробітників.

Резюме:

Дуже рекомендую всім, хто не знає, із чого почати, хоче спробувати свої сили в керуванні бізнес-процесами й прагне одержати інструмент, заточений під ваші унікальні процеси.

ELMA BPM

Ізюминка програми полягає в можливості інтеграції із платформою 1С, що, безумовно, досить привабливо для українських компаній. Що це значить? Це значить, що все відбувається в 1С буде відбите в ELMA. І навпаки.

ELMA дозволяє виконувати й відслідковувати виконання процесів у реальному часі. Для побудови моделей використовується нотація BPMN 2.0. До речі, саме завдяки співробітникам ELMA нотація була перекладена на українську мову. За що їм велике людське спасибі.

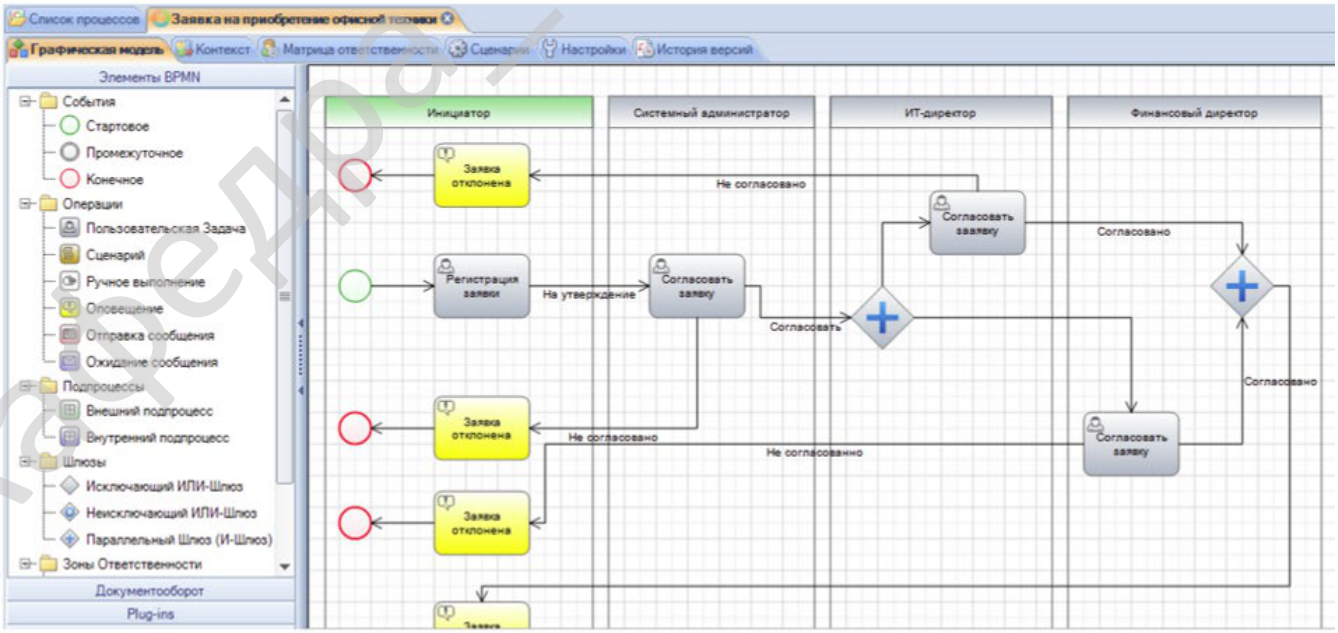

Рисунок 2.1 – Інтерфейс користувача ELMA

Дуже могутньо працює система документообігу в системі. Всі документи мають класифікацію по типах, розсортовані по папках, мають правила створення й роботи й т.д. Звичайно, буде потрібно час, щоб привести ваші документи в необхідний вид і відповідність системі, але воно того коштує. Якщо все зроблено правильно, то ви запросто зможете відстежити життєвий цикл будь-якого документа.

Існують додаткові модулі – Проекти, CRM і т.д. Але їх не пробував, тому нічого не можу сказати.

Інтеграцією й навчанням по роботі з ELMA компанія займається самостійно. Судячи з реалізованих проектів, можна сказати, що вони знають свою справу.

Функціонал і особливості:

– Побудова моделей бізнес-процесів.

– Призначення ролей бізнес-процесів співробітникам.

– Виконання й відстеження процесів у реальному часі.

– Системна робота з документообігом.

– Зручна “довідка”.

– Відмінна підтримка.

– Інтеграція з 1С.

Вартість:

– 77 000 грн. за 10 ліцензій ELMA Standart. Це мінімальна кількість. На мій погляд, вартість цілком адекватна функціоналові.

Резюме:

Ви твердо вирішили займатися керуванням бізнес-процесами, їхньою автоматизацією й поліпшенням? Ви прив'язані до 1С? Тоді ELMA – це те, що потрібно.

Business Studio

Так само як і ELMA, це українська розробка. Напевно, самий розкручений інструмент для керування бізнес-процесами на вітчизняному ринку. Перша версія

побачила світло в 2004 році. Уперше я зштовхнувся із цією програмою в 2006. На той момент це було найкраще рішення.

У принципі, у програмі все довольно стандартно – визначаємо цілі компанії, моделюємо процеси, які дозволяють досягати цілей, призначаємо відповідальних з дерева оргструктури, відзначаємо використовувані в процесах ресурси.

Дуже примітно, що для постановки цілей використовується концепція Системи збалансованих показників. Це одна із самих успішних методик перекладу стратегії компанії у відчутний і зрозумілий вид.

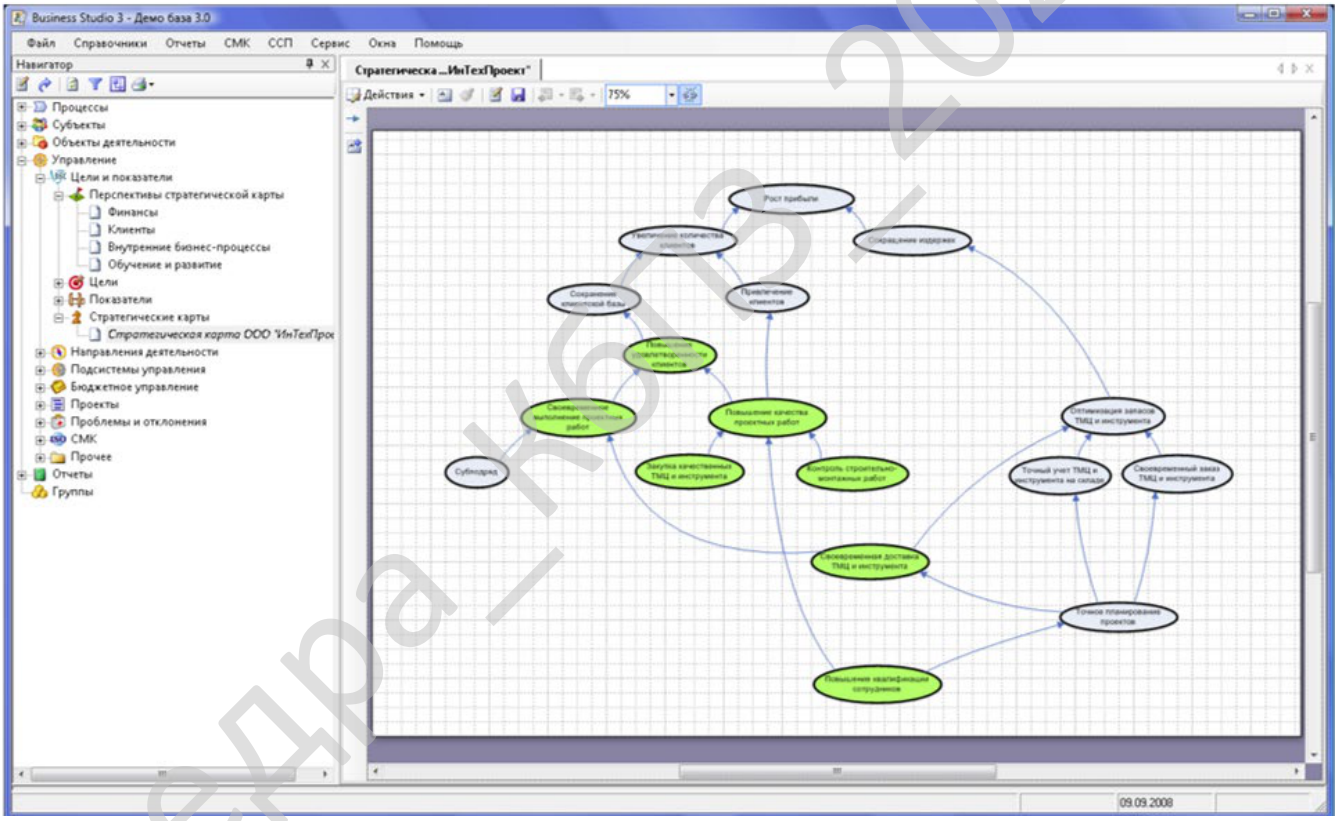

Рисунок 2.2 – Інтерфейс користувача Business Studio

Побудова бізнес-процесів, як часто відбувається, виробляється зверху долілиць. Програма підтримує кілька нотацій моделювання: IDEF, eEPC, BPMN і ще трохи інших.

Є присутнім можливість імітаційного моделювання, проведення функціонально-вартісного аналізу й автоматичної генерації документів, наприклад, посадових інструкцій. Документи відповідають вимогам законодавства, що істотно полегшує роботу. Виконання й моніторинг процесів відбувається через інтеграцію з іншими системами, наприклад, ELMA.

Функціонал і особливості:

– Моделювання процесів у різних нотаціях.

– Автоматична генерація документів.

– Постановка цілей компанії по Системі збалансованих показників.

– Інтеграція зі сторонніми системами..

– Контроль виконання процесів.

– База знань.

Вартість:

– Ціноутворення гнучке, так що для визначення вартості необхідно звернутися до консультантів компанії. Т.к. я не зіштовхувався з покупкою даного ПО в останні парі років, то порядок цифр мені невідомий.

Резюме:

Система потужна. Але складна. Будуть потрібні серйозні витрати, у першу чергу тимчасові – для налагодження й інтеграції системи. Найкраще, якщо у вас буде відділ або просто трохи бізнес-аналітиків, які візьмуть на себе цю роботу. Робота із програмою вимагає глибокого розуміння методик і специфіки програми.

Моделювання бізнес-процесів

Visual Paradigm

Скажу відверто, це краща програма для моделювання й опису бізнес-процесів. Більше зручного, функціонального й гнучкого інструмента для моделювання я не зустрічав. Почнемо з того, що VP підтримує велику кількість нотацій, блок-схем і моделей. Починаючи від стандартних нотацій IDEF, eEPC і BPMN і закінчуючи схемами баз даних, діаграм взаємодії й матриць.

Безпосереднє моделювання виконане дуже зручно. У програмі повністю відсутні недоліки, властиві іншим, наприклад: напливи елементів діаграми один на одного, перетинання стрілок, збої при перетаскуванні об'єктів, пулів і т.д. Інтерфейс зручний, зрозумілий і може налаштуватися користувачем.

Всі моделі можуть бути зв'язані один з одним, так що провести моделювання всієї системи бізнесу не проблема. Крім того, можливо провести імітаційне моделювання й перевірку діаграм.

VP дозволяє детально управляти атрибутами елементів, що, у свою чергу, дозволяє автоматично генерувати відмінні описи. Т.к. програма споконвічно орієнтована на розроблювачів інформаційних систем, кожному елементу можна задати умови поводження в системі, бізнес-правила й т.буд. До речі, шаблони документів також настроюються.

І нарешті, програма дозволяє вивантажувати отримані моделі у вигляді програмного коду. Причому в різних мовах! Безумовно, дана функція має високу цінність при розробці інформаційних систем і автоматизації бізнес-процесів.

Функціонал і особливості:

- Моделювання бізнес-процесів у різних нотаціях.
- Побудова інших моделей.
- Перевірка моделей.
- Автоматична генерація документів.
- Керування атрибутами елементів моделей.
- Створення й призначення правил поведінки моделей.
- Можливість додавати свої елементи в моделі.
- Взаємозв'язок моделей.
- Вивантаження моделей у вигляді програмного коду.
- Вивантаження моделі в графічному виді.
- Версія для Mac OS X.

Вартість:

- По підписці – 35$ на місяць.

– Повна ліцензія – 800$..

Резюме:

Краща програма для моделювання й опису бізнес-процесів.

BizAgi Modeler

Це частина вищезгаданого BizAgi Suite. Програма незалежна від повного комплекту й може бути поставлена окремо.

Дуже простий, лаконічний і зручний інтерфейс.

Гарний робочий інструмент для моделювання, що до того ж часто обновляється й удосконалюється. Моделі, побудовані в BizAgi Modeler, повністю сумісні з повною версією – Suite. Існують певній і властиві тільки цій програмі обмеження при моделюванні, яких немає в нотації BPMN, але вони в принципі обходяться.

Працювати з моделями досить зручно. Правда, іноді можуть виникати прикрі зсуви елементів моделі. Особливо при перетаскуванні великої кількості елементів. На мій погляд, недостатньо пророблена оптимізація розташування стрілок і елементів. Це приводить до того, що іноді доводиться небагато повозитися для гармонічного розташування елементів.

Недостатньо пророблений взаємозв'язок діаграм. Тобто зв'язати можна, але не прямо. Атрибути елементам можна призначати будь-які – ви самі визначаєте назву й властивості атрибута.

Можлива перевірка моделей і генерація опису по шаблоні.

Незважаючи на деякі недоліки, даний інструмент заслуговує тверду п'ятірку й підійде невеликим компаніям. Особливо у світлі того, що інструмент повністю безкоштовний.

Функціонал і особливості:

– Нотація BPMN.

– Перевірка моделей.

– Автоматична генерація документів.

– Керування атрибутами елементів моделей.

– Можливість додавати свої елементи в моделі.

– Вивантаження моделі в графічному виді.

– Зручний інтерфейс.

– Українською мовою.

– Можлива спільна робота над моделями.

Вартість:

– Повністю безкоштовно.

Резюме:

Підійде як починаючим, так і компаніям, що вже займається моделюванням і описом процесів. Простий в освоєнні. Дуже рекомендую.

ARIS Express

Безкоштовна й проста “рисовалка” процесів від монстра по ім'ю ARIS. А точніше, Software AG.

У своєму розпорядженні має кілька варіантів моделей – зокрема: моделі бізнес-процесів у нотації eEPC і BPMN, організаційні моделі, карти процесів і т.д. Примітна наявністю функції Smart Design, що дозволяє швидко забити необхідні дані в таблицю й програма самостійно створить діаграму. Для швидких начерків досить зручно.

На жаль, Express цей тільки графічний засіб. Моделі не можна зв'язати один з одним, атрибути не призначиш тощо. Состав елементів діаграм досить обмежений, так що не вийде створити модель в Express і експортувати в ARIS BA. До речі, у жодному разі не використовуйте це ПО для роботи з нотацією BPMN. Незважаючи на те, що такі моделі можна тут створювати, їхня обмеженість задає кардинально невірне враження про функціонала BPMN.

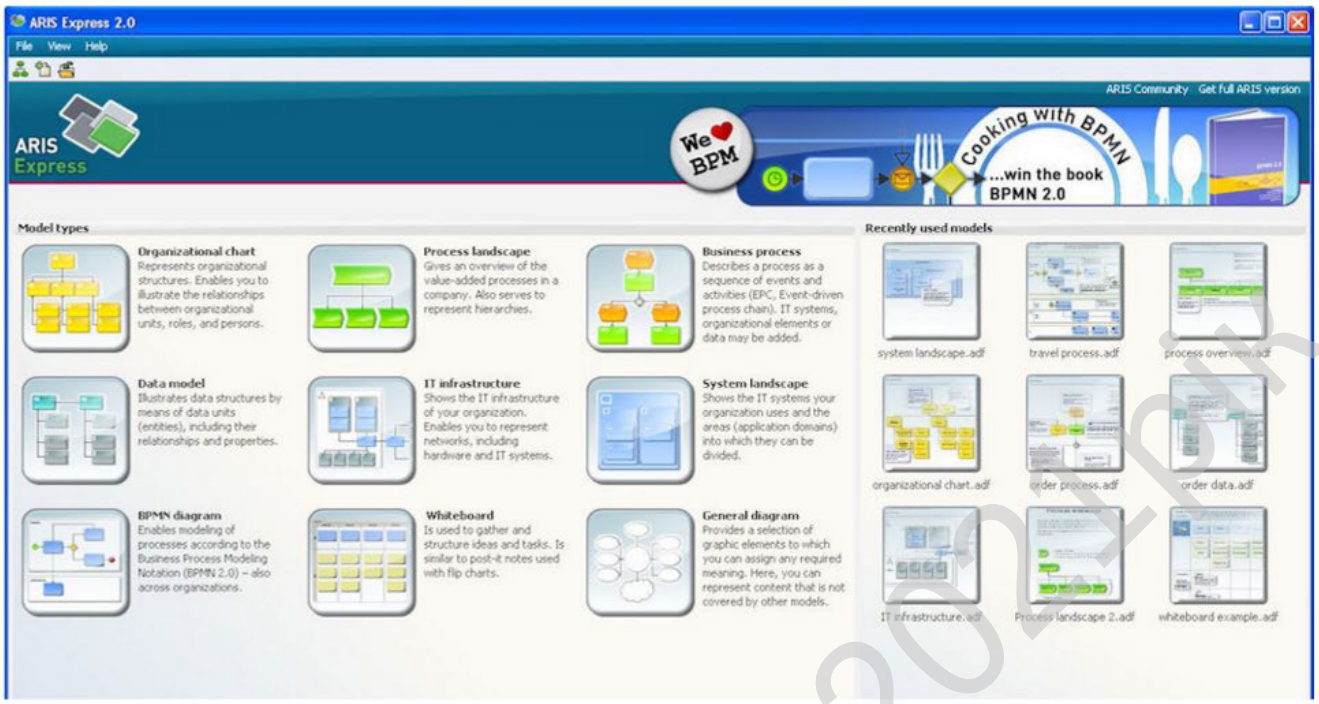

Рисунок 2.3 – Інтерфейс користувача ARIS Express

Однак мені відомі досить серйозні компанії, які використовують цей інструмент. Причому деякі затверджують, що він зручніше MS Visio. Це не так. Visio – потужний інструмент, що дозволяє фактично створити своє середовище для керування процесами. Але про це як-небудь іншим разом.

Функціонал і особливості:

– Нотації eEPC і BPMN.
– Карта процесів.
– Організаційна структура.
– Функція Smart Design.
– Вивантаження моделі в графічному виді.
– Простий інтерфейс.

Вартість:

– Повністю безкоштовно.

Резюме:

Вибирайте ARIS Express, якщо всі перераховані вище обмеження вас не хвилюють. Ну і якщо ви віддаєте перевагу нотації eEPC.

Онлайн-сервіси для моделювання бізнес-процесів

Gliffy

Відмінний сервіс із різноманітним функціоналом. Дозволяє створювати не тільки моделі в нотації BPMN, але й робочі потоки, проектувати користувальницький інтерфейс, створювати діаграми UML, організаційні діаграми, карти сайтів і т.д.

Що дуже важливо, сервіс дозволяє проводити колективну роботу над діаграмами, притім зберігаються всі версії моделі. Крім того, ви можете вставити діаграму у вигляді шорткоду на ваш сайт. До речі, моя карта статей зроблена саме в цьому сервісі.

При моделюванні процесів можливо зв'язувати діаграми один з одним за допомогою гіперпосилань, адже одна діаграма – це, по суті, одна сторінка.

Всі елементи нотації BPMN уже присутні в сервісі. Також можливо самостійно змінювати зовнішній вигляд елементів і додавати свої. У безкоштовній версії експортувати діаграми можна тільки у вигляді графічних файлів.

Функціонал і особливості:

– Повна підтримка BPMN.
– Взаємозв'язку моделей через гіперпосилання.
– Зручна побудова моделей.
– Гнучке налаштування зовнішнього вигляду елементів.

Вартість:

– Безкоштовно з невеликими обмеженнями..
– 4.95$ на місяць для стандартної версії й 9.95$ для бізнес-версії..

Резюме:

Зручний і функціональний сервіс для створення діаграм бізнес-процесів і не тільки.

BPsimulator

Ну дуже цікавий сервіс, у якому упор зроблений не на моделі, а на симуляцію й оцінку моделі.

Працює це в такий спосіб: моделюєте процес -> задаєте властивості потоків, вартості, тривалості й зайнятості співробітників -> запускаєте симуляцію -> дивитеся показники процесу за результатами симуляції.

Що це дає? Насправді багато чого. Симуляція дозволяє з легкістю виявляти вузькі місця процесу, розрахувати вартість ресурсів у процесі, оцінити завантаження ресурсів і т.д.

Симулятор нескладний, точніше, має певні обмеження, але користь із нього витягти можна. А при вмінні й чималу.

Керування досить зручне. Стрілки мають тунелі (я завжди звертаю увагу на цей момент). Отримані звіти й моделі можна зберегти на комп'ютер, Google Drive або One Drive.

Функціонал і особливості:

- Моделювання процесу.
- Оцінка вартості / тривалості процесу.
- Симуляція.
- Зручна побудова моделей.
- Звіти.
- Збереження моделей в Google Drive або One Drive.

Вартість:

- Безкоштовно з рекламою..
- 300 грн./міс без реклами й з невеликими булочками..

Резюме:

Дуже раджу спробувати.

Draw io

Сервіс дозволяє будувати величезна кількість діаграм і має великий набір елементів. У тому числі набори для побудови BPMN і eEPC діаграм.

Можливо зв'язувати моделі через гіперпосилання. Крім того, можна до елементів приєднувати файли із хмарних сховищ даних.

Робота з моделями відносно зручна. Можна всіляко набудовувати зовнішній вигляд елементів. Але й це незручно, відсутнє тунелювання стрілок, а також відштовхування об'єктів. Тобто один елемент може розміщатися на іншому. Що приводить до того, що необхідно витрачати час на ручне розміщення елементів діаграми.

Сервіс дозволяє зберігати моделі в Google Drive, Dropbox, One Drive або на комп'ютер. Можливий експорт моделей у форматах графічних файлів, PDF, HTML, XLS.

Функціонал і особливості:

– Побудова різних діаграм.

– Збереження моделей в Google Drive, Dropbox або One Drive.

– Відсутня можливість колективної роботи.

Вартість:

– Безкоштовно.

Резюме:

Проста й безкоштовна рисовалка. Завдяки інтеграції із хмарними сховищами може бути використана в рамках групи співробітників.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero RAD Studio Delphi 10.3.2 Rio Architect – це найшвидший спосіб створювати й обновляти інтенсивно працюючі з даними, сильно взаємодіючі застосунки з візуально насиченим користувальницьким інтерфейсом

для Windows 10, Mac, мобільних пристроїв, IoT і інших платформ за допомогою Object Pascal і C++. Широкий вибір функцій підтримки Windows 10, у тому числі нові компоненти VCL для Windows 10, стилі для VCL і FMX, а також служби UWP (універсальної платформи Windows), наприклад повідомлення, дозволяють легко й швидко перенести застосунки в Windows 10, зберігши користувачів. Нова платформа дозволяє підтримувати великі проекти на більшому числі платформ із подвоєним обсягом пам'яті в середовищі розробки й удвічі більшим розміром підтримуваних проектів. Крім того, підтримка декількох моніторів і десятки нових функцій середовища розробки, призначених для прискорення створення коду, зроблять роботу як ніколи ефективною. За допомогою RAD Studio 10 розроблювачі зможуть створювати застосунки в 5 разів швидше в порівнянні з іншими інструментами, а розробка застосунків для декількох настільних, мобільних, хмарних платформ і платформ баз даних, включаючи 32і 64 -розрядні версії Windows 10, Mac OS X, iOS і Android, стане ще швидше.

Зміни у версії 10.3 Rio:

– Створюйте міжплатформені застосунки. 80% інтернет користувачів мають смартфони й застосунки доступу, а також дані з мобільного пристрою й ноутбука / настільного комп'ютера, саме тому так важливо в цей час, щоб застосунки працювали в будь-якому пристрої.

– У всіх версіях Professional, Enterprise і Architect RAD Studi o 10.3 надається підтримка процесу розробки застосунків для мобільних пристроїв. Розроблювачі RAD Studio кодують лише один раз, компілюють споконвічно для кожної платформи, що скорочує час і трудозатрати на вивчення декількох мов і дозволяє паралельно управляти циклами розробки.

– Підтримка Android API26, відповідність вимогам Google Play Store відносно нових застосунків із серпня 2018 року й відновлення застосунків з листопада 2018 року.

– Власні елементи керування Android і стилізовані елементи керування FMX в одній і тій же формі Android, включаючи тему матеріального дизайну для Android 5.0 або вище.

– Підтримка iOS 12 (32і 64-біт) для створення App Store і корпоративних застосунків.

– Підтримка смайликів Юнікод.

– Програмуйте по-своєму. Завдяки двом новим темам са мостійне налаштування інтегрованого середовища розробки для відповідності вашому стилю кодування ще ніколи не була настільки простій.

– Темне й світле оформлення Незалежно від того чи волієте ви кодувати вночі або у світлий час доби, завдяки темному й світло му оформленню RAD Studio ви можете вибрати потрібний вам стиль. Було доведено, що темне оформлення допомагає знизити зорову напругу в умовах низького освітлення, дозволяючи вам працювати більш продуктивно вночі. Немає нічого простіше, ніж перейти від темного до світлого оформлення й навпаки за допомогою меню панелі інструментів.

– Виконаєте користувальницьке налаштування свого середовища розробки Поліпшена програма установки інтерфейсу користувача й менеджера ліцензій інтерфейсу користувача дозволяє визначити ті можливості, які необхідні й опустити непотрібні, незалежно від того чи розробляєте ви застосунки для декількох платформ або всього однієї.

– Чистий, оновлений інтерфейс користувача інтегрованого середовища розробки Знайдіть потрібні можливості. Швидко. Головне вікно інтегрованого середовища розробки відцентровано й відрізняється високим ступенем читаності. Ви з легкістю визначите, де перебуває область фокусування клавіатури з оновленими змінами фонових квітів фокуса. Вкладки редактора більше, що полегшує читання шрифтів, тому ви можете швидко внести зміни й зберегти кодування.

– Чудові застосунки Windows з VCL. Бібліотека візуальних компонентів (Visual Component Library, VCL) пропонує просту й візуальну розробку користувальницького інтерфейсу застосунки, у версії 10.3 представлені нові відновлення, які дозволять вашим додаткам виглядати сучасними й свіжими.

– Розширена підтримка HighDPI. Завдяки новому елементу керування VCL High DPI ImageList у версії 10.3 розроблювачі, що створюють нові застосунки VCL для Win dows або оновлюючі існуючі застосунки для High DPI дисплеїв, можуть повністю підтримувати зроблені до рівня пікселів зображення зі змінною розв'язною здатністю на всіх елементах керування, а також будь-яке користувальницьке креслення, що вимагає масштабованих зображень для моніторів з різною розв'язною здатністю.

– Підтримка Per Monitor V2. Переконаєтеся, що ваш застосунок масштабується правильно для всіх типів масштабування в Windows, реагуючи на зміни масштабування DPI на різних екранах під час виконання.

– Розширена підтримка Windows 10 і WinRT API. Сюди відноситься ряд ключових API-інтерфейсів WinRT і останні API-інтерфейси Windows 10, включаючи готові до використання компонента для убудованих у застосунки покупок і випробувань у магазині Windows 10 Store.

– Розгортання застосунків на основі служб за допомогою RAD Server. Продуктивність RAD Server була значно поліпшена завдяки десятикратному збільшенню потужності відносно простих операцій.

– Нові компоненти обробки JSON допоміжного засобу.

– Розширена підтримка RAD Server для клієнта Ext JS. Об'єднайте зовнішній інтерфейс javascript і веб-службу, підтримувану REST Server REST. (У версії Architect тепер включена ліцензія ExtJS Professional).

– Версії Enterprise включають ліцензію для одиничного розгортання RAD Server.

– Версії Architect включають ліцензію для розподіленого розгортання RAD Server.

– Що нового в C++? Підтримка C++17 Win32 збільшує продуктивність, поліпшує роботу компілятора й прискорює процес кодування. Були оновлені RTL і STL.

– Нова версія STL/Dinkumware 2018 для Win32 і Win64.

– Поліпшене автодоповнення коду Автодоповнення коду для даного компілятора тепер асинхронне, швидше й із кращими результатами, чим у попередньому автодоповненні коду C++. Уведення тексту не буде припинятися, поки виконується розрахунок.

– Тепер є підтримка налагодження для оптимізації компонувань.

– 2X швидкість математичної продуктивності для Win64.

– Нові додаткові лабораторії C++ в GetIt.

– Нові й поліпшені можливості роботи з базами даних. InterBase 2017 / IBToGo 2017 в RAD Studio. Версії Professional включають ліцензію розроблювача InterBase 2017, у той час як версії Enterprise і Architect містять у собі ліцензії InterBase ToGo. InterBase ToGo доповнена можливістю шифрування, функціями зміни подань, призначених для простої синхронізації даних застосунку по підписці без обмежень за розміром файлу бази даних.

– Поліпшена й оновлена підтримка для популярних баз даних, включаючи mySQL v8.0, MariaDB 10.3, SQL Server 2017, PostgreSQL v10, Firebird v3.0, MongoDB, InterBase, SQLite 3.23.1, SQL Anywhere і багатьох інших.

– Удосконалення DataSnap.

– Поліпшення REST. Підтримка додаткових родинних REST методів, типів і властивостей.

– Повністю оновлений модуль живлення версії Architect. Одержіть більше від версії Architect, включаючи ці ліцензії сімейства Idera.

– Ліцензія Sencha ExtJS Professional: Створіть свій ідеальний мережний вхідний інтерфейс за допомогою javascript і ExtJS.

– Ліцензія на розгортання InterBase ToGo. Додайте сховище даних у свої застосунки за допомогою цієї гнучкої, зашифрованої бази даних, що вбудовується.

– Ліцензія для розподіленого розгортання RAD Server. Ідеально підходить для серверного застосунку архітектури мікросервісів.

– Ліцензія AquaData Studio. Вражаючий аналіз бази даних.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Аналіз бізнес-процесів

Аналіз бізнес-процесів (Business Process Analysis) – це систематичне одержання даних з метою ідентифікації, визначення, оцінки й подання процесу як основи для його організації й поліпшення.

Приводом для проведення аналізу, як правило, є конкурентне положення компанії на ринку. Порівняння цін, витрат і продуктів/послуг може прояснити необхідні вимоги й підштовхнути до поліпшення. Індикаторами фактичної ситуації можуть служити:

– тривалий час поставки продукції й виникаючих проблем зі строками виконання замовлень;

– непрозорий хід процесу й недостатня його глибина;

– надмірно широкий спектр продуктів і деталей;

– часта зміна місць виникнення витрат при проходженні замовлення;

– значні внутрифірмові транспортні й складські витрати, заморожування матеріалів і площ;

– високі витрати на переустаткування при зміні продукту або технології;

– низька частка часу обробки в загальному часі проходження замовлення;

– високі витрати й високе завантаження потужностей; поява «вузьких місць» і ін.

Названі індикатори ставляться, переважно, до ключових процесів. Однак це не означає, що все дослідження повинне бути зосереджене винятково на них. Більший результат приносить аналіз всіх видів бізнес-процесів – ключових, управлінських, підтримуючих.

Структура процесу

Для ідентифікації процесу як необхідної умови його поліпшення, потрібне визначення його структури. Тут можуть використовуватися наступні дані:

– вимоги (кількісні, якісні, економічні, екологічні, тимчасові);

– технологічна послідовність подій і дій (трансформацій), що визначає будову й характеризує процес по виду й меті;

– актуальна структура (послідовність виконуваної роботи), як просторово-логічна послідовність проходження замовлення через організаційні одиниці й робітники системи;

– процесно-орієнтовані дані, такі як тривалість процесу (тривалість обробки замовлення), використання персоналу, площі, витрати на що створюють і не створюють вартість події (транспортування, зберігання й складування).

Для визначення даних необхідні наступні інструменти:

– виробнича документація й регламент;

– проведення аудита;

– проведення інтерв'ю й самоопис працівників;

– опис послідовності виконання робіт;

– workshop з учасниками процесу.

При вивченні даних виходять відповіді на наступні питання:

– який процес аналізується? які функціональні області або організаційні одиниці беруть участь? коли і які функції повинні виконуватися? Як виглядають результати цих функцій? Які наслідки повинні виходити із цих результатів?

– щодо ходу роботи – які етапи і як вони повинні виконуватися? який час проходження замовлення? які витрати? за допомогою чого виконуються робочі етапи? Які існують вимоги до якості?

– щодо матеріального потоку – які види ресурсів? яка потреба в потужностях? який обсяг потужностей у наявність? які потужності не задіяні? яка частота коливань у використанні потужностей? Яка матриця надходження – передачі матеріалів?

– щодо інформаційного потоку – звідки надходить інформація (вхід)? які дані? по якому шляху надходять? Як обробляються дані? куди поставляється вихідна інформація (вихід)?

Показники, використовувані для аналізу бізнес-процесів

Аналіз означає в першу чергу оцінку результативності, що виходить за допомогою показників. Для безперервного процесу поліпшення також як і для ефективного керування необхідна побудова системи показників, що складається із двох і більше факторів. Розрізняють три види показників: структурні, відносні й вимірювані або індекси.

Структурний показник являє собою відношення частини до цілого, причому цифра в чисельнику є частиною цифри в знаменнику.

Відносні показники відбивають відношення різних даних друг до друга.

У коефіцієнтах і індексах співвідносяться цифри в рівних одиницях виміру, але з різних, здебільшого рівних по величині періодів або стосовні до різних моментів часу.

Якщо виразити показник А в відсотках від показника Б, тоді говорять про індекс:

Для оцінки інтегральної результативності впровадження бізнес-процесів підприємства існує європейська модель оцінки досконалості EFQM, що була заснована в 1988 році чотирнадцятьома провідними підприємствами Європи на паритетних засадах з метою розробки для поліпшення й спрощення виробничих процесів на підприємствах. Один раз у рік одне підприємство нагороджується Європейською нагородою якості (European Quality Award). Одержати її може одне з підприємств, що впровадили в себе модель EFQM і яке отримало найкращі результати.

Моделювання бізнес-процесів

Що таке моделювання бізнес-процесів? Моделювання бізнес-процесів (Business Process Modeling) – орієнтоване по цілям, розроблене по певній систематиці й формі подання відображення бізнес-процесу.

Структура моделі відбиває по суті логічну предметно-тимчасову послідовність функцій, розглянутих у рамках певного процесу. Загальні характеристики моделі є основою для документації, аналізу, організації, автоматизованої обробки й підтримки процесів, а також для їхнього сприяння й комунікації.

Цілі моделювання бізнес-процесів

1. Документування бізнес-процесів підприємства для того:

– щоб вчасно одержувати дані;

– щоб представляти дійсну ситуацію в організаційній одиниці підприємства;

– щоб переміщати бізнес-процеси в інші підрозділи;

– щоб регулювати робочі процеси й методи через механізм зовнішнього керування;

– щоб виконувати обов'язки перед бізнес-партнерами або бізнесом-співтовариством (наприклад, по сертифікації підприємства);

– щоб задовольняти діючим правовим нормам;

– щоб навчати співробітників або уводити в суть справи;

– щоб уникати втрат знань (наприклад, при звільненні співробітника);

– щоб підтримувати менеджмент якості й керування охороною навколишнього середовища.

2. Підготовка / проведення оптимізації бізнес-процесів:

– щоб уводити нові організаційні структури;

– щоб змінювати при зміні ринкових умов завдання підприємства;

– щоб перебудовувати або поліпшувати процеси підприємства.

3. Підготовка автоматизації й впровадження інформаційних технологій,

4. Установлення показників процесу й контролю результативності,

5. Проведення benchmarking між підрозділами підприємства, партнерами й конкурентами,

6. Знаходження Best Practice (кращого досвіду в компанії, регіоні, галузі),

7. Супровід організаційних змін

– як продаж або частковий продаж;

– як додаткова покупка й інтеграція підприємства або схилів підприємства;

– як впровадження(вступ) або зміна(перемикання) систем IT або організаційних структур;

8. Участь у конкурсах (наприклад, конкурс EFQM, премія фонду Людвіга Ерхарда й ін.)

Загальною тенденцією сучасної організації й моделювання бізнес-процесів на підприємстві є перехід від функціонально-орієнтованої до процесно-орієнтованої моделі.

Моделювання бізнес-процесів припускає розгляд не тільки їхню типологію, але й облік рівня.

Подання бізнес-процесів припускає використання відповідних інструментів: символи, показники, графіки, діаграми, графи, бланки, а також таких рішень, як спеціальне програмне забезпечення.

Використання автоматизованих методів дозволяє знизити витрати, провести симуляцію й візуалізацію можливих процесних рішень, а також застосовувати й модифікувати раніше розроблені рішення. Існують різні програмні продукти для рішення завдань аналізу й організації процесів. Їх можна розділити на 3 групи:

– Стандартні графічні пакети, для подання процесів в електронному виді (візуалізація). Наприклад: ABC-FlowCharter, CorelFlow, Visio.

– Програмне забезпечення для аналізу процесів побудовано на базі графічних пакетів і дозволяє, поряд з візуалізацією, обробляти деякі дані процесів. Наприклад: Ablauf-Profi, Proplan, Vamos-BE.

– Процесно-орієнтоване програмне забезпечення. Ця група пропонує широкі функціональні можливості. Звичайно в даних продуктах реалізовані модулі для аналізу, моделювання й візуалізації процесів, а також підтримуються

оцінка й документація. Деякі системи дозволяють будувати анімаційні моделі. Наприклад: SYCAT, ARISToolset, AENEIS, AIBAS.

Опис бізнес-процесів

Опис бізнес-процесів (Business Process Description) – для опису процесу з якісно-кількісної, просторово-організаційної й технічно-технологічної точок зору використовуються характеристики (параметри), які задані стандартом ENISO 9001:2000. До таких параметрів ставляться:

- Цілі процесу.
- Об'єкт впливу (вид, кількість, розміри, ...)
- Місце впливу (організаційна одиниця, робоча система).
- Вид і послідовність подій, структура.
- Застосовувані засоби виробництва (вид, кількість, продуктивність, ...).
- Хід процесу (горизонтальний, вертикальний; просторово-тимчасовий).
- Участь персоналу (кількість, кваліфікація, ...).
- Витрати часу на події й загальний процес (тривалість процесу).
- Робочі умови, вимоги й завдання.
- Застосовувані технології, режими роботи.
- Результати.
- Невизначені стани й події.

Опис процесів вимагає визначення відповідної області розгляду, тобто вибору стартової й кінцевої крапки, що обмежують область, що цікавить.

Приклад

Для визначення типового шляху проходження замовлення в компанії область розгляду повинна охоплювати всі події від стартової крапки – «надходження замовлення» – до кінцевої крапки – «поставка замовлення клієнтові». Нехай, рівнем розгляду буде рівень структурного підрозділ і частина загального процесу. Необхідно визначити проходження замовлення в «Конструкторському бюро», тоді стартовою крапкою для опису й аналізу є «надходження замовлення в конструкторське бюро», а кінцевої – «передача

замовлення на виробництво». Цей приклад показує, що процеси можуть мати різний ступінь складності, що визначає витрати на їхній аналіз і організацію. Тому при реалізації подібних проектів, насамперед, визначають ступінь деталізації.

Подання бізнес-процесів припускає використання відповідних інструментів: символи, показники, графіки, діаграми, графи, бланки, а також таких рішень, як спеціальне програмне забезпечення.

Розповсюдженою формою подання може бути графічна блок-схема.

Зв'язок організаційних одиниць, що беруть участь, відзначається лінією, що підвищує наочність подання, однак, не несе необхідної інформації про час проходження, витратах на обробку або перервах. Подібний вид опису є простим і низькозатратним, однак надає таку корисну інформацію, як проходження через організаційні одиниці, час проходження замовлення, витрати на обробку й ін.. Крім того, він дозволяє зробити виводи про високу частку часу проміжного зберігання, можливих «вузьких місцях» у виробництві, що повторюється зміні відповідальності під час обробки замовлення.

Оптимізація бізнес-процесів

Оптимізація бізнес-процесів (Business Process Optimization) – безпосередня розробка й реалізація заходів щодо вдосконалювання (реорганізації) бізнес-процесів компанії.

Дослідження їхнього фактичного стану дозволяє сформулювати цілі по вдосконалюванню (реорганізації). Наприклад, завоювання частки ринку, зниження часу проходження замовлення, зменшення матеріальних запасів і ін.

Приклад

Компанія А робить спортивні товари й товари для відпочинку. На основі дослідження поточного положення на ринку в компанії були сформульовані наступні цілі:

– Зниження витрат до 25 Євро в середньому на виріб.

– Виготовлення й поставка продукту за замовленням клієнта протягом 5 днів з моменту замовлення в магазині.

– Підтримка номенклатури продукції в кількості 1300 штук.

Керівництво компанії прийняло пропозицію по дослідженню й поліпшенню процесів і, як результат проведеного дослідження, установило відповідні цілі. Для досягнення запланованих результатів було необхідно не тільки докорінно поліпшити процес проходження замовлення, але й внести зміни в розробки продуктів, організацію роботи на ділянці монтажу, змінити функції працівників і організувати роботу зі збуту.

Оптимізація направляється на реалізацію поставлених цілей і містить заходу, що усувають виявлені проблеми. До них можуть ставитися: питання сполучення змінених технологій, змінених робочих систем, занадто велике число рівнів керування, простої, невикористовувані потужності, дублювання робочих завдань, помилки в передачі інформації, втрата інформації, помилки в документації м ін.

При розробці заходів щодо оптимізації варто враховувати параметри впливу: логістичні, економічні, тимчасові, просторові, персональні.

Логічні – це кількість етапів процесу, технологічна реалізуємість, послідовність подій, організаційна взаємодія.

Економічні – низькі витрати, високе завантаження потужностей, низький рівень запасів, економічна глибина процесу, гнучкість, висока частка створення вартості.

Часові – короткий час проходження замовлення, низька частка допоміжного часу, низька частка часу переналагодження, гнучкість виробничого часу.

Просторові – можливість розташування необхідних робочих місць, можливість упорядкування робочих систем, мінімальні транспортні шляхи, можливість зміни порядку розташування робочих систем.

Персональні – обсяг роботи й потреба в персоналі, забезпечення необхідної кваліфікації, підвищення кваліфікації, гнучкий робочий час для персоналу.

Методи оптимізації бізнес-процесів

Методи оптимізації бізнес-процесів можуть бути різними, залежно від рішення виявлених проблем. Схема 2. показує можливі підходи по їхньому поліпшенню.

Метод виключати позначає зменшення рівнів процесу, ліквідацію причин перешкод, скорочення транспортних шляхів, виключення вхідного контролю.

Спрощувати припускає зменшення складності в проходженні замовлення, зниження комплексності структури продукту, організацію роботи, поділ робіт

Стандартизувати – програми, технології, методи, продукти, що комплектують, етапи.

Скорочувати – місця виникнення витрат, кількість і тривалість подій, деталей, виробничі витрати.

Прискорювати – паралельний інжиніринг, симуляцію, швидке проектування зразків, автоматизацію.

Змінювати – необхідні матеріали, технології, методи роботи, розташування, робочі системи, обсяг замовлення/партії, порядок обробки.

Забезпечувати взаємодія організаційних одиниць, робочих систем, працівників.

Виділяти й включати – необхідні процеси, що комплектують.

Організація бізнес-процесів

Організація бізнес-процесів (Business Process Organization) – поєднує заходу щодо встановлення їхньої внутрішньої структури (технологічної, тимчасовий, просторової, організаційної) з урахуванням конкретних умов компанії для певної області. Результатом є план, модель, опис процесів як основа для їхньої реалізації.

У заходи щодо організації входять: визначення ходу процесу й оргструктури, визначення ресурсів, установлення керівництва, формування процесних даних і документів, розробка інформаційного обслуговування й інші аспекти.

Шість кроків системного підходу

Системний підхід до організації процесів базується на шести кроках і припускає метод, що складається із шести пунктів.

1. Дослідження вихідної ситуації.
2. Аналіз і оцінка.
3. Розробка концепції.
4. Деталізація процесного рішення.
5. Впровадження.
6. Застосування.

У Кроці 1 проводиться дослідження фактичного стану процесів з використанням різних інструментів і методів, а також його аналіз. Фактичний стан можуть відбивати наступні дані: володіння процесом і результати, тривалість проходження замовлення (робочих днів, змін), витрати на обробку замовлення (годин/замовлення, хвилин/замовлення), кількість подій і робочих систем, що беруть участь, частка подій, що створюють і не створюють вартість, кількість організаційних рівнів, використання площ, завантаження робочих систем/потужностей, затримки, час очікування, умови роботи; керування перешкодами й ін. На основі причин і стимулів виробляються необхідні цілі.

У Кроці 2 зібрані дані необхідно проаналізувати відповідним чином, підготувати, тобто впорядкувати, перевірити на повноту, обробити й оцінити.

З урахуванням виявлених причин неефективності починається розробка заходів щодо зміни. Як альтернативні варіанти змін виступають: повна (ре)організація процесу або його поступове поліпшення.

У Кроці 3 проробляються варіанти можливих рішень, уточнюються вимоги й необхідні переваги. Тут формуються заходи щодо організації процесів,

насамперед, у формі загального планування можливих варіантів рішень. При цьому справедливо наступне основне правило: чим більше що змістовно відрізняються друг від друга варіантів буде знайдено, тим більша ймовірність досягнення поставленої цілі. Варіанти рішень рівняються по:

– результатам, що досягаються;

– вимогам до реалізації;

– витратам і строкам реалізації;

– необхідності навчання й перекваліфікації працівників і т.д.

Наприкінці кроку приймається остаточне рішення про впровадження одного із запропонованого варіанта. Тому що на попередньому вужі визначені вимоги й необхідні заходи, то далі переходять до Кроку 4 – деталізації процесного рішення. Детальне планування припускає: властиво деталізацію обраного рішення; організацію, переміщення й зміну робочих систем, іноді робочих місць; розробку необхідних заходів щодо реалізації (проведення перекваліфікації, організація робочого часу й системи винагороди, зміна кооперації, розробка процесних інструкцій і документації).

У Кроці 5 реалізуються необхідні підготовчі заходи й заходи щодо зміни:

– розміщення процесу;

– організація матеріального потоку;

– перекваліфікація працівників;

– зміна організації роботи, а також методів, засобів виробництва.

Далі виконується властиво впровадження на підприємство обраного рішення. Для виявлення можливих недоліків і слабких місць проводиться пілотний проект, що означає послідовний прогін процесу до досягнення запланованих результатів.

Бізнес-процес починає функціонувати по-новому в Кроці 6. Одержувані результати необхідно зіставляти із установленими цілями для виявлення можливих відхилень і визначення можливих коректувань. Отримані результати й досвід повинні оброблятися й зберігатися. У Кроці 6 необхідно здійснювати

постійне поліпшення й удосконалювання, для чого використовуються методи в області планування, керування й організації процесів.

Схема бізнес-процесу

Схема бізнес-процесу (Business Process Diagram) – це подання покрокових процесів, де схеми звичайно створюються як блок-схеми, у яких фігури представляють етапи процесу, а послідовність етапів позначається стрілками.

Багато українських компаній використовують у своїй діяльності текстовий опис бізнес-процесів у документах, які є процесними регламентами. Але для цілей аналізу й оптимізації діяльності компанії даний варіант не ідеальний. Опис бізнес-процесу в текстовому виді складно представити й аналізувати системно. При сприйнятті й аналізі текстової інформації людський мозок розкладає її на ряд образів, на що йдуть додатковий час і розумові зусилля.

Види схем бізнес-процесів

Схеми розробляються за допомогою цілого ряду різних методик: без символів і діаграм; з використанням символів і діаграм; побудови залежно від пріоритетів; графічно-описове подання процесів.

Схеми можуть бути побудовані з використанням графів пріоритетів. Графи пріоритетів – це подання за допомогою мережного плану часткових завдань монтажу, причому часткові завдання представляються як вузли, а взаємини між ними як єднальні лінії.

Схеми на основі графічно-описового подання є більше зручним для реалізації.

3.2 Розробка структурної схеми

Під цифровою трансформацією бізнесу звичайно розуміють впровадження нових технологій, які дозволять істотно поліпшити бізнес-процеси компанії. У даній роботі під цифровою трансформацією мається на увазі впровадження хмарних технологій у бізнес-процеси підприємства. При цьому IT-менеджери

вважають, що реінжинірингом бізнес-процесів будуть займатися представники бізнесу. А бізнес-керівництво вважає, що співробітники ІТ впровадять новітні ІТ-технології, які самі поліпшать доходи бізнесу, скоротять витрати й знизять ризики.

Цифрова трансформація бізнесу доречна, якщо треба:

– треба дуже сильно, у кілька разів, збільшити вигоди бізнесу від використання ІТ;

– є й гроші, і час, і бажання бізнес -керівництва займатися реінжинірингом бізнес-процесів, створивши до цього ще і єдину цифрову ІТ-платформу (як мінімум, логічно єдині дані по вашій компанії).

Цифрова трансформація бізнесу: що це таке?

Загальноприйнятих визначень, що таке цифрова трансформація бізнесу, немає. Є різні точки зору на це, що сильно залежать від того, хто відповідає на це питання.

Цифрова трансформація бізнесу: як її бачать гендиректори українських компаній:

– Істотне збільшення вигід, які інформаційні технології дають бізнесу при непідвищенні ризиків ІТ.

– У деяких випадках є розуміння, що за ІТ прийде платити в кілька разів більше, але майже всі гендиректори намагаються прикинути «валянками» і спробувати провести цифрову трансформацію бізнесу без збільшення витрат на ІТ.

Конкретні ІТ-технології, які гендиректори українських компаній, звичайно асоціюють із цифровою трансформацією бізнесу:

– Продажі через Інтернет.

– Омніканальність (робота із замовниками через різні канали).

– Мобільний доступ до корпоративних інформаційних систем.

– Налаштування виробництва під конкретні замовлення.

– Аналіз і прогноз поводження замовників.

– Цифрове проектування й моделювання.

– Соціальні мережі: продажу й робота із претензіями.

– Автоматизація керування логістикою.

– 3D-друк.

– Блокчейн.

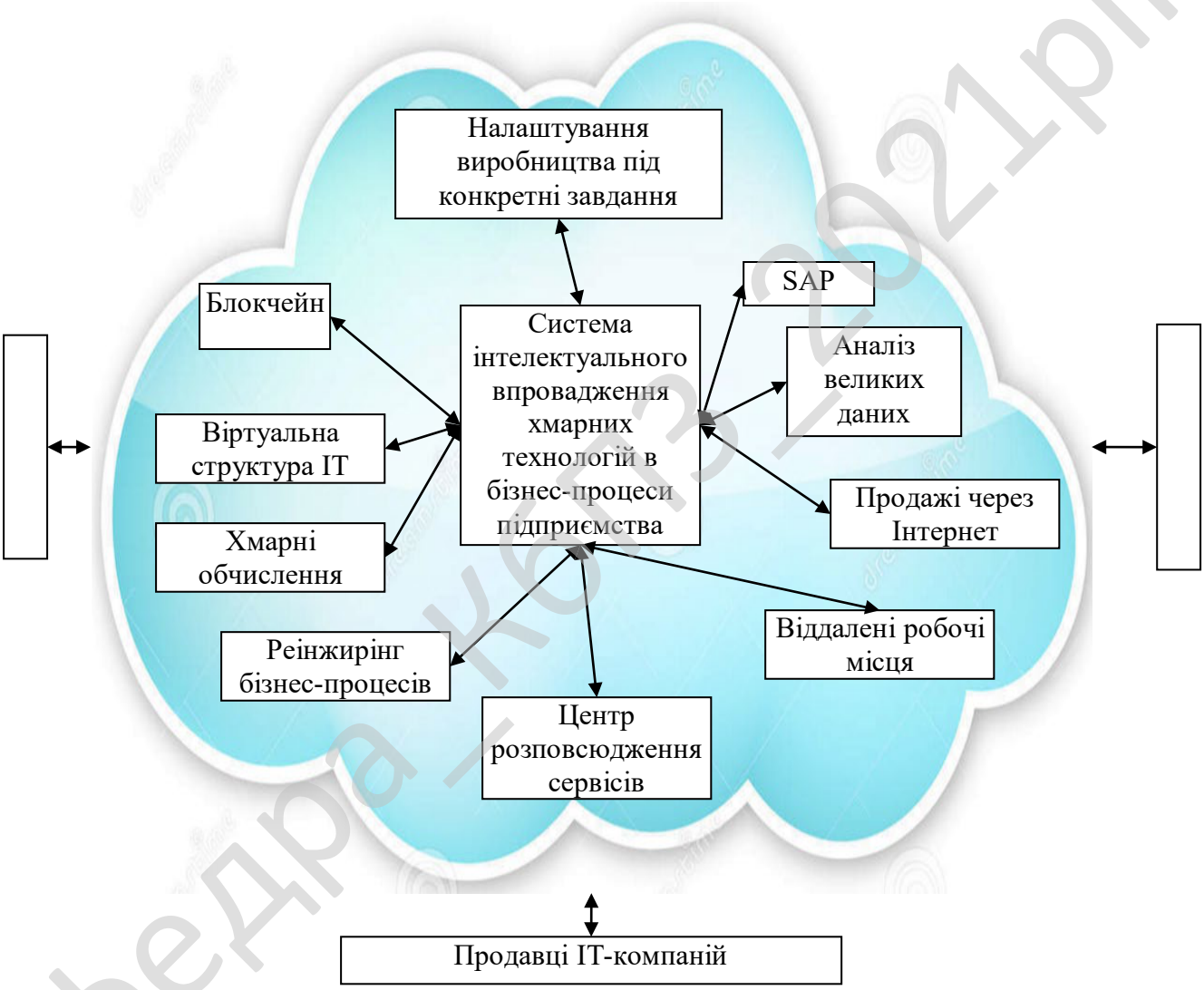

Рисунок 3.1 – Структурна схема системи

IT-керівники хочуть і далі продовжувати впровадження нових IT-технологій, уважаючи що цифрова трансформація бізнесу – це завдання бізнес-

менеджерів по реінжинірингу бізнес-процесів на базі IT-технологій (особливо нових).

Цифрова трансформація бізнесу: як її бачать керівники IT-служб українських компаній:

– «Хмарні обчислення».
– Віртуальна інфраструктура IT.
– Мобільний доступ.
– Вилучені робочі місця.
– SaaS, PaaS, IaaS.
– Аналіз більших даних.
– Омніканальність.
– Реінжиніринг бізнес-процесів.
– Автоматизація служби підтримки клієнтів компанії.
– Майнинг криптовалют.

Продавці хочуть вигідно продати що те зараз, ну або через півроку.

Цифрова трансформація бізнесу: як її бачать продавці IT-компаній в Україні:

– «Хмарні обчислення».
– Впровадження продуктів SAP.
– Впровадження продуктів IBM.
– Впровадження продуктів Oracle.
– Аналіз великих даних.
– ЦОД.
– Віртуальна інфраструктура IT.
– Інтернет речей.
– Штучний інтелект.
– Shares Service Center.

Нові комп'ютерні технології часто виявляються старими розробками, але під новою рекламною оболонкою.

Проблеми, що виникають при цифровій трансформації бізнесу:

– Нереально виконати вимоги багатьох українських гендиректорів (швидко, якісно, безкоштовно, з нульовими ризиками, а також не міняючи бізнес-процесів (і не задіючи співробітників бізнес-підрозділів);

– Нереально виконати вимоги багатьох українських гендиректорів (швидко, якісно, безкоштовно, з нульовими ризиками, а також не міняючи бізнес-процесів (і не задіючи співробітників бізнес-підрозділів);

– Продавці IT-компаній просто хочуть щось продати, не піклуючись про результати впровадження.

Етапи еволюції IT до цифрової трансформації бізнесу

На мій погляд можна виділити наступні етапи залежності бізнесу від IT:

– до 1965 року: бізнес взагалі не залежав від IT, хіба що папір і рахівницю за IT уважати;

– с 1965 року: IT виконують вимоги бізнесу до IT. Рішення по поліпшенню IT: оптимізація окремих елементів IT. Ефективність +5+10% вигід від IT;

– с 2000 року: IT виконують вимоги бізнесу й дають нові можливості бізнесу. Рішення: IT-стратегія. Вигоди +10+20%;

– с 2010 року: єдині цифрові платформи бізнесу. Рішення: інтегровані інформаційні системи, центри даних, обробки даних, єдині класифікатори інформації, централізовані служби підтримки IT. Вигоди: +15+30%;

– с 2015 року: IT радикально трансформують бізнес. Рішення: цифрова трансформація бізнесу, +30+50% збільшення вигід від IT.

Вивід: Для планування цифрової трансформації бізнесу треба визначитися, чи готова ваша компанія не тільки до більших можливих вигід від цифрової трансформації бізнесу, але й більшим витратам і ризикам неуспіхів.

Краще на початку провести аудит готовності до цифрової трансформації бізнесу й/або розробити IT-стратегію або стратегію створення цифрової трансформації бізнесу.

Планування цифрової трансформації бізнесу

Роботи на етапі 1: Виявлення суті бізнесу й перспектив його розвитку, пріоритетів бізнесу, вимог бізнесу до IT:

– Вибір доцільних нових IT-технологій. Аналіз можливостей нових IT-технологій.

– Аналіз поточного стану IT (інформаційні системи, інфраструктура IT, керування IT).

– Облік розміру компанії, галузі, інших особливостей.

– Визначення відповідальних за цифрову трансформацію бізнесу.

– Визначення рамок проекту по цифровій трансформації бізнесу: час, гроші, люди, методики, елементи IT і бізнесу.

– Розробка стратегії цифрової трансформації бізнесу.

Оцінити впровадження нових IT-технологій треба з урахуванням їх вигід, витрат і можливих ризиків. На жаль ризики впровадження зовсім нових IT-технологій можуть бути дуже великі.

Відповідальні за цифрову трансформацію бізнесу

От типові варіанти відповідальних за цифрову трансформацію бізнесу (CDO):

0) Спеціальних відповідальних немає.

1) На одному рівні з IT-директором.

2) Новий керівник IT-директори.

3) Уже наявний куратор IT від бізнесу.

4) Новий менеджер, підлеглий IT-директорові.

5) Нова функція IT-директори.

6) Новий радник і/або робоча група по цифровій трансформації.

7) Новий IT-директор, що займеться й цифровою трансформацією.

Виводи:

– Треба вибрати оптимальний саме для вашої компанії варіант відповідального (відповідальних) за цифрову трансформацію бізнесу (тому що це нова область і в багатьох компаніях поки немає відповідальних за неї).

– Планувати цифрову трансформацію бізнесу треба з обліком того, хто буде за це відповідати (ну або враховувати при плануванні цифрової трансформації бізнесу що треба буде вибрати відповідального за цей напрямок).

– Краще на початку провести аудит готовності до цифрової трансформації бізнесу й/або розробити IT-стратегію або стратегію створення цифрової платформи бізнесу.

Роботодавці й кадровики вже досліджували це питання й от відповідь. IT-директор, якого з руками відірвуть компанії, у яких вам хочеться працювати, це:

Цифровий і командний геній: харизматичний менеджер зі стратегічним мисленням, що провидить, як розвивати бізнес через цифрові технології. Проекти його команди повинні перегравати по ефективності, красі й безпеці все те, що можуть дати зовнішні IT-провайдери.

Джерело мегадоходи й економії: відрізняє тренд від хайпа, вчасно бачить і ощадливо забезпечує унікальні конкурентні технологічні переваги, впроваджує проривні рішення й мінімізує витрати бізнесу розумної й своєчасної цифровізацією.

Футуролог, продажник, політик: пророкує, які IT-інструменти знадобляться завтра всім іншим директорам і клієнтам, уміє оптимізувати цифрою спільні рішення й надихаюче «продавати» їхнім колегам, керівництву й клієнтам, на зрозумілому для них мові.

Чоловік мрії: win-win фахівець із безперервних змін, схильний до постійного апгрейду свого мислення й консервативному керуванню ризиками компанії, гнучкий менеджер талантів і геній комунікації зі складними клієнтами, терплячий як Будда й надійний як Бандера …

Загалом, **ідеальний CIO** для компанії – це оцифрованний від голови до ніг підприємець, продажник і політик, а також генерал, поет і трошки бог: «Юрій Гагарін цифрової епохи». Довідаєтеся в цьому портреті себе або?.. Не зовсім?.. Так ніхто не ідеальний! Ідеали потрібні, щоб постійно рости над собою.

Вибір за вами, шановні IT -директори 21 століття, тому що навіть поверхневий data analysis показує, що ви навіть не усвідомлюєте, що не володієте:

1. Ні стратегічним і футурологічним мисленням 21 вік.

2. Ні навичками політичного просування рішень цифрового розвитку бізнесу.

3. Ні технологією керування інноваційними змінами й ризиками.

4. Ні методами роботи з міждисциплінарними командами й наставництва.

5. Ні мистецтвом творчого мислення й дії в бізнесі.

6. Ні навичками продажів, переговорів, презентацій, сторителінгу.

7. Ні навичками емоційної компетентності й керування стресом.

8. Ні методикою розвитку власної кар'єри з CIO в GlobalCIO.

Від IT-стратегії до стратегії цифрової трансформації бізнесу

Можлива структура стратегії цифрової трансформації бізнесу:

– Доцільна цифрова трансформація бізнес-процесів (вимоги й побажання з боку бізнесу).

– IT-технології, доречні для цифрової трансформації бізнесу.

– Інфраструктура IT (необхідне через 1-5 років стан).

– Інформаційні системи й дані (необхідне через 1-5 років стан).

– Керування IT (необхідне через 1-5 років стан).

– План проектів по IT (на 1 рік і 2-5 років). Бюджет IT.

– Сценарії розвитку IT (альтернативні варіанти розвитку IT і цифрової трансформації бізнесу).

Істотне питання, що цікавив передплатників мого сайту, якщо стратегія IT є, то, якщо бізнес вимагає стратегію цифрової трансформації, що робити. На мій

погляд, потрібно з наявної IT-стратегії взяти як основу шматочки. Тобто зрозуміти, що бізнес хоче від цифрової трансформації, що він хоче через рік і два й постаратися зрозуміти й формалізувати. Нехай з боку бізнесу письмово це напишуть.

З боку IT і бізнесу вибрати нові IT -технології, доробити власну IT-стратегію: цілі IT, необхідний стан, інфраструктуру, інформаційні системи, керування, плани проектів. Ну й зрозуміло, що треба врахувати розмір компанії, галузі й інші фактори. І далі, на мій погляд, робити не стратегію цифрової трансформації, а стратегію створення єдиної цифрової платформи, що реально.

3.3 Розробка функціональної схеми

Бізнес-процес (Business Process) – установлена послідовність дій, що вимагає певного входу, що досягає певного виходу й використовує певних ресурсів, що служить для реалізації роботи або послуги для клієнта. В англомовній літературі бізнес-процес представляється як безліч із однієї або декількох зв'язаних операцій або процедур, у сукупності які реалізують деяку ціль виробничої діяльності, здійснюваної звичайно в рамках заздалегідь певної організаційної структури, що відбиває відносини між учасниками.

Поняття бізнес-процесу

Поняття одержало поширення у зв'язку з переходом до процесно-орієнтованої організації й процесно-орієнтованому менеджменту підприємства. Характерними для компаній бізнес-процесами є виконання замовлення, розробка продукту, керування компанією, доставка продукції. На практиці в кожній компанії існують типові для їхньої сфери й взаємозалежні один з одним бізнес-процеси, що мають своєю метою створення й реалізацію вартості продуктів і послуг. Обов'язково ознайомтеся зі статтею "Як побудувати бізнес-процес у компанії – інструкція в 4 кроки", щоб зрозуміти, як створюються бізнес-процеси на практиці. Складне стане наочним і зрозумілим.

У відповідності зі стандартом ENISO 9001:2000 процес – це набір взаємозалежних засобів і дій, що перетворять вхід у результат. Процеси викликають зміни відповідного об'єкта.

У компаніях існують процеси різних видів, які можуть залежати друг від друга й у той же час розрізнятися по багатьом параметрам. Такими параметрами є:

– Вид діяльності: виробничі процеси або надання послуг. Результатом виробничого процесу є матеріальний продукт (наприклад, верстати), надання послуг – нематеріальний (наприклад, інформація).

– Вид виконання: оперативні, тобто виконувані процеси, такі як обробка, розрахунок, або диспозитивні процеси, як планування, керування.

– Створення вартості: прямі, що змінюють об'єкт (монтувати), або непрямі процеси створення вартості, підготовчі, або підтримуючі процеси (перевіряти, транспортувати).

– Комплексність: макропроцеси або мікропроцеси. Макропроцеси описують загальні процеси на підприємствах (виробництво легкових автомобілів). Мікропроцеси – їхні складові частини (виробництво кузова).

– Комерційний успіх: ключові, управлінські й підтримуючі процеси. Ключові процеси є специфічними для компанії й мають особливе значення для досягнення загальної цілі компанії.

Бізнес-процеси часто являють собою комбінацію ключових, управлінських і підтримуючих процесів.

Ключові процеси (створення вартості) поєднують завдання й роботу для виконання певних вимог клієнта із застосуванням ключових виробничих компетенцій. Вони є стратегічно важливими й у той же час специфічними (унікальними, тому що, наприклад, внаслідок застосування фірмових знань їх складно скопіювати). До них ставляться:

– обробка й виконання замовлення;

– розробка, проектування й дизайн продукту;

– виробництво й монтаж і ін.

Управлінські процеси містять у собі завдання й діяльність, спрямовані на довгостроковий розвиток компанії й реалізацію цілей компанії. До них ставляться:

– стратегічний розвиток компанії;

– довго- і середньострокове планування в компанії;

– розвиток персоналу;

– інвестиційне планування;

– мотивація персоналу й ін.

Підтримуючі процеси містять необхідні завдання й роботи для підтримки ключових процесів, але не приводять до безпосередньої цінності для клієнта, наприклад:

– обробка даних;

– технічне обслуговування;

– логістика;

– адміністративні процеси й ін.

На рисунку 3.2 наведена функціональна схема системи.

Формування й структурування припускає розгляд не тільки типології, але й облік рівня процесу.

Для опису процесу з якісно-кількісної, просторово-організаційної й технічно-технологічної точок зору використовуються характеристики (параметри), які задані стандартом ENISO 9001:2000. Параметри процесу – дані для позначення результативності й ефективності, наприклад, витрати, час виконання, якість, точність.

Рисунок 3.2 – Функціональна схема системи

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей.

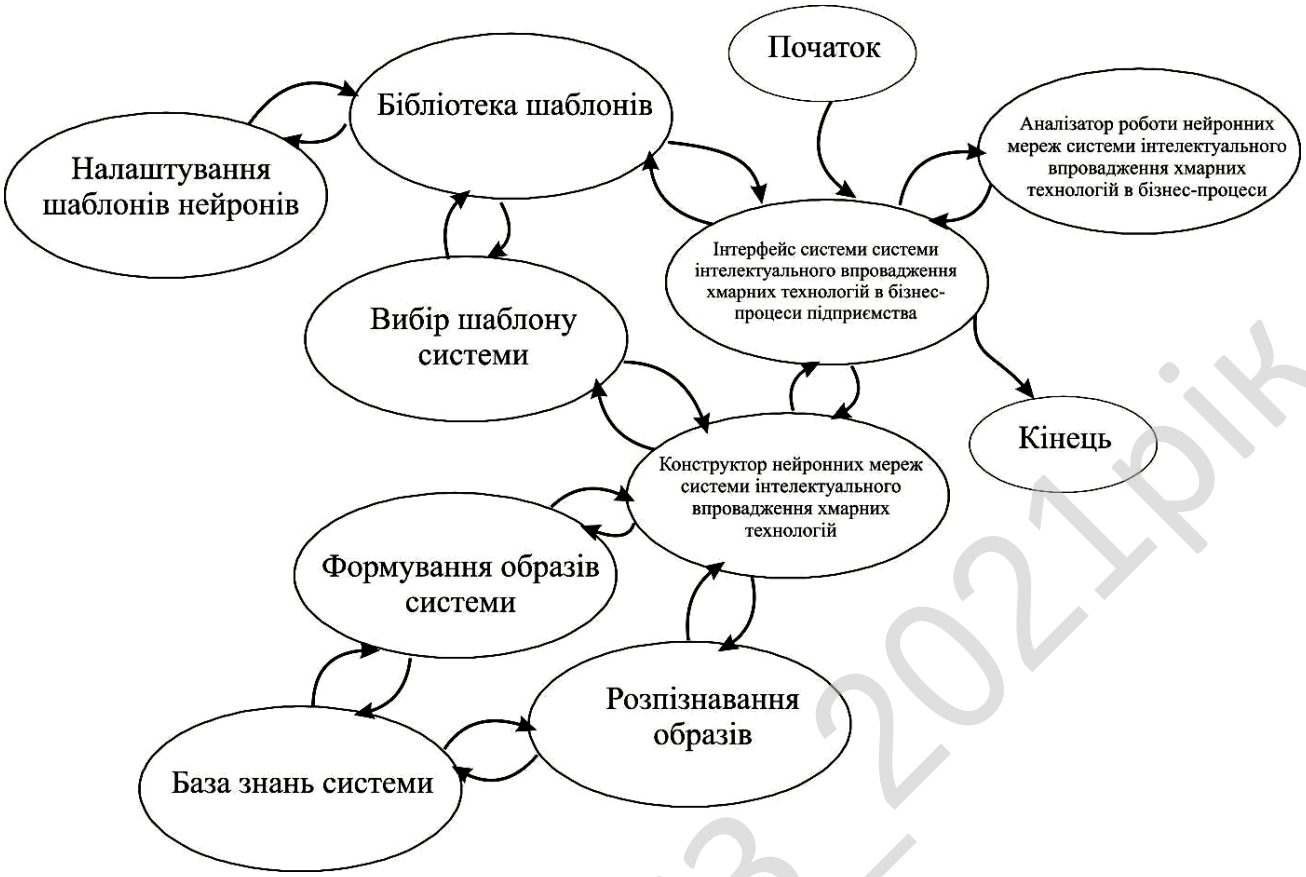

Рисунок 3.3 – Діаграма взаємодії процесів

Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію магістерської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень.

Було створено блок-схеми роботи системи. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується. Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач. Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції. Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу.

Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується.

Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи. Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох оп ерацій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані. Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи). З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми.

Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Систе́ма керування ба́зами да́них (СКБД, Database Management System, DBMS) – набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.

Першим поколінням СКБД прийнято вважати ієрархічні й мережеві системи. Ці системи отримали широке поширення в 1970-х роках, а першою комерційною системою цього типу була система IMS компанії IBM.

У 1980-х роках ці системи були витіснені системами другого покоління – повсюдно використовуваними і донині реляційними СКБД.

У цих системах використовувалися непроцедурні мови управління даними (SQL) і передбачався значний ступінь незалежності даних. Реляційні системи внесли значні удосконалення в управління даними: графічний користувацький інтерфейс (GUI), клієнт-серверні застосунки, розподілені бази даних, паралельний пошук даних та інтелектуальний аналіз даних.

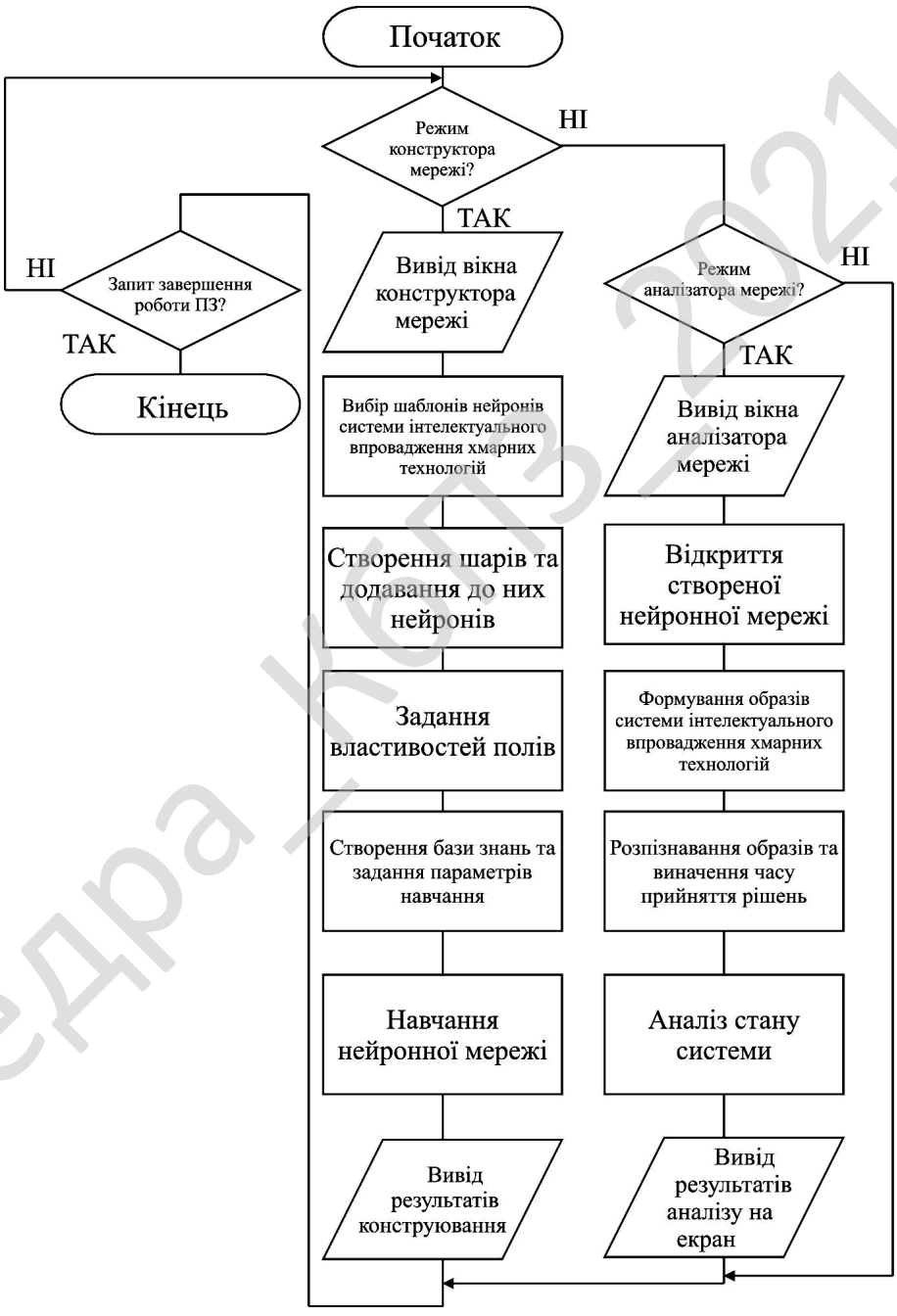

Рисунок 4.1 – Блок-схема основної програми

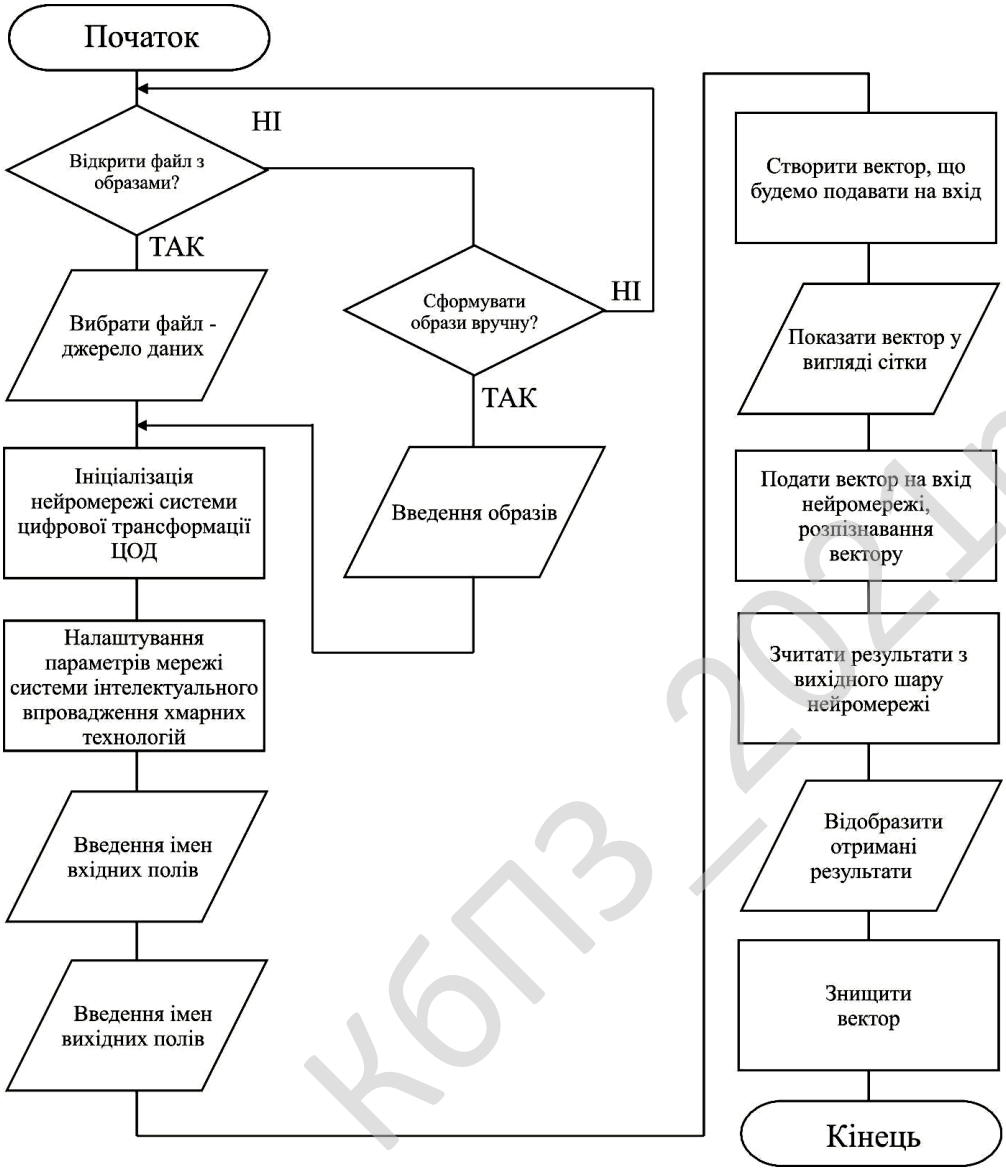

Рисунок 4.2 – Блок-схема роботи підпрограми

Але вже до кінця 1980-х років існуюча тоді реляційна модель перестала задовольняти розробників в силу низки обмежень. Відповіддю на зростаючу складність програм баз даних стали два нових напрямки розвитку СКБД: об'єктно-орієнтовані СКБД і об'єктно-реляційні СКБД.

У 1991 був утворений консорціум ODMG, основною метою якого стало вироблення промислового стандарту об'єктно-орієнтованих баз даних. Між 1993 та 2001 роками ODMG опублікувала п'ять ревізій своїх специфікацій. Остання версія стандарту має індекс 3.0, після чого група розпустилася. До кінця 1990-х

існувало близько десяти компаній, що виробляли комерційні продукти, що позиціонуються на ринку як ООСКБД. Найбільш відомими системами даного класу стали Objectivity, Versant виробництва однойменних компаній, а також СКБД Jasmine, випущена компанією CA. Незважаючи на переваги, що дозволяють ефективніше вирішувати певний ряд завдань, об'єктно-орієнтовані системи так і не змогли завоювати значущу частку ринку СКБД, залишившись «нішевим» продуктом.

Постачальниками традиційних реляційних СКБД також була проведена значна робота з об'єднання об'єктно-орієнтованих і реляційних систем. Розробники постаралися розширити мову SQL, щоб включити в неї концепції об'єктно-орієнтованого підходу, зберігаючи переваги реляційної моделі (об'єктні розширення мови SQL були зафіксовані в стандарті SQL:1999).

Основний принцип – це еволюційний розвиток можливостей СКБД без поломки попередніх підходів та зі збереженням наступності з системами попереднього покоління.

Поняття СКБД третього покоління, якими, власне кажучи, і є об'єктно-реляційні СКБД, з'явилося після опублікування групою відомих фахівців в області баз даних «Маніфесту систем баз даних третього покоління». Основні принципи СКБД третього покоління, позначені в маніфесті:

Крім традиційних послуг з управління даними, СКБД третього покоління повинні забезпечити підтримку розвиненіших структур об'єктів і правил. Розвинутіша структура об'єктів характеризує засоби, необхідні для зберігання і маніпулювання нетрадиційними елементами даних (тексти, просторові дані, мультимедіа).

СКБД третього покоління повинні включити в себе СКБД другого покоління. Системи другого покоління внесли вирішальний вклад у двох областях – непроцедурний доступ за допомогою мови запитів SQL і незалежність даних. Ці досягнення обов'язково повинні враховуватися в системах третього покоління.

СКБД третього покоління повинні бути відкриті для інших підсистем. Це включає оснащення різноманітними інструментами підтримки прийняття рішень, доступом з багатьох мов програмування, інтерфейсами до існуючих популярних систем і бізнес-застосунків, можливістю запуску програм з бази даних на іншій машині і розподілені СКБД. Весь набір інструментів і СКБД має ефективно функціонувати на різноманітних апаратних платформах з різними операційними системами.

Крім того, СКБД, що розраховує на широку сферу застосування, повинна бути оснащена мовою четвертого покоління (4GL).

У середині 1990 років було лише кілька дослідних прототипів СКБД, які поєднали найкращі риси реляційних і об'єктно-орієнтованих СКБД. Першим комерційним продуктом, якому були властиві об'єктно-реляційні риси, став Universal Server компанії Informix(згодом була поглинена IBM). В даний час більшість цих ідей вже втілено в реальних комерційних рішеннях, в тому числі і в продуктах основних постачальників СКБД (Oracle Database і IBM DB2).

Розвиток індустрії систем керування базами даних базується на значних фундаментальних наукових дослідженнях. Найчастіше, між самими дослідженнями та їхньою конкретною реалізацією в прикладних рішеннях минають роки, а іноді й десятиліття. Роботу в області управління даними проводять як університетські дослідницькі групи (MIT, Berkeley), так і центри розробок основних постачальників СКБД (Oracle, IBM, Microsoft). Інвестування в управління даними – це довгострокове, і разом з тим, вигідне вкладення коштів. В даний час дослідники мають у своєму розпорядженні засоби, що дозволяють ефективно реалізувати найскладніші запити, що маніпулюють терабайтами й петабайтами різних даних.

Основними тенденціями, які дали привід для проведення різних масштабних досліджень в області баз даних стали:

Експонентний ріст даних. Обсяг даних, у тому числі синтетичних, що генеруються автоматизованими системами, значно зріс. Збільшилося і число

прикладних областей, в яких вимагається обробка великих обсягів даних. До таких областей тепер відносяться не тільки традиційні корпоративні програми, пошук у веб, але також і наукові дослідження, обробка природних мов, аналіз соціальних мереж тощо.

Значне ускладнення структур використовуваних даних. Прості види даних у вигляді чисел і символьних рядків стали доповняться численною мультимедійною інформацією, просторовими, процедурними даними та великою кількістю інших складних форматів.

Широке поширення дешевих високопродуктивних апаратних засобів. Щорічно ми спостерігаємо зростання обчислювальних можливостей мікропроцесорів, збільшення ємності і зниження вартості доступних і зручних в експлуатації пристроїв дискової оперативної пам'яті.

Активний розвиток засобів комунікації та «всесвітньої павутини» World Wide Web. WWW стає єдиним інформаційним середовищем, що пронизує весь світ і об'єднує величезне число користувачів та електронних пристроїв.

Поява нових важливих областей застосування СКБД. У першу чергу, це пов'язано з інтелектуальним аналізом даних, сховищами даних, а останнім часом – з паралельними обчисленнями і хмарними технологіями.

Основні характеристики СКБД

1. Контроль за надлишковістю даних.
2. Несуперечливість даних.
3. Підтримка цілісності бази даних (коректність та несуперечливість).
4. Цілісність описується за допомогою обмежень.
5. Незалежність прикладних програм від даних.
6. Спільне використання даних.
7. Підвищений рівень безпеки.

Можливості СКБД

1. Дозволяється створювати БД (здійснюється за допомогою мови визначення даних DDL (Data Definition Language)).

2. Дозволяється додавання, оновлення, видалення та читання інформації з БД (за допомогою мови маніпулювання даними DML, яку часто називають мовою запитів)

3. Можна надавати контрольований доступ до БД за допомогою: Системи забезпечення захисту, яка запобігає несанкціонованому доступу до БД; Системи керування паралельною роботою прикладних програм, яка контролює процеси спільного доступу до БД; Система відновлення – дозволяє відновлювати БД до попереднього несуперечливого стану, що був порушений в результаті збою апаратного або програмного забезпечення.

Основні компоненти середовища СКБД

1. Апаратне забезпечення.

2. Програмне забезпечення.

3. Дані.

4. Процедури – інструкції та правила, які повинні враховуватись при проектуванні та використанні БД.

5. Користувачі: адміністратори даних (керування даними, проектування БД, розробка алгоритмів, процедур) та БД (фізичне проектування, відповідальність за безпеку та цілісність даних); розробники БД; прикладні програмісти; кінцеві користувачі.

Архітектура СКБД

Існує трирівнева система організації СКБД ANSI-SPARC, при якій існує незалежний рівень для ізоляції програми від особливостей представлення даних на нижчому рівні.

Рівні:

1. Зовнішній – представлення БД з точки зору користувача.

2. Концептуальний – узагальнене представлення БД, описує які дані зберігаються в БД і зв'язки між ними. Підтримує зовнішні представлення, підтримується внутрішнім рівнем.

3. Внутрішній – фізичне представлення БД в комп'ютері.

Логічна незалежність – повна захищеність зовнішніх моделей від зм ін, що вносяться в концептуальну модель.

Фізична незалежність – захищеність концептуальної моделі від змін, які вносяться у внутрішню модель.

Було використано **MongoDB** – документо-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми таблиць.

MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

Код MongoDB написаний на мові C++ і поширюється в рамках ліцензії AGPLv3.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має досить гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів, підтримує журналювання операцій зі зміни і додавання даних в БД, може працювати відповідно до парадигми Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій.

У MongoDB є вбудовані засоби із забезпечення шардінгу (розподіл набору даних по серверах на основі певного ключа), комбінуючи який з реплікацією даних можна побудувати горизонтально масштабований кластер зберігання, в якому відсутня єдина точка відмови (збій будь-якого вузла не позначається на роботі БД), підтримується автоматичне відновлення після збою і перенесення навантаження з вузла, який вийшов з ладу.

Розширення кластера або перетворення одного сервера на кластер проводиться без зупинки роботи БД простим додаванням нових машин.

При розробці автори виходили з необхідності спеціалізації баз даних, завдяки чому їм вдалося відійти від принципу «один розмір під усе». За рахунок мінімізації семантики для роботи з транзакціями з'являється можливість

вирішення цілого ряду проблем, пов'язаних з нестачею продуктивності, причому горизонтальне масштабування стає простішим. Використовувана модель документів зберігання даних (JSON/BSON) простіше кодується, простіше управляється (у тому числі за рахунок застосування так званого безсхемного стилю (schemaless style), а внутрішнє угруповання релевантних даних забезпечує додатковий виграш в швидкодії.

Нереляційний підхід досить зручний для створення баз даних, у яких горизонтальне масштабування означає розгортання на множині машин. Можливість забезпечувати найкращу продуктивність повинна існувати паралельно з підтримкою більшої функціональності, ніж це дозволяє використання пар «ключ-значення» (у чистому вигляді).

Технологія баз даних має працювати скрізь, починаючи з серверів користувача та віртуальних машин і закінчуючи хмарними технологіями.

MongoDB, на думку розробників, має заповнити розрив між простими сховищами даних типу «ключ-значення» (швидкими і легко масштабованими) і великими СКБД (зі структурними схемами і потужними запитами).

Основні можливості MongoDB:

1. Документо-орієнтоване сховище (проста та потужна JSON-подібна схема даних).

2. Досить гнучка мова для формування запитів.

3. Динамічні запити.

4. Повна підтримка індексів.

5. Профілювання запитів.

7. Швидкі оновлення «на місці».

8. Ефективне зберігання бінарних даних великих обсягів, наприклад, фото та відео.

9. Журналювання операцій, що модифікують дані в БД.

10. Підтримка відмовостійкості і масштабованості: асинхронна реплікація, набір реплік і шардінг.

11. Може працювати відповідно до парадигми MapReduce.

СКБД управляє наборами JSON-подібних документів, що зберігаються в бінарному форматі BSON. Зберігання і пошук файлів в MongoDB відбувається завдяки викликам протоколу GridFS.

Подібно до інших документо-орієнтованих СКБД (CouchDB тощо), MongoDB не є реляційною СКБД.

Є докладна і якісна документація, велике число прикладів і драйверів для популярних мов Java, C++, C#, PHP, Python, Perl, Ruby.

При випуску одразу було заявлено, що реліз MongoDB 1.0 готовий до використання у виробництві як одиничний хост, так і у зв'язках master/slave.

Код цього релізу досить стабільний і перевірений в промисловій експлуатації протягом 1,5 років. MongoDB рекомендується розгортати мінімум на двох серверах використовуючи реплікацію Master/Slave.

Це забезпечує наявність актуальних даних при виході з ладу однієї з СКБД. MongoDB – продукт досить молодий, і відтак у ньому зустрічаються помилки, з'являються нові можливості тощо. Характерний високий темп розробки (проект пишуть не тільки волонтери, а й компанія людей на повній зайнятості). Компанія-розробник надає платні підтримку, хостинг, консультації.

Приклади запитів

Запити можуть витягати дані з вбудованих об'єктів та масивів. Якщо в колекцію users вставлений такий об'єкт:

```
{
   "username" : "bob",
   "address" : {
      "street" : "123 Main Street",
      "city" : "Springfield",
      "state" : "NY"
   }
}
```

Ми можемо запитати цей документ (і всі документи з адресою в Нью - Йорку) за допомогою:

```
> db.users.find({"address.state" : "NY"})
```

Можна також запитати елементи масиву:

```
> db.food.insert({"fruit" : ["peach", "plum", "pear"]})
> db.food.find({"fruit" : "pear"})
```

Jira – була використана комерційна система відслідковування помилок, призначена для організації взаємодії з користувачами, хоча в деяких випадках використовується і для управління проектами. Розроблено компанією Atlassian, є одним з двох її основних продуктів (поряд з вікі -системою Confluence). Має веб-інтерфейс.

Назва системи отримано шляхом усічення слова «Gojira» – Японського імені монстра Годзилла, що, в свою чергу, є відсиланням до назви конкурую чого продукту – Bugzilla; створювалася в якості заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проектами. Для кожного з проектів створює і веде схеми безпеки і схеми оповіщення.

До версії 3.13.5 (включно) розрізнялися редакції Enterprise, Professional і Standard, після – Залишилася тільки редакція Enterprise (для великих організацій).

Система заснована на Java EE і працює на кількох популярних системах управління базами даних і операційних системах.

Основний елемент обліку в системі – завдання (ticket або issue). Завдання містить назву проекту, тему, тип, пріоритет, компоненти і зміст. Завдання може бути розширена додатковими полями (також і нові призначені для користувача поля можуть бути визначені), додатками (наприклад – Фотографіями, скриншотами) або коментарями. Завдання може редагуватися або просто змінювати статус, наприклад, з «відкритий» в «закритий». Які переходи між станами можливі, визначається через настроюється потік операцій. Будь-які зміни в задачі записуються в журнал.

Jira має велику кількість можливостей конфігурації: для кожної програми може бути визначений окремий тип завдання з власним workflow, набором статусів, одним або декількома видами уявлення (screens). Крім того, за допомогою так званих «схем» можна визначити для кожного індивідуального Jira-проекту власні права доступу, поведінку і видимість полів і багато іншого.

Завдяки універсальному підходу можна пристосувати Jira для багатьох непрофільних завдань, наприклад, керування вимогами, керування ризиками, аж до реалізації невеликої системи бронювання, автоматизації процесу рекрутингу.

Для інтеграції з зовнішніми системами підтримує інтерфейси SOAP, XML-RPC і REST. Поставляється із засобами інтеграції з такими системами управління версіями, як Subversion, CVS, Git, Clearcase, Team Foundation Server, Mercurial і Perforce.

Існують доповнення, що дозволяють вбудувати Jira в інтегровані середовища розробки, в тому числі Eclipse і IntelliJ IDEA. Перекладена багатьма мовами, включаючи російську, англійську, японську, німецьку, французьку, іспанську.

Для сторонніх розробників надаються кошти розробки розширень системи – плагінів. Розробники розширень можуть викладати плагіни для продажу на спеціальний розділ сайту Atlassian.

Є комерційним продуктом, який може бути ліцензований для роботи на локальному сервері або доступний в якості віддаленого додатки. Ціноутворення залежить від максимального числа користувачів, при цьому близько $50 за користувача для локального і $7 на місяць за користувача для віддаленого доступу є типовими цінами.

Для академічних і комерційних клієнтів доступний повний вихідний код під ліцензією розробника.

Для проектів з відкритим вихідним кодом Atlassian надає спеціальну безкоштовну ліцензію при дотриманні наступних правил:

– проект використовує ліцензії, схвалені Open Source Initiative;

– вихідний код проекту доступний для скачування;

– у проекту є публічно доступна веб-сайт;

– програмне забезпечення від Atlassian є на веб-сайті проекту.

Хоча я реалізовував програму сам, було використано підходи Scrum для саморозвитку та пришвидшенню розробки, розглянемо цей метод. Scrum – підхід

управління проектами для гнучкої розробки програмного забезпечення. Скрам чітко робить акцент на якісному контролі процесу розробки.

Підхід вперше описали Гіротака Такеучі та Ікуджіро Нонака в статті The New New Product Development Game (Гарвардський Діловий Огляд, січ–лют 1986). Вони відзначили, що проекти, над якими працюють невеликі, крос–функціональні команди, зазвичай систематично продукують кращі результати, і пояснили це, як «підхід регбі». У 1991 році ДеҐрейс та Шталь у книжці Злі проблеми, справедливі рішення послалися на цей підхід, як на Scrum (штовханина; сутичка навколо м'яча (у регбі)), спортивний термін, згаданий в статті Такеучі і Нонака. Кен Швабер на початку 1990–х використовував підхід який привів Scrum в його компанію.

Вперше метод Scrum було представлено на загальний огляд задокументованим, чітко сформульованим та описаним спільно Сазерлендом та Швабером на OOPSLA'96 в Остіні. Швабер та Сазерленд протягом наступних років працювали разом щоб обробити та описати весь їхній досвід та найкращі практичні зразки для індустрії в одне ціле, в ту методологію, що відома сьогодні як Scrum. Швабер об'єднав зусилля з Майком Бідлом в 2001, щоб детально описати метод в книжці Agile Software Development with SCRUM. Не зважаючи на те, що для Scrum нарікли долю управління проектами з розробки ПЗ, він може також використовуватися в роботі команд обслуговувань програмного забезпечення (software maintenance teams), або як підхід управління розробкою і супроводом програм: Scrum of Scrums.

Scrum – це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи – ScrumMaster, той хто опікується процесами, веде їх і працює як керівник проекту, Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників.

Протягом кожного спринту, 15-30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog (документація запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана.

Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.

Product backlog – це документ, який має список вимог до функціональності, які упорядковані згідно зі ступенем важливості. Product backlog представляє список того, що повинно бути реалізовано. Елементи цього списку називається «історіями» (user story) або елементами backlog–у (backlog items). Product backlog відкритий для редагування усім учасникам Scrum–процесу.

Обов'язкові поля:

1. ID – унікальний ідентифікатор, порядковий номер, який використовується для ідентифікації історій у разі їх перейменування.

2. Назва (Name) – стислий опис історії. Він повинен бути однозначним, щоб і розробники і product owner могли зрозуміти, про що йдеться і відрізнити одну історію від іншої.

3. Важливість (Importance) – ступінь важливості даної історії на погляд product owner ’а. Зазвичай являє собою натуральне число, іноді для цієї цілі використовуються числа Фібоначчі. Чим більше значення, тим більше пріоритет.

4. Попередня оцінка (initial estimate) – початкова оцінка об'єму робіт, необхідного для реалізації історії порівняно з іншими історіями. Вимірюється у story point'ах. Приблизно відповідає числу «ідеальних людино–днів».

5. Як продемонструвати (how to demo) – стисле пояснення того, як завершена задача буде продемонстрована у кінці спринта. Дане поле може являти собою код автоматизованого приймального тесту.

Додаткові поля. Іноді, також, використовуються додаткові поля у product backlog, в основному для того, щоб допомогти product owner'у визначитися з його пріоритетами.

Категорія (track). Наприклад, «панель управління» чи «оптимізація». За допомогою цього поля product owner може легко вибрати усі пункти категорії «оптимізація» і задати їм низький пріоритет.

Компоненти (components) – указує, які компоненти (наприклад, база даних, сервер, клієнт) будуть зачеплені при реалізації історії. Дане поле складається з групи checkbox'ів, які відмічаються, якщо відповідні компоненти потребують змін.

Ініціатор запиту (requestor). Product owner може захотіти зберігати інформацію про усіх замовників, зацікавлених у даній задачі. Це потрібно для того, щоб тримати їх у курсі діла про хід виконання робіт.

ID у системі обліку помилок (bug tracking ID) – якщо ви використовуєте окрему систему обліку помилок, тоді у описі історії корисно зберігати посилання на всі дефекти, які до неї відносяться.

Sprint backlog – містить функціональність, обрану Product Owner із Product Backlog. Всі функ ції розбиті по задачах, кожна з яких оцінюється командою. Кожен день команда оцінює об'єм роботи, який необхідно провести для завершення задачі.

Burndown chart – показує, скільки вже виконано і скільки ще залишається зробити.

Планування спринта (Sprint Planning Meeting)

Проходить на початку нової ітерації Спринта:

– Із Product Backlog обираються задачі, зобов'язання по виконанню яких за спринт приймає на себе команда;

– На основі обраних задач створюється Sprint Backlog. Кожна задача оцінюється у ідеальних людино–годинах;

– Рішення задачі не повинно займати більше 12 годин або одного дня. При необхідності задача розбивається на підзадачі;

– Обговорюється та визначається, яким чином буде реалізовано цей об'єм робіт;

– Тривалість наради обмежена зверху 4–8 годинами в залежності від тривалості ітерації, досвіду команди тощо;

– (перша частина наради) Беруть участь Product Owner + Команда: обирають задачі із Product Backlog;

– (друга частина наради) Бере участь лише команда: обговорюють технічні деталі реалізації, наповнюють Sprint Backlog.

Щоденна нарада (Daily Scrum Meeting)

Відбувається кожен день протягом спринта. Є «пульсом» ходу спринта. Нараді властиві наступні обмеження:

– починається точно вчасно;

– всі можуть спостерігати, але говорять тільки обрані;

– триває не більш ніж 15 хвилин;

– проводиться в одному і тому ж місці протягом одного спринта.

Протягом наради кожен член команди відповідає на 3 запитання:

– Що зроблено з моменту попередньої щоденної наради?

– Що буде зроблено з моменту поточної наради до наступної?

– Які проблеми заважають досягненню цілей спринта? (Над рішенням цих проблем працює ScrumMaster. Зазвичай це рішення проходить за рамками щоденної наради і у складі осіб, що безпосередньо займаються даною перешкодою.)

Демонстрація (Sprint Review Meeting):

– Проходить у кінці ітерації (спринта).

– Команда демонструє внесок функціональності до продукту всім зацікавленим особам.

– Залучається максимальна кількість глядачів.

– Усі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт).

– Обмежена 4–ма годинами в залежності від тривалості ітерації і змін у продукті.

Ретроспектива (Sprint Retrospective):

– Члени команди висловлюють свою думку про минулий спринт.

– Відповідають на два основних запитання: Що було зроблено добре у минулому спринті?; Що потрібно покращити в наступному?.

– Виконують покращення процесу розробки (вирішують питання та фіксують вдалі рішення).

– Обмежена 1-3ма годинами.

Розглянемо визначення API. Це прикладни́й програ́мний інтерфе́йс (Application Programming Interface, API) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

Спрощено – це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Одним з найпоширеніших призначень API є надання набору широко використовуваних функцій, наприклад для малювання вікна чи іконок на екрані.

API є абстрактним поняттям – програмне забезпечення, що пропонує деякий API, часто називають реалізацією (implementation) даного API. У багатьох випадках API є частиною набору розробки програмного забезпечення, водночас, набір розробки може включати як API, так і інші інструменти/апаратне забезпечення, отже ці два терміни не є взаємозамінювані.

Високорівневі API часто програють у гнучкості. Виконання деяких функцій нижчого рівня стає набагато складнішим, або навіть неможливим.

В об'єктно-орієнтованих мовах, прикладний програмний інтерфейс зазвичай включає в себе опис набору визначень класу, з набором форм поведінки, пов'язаних з цими класами. Це абстрактне поняття пов'язане з реальними функціями, які надані або надаватимуться, класами, які реалізуються в методах класу.

Прикладний програмний інтерфейс в даному випадку можна розглядати як сукупність всіх методів, які публічно доступні в класах (зазвичай званий інтерфейс класу). Це означає, що прикладний програмний інтерфейс вказує методи, за допомогою яких взаємодіє з об'єктами, отриманими з визначень класів і обробляє їх.

У більш загальному плані можна визначити Прикладний Програмний Інтерфейс як сукупність усіх видів об'єктів, які можна вивести з визначення класу, і пов'язаних з ними можливих варіантів поведінки.

Наприклад: клас, що представляє Stack, може просто виставити публічно два методи Push() (для додавання нового елемента в стек) і Pop() (для вилучення останнього пункту, ідеально розташований на вершині стека).

У цьому випадку Прикладний Програмний Інтерфейс може бути інтерпретованим як два методи pop() і push(), або, більш широко, використовується варіант, коли можна використовувати елемент типу Stack, який реалізує поведінку стека, надаючи йому можливість для додавання / видалення елементів з вершини. Друга інтерпретація видається більш доречною в дусі об'єктно-орієнтованого підходу.

Якість документації, пов'язаної з Прикладним Програмним Інтерфейсом, є часто ключовим фактором, що визначає його успішність з точки зору простоти використання.

ППІ, як правило, пов'язаний із бібліотеками програмного забезпечення: ППІ описує і вказує очікувану поведінку в той час, як бібліотека є фактичною реалізацією даного набору правил. Один ППІ може мати декілька реалізацій (або жодної, будучи абстрактним) у вигляді різних бібліотек, які мають такий же інтерфейс.

Прикладний програмний інтерфейс також може бути пов'язаним з платформами програмування: платформа може бути заснована на кількох бібліотеках реалізує декілька інтерфейсів ППІ, але на відміну від звичайного використання ППІ, доступ до поведінки вбудований в платформу опосередкований шляхом розширення його змісту новими класами і вставлений в саму платформу. Крім того, загальний потік управління програми може бути під контролем абонента.

Прикладний програмний інтерфейс може бути також реалізацією протоколу.

Коли ППІ реалізує протокол, він може бути заснованим на проксі-методах віддалених викликів, що засновані на протоколі зв'язку. Роль ППІ може заключатися саме в тому, щоб приховати деталі транспортного протоколу. Наприклад: RMI є ППІ, який реалізує протокол або JRMP IIOP як RMI-IIOP.

Протоколи, як правило, розподіляються між різними технологіями і зазвичай дозволяють різним технологіям обмінюватися інформацією, діючи як абстракція між двома світами. ППІ, як правило, є специфічним для конкретної технології: звідси, інтерфейси даної мови не можуть бути використані на інших мовах, якщо виклики функції не будуть перетворені з конкретної адаптації бібліотеки.

Деякі мови, серед яких такі, що працюють на віртуальних машинах (наприклад: мови, сумісні з NET CLI середовища CLR і JVM сумісних мов у віртуальній машині Java) можуть ділитися програмними інтерфейсами.

У цьому випадку віртуальна машина дозволяє мові взаємодії завдяки спільному знаменнику віртуальної машини, що абстрагується від конкретної мови, використовувати проміжний байт-код і його мову.

При використанні прикладного програмного інтерфейсу в контексті веб-розробки, як правило, ППІ визначається набором повідомлень запиту HTTP, також визначається структура повідомлень-відповідей, зазвичай у розширенні мови розмітки XML або в форматі об'єктного запису JavaScript (JSON). У той час як прикладний програмний інтерфейс у Web історично був практично синонімом для веб-служби, останнім часом тенденція змінилась (так званий Web 2.0) на відхід від Simple Object Access Protocol (SOAP) на основі веб-сервісів і сервіс-орієнтованої архітектури (SOA) на більш прямі передачі репрезентативного стану (REST) стилів веб-ресурсів та ресурсів-орієнтованої архітектури (ROA).

Частина цієї тенденції пов'язана з рухом Семантичного веб-ресурсу до Опису Платформ (RDF), Концепції розвитку веб-технологій інженерних онтологій. Прикладні програмні інтерфейси у Web, що дозволяють комбінувати декількома прикладними програмними інтерфейсами в нові додатки називають гібридними.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм LOKI_91. Механізм алгоритму LOKI_91 подібний DES (рисунок 4.3). Блок даних розщеплюється на ліву й праву половини й проходить 16 раундів, що досить нагадує DES. У кожному раунді права половина спочатку піддається операції XOR із частиною ключа, а потім розширювальній перестановці (таблиця 4.1).

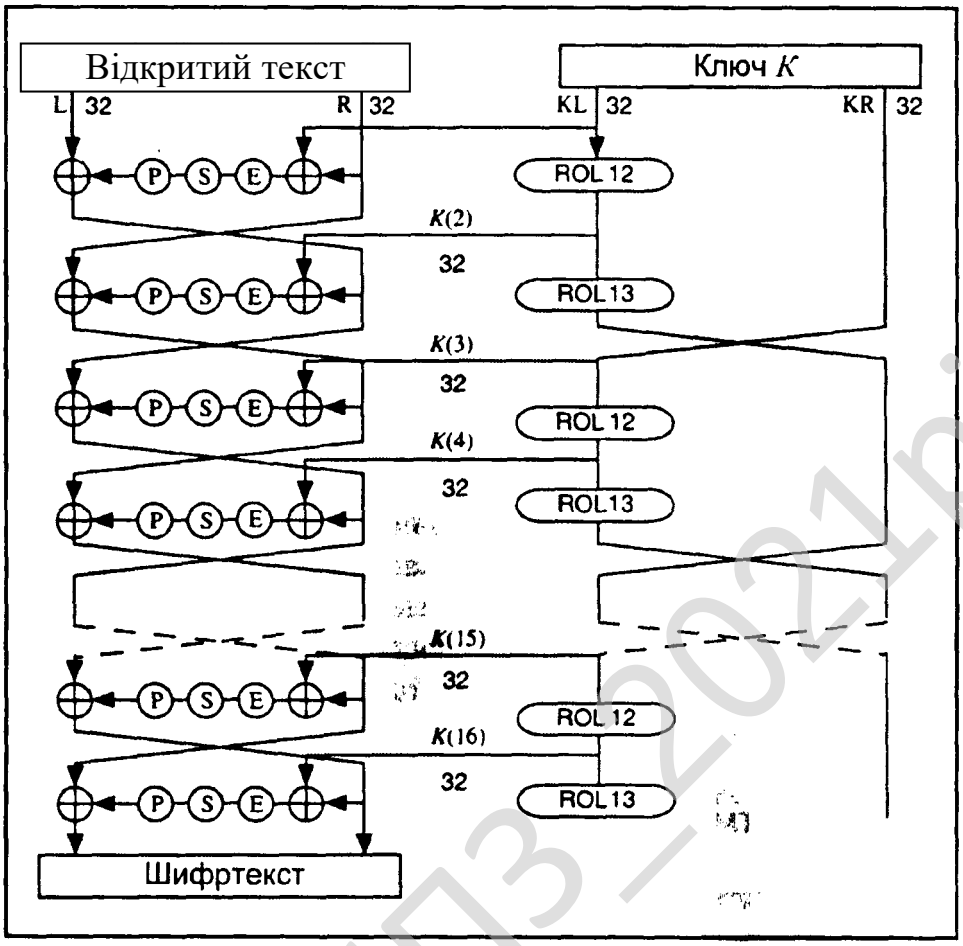

Рисунок 4.3 – Алгоритм LOKI91

Таблиця 4.1 – Перестановка з розширенням

4	3	2	1	32	31	30	29	28	27	26	25
28	27	26	25	24	23	22	21	20	19	18	17
20	19	18	17	16	15	14	13	12	11	10	9
12	11	10	9	8	7	6	5	4	3	2	1

48-бітовий вихід розділяється на чотири 12-бітових блоки. У кожному блоці виконується така підстановка з використанням S-блоку: береться кожний 12-бітовий вхід, 2 старших і 2 молодших біти використовуються для утворення

номера r, а вісім внутрішніх біт утворять номер c. Вихід S-блоку, O, має наступне значення: $O(r,c) = (c + ((r*17) \oplus 0xff) \ \& \ 0xff)^{31} \bmod P_r$.

Таблиця 4.2 – Значення P_r

r	1	2	3	4	5	6	7	8
P_r	375	379	391	395	397	415	419	425
r	9	10	11	12	13	14	15	16
P_r	433	445	451	463	471	477	487	499

Після цього чотири 8-бітових результати знову поєднуються, утворюючи 32-бітове число, що піддається операції перестановки, описаній в таблиці 3. Нарешті, для одержання нової лівої половини виконується операція XOR правої половини з колишньою лівою половиною, а ліва половина стає новою правою половиною. Після 16 раундів для одержання остаточного шифртексту знову виконується операція XOR над блоком і ключем.

Таблиця 4.3 – Перестановка за допомогою P-блоку

32	24	16	8	31	23	15	7	30	22	14	6	29	21	13	5
2	20	12	4	27	19	11	3	26	18	10	2	25	17	9	1

Підключи генеруються із ключа досить прямолінійно. 64-бітовий ключ розбивається на ліву й праву половини. На кожному раунді підключем служить ліва половина. Далі вона циклічно зрушується вліво на 12 або 13 біт, потім після кожних двох раундів ліва й права половини міняються місцями. Як і в DES, для зашифрування й розшифрування використовується один й той самий алгоритм із деякими змінами у використанні підключів.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у магістерської дипломної роботі системf інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Розділу виведення результату роботи системи; Функціональних кнопок ПЗ; Навігаційного меню; Верхнього меню.

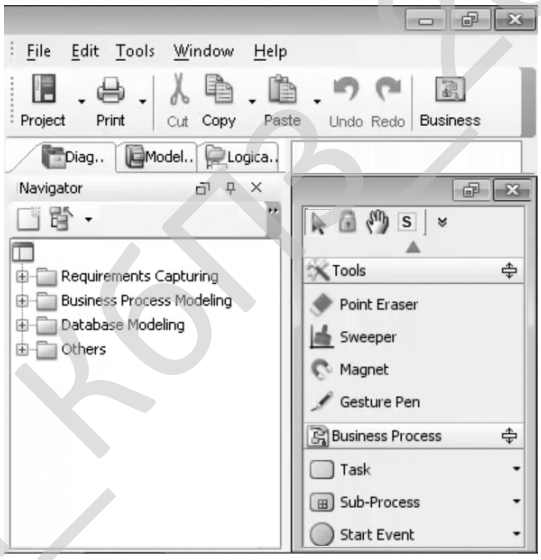

Рисунок 5.1 – Головне вікно ПЗ

Інфраструктурні хмарні сервіси, керовані сервіси в області інфраструктури й експертиза по побудові приватних хмар були об'єднані в підрозділі «Хмарні Сервіси». Для продуктового бізнесу характерні інші правила гри й процеси, ніж ті, що властиві проектній діяльності. Якщо по завершенні проекту спілкування із замовником зводиться до мінімуму, то при наданні сервісів із замовником доводиться спілкуватися щодня.

Виділення в окремий бізнес дало крім того більшу свободу дій і дозволило прискорити прийняття рішень. Початий редизайн внутрішніх процесів дозволив оптимізувати роботу й, як затверджується, «позитивно позначився на ціновій політиці».

Хмарні сервіси дозволяють залучати нових замовників, з ким до того ніколи не працював – по внутрішній статистиці 70% це саме нові замовники. При цьому більше 80% замовників розміщають у хмарі бізнес -критичні сервіси. Для забезпечення необхідного ступеня надійності хмара розміщається у власному ЦОД, сертифікованому на відповідність вимогам і Tier 3. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

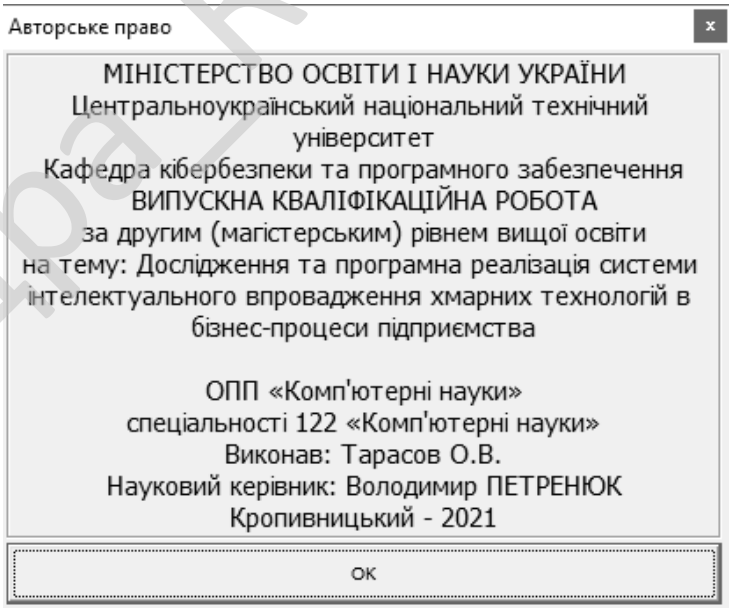

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих

процедур. Таким чином, розроблене ІТ рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТс;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware. Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Метою розробки є дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Об'єктом дослідження є процес інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Предметом дослідження є методи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Методи дослідження базуються на методах хмарних технологій, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

– Розроблено вітчизняний продукт інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи реалізації системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність

Таблиця 7.1 – Початкові данні

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	40 (2 ост. цифри № зал*10)
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	_	2
13. Рівень спрацьованості колективу (1-6)	_	2
14. Ступінь вимірності процесів (1-6)	_	3
15. Необхідна надійність програмного забезпечення (1-6)	_	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
17. Складність кінцевого програмного продукту (1-6)	_	2
18. Необхідний рівень забезпечення повторного використання (1-6)	_	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
20. Вимоги до швидкодії ПП (1-6)	_	2
21. Обмеження на розміри основного сховища даних (1-6)	_	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
23. Професійний рівень аналітиків (1-6)	_	2
24. Професійний рівень програмістів (1-6)	_	2
25. Постійність складу команди розробників (1-6)	_	2
26. Досвід розробки додатків (1-6)	_	2
27. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження табл. 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	40000 (2 ост. цифри № зал*10^4)
33. Норматив додаткової зарплати, % :	Нд	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B , \qquad (7.1)$$

де A – коефіцієнт Боема, A=2,45; Size – загальний об'єм відлагодженого програмного коду, тис. рядків; B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \qquad (7.2)$$

де Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн}=T_{ном}\, ПВ_j, \qquad (7.3)$$

де $ПВj$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S, \qquad (7.4)$$

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot100=168 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де Fpq – плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3{,}8 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість обладнання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2} \quad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 3}{1,2} = 567,5 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=567,5/(60·8)=1,2 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	Кількість штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, серверу доступу АДСЛ (ОС Linux), Wi-Fi налаштування ADSL, VPN, PPPoE, Frame Relay	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів д мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців:

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	1	12000	36000
Продакт-менеджер	0,25	8000	6000
Інженер-програміст	3,8	8000	91200
Інженер-електронщик	1,2	6000	21600
Інженер-системотехнік	0,25	6000	4500
Адміністратор мережі	0,5	6000	9000
Системний програміст	0,25	6000	4500
Дизайнер WEB	0,25	8000	6000
Інженер-верстальник	0,25	6000	4500
Бухгалтер-економіст	0,5	10000	15000
Всього за період розробки	$R_{сп}$=8,25	-	$Ф_{роб}$=198300

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де Фроб – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{198300}{8,25 \cdot 60} = 400 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць. Sу – питома площа на одне робоче місце, $м^2$; Цпл – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де Цм – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет магазину Компбест за 28.10.21 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок	Fujitsu P720 Tower	7347
Процесор	Intel Core i3-4130 (2 (4) ядра по 3.40 GHz); Cache Memory 3	-
Системна плата	Fujitsu D3221-A1 Intel Haswell с TDP до 95 Вт	-
Відеокарта	AMD Radeon RX 550 4GB GDDR5 Re Dragon PowerColor (AXRX 550 4GBD5-DH)	-
Жорсткий диск	HDD Seagate Barracuda 750 Gb 7200 32Mb SATAII ST3750528AS (ST3750528AS)	-
Оперативна пам’ять	DIMM 4096Mb DDR3 PC3-10600 CL9 Transcend JetRam, non-Reg., no-ECC , CL 9 (2 модулі)	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	-
Корпус	GRESSO GE-7525, 500W (120mm big fan), 2xIDE, full-ATX,БЖ 2xSATA, 1xFDD, Air Duct, 2xUSB 2.0, Mic+Audio, silver/black	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR -35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	-
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	Samsung S22R350FHI (LS22R350FHIXCI) / 21.5" (1920x1080) IPS LED / HDMI, VGA	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струменевий	Epson Stylus Photo P50 (C11CA45341) + USI cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	40000	10	4000
Разом	$К_р$= 1596075		$А_р$= 140140

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_e}, \qquad (7.11)$$

де Ne – Кількість екземплярів програм, шт.

$$З_о = 400 \cdot 209/40 = 2090 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 2090 \cdot 10 \cdot 0{,}01 = 209 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(2090+209) = 506 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом Нг=15% від основної зарплати

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 2090 \cdot 15 \cdot 0{,}01 = 314 \text{ грн}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картріджей, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм $n_{міс}$ приймаємо 0,33 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=105 грн., визначаємо вартість паперу за період розробки $N_м$ =3 міс:

$$З_{М1} = Ц_п \cdot N_м \, n_{міс}. \quad (7.16)$$

$$З_{М1} = 105 \cdot 3 \cdot 0{,}33 = 105 \text{ грн}.$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_{д}., \quad (7.17)$$

де Цд – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 2 грн/шт., DVD-R LG 4,7Gb, 16x speed Cake box – 2 грн/шт.

$$З_{М2} = 41 \cdot 12 = 492 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_{з}., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570=1702 \text{ грн.}$$

$$З_{М} = (105 + 492 + 1702)/40=57 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом (Нп = 15%) від основної зарплати виконавців

$$О_{п} = З_{о} \cdot Н_{п} \cdot 0{,}01, \quad (7.19)$$

де Нп – норматив витрат на освоєння нових мов програмування, %

$$О_{п} = 2090 \cdot 15 \cdot 0{,}01=314 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 40 прим.)

$$А_{м} = \frac{А_{р} \cdot N_{міс}}{N_{е} \cdot 12}, \quad (7.20)$$

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_{м} = 140140 \cdot 3/(40 \cdot 12)=876 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 2090+209+506+314+57+314+876 = 4366 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 55%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де Рс – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 4366 = 2401 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1. Основна зарплата виконавців	$З_о$	2090
2. Додаткова зарплата виконавців	$З_д$	209
3. Відрахування на соціальні потреби	$С_{оц}$	506
4. Загальногосподарські витрати	$Г_{осп}$	314
5. Витрати на матеріали	$З_М$	57
6. Освоєння нових операційних систем, мов програмування	$О_п$	314
7. Амортизація основних фондів	$А_м$	876
8. Повна собівартість програмного забезпечення	$С_п$	4366
9. Плановий прибуток	$П_р$	2401
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	6767
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1353,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	9168

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	9168
Всього капітальних витрат	–	9168

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	19290	3617
2. Витрати на електроенергію	$З_{ел}$	2099	1049
3. Витрати на амортизацію	$З_{ам}$	0	4584
Всього витрат за рік	$І$	21389	9250

Витрати на оплату праці:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \qquad (7.23)$$

де $Т_р$ – кількість годин обслуговування системи за рік, *год.*; $З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 800 годин на рік до 150 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\ баз} = 800 \cdot 16 \cdot 1{,}1 \cdot 1{,}37 = 19290\ грн.$$

до

$$З_{р\ нов} = 150 \cdot 16 \cdot 1{,}1 \cdot 1{,}37 = 3617\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$).

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

Зел баз=·0,475·2455·1,8 = 2099 грн

Зел нов=·0,475·1227·1,8 = 1049 грн

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	9168	–	4584
Всього відрахувань	-	–	9168	–	4584

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (6767 - 4366) \cdot 40 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 40000) \cdot 3/12 = 61005 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{188075}{(6767 - 4366) \cdot 40 \cdot 12 / 3} = 0,5 \; років.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н (K_н - K_б), \quad (7.27)$$

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (21389 - 9250) - 0,5 \cdot 9168 = 7555 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{9168}{21389 - 9250} = 0,8 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	40
2. Повна собівартість розробленої програми	Грн	4366
3. Ціна розробленої програми	Грн.	6767
4. Плановий прибуток від реалізації розробленої програми	Грн.	2401
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1596075
7. Загальний прибуток від реалізації програмної продукції	Грн.	96040
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	61005
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	9168
11.Величина економічного ефекту у користувача програмної продукції	Грн.	7555
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,8

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

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

Охорона праці є складовою частиною безпеки життєдіяльності [3].

Законом України “Про охорону праці” [2] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та «Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста включають наступні фактори:

– параметри монітора (випромінювання);

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

8.2 Аналіз умов праці

Фірма дотримується всіх правил з охорони праці і слідкує за їх дотриманням.

При виконанні робіт на комп'ютерах працівникам необхідно дотримуватись вимог загальної інструкції з охорони праці.

До роботи на комп'ютерах допускаються особи, які пройшли: медичний огляд, навчання по професії, вступний інструктаж з охорони праці та первинний інструктаж з охорони праці на робочому місці. В подальшому вони проходять повторні інструктажі з охорони праці на робочому місці один раз на півріччя, періодичні медичні огляди один раз на два роки.

Основним обладнанням робочого місця є монітор, системний блок, миша та клавіатура.

Робочі місця розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін на відстані 1м та між собою на відстані не менше 1,5 м. Відносно вікон робоче місце доцільно розташовувати таким чином, щоб природне світло падало на нього збоку, переважно зліва.

Монітор розташований на робочому місці так, щоб поверхня екрана знаходилася в центрі поля зору на відстані 400-700 мм від очей користувача. Елементи робочого місця розміщуються так, щоб витримувалася однакова відстань очей від екрана, клавіатури, тексту.

Джерела освітлення розташовані з обох боків екрану паралельно напрямку погляду. Для уникнення світлових відблисків екрану, клавіатури в напрямку очей користувача, від світильників загального освітлення або сонячних променів, також використовують антиблікові сітки, спеціальні фільтри для екранів, захисні козирки, на вікнах – жалюзі.

Використовуються скляні поляризаційні фільтри вони забезпечують найкращу якість зображення. Вони усувають практично всі відблиски, роблять зображення чітким і контрастним.

Зручна робоча поза при роботі з комп'ютером забезпечується регулюванням висоти робочого столу, крісла та підставки для ніг. Працівники мають пам’ятки, що раціональною робочою позою вважається положення, при якому ступні працівника розташовані горизонтально на підлозі або підставці для ніг, стегна зорієнтовані у горизонтальній площині, верхні частини рук – вертикальні. Кут ліктєвого суглоба коливається в межах 70 – 90°, зап'ястя зігнуті під кутом не більше ніж 20°, нахил голови 15 – 20°.

Для нейтралізації зарядів статичної електрики в приміщенні, де виконується робота на комп'ютерах, в тому числі на лазерних та світлодіодних принтерах, збільшується вологість повітря за допомогою кімнатних зволожувачів. Не рекомендується носити одяг з синтетичних матеріалів.

Для попередження травм усе електричне обладнання заземлене. Приступаючи до роботи працівникам необхідно перевірити справність обладнання. В разі виявлення порушень їм треба негайно повідомити про це свого начальника для вжиття заходів щодо усунення несправності. Проводити самому ремонт електроустаткування забороняється.

8.3 Розробка заходів з охорони праці

Працівники повинні дотримуватися статті 18 Закону України ”Про охорону праці” згідно цій статті працівники зобов'язані:

− знати і виконувати вимоги нормативних актів про охорону праці, правила поводження з устаткуванням та іншими засобами виробництва, користуватися засобами колективного та індивідуального захисту;

− співробітничати з власником у справі організації безпечних і нешкідливих умов праці, особисто вживати посильних заходів щодо усунення будь-якої виробничої ситуації, яка створює загрозу його життю чи здоров'ю, або людей, які його оточують, повідомляти про небезпеку свого безпосереднього керівника або іншу посадову особу;

− дотримуватись зобов'язань щодо охорони праці, передбачених колективним договором та правилами внутрішнього трудового розпорядку підприємства.

Також повинні виконуватися вимоги безпеки перед початком роботи, працівникам потрібно:

− увімкнути систему кондиціювання в приміщенні;

− перевірити надійність встановлення апаратури на робочому столі. Повернути монітор так, щоб було зручно дивитися на екран – під прямим кутом (а не збоку) і трохи зверху вниз, при цьому екран має бути трохи нахиленим, нижній його край ближче до оператора;

− перевірити загальний стан апаратури, перевірити справність електропроводки, з'єднувальних шнурів, штепсельних вилок, розеток, заземлення захисного екрана;

− відрегулювати освітленість робочого місця;

− відрегулювати та зафіксувати висоту крісла, зручний для користувача нахил його спинки;

− приєднати до системного блоку необхідну апаратуру. Усі кабелі, що

з’єднують системний блок з іншими пристроями, слід вставляти та виймати при вимкненому комп'ютері;

− ввімкнути апаратуру комп'ютера вимикачами на корпусах в послідовності: монітор, системний блок, принтер (якщо передбачається друкування);

− відрегулювати яскравість монітора, мінімальний розмір світної точки, фокусування, контрастність.

Працівники повинні дотримуватися рекомендацій:

− яскравість монітору – не менше 100Kg/M2;

− відношення яскравості монітора до яскравості оточуючих його поверхонь в робочій зоні – не більше 3:1;

− мінімальний розмір точки свічення не більше 0,4 мм для монохромного монітора і не менше 0,6 мм для кольорового, контрастність зображення знаку – не менше 0,8.

При виявленні будь-яких неполадок роботу не розпочинати, повідомити про це керівника.

Працівникам потрібно дотримуватися вимог безпеки під час виконання роботи:

− необхідно стійко розташовувати клавіатуру на робочому столі, не опускати її хитання. Під час роботи на клавіатурі сидіти прямо, не напружуватися;

− для забезпечення несприятливого впливу на користувача пристроїв типу ”миша” належить забезпечувати вільну велику поверхню столу для переміщення ”миші” і зручного упору ліктьового суглоба;

− не дозволяються сторонні розмови, подразнюючі шуми;

− періодично при вимкненому комп'ютері прибирати ледь змоченою мильним розчином бавовняною ганчіркою порох з поверхонь апаратури. Екран ВДТ та захисний екран протирають ганчіркою, змоченою у спирті. Не

дозволяється використовувати рідинні або аерозольні засоби чищення поверхонь комп'ютера.

Працівники не повинні порушувати правил з охорони праці та їм забороняється:

– самостійно ремонтувати апаратуру. Ремонт апаратури здійснюється спеціалістами з технічного обслуговування комп'ютера, 1 раз на півроку повинні відкривати процесор і вилучати пилососом пил і бруд, що накопичилися;

– класти будь-яку предмети на апаратуру комп'ютера;

– закривати будь-чим вентиляційні отвори апаратури, що може призвести до її перегрівання і виходу з ладу.

Для зняття статичної електрики рекомендується час від часу доторкатися до металевих поверхонь.

Розташувати принтер необхідно поруч з системним блоком таким чином, щоб з'єднувальний шнур не був натягнутий. Забороняється ставити принтери на системний блок.

Для досягнення найбільш чистих, з високою роздільністю зображень і щоб не зіпсувати апарат, має використовуватися папір, вказаний в інструкції до принтера. При зминанні паперу потрібно відкрити кришку і обережно витягнути лоток з папером.

Працівника потрібно дотримуватися вимоги безпеки після закінчення роботи:

– закінчити та записати у пам'ять комп'ютера файл, що знаходиться в роботі;

– вимкнути принтер та інші периферійні пристрої. Штепсельні вилки витягнути з розеток. Накрити клавіатуру кришкою запобігання попаданню в неї пилу;

– прибрати робоче місце;

– ретельно вимити руки теплою водою з милом;

– вимкнути кондиціонер, освітлення і загальне електроживлення;

− пройти в спеціально обладнаному приміщенні сеанс психофізіологічного розвантаження і зняття втоми з виконанням спеціальних вправ аутогенного тренування.

8.4 Пожежна безпека

Пожежі в приміщеннях з оргтехнікою становлять особливу небезпеку, бо поєднані з великими матеріальними збитками. Пожежа може виникнути при взаємодії горючих речовин і джерел запалювання. Горючими речовинами є будівельні та опоряджувальні матеріали, пластмасові корпуси техніки, шнури тощо. Джерелами запалювання можуть бути електронні схеми комп'ютерів, принтерів, пристроїв електроживлення, де внаслідок різних порушень виникає перегрівання елементів, утворюються електричні іскри та дуги, здатні спричинити займання горючих матеріалів.

З метою виявлення початкової стадії займання необхідно використовувати пристрої систем автоматичного пожежогасіння там, де цього вимагають правила пожежної безпеки.

При обслуговуванні, ремонтних та профілактичних роботах використовуються різні легкозаймисті рідини, прокладаються тимчасові електропровідники, здійснюється паяння. Виникає додаткова пожежна небезпека, яка потребує відповідних заходів пожежного захисту. До засобів гасіння пожежі, призначених для локалізації невеликих займань, належать вогнегасники, сухий пісок, азбестові ковдри. Приміщення, в який встановлено комп'ютери і де немає необхідності влаштування систем автоматичного пожежогасіння, необхідно оснащувати переносними вуглекислотними з розрахунку 2 шт. на кожні 20 м^2 в приміщеннях. Звуковбирне облицювання стін, стель приміщень треба виконувати з негорючих та важко горючих матеріалів.

Електроустановки (можливість їх застосування, монтаж, накладка експлуатація) повинні відповідати вимогам чинних правил улаштування

електроустановок, правил технічної експлуатації, електроустановок та інших нормативних документів.

Ймовірність виникнення пожежі від електротехнічного та іншого одиничного виробу не повинна перевищувати 10^{-6} на рік. При короткому замиканні в місцях з'єднання проводів опір практично дорівнює нулю, звідси величина струму досягає дуже великих значень.

Персональні комп'ютери після закінчення роботи повинні відключатися від мережі не рідше 1 разу на квартал, необхідно очищати від пилу агрегати та вузли, кабельні канали та простір між підлогами. Не дозволяється розміщувати комп'ютерні зали ЕОМ у підвалах; проводити ремонт вузлів (блоків) ЕОМ безпосередньо у залах, де знаходяться ПК (персональні комп'ютери), залишати без нагляду ввімкнену в мережу електронну апаратуру, яка використовується для контролю ЕОМ.

Електричний струм силою 0,1 А є небезпечним для людини. Для попередження травм усе електричне обладнання повинне бути заземлене. Приступаючи до роботи необхідно перевірити справність обладнання, ізоляцію проводів і надійність заземлення. Доторкання до оголених струмоведучих і незахищених частин в електроустаткуванні забороняється. В разі виявлення порушень ізоляції електропроводів, відкритих струмоведучих частин електроустаткування або порушення заземлення треба негайно повідомити про це свого начальника для вжиття заходів щодо усунення несправності. Проводити самому ремонт електроустаткування забороняється.

8.5 Розрахункова частина

Початкові данні для розрахунку захисного заземлення:

Тип заземлення: робоче заземлення нульової точки трансформатора. Заземленню підлягає виробниче обладнання організації. Напруга – 220/380 В. Розташування заземлюючих електродів – по контуру.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача методом коефіцієнта використання заземлювачів.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина. Умовна товщина верхнього шару грунта: *H=0,8 м*. Для захисного заземлення: застосовуються вертикальні електроди – пруток довжиною *L=2 м.* Відстань між вертикальними заземлювачами (електродами) *A*=2 *м*. Діаметр ве ртикальнього електрода (прутка) *D=60 мм*, Тип горизонтального заземлювача: металева полоса. Розміри перетину з'єднуючої полоси: *60 х 6 мм*. (*b=60 мм*.). Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Глибина закладення горизонтального контура заземлення *t= 0,7 м*.

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр.13-16 http://dspace.kntu.kr.ua/jspui/handle/123456789/4358[4], або будь якої відповідної іншої.

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,7+2/2=1,7 \text{ м.}$$

Еквівалентний питомий опір грунта:

$$\rho_{екв}= \psi \rho_1 \rho_2 L /[\rho_1 \psi(L-H+t)+ \rho_2 \psi(H-t)]=96,77 \text{ Ом}$$

де

$\psi = 1,2$ – табличне значення коефіцієнта сезонності для відповідной климатичной зони у багатошаровому грунті (джерело: http://ftemk.mpei.ac.ru/bgd/_private/Grunt_/t_goda.htm);

$\rho_1 = 50$ *Ом*м.* – табличне значення питомого опору верхнього шару грунта (джерело: https://el-line.ru/udelnoe_soprotivlenie_grunta.shtml);

$\rho_2 = 100$ *Ом*м.* – табличне значення питомого опору нижнього шару грунта (джерело: https://zandz.com/ru/udelnoe_soprotivlenie_grunta).

Опір розтіканню електричного струму одного електрода вертикального заземлювача:

$$R_O = [\rho_1/(2\pi*L)]*[(ln)2L/D)+0,5ln[(4T+L)/(4T-L)]=43,16 \text{ Ом}.$$

Опір розтіканню електричного струму заземлювача, який нормується, $R=1$ якщо $\rho_{екв} <100$ *(у нас* $\rho_{екв} = 96,77 <100$*),* *та* $R= R_{ЗН} *\rho_{екв}/100$, *якщо* $\rho_{екв} >100$.

Опір розтіканню електричного струму горизонтального заземлювача (полоси):

$$R_П = 0,366(\rho_{екв}\ \psi / L_П \cdot \eta_П)\lg(2/ L_П * L_П/bt)=40,32 \text{ Ом}.$$

де

$\eta_П = 0,3$ – табличне значення коефіцієнта використання горизонтального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення *A/L*;

довжина горизонтального заземлювача (полоси) при розташуванні заземлювачів по контуру:

$$L_П = A*n = 2*22 = 44 \text{ м.};$$

де n – ітераційна кількість вертикальних заземлювачів.

Загальний опір штучного заземлюючого пристрою розтіканню електричного струму на землю:

$$R_B = R_П\ R / (R_П - R) = 4,4 \text{ Ом}.$$

Кількість вертикальних заземлювачів:

$$n = R_O / R_B * \eta_B = 22 \text{шт}.$$

де $\eta_B = 0,6$ – табличне значення коефіцієнта використання вертикального заземлювача, залежить від розташованя (в ряд або по контуру) та співвідношення A/L.

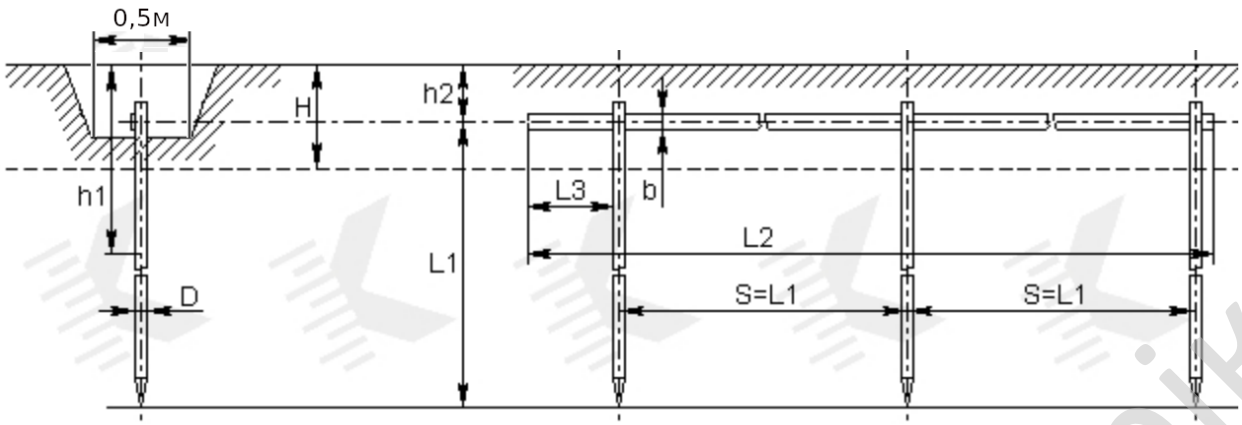

Рисунок 8.1 – Штучний заземлювач

8.6 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

– Досліджена система інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

– На основі отриманих результатів досліджень створена програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Embarcadero RAD Studio Delphi 10.3.2. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм LOKI_91.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 7555 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,8 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тарасов О.В. Дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.

2. Жданов А.А. Принцип автономного адаптивного управления. Диссертация на соискание ученой степени доктора физико-математических наук. ВЦ РАН. Москва, 1993. 318 с.

3. Жданов А.А. О подходе к моделированию управляемых объектов. Препринт ВЦ АН СССР. Сообщения по прикладной математике. Москва, 1991. 44 с.

4. Zhdanov A.A. A principle of Pattern Formation and Recognition// Pattern Recognition and Image Analysis. Vol.2. N3. 1992. – P. 249-264. (ISSN: 1054-6618).

5. Zhdanov A.A. Application of Pattern Recognition Procedure to the Acquisition and Use of Data in Control// Pattern Recognition and Image Analysis. Vol.2. N2. 1992. – P. 180-194. (ISSN: 1054-6618).

6. Жданов А.А. Накопление и использование информации при управлении в условиях неопределенности// Сб.науч.тр. Информационная технология и числение методы анализа распределених систем. – М.: ИФТП. 1992. С. 112-133.

7. Жданов А.А. Об одном подходе к адаптивному управлению// Сб. науч. тр. Анализ и оптимизация кибернетических систем, – М.: ГосИФТП, 1996. С. 42-64.

8. Жданов А.А. Об одном имитационном подходе к адаптивному управлению// Сб. Вопросы кибернетики. – М.: 1996. С. 171 – 206.

9. Жданов А.А. Формальная модель нейронна и нейромережі в методологии автономного адаптивного управления// Сб. Вопросы кибернетики. Вып. 3. М.: 1997. С. 258-274.

10. Жданов А.А., Б.Б.Беляев, В.В.Мамаев. Использование принципа автономного адаптивного управления в системе угловой стабилизации космического аппарата "Спектр РГ"// Сб.науч.тр. Информационная бионика и моделирование, – М.: ГосИФТП, 1995. С. 87-114.

11. В.В.Боровко. Математическая статистика.

12. В.Брауэр. Введение в теорию конечных автоматов. М, "Радио и связь":1987. 392 с.

13. В.А.Евстигнеев и др. Теория графов. Алгоритмы обработки деревьев. ВО "Наука",Новосибирск:1994. 360 с.

14. Жданов А.А. Формальная модель нейронна и нейромережі в методологии автономного адаптивного управления. Сборник «Вопросы кибернетики» №3. Научный совет по комплексной проблеме «Кибернетика» РАН. сс.258-273.

15. Michael I. Jordan, Cristopher M. Bishop. Neural Networks. Massachusets Institute of Technology. AI Memo No. 1562. Anonymous ftp://publications.ai.mit.edu.

16. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns. Elements of reusable object-oriented software. Addison Wesley. 1994. 395 с.

17. Valentin F. Turchin. The phenomenon of the science, a cybernatic approach to human evolution. Addison Wesley.

18. McCulloch W.W., Pitts W. 1943. A logical calculus of the ideas imminent in nervous activiti. Bulletinn of Mathematical biophysics 5: 115-33. (Русский перевод: Маккалок У.С., Питтс У. Логическое исчисление идей, относящихся к нервной деятельности. Автоматы. – М: ИЛ. – 1956.)

19. «Итоги науки и техники». Вычислительные науки, 1991, том 8, сс. 15-16, 26.

20. Уоссермен Ф.. Нейрокомпьютерная техника. – М.: Мир, 1992.

21. Круглов Владимир Васильевич, Борисов Вадим Владимирович Искусственние нейроние мережі. Теория и практика. – 1-е. – М.: Горячая линия – Телеком, 2001. – С. 382. – ISBN 5-93517-031-О.

22. В. А. Терехов, Д. В. Ефимов, И. Ю. Тюкин Нейросетевые системы управления. – 1-е. – Высшая школа, 2002. – С. 184. – ISBN 5-06-004094-1.

23. Уоссермен, Ф. Нейрокомпьютерная техника: Теория и практика = Neural Computing. Theory and Practice. – М.: Мир, 1992. – 240 с. – ISBN 5-03-002115-9.

24. Саймон Хайкин Нейроние мережі: полный курс = Neural Networks: A Comprehensive Foundation. – 2-е. – М.: «Вильямс», 2006. – С. 1104. – ISBN 0-13-273350-1.

25. Роберт Каллан Основные концепции нейроних сетей = The Essence of Neural Networks First Edition. – 1-е. – «Вильямс», 2001. – С. 288. – ISBN 5-8459-0210-X.

26. Л.Н. Ясницкий Введение в искусственний интеллект. – 1-е. – Издательский центр "Академия", 2005. – С. 176. – ISBN 5-7695-1958-4.

27. Г. К. Вороновский, К. В. Махотило, С. Н. Петрашев, С. А. Сергеев Генетические алгоритмы, искусственние нейроние мережі и проблемы виртуальной реальности. – заказное. – Х.: ОСНОВА, 1997. – С. 112. – ISBN 5-7768-0293-8.

28. Миркес Е. М., Нейрокомпьютер. Проект стандарта. – Новосибирск: Наука, 1999. – 337 с. ISBN 5-02-031409-9.

29. Коваленко А.С. Задачи распознавания ситуаций в ERP системах / А.В. Коваленко ., А.А. Смірнов, А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2014. – Вип. 4(120). – С. 161-164.

30. Коваленко А.С. Підсистема технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко , О.А.Смірнов, О.В. Коваленко // Системи озброєння і військова техніка.– Х.: ХУПС, 2014. – № 1(37). – С. 126-129.

31. Коваленко А.С. Анализ эффективности использования экспертной системы технической диагностики с традиционной структурой / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 2(38). – С. 106-108.

32. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

33. Коваленко А.С. Разработка структуры экспертной системы технической диагностики интегрированной информационной системы / А.С. Коваленко, А.А. Смирнов, А.В. Коваленко // Наука і техніка Повітряних Сил Збройних Сил України. – Харків: ХУПС, 2014. – № 2(15). – С.154-157.

34. Коваленко А.С. Структура системи технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Кіровоградського національного технічного університету / техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кіровоград: Вид-во КНТУ, 2014. – Вип. 27. – С. 245-251.

35. Коваленко А.С. Дослідження будови інтегрованої інформаційної системи та її елементів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2014. – № 4(40). – С. 85-88.

36. Коваленко А.С. Розробка структури бази даних для обліку технічного стану елементів інтегрованої інформаційної системи з урахуванням вимог споживачів інформації / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2015. – Вип. 1(126). – С. 75-79.

37. Коваленко А.С. Обґрунтування набору даних для оцінки технічного стану інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2015. – Вип. 1(42). – С.39-41.

38. Коваленко А.С. Експертна система технічного діагностування інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Системи озброєння і військова техніка. – Х.: ХУПС, 2015. – № 1(41). – С. 106-111.

39. Коваленко А.С. Удосконалення методу технічного обслуговування об'єктів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко, О.П. Доренський // Системи озброєння і військова техніка. – Х.: ХУПС, 2016. – № 2(46). – С. 109-114.

40. Коваленко А.С. Метод визначення оптимального комплексу робіт з відновлення працездатності інтегрованої системи технічної діагностики в умовах ресурсних обмежень / А.С. Коваленко // Системи обробки інформації. – Х.: ХУПС, 2016. – Вип. 3(140). – С. 69-72.

41. Kovalenko A.S. Information model and its element for displaying information on technical condition of objects of integrated information system / A.S. Kovalenko, A.A. Smirnov, A.V. Kovalenko, A.P. Dorensky // International Journal of Computational Engineering Research (IJCER). – India: Delhi, 2016. – Volume 6, Issue 1. – P. 21-27.

42. Кожанова А.С. Система технічної діагностики інтегрованих інформаційних систем – обґрунтування необхідності створення, визначення понятійного апарату та напрямів досліджень / А.С. Кожанова, О.А. Смірнов, М.П. Савченко, Д.М. Ізосімов, В.В. Мороз // Створення та модернізація озброєння і військової техніки в сучасних умовах: Тринадцята наук.-техн. конф., 5-6 вер. 2013 р., м. Феодосія: тези доп. – Феодосія: ДНВЦ, 2013. – С. 187-188.

43. Кожанова А.С. Визначення основних напрямків досліджень щодо створення системи технічної діагностики інтегрованих інформаційних систем / А.С. Кожанова, О.А. Смірнов, А.В. Челпанов // Проблемні питання розвитку озброєння та військової техніки Збройних Сил України: IV наук.-техн. конф., 16-20 груд. 2013 р., м. Київ: зб. тез. – Київ: ЦНДІ ОВТ ЗСУ, 2013. – С. 293.

44. Коваленко А.С. Обґрунтування необхідності створення систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Інформатика та системні науки : V Всеукр. наук.-практ. конф., 13–15 бер. 2014 р., м. Полтава : зб. тез. – Полтава: ПУЕТ, 2014. – С. 292-294.

45. Коваленко А.С. Задачи распознавания ситуаций в системах организационной стратегии интеграции производства и операций / А.С. Коваленко, А.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVI міжнар. наук.-практ. сем., 11-12 квіт. 2014 р., м. Кіровоград: зб. тез. – Кіровоград: КНТУ, 2014. – С. 53-55.

46. Коваленко А.С. Створення систем технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VI між нар. наук.-практ. конф., 17-18 квіт. 2014 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2014. – С. 241.

47. Коваленко А.С. Визначення понятійного апарату та напрямів досліджень для синтезу систем технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комп'ютерне моделювання у наукоємних технологіях (КМНТ-2014): наук.-техн. конф. з міжнар. участю, 28 -31 трав. 2014 р., м. Харків: зб. наук. праць. – Харків: ХНУ, 2014. – С. 190-193.

48. Коваленко А.С. Основні складові та функції системи технічної діагностики інтегрованих інформаційних систем / Коваленко А.С. // Інформаційні технології та комп'ютерна інженерія: наук.-практ. конф., 4 груд. 2014 р., м. Кіровоград: зб. тез доп. – Кіровоград: КНТУ, 2014. – С. 236.

49. Коваленко А.С. Розробка структури бази даних інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Проблеми і перспективи розвитку ІТ-індустрії: VII міжнар. наук.-практ. конф., 17-18 квіт. 2015 р., м. Харків: зб. тез. – Харків: ХНЕУ, 2015. – С. 15.

50. Коваленко А.С. Дослідження елементів інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Комбінаторні конфігурації та їх застосування: XVII між нар. наук.-практ. сем., 17-18 квіт. 2015 р., м. Кіровоград: зб. тез – Кіровоград: КНТУ, 2015. – С. 5.

51. Коваленко А.С. Метод автоматизованої перевірки результатів вимірювання параметрів об'єкті в інтегрованої інформаційної системи / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Стратегія якості у промисловості і освіті: XI міжнар. конф., 1 – 5 черв. 2015 р., м. Варна, Болгарія.: зб. матер. – Варна: ТУВ, 2015. – С. 423-426.

52. Коваленко А.С. Обґрунтування необхідності створення розподіленої бази даних для забезпечення захисту рухомих повітряних об`єктів / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Перспективні напрями захисту інформації: I всеукр. наук.-практ. конф., 07 вер. 2015 р., м. Одеса: зб. тез доп. – Одеса: ОНАЗ, 2015. – С. 35-39.

53. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

54. Про охорону праці: Закон України від 14.10.1992 р. № 2694-XII. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

55. . Щодо организации роботи по вопросам охорони праці та безпеки життєдіяльності у дошкільних навчальних закладах: Лист МОН України від 23.09.2014р. № 1 / 9-482. Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v-482729-14#Text

56. Охорона праці. Ч. 1. Захисне заземлення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака] – Кіровоград : КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 39-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Embarcadero RAD Studio Delphi 10.3.2.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинне бути розглянутий аналіз умов праці.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 132 аркуші.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2021 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 21.12.2021 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Петренюк В.І.

Дослідження та програмна реалізація системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 63

Літера: РП

Кропивницький – 2021 року

Основна програма

NeuralBaseTypes.pas – ініціалізація базових констант

```
unit NeuralBaseTypes;

interface

const

  DefaultAlpha = 1;               // Параметр активаційної функції
  DefaultEpochCount = 10000;      // Кількість етапів для навчання
  DefaultErrorValue = 0.05;       // Помилка за замовчуванням для всіх видів
  DefaultHopfLayerCount = 2;      // Кількість шарів у мережі Хопфилда
  DefaultLayerCount = 0;          // Мінімальна кількість шарів у мережі back-
propagation
  DefaultMaxIterCount = 10;       // Максимальна кількість ітерацій в алгоритмі
Хопфилда
  DefaultMomentum = 0.9;          // Імпульс - момент
  DefaultNeuronCount = 0;         // Кількість нейронів у прихованому шарі за
замовчуванням
  DefaultPatternCount = 0;        // Кількість прикладів
  DefaultTeachRate = 0.1;         // Швидкість навчання
  DefaultTeachIdentCount = 100;   // Необхідний відсоток розпізнаних прикладів з
навчальної множини
  DefaultTestIdentCount = 100;    // Необхідний відсоток розпізнаних прикладів з
тестової множини
  DefaultUseForTeach = 100;       // Відсоток прикладів використовуваних як
навчальна множина
  DefaultDeltaBarAcceleratingConst = 0.095;
  DefaultDeltaBarDecFactor = 0.85;
  DefaultRPropInitValue = 0.1;
  DefaultRPropMaxStepSize = 50;
  DefaultRPropMinStepSize = 1 E-6;
  DefaultRPropDecFactor = 0.5;
  DefaultRPropIncFactor = 1.2;
  DefaultSuperSABDecFactor = 0.5;
  DefaultSuperSABIncFactor = 1.05;

  SensorLayer = 0;                // Сенсорний шар
  BiasNeuron = 1;                 // Зсув у мережі back-propagation

  Separators = ['.', ','];
  Letters    = ['a'..'z'];
  Capitals   = ['A'..'Z'];
  DigitChars = ['0'..'9'];
  SpaceChar = #9;

resourcestring

  SFieldNorm = 'Не існує типу нормалізації %d';
  SFieldKind = 'Не існує типу поля %d';
  SFieldIndexRange = 'Неправильно зазначений номер поля %d';
  SInFieldCount = 'Неправильно встановлена кількість вхідних полів';
  SInNeuronCount = 'Неправильно встановлена кількість вхідних нейронів';
  SInVectorCount = 'Неправильно встановлена розмірність вхідного вектора';
  SLayerRangeIndex = 'Неправильно зазначений номер шару %d';
  SNeuronRangeIndex = 'Неправильно зазначений номер нейрона %d';
  SNeuronCount = 'Неправильно зазначена кількість нейронів';
  SOutFieldCount = 'Неправильно встановлена кількість вихідних полів';
  SOutNeuronCount = 'Неправильно встановлена кількість вихідних нейронів';
  SOutVectorCount = 'Неправильно встановлена розмірність вихідного вектора';
  SPatternRangeIndex = 'Вихід за межі масиву прикладів %d';
  SStreamCannotRead = 'Помилка читання з потоку';
  SWeightRangeIndex = 'Неправильно зазначений номер ваги %d';
  SWrongFileName = 'Неправильно зазначене ім'я файлу %s';
  SCannotBeNumber = 'Помилка, вираз %s неможливо привести до числового типу';
```

```
  SBPStopCondition = 'Не задана умова зупинки процесу навчання';

type

  TVectorInt = array of integer;
  TVectorFloat = array of double;
  TVectorString = array of string;
  TMatrixInt = array of array of integer;
  TMatrixFloat = array of array of double;
  TNormalize = (nrmLinear, nrmSigmoid, nrmAuto,  nrmNone,
                nrmLinearOut, nrmAutoOut);
  TNeuroFieldType = (fdInput, fdOutput, fdNone);

implementation


end.
```

NeuralBaseEditor.pas – проектування нейронних мереж системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства

```
unit NeuralBaseEditor;

interface

uses
  Classes,
  DesignIntf, DesignEditors,
  NeuralBaseComp, NeuralBaseEditorForm,
  SysUtils, Dialogs, Controls, NeuralBaseEditorFieldsForm;

type
  TNeuronsInLayerFieldsProperty = class(TStringProperty)
  public
    function GetAttributes : TPropertyAttributes; override;
    function GetValue : string; override;
    procedure Edit; override;
  end;

  TFileNameFieldsProperty = class(TStringProperty)
  public
    function GetAttributes : TPropertyAttributes; override;
    procedure Edit; override;
  end;

  TOptionsFieldsProperty = class(TStringProperty)
  public
    function GetAttributes : TPropertyAttributes; override;
    function GetValue : string; override;
    procedure Edit; override;
  end;

procedure Register;

implementation

function TOptionsFieldsProperty.GetAttributes;
begin
  Result := [paDialog];
end;

procedure TOptionsFieldsProperty.Edit;
var
  i: integer;
  xNeuralNetExtended: TNeuralNetExtended;
  frmNeuroFields: TfrmNeuroFields;
  xCurrent: integer;
begin
  frmNeuroFields := TfrmNeuroFields.Create(nil);
  try
    with frmNeuroFields do
    begin
      xNeuralNetExtended := (GetComponent(0) as TNeuralNetExtended);
      for i := 0 to xNeuralNetExtended.AvailableFieldsCount - 1 do
        ltbFieldName.Items.Add(xNeuralNetExtended.Fields[i].Name);
      xCurrent := 0;
      rdgFieldType.ItemIndex := xNeuralNetExtended.Fields[xCurrent].Kind;
      rdgNormType.ItemIndex := xNeuralNetExtended.Fields[xCurrent].NormType;
      edtAlpha.Text := FloatToStr(xNeuralNetExtended.Fields[xCurrent].Alpha);
      sttMin.Caption :=
FloatToStr(xNeuralNetExtended.Fields[xCurrent].ValueMin);
      sttMax.Caption :=
FloatToStr(xNeuralNetExtended.Fields[xCurrent].ValueMax);
      NeuralNetExtended := xNeuralNetExtended;
      ShowModal;
```

```
    end;
  except
    frmNeuroFields.Free;
  end;
end;

function TOptionsFieldsProperty.GetValue;
var
  i: integer;
begin
  // внести зміни
  Result := '[';
  with (GetComponent(0) as TNeuralNetExtended) do
    if AvailableFieldsCount > 1 then
    begin
      for i := 0 to AvailableFieldsCount - 1 do
        Result := Result + Fields[i].Name + ',';
      Result[Length(Result)] := ']'
    end
    else
      Result[Length(Result) + 1] := ']';
end;

function TNeuronsInLayerFieldsProperty.GetAttributes;
begin
  Result := [paDialog];
end;

function TNeuronsInLayerFieldsProperty.GetValue;
var
  i: integer;
begin
  // внести зміни
  Result := '[';
  with (GetComponent(0) as TNeuralNetBP) do
    if LayerCount > 0 then
    begin
      for i := 0 to LayerCount - 1 do
        Result := Result + IntToStr(LayersBP[i].NeuronCount) + ',';
      Result[Length(Result)] := ']';
    end
    else
      Result[Length(Result) + 1] := ']';
end;

procedure TNeuronsInLayerFieldsProperty.Edit;
var
  i: integer;
  xPreviousCount: integer;
  xNeuralNetBP: TNeuralNetBP;
  frmNeuronsInLayer: TfrmNeuronsInLayer;
  xChangesMade: boolean;
begin
  xChangesMade := False;
  frmNeuronsInLayer := TfrmNeuronsInLayer.Create(nil);
  try
    with frmNeuronsInLayer do
    begin
      xNeuralNetBP := (GetComponent(0) as TNeuralNetBP);
      speLayers.Value := xNeuralNetBP.LayerCount;
      stgNeuronsInLayer.RowCount := xNeuralNetBP.LayerCount + 1;
      for i := 0 to xNeuralNetBP.LayerCount - 1 do
      begin
        stgNeuronsInLayer.Cells[0, i + 1] := IntToStr(i);
        stgNeuronsInLayer.Cells[1, i + 1] :=
IntToStr(xNeuralNetBP.LayersBP[i].NeuronCount);
      end;
      if ShowModal = mrOk then
      begin
```

```
        xNeuralNetBP.ResetLayers;
        if speLayers.Value = 0 then
        begin
          xNeuralNetBP.ResetLayers;
          Exit;
        end;
        if xNeuralNetBP.LayerCount <> speLayers.Value then
          xChangesMade := True
        else
          for i := 0 to xNeuralNetBP.LayerCount - 1 do
            if xNeuralNetBP.LayersBP[i].NeuronCount <>
StrToInt(stgNeuronsInLayer.Cells[1, i + 1]) then
            begin
              xChangesMade := True;
              Break;
            end;
        if xChangesMade then
        begin
          if xNeuralNetBP.LayerCount = speLayers.Value then
            for i := 0 to speLayers.Value - 1 do
              xNeuralNetBP.NeuronsInLayer[i] := stgNeuronsInLayer.Cells[1, i +
1]
          else
            if xNeuralNetBP.LayerCount < speLayers.Value then
            begin
              for i := 0 to xNeuralNetBP.LayerCount - 1 do
                xNeuralNetBP.NeuronsInLayer[i] := stgNeuronsInLayer.Cells[1, i +
1];
              for i := xNeuralNetBP.LayerCount to speLayers.Value - 1 do
                xNeuralNetBP.AddLayer(StrToInt(stgNeuronsInLayer.Cells[1, i +
1]));
            end
            else
            begin
              if speLayers.Value > 0 then
                for i := 0 to speLayers.Value - 1 do
                  xNeuralNetBP.NeuronsInLayer[i] := stgNeuronsInLayer.Cells[1, i
+ 1];
              xPreviousCount := xNeuralNetBP.LayerCount -  speLayers.Value;
              for i := 1 to xPreviousCount do
                xNeuralNetBP.DeleteLayer(xNeuralNetBP.LayerCount - 1);
            end;
          if not xNeuralNetBP.AutoInit then
            xNeuralNetBP.AutoInit := True;
        end;
      end;
    end;
  except
    frmNeuronsInLayer.Free;
  end;
end;

function TFileNameFieldsProperty.GetAttributes;
begin
  Result := [paDialog];
end;

procedure TFileNameFieldsProperty.Edit;
var
  xOpenDialog: TOpenDialog;
begin
  xOpenDialog := TOpenDialog.Create(nil);
  if UpperCase(GetName) = 'FILENAME' then
    xOpenDialog.Filter := 'Нейронна мережа(*.nnw)|*.nnw|всі файли (*.*)|*.*';
  if UpperCase(GetName) = 'SOURCEFILENAME' then
    xOpenDialog.Filter := 'Текстові файли (*.txt)|*.txt|всі файли (*.*)|*.*';

  if xOpenDialog.Execute then
  begin
```

```
    if UpperCase(GetName) = 'FILENAME' then
      (GetComponent(0) as TNeuralNetExtended).FileName := xOpenDialog.FileName;
    if UpperCase(GetName) = 'SOURCEFILENAME' then
      (GetComponent(0) as TNeuralNetExtended).SourceFileName :=
xOpenDialog.FileName;
  end;
  xOpenDialog.Free;
end;

procedure Register;
begin
  RegisterPropertyEditor(TypeInfo(TStrings), TNeuralNetBP, 'NeuronsInLayer',
TNeuronsInLayerFieldsProperty);
  RegisterPropertyEditor(TypeInfo(TFileName), TNeuralNetExtended, '',
TFileNameFieldsProperty);
  RegisterPropertyEditor(TypeInfo(string), TNeuralNetExtended, 'Options',
TOptionsFieldsProperty);
end;

end.
```

NeuralBaseEditorForm.pas – редактор нейронних мереж системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства

```
unit NeuralBaseEditorForm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, StdCtrls, ExtCtrls, NeuralBaseComp, Spin, NeuralBaseTypes;

type
  TfrmNeuronsInLayer = class(TForm)
    Bevel1: TBevel;
    btnOk: TButton;
    btnCancel: TButton;
    stgNeuronsInLayer: TStringGrid;
    Label1: TLabel;
    speLayers: TSpinEdit;
    procedure stgNeuronsInLayerGetEditMask(Sender: TObject; ACol,
      ARow: Integer; var Value: String);
    procedure stgNeuronsInLayerSetEditText(Sender: TObject; ACol,
      ARow: Integer; const Value: String);
    procedure speLayersChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmNeuronsInLayer: TfrmNeuronsInLayer;

implementation

{$R *.DFM}

procedure TfrmNeuronsInLayer.stgNeuronsInLayerGetEditMask(Sender: TObject;
  ACol, ARow: Integer; var Value: String);
begin
  Value := '0000';
end;

procedure TfrmNeuronsInLayer.stgNeuronsInLayerSetEditText(Sender: TObject;
  ACol, ARow: Integer; const Value: String);
begin
{  with stgNeuronsInLayer do
    try
      Cells[ACol, ARow] := IntToStr(StrToInt(trim(Value)));
    except
      Cells[ACol, ARow] := IntToStr(DefaultNeuronCount);
    end;}
end;

procedure TfrmNeuronsInLayer.speLayersChange(Sender: TObject);
var
  i: integer;
begin
  stgNeuronsInLayer.RowCount := speLayers.Value + 1;
  for i := 1 to stgNeuronsInLayer.RowCount do
    if trim(stgNeuronsInLayer.Cells[1, i]) = '' then
    begin
      stgNeuronsInLayer.Cells[0, i] := IntToStr(i);
      stgNeuronsInLayer.Cells[1, i] := IntToStr(DefaultNeuronCount);
    end;
end;
```

```
procedure TfrmNeuronsInLayer.FormCreate(Sender: TObject);
begin
  stgNeuronsInLayer.Cells[0,0] := '# шару';
  stgNeuronsInLayer.Cells[1,0] := 'нейронів';
end;

end.
```

NeuralBaseEditorFieldsForm.pas – редактор властивостей полів нейроної мережі системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства

```
unit NeuralBaseEditorFieldsForm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, NeuralBaseComp;

type
  TfrmNeuroFields = class(TForm)
    ltbFieldName: TListBox;
    rdgFieldType: TRadioGroup;
    rdgNormType: TRadioGroup;
    Bevel1: TBevel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    sttMin: TStaticText;
    sttMax: TStaticText;
    edtAlpha: TEdit;
    btnOk: TButton;
    btnCancel: TButton;
    Bevel2: TBevel;
    Label4: TLabel;
    procedure ltbFieldNameClick(Sender: TObject);
    procedure rdgFieldTypeClick(Sender: TObject);
    procedure rdgNormTypeClick(Sender: TObject);
    procedure edtAlphaChange(Sender: TObject);
  private
    { Private declarations }
  public
    NeuralNetExtended: TNeuralNetExtended;
    { Public declarations }
  end;

var
  frmNeuroFields: TfrmNeuroFields;

implementation

{$R *.DFM}

procedure TfrmNeuroFields.ltbFieldNameClick(Sender: TObject);
begin
  rdgFieldType.ItemIndex :=
NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Kind;
  rdgNormType.ItemIndex :=
NeuralNetExtended.Fields[ltbFieldName.ItemIndex].NormType;
  edtAlpha.Text :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Alpha);
  sttMin.Caption :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].ValueMin);
  sttMax.Caption :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].ValueMax);
end;

procedure TfrmNeuroFields.rdgFieldTypeClick(Sender: TObject);
var
  i: integer;
begin
  if ltbFieldName.SelCount = 1 then
    NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Kind :=
rdgFieldType.ItemIndex
```

```
  else
    if ltbFieldName.SelCount > 1 then
    for i := 0 to ltbFieldName.Items.Count - 1 do
      if ltbFieldName.Selected[i] then
        NeuralNetExtended.Fields[i].Kind := rdgFieldType.ItemIndex;
end;

procedure TfrmNeuroFields.rdgNormTypeClick(Sender: TObject);
var
  i: integer;
begin
  if ltbFieldName.SelCount = 1 then
    NeuralNetExtended.Fields[ltbFieldName.ItemIndex].NormType :=
rdgNormType.ItemIndex
  else
    if ltbFieldName.SelCount > 1 then
    for i := 0 to ltbFieldName.Items.Count - 1 do
      if ltbFieldName.Selected[i] then
        NeuralNetExtended.Fields[i].NormType := rdgNormType.ItemIndex;
end;

procedure TfrmNeuroFields.edtAlphaChange(Sender: TObject);
begin
  NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Alpha :=
StrToFloat(edtAlpha.Text);
end;

end.
```

NeuralBaseComp.pas – формування бібліотеки шаблонів та дослідження нейронних мереж системи інтелектуального впровадження хмарних технологій в бізнес-процеси підприємства

```
unit NeuralBaseComp;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
  Forms, Dialogs, PumpData, NeuralBaseTypes, IniFiles, Math;

type

  // Класи виключень
  EInOutDimensionError = class(Exception);
  ENeuronCountError  = class(Exception);
  ENeuronNotEqualFieldError  = class(Exception);
  EBPStopCondition  = class(Exception);

  // Процедурні типи
  TActivation = function (Value: double): double of object;

  // Упереджуюче оголошення класів
  TNeuron = class;
  TLayer = class;

  // Базовий клас нейрона

  TNeuron = class(TObject)
  private
    FOutput: double;
    // Вектор ваг
    FWeights: TVectorFloat;
    // Покажчик на шар, у якому перебуває нейрон
    Layer: TLayer;
    function GetWeights(Index: integer): double;
    procedure SetWeights(Index: integer; Value: double);
    procedure SetWeightCount(const Value: integer);
  public
    constructor Create(ALayer: TLayer); virtual;
    destructor Destroy; override;
    // Ініціалізація ваг
    procedure InitWeights;  virtual;
    // Зважена сума
    procedure ComputeOut(const AInputs: TVectorFloat); virtual;
    property Output: double read FOutput write FOutput;
    property WeightCount: integer  write SetWeightCount;
    property Weights[Index: integer]: double read GetWeights write SetWeights;
  end;

  // Клас нейрона для мережі Хопфілда

  TNeuronHopf = class(TNeuron)
  public
    procedure ComputeOut(const AInputs: TVectorFloat); override;
  end;

  // Клас нейрона для мережі

  TNeuronBP = class(TNeuron)
  private
    // Локальна помилка
    FDelta: double;
    // Значення швидкості навчання на попередньому етапі
    FLearningRate: TVectorFloat;
    // Значення частинної похідної на попередньому етапі
```

```
    FPrevDerivative: TVectorFloat;
    // Значення корекції ваги на попередньому етапі
    FPrevUpdate: TVectorFloat;
    // Функція активації
    FOnActivation: TActivation;
    // Похідна функції активації
    FOnActivation: TActivation;
    function GetPrevUpdate(Index: integer): double;
    function GetPrevDerivative(Index: integer): double;
    function GetLearningRate(Index: integer): double;
    function GetPrevUpdateCount: integer;
    procedure SetPrevDerivative(Index: integer; const Value: double);
    procedure SetPrevDerivativeCount(const Value: integer);
    procedure SetDelta(Value: double);
    procedure SetPrevUpdate(Index: integer; Value: double);
    procedure SetPrevUpdateCount(const Value: integer);
    procedure SetLearningRate(Index: integer; const Value: double);
    procedure SetLearningRateCount(const Value: integer);
  public
    destructor Destroy; override;
    procedure ComputeOut(const AInputs: TVectorFloat); override;
    property Delta: double read FDelta write SetDelta;
    property LearningRate[Index: integer]: double read GetLearningRate write
SetLearningRate; //
    property LearningRateCount: integer write SetLearningRateCount;
    property PrevDerivativeCount: integer write SetPrevDerivativeCount;
    property PrevDerivative[Index: integer]: double read GetPrevDerivative write
SetPrevDerivative; //
    property PrevUpdateCount: integer read GetPrevUpdateCount write
SetPrevUpdateCount;
    property PrevUpdate[Index: integer]: double read GetPrevUpdate write
SetPrevUpdate;
    property OnActivation: TActivation read FOnActivation write FOnActivation;
    property OnActivation: TActivation read FOnActivation write FOnActivation;
  end;

  // Базовий клас шару
  TLayer = class(TPersistent)
  private
    FNumber: integer;
    // Розмірність NeuronCount
    FNeurons: array of TNeuron;
    function GetNeurons(Index: integer): TNeuron;
    function GetNeuronCount: integer;
    procedure SetNeurons(Index: integer; Value: TNeuron);
    procedure SetNeuronCount(Value: integer);
  public
    constructor Create(ALayerNumber: integer; ANeuronCount: integer); virtual;
    destructor Destroy; override;
    procedure Assign(Source: TPersistent); override;
    property Neurons[Index: integer]: TNeuron read GetNeurons write SetNeurons;
    property NeuronCount: integer read GetNeuronCount write SetNeuronCount;
  end;

  // Клас шару для мережі Хопфілда
  TLayerHopf = class(TLayer)
  public
    constructor Create(ALayerNumber: integer; ANeuronCount: integer); override;
  end;

  // Клас шару для мережі
  TLayerBP = class(TLayer)
  private
    function GetNeuronsBP(Index: integer): TNeuronBP;
    procedure SetNeuronsBP(Index: integer; Value: TNeuronBP);
  public
    constructor Create(ALayerNumber: integer; ANeuronCount: integer); override;
    destructor Destroy; override;
    procedure Assign(Source: TPersistent); override;
```

```
    property NeuronsBP[Index: integer]: TNeuronBP read GetNeuronsBP write
SetNeuronsBP;
  end;

  // Базовий клас мережі
  TNeuralNet = class(TComponent)
  private
    // Масив шарів
    FLayers: array of TLayer;
    // Число вибірок
    FPatternCount: integer;
    // Розмірність FPatternCount, InputNeuronCount
    FPatternsInput: TMatrixFloat;
    // Розмірність FPatternCount, OutputNeuronCount
    FPatternsOutput: TMatrixFloat;
    function GetLayers(Index: integer): TLayer;
    function GetOutputNeuronCount: integer;
    function GetPatternsOutput(PatternIndex: integer; OutputIndex: integer):
double;
    function GetPatternsInput(PatternIndex: integer; InputIndex: integer):
double;
    procedure SetLayers(Index: integer; Value: TLayer);
    procedure SetPatternsInput(PatternIndex: integer; InputIndex: integer;
Value: double);
    procedure SetPatternsOutput(PatternIndex: integer; InputIndex: integer;
Value: double);
  protected
    function GetLayerCount: integer; virtual;
    function GetInputNeuronCount: integer; virtual;
    procedure Clear; virtual;
    procedure ResizeInputDim; virtual;
    procedure ResizeOutputDim; virtual;
    procedure SetPatternCount(const Value: integer); virtual;
    procedure SetLayerCount(Value: integer); virtual;
    property PatternCount: integer read FPatternCount write SetPatternCount;
  public
    destructor Destroy; override;
    procedure AddLayer(ANeurons: integer); virtual; abstract;
    procedure AddPattern(const AInputs: TVectorFloat; const AOutputs:
TVectorFloat); overload; virtual;
    procedure DeleteLayer(Index: integer); virtual; abstract;
    procedure DeletePattern(Index: integer); virtual;
    procedure Init(const ANeuronsInLayer: TVectorInt); overload; virtual;
    property InputNeuronCount: integer read GetInputNeuronCount;
    property LayerCount: integer read GetLayerCount write SetLayerCount;
    property Layers[Index: integer]: TLayer read GetLayers write SetLayers;
    property OutputNeuronCount: integer read GetOutputNeuronCount;
    property PatternsInput[PatternIndex: integer; InputIndex: integer]: double
read GetPatternsInput write SetPatternsInput;
    property PatternsOutput[PatternIndex: integer; InputIndex: integer]: double
read GetPatternsOutput write SetPatternsOutput;
    procedure ResetPatterns; virtual;
  end;

  // Клас мережі Хопфілда
  TNeuralNetHopf = class(TNeuralNet)
  private
    FAutoInit: boolean;
    FInputNeuronCount: integer;
    FMaxIterCount: integer;
    FPatternCount: integer;
    FPatterns: TMatrixInt;
    FOnAfterInit: TNotifyEvent;
    FOnBeforeInit: TNotifyEvent;
    FOnPatternRecognized: TNotifyEvent;
    function GetInput(Index: integer): double;
    function GetPatterns(InputIndex: integer; PatternIndex: integer): integer;
    function Stabled: boolean;
    procedure SetInput(Index: integer; Value: double);
```

```
    procedure SetPatterns(InputIndex: integer; PatternIndex: integer; Value:
integer);
  protected
    function GetInputNeuronCount: integer; override;
    function GetLayerCount: integer; override;
    procedure SetInputNeuronCount(Value: integer);
    procedure SetPatternCount(const Value: integer); override;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure AddPattern(const ANewPattern: TVectorInt); reintroduce; overload;
    procedure Calc; virtual;
    procedure DeletePattern(Index: integer); override;
    procedure Init; reintroduce; overload;
    procedure InitWeights; virtual;
    procedure ResetPatterns; override;
    procedure ResizePatternsDim; virtual;
    property Input[Index: integer]: double read GetInput write SetInput;
    property LayerCount: integer read GetLayerCount write SetLayerCount;
    property Patterns[InputIndex: integer; PatternIndex: integer]: integer read
GetPatterns write SetPatterns;
  published
    property AutoInit: boolean read FAutoInit write FAutoInit;
    property InputNeuronCount: integer read GetInputNeuronCount write
SetInputNeuronCount;
    property MaxIterCount: integer read FMaxIterCount write FMaxIterCount;
    property OnAfterInit: TNotifyEvent read FOnAfterInit write FOnAfterInit;
    property OnBeforeInit: TNotifyEvent read FOnBeforeInit write FOnBeforeInit;
    property OnPatternRecognized: TNotifyEvent read FOnPatternRecognized write
FOnPatternRecognized;
    property PatternCount: integer read FPatternCount write SetPatternCount;
  end;

  // Клас мережі
  TNeuralNetBP = class(TNeuralNet)
  private
    // Коефіцієнт крутості граничної сигмоїдальної функції
    FAlpha: double;
    // Прапор автоініціалізації топології мережі
    FAutoInit: boolean;
    // Прапор продовження навчання
    FContinueTeach: boolean;
    // Бажаний вихід нейромережі розмірність OutputNeuronCount
    FDesiredOut: TVectorFloat;
    // Прапор зупинки при досягненні FEpochCount
    FEpoch: boolean;
    // Лічильник етапів (пред'явлення мережі всіх прикладів з навчальної
вибірки)
    FEpochCount: integer;
    // Номер поточної епохи
    FEpochCurrent: integer;
    // Значення помилки, при якій приклад вважається розпізнаним
    FIdentError: double;
    // Значення максимальної помилки на навчальній множині
    FMaxTeachResidual: double;
    // Значення максимальної помилки на тестовій множині
    FMaxTestResidual: double;
    // Значення середньої помилки на навчальній множині
    FMidTeachResidual: double;
    // Значення середньої помилки на тестовій множині
    FMidTestResidual: double;
    // Помилка на навчальній множині
    FTeachError: double;
    // Коефіцієнт інерційності
    FMomentum: double;
    // Кількість нейронів у шарах
    FNeuronsInLayer: TStrings;
    // Подія після ініціалізації
    FOnAfterInit: TNotifyEvent;
```

```
    FOnAfterNeuronCreated: TNotifyEvent;
    // Подія після навчання
    FOnAfterTeach: TNotifyEvent;
    // Подія до ініціалізації
    FOnBeforeInit: TNotifyEvent;
    // Подія до початку навчання
    FOnBeforeTeach: TNotifyEvent;
    // Подія після проходження одного етапу
    FOnEpochPassed: TNotifyEvent;
    // Число прикладів у навчальній безлічі
    FPatternCount: integer;
    // Масив утримуючий псевдовипадкову послідовність
    FRandomOrder: TVectorInt;
    // Лічильник розпізнаних прикладів на навчальній множині
    FRecognizedTeachCount: integer;
    // Лічильник розпізнаних прикладів на навчальній множині
    FRecognizedTestCount: integer;
    // Прапор зупинки навчання
    FStopTeach: boolean;
    FTeachStopped: boolean;
    // Коефіцієнт швидкості навчання - величина градієнтного кроку
    FTeachRate: double;
    // Число прикладів у тестовій множині
    FTestSetPatternCount: integer;
    // Розмірність FTestSetPatternCount, InputNeuronCount
    FTestSetPatterns: TMatrixFloat;
    // Розмірність FTestSetPatternCount, InputNeuronCount
    FTestSetPatternsOut: TMatrixFloat;
    function GetDesiredOut(Index: integer): double;
    function GetLayersBP(Index: integer): TLayerBP;
    function GetTestSetPatterns(InputIndex, PatternIndex: integer): double;
    function GetTestSetPatternsOut(InputIndex, PatternIndex: integer): double;
    procedure NeuronCountError;
    procedure NeuronsInLayerChange(Sender: TObject);
    procedure SetAlpha(Value: double);
    procedure SetDesiredOut(Index: integer; Value: double);
    procedure SetEpochCount(Value: integer);
    procedure SetLayersBP(Index: integer; Value: TLayerBP);
    procedure SetMomentum(Value: double);
    procedure SetTeachRate(Value: double);
    procedure SetTestSetPatternCount(const Value: integer);
    procedure SetTestSetPatterns(InputIndex, PatternIndex: integer; const Value:
double);
    procedure SetTestSetPatternsOut(InputIndex, PatternIndex: integer; const
Value: double);
    // Перетасування набору даних
    procedure Shuffle;
  protected
    function GetLayerCount: integer; override;
    function GetOutput(Index: integer): double; virtual;
    // Активаційна функція
    function Activation(Value: double): double; virtual;
    // Похідна активаційної функції
    function Activation(Value: double): double; virtual;
    // Середня квадратична помилка
    function QuadError: double; virtual;
    // Підстроювання ваг
    procedure AdjustWeights; virtual;
    // Розраховує  локальну помилку - дельту
    procedure CalcLocalError; virtual;
    // Перевірка мережі на тестовій множині
    procedure CheckTestSet; virtual;
    procedure DoOnAfterInit; virtual;
    procedure DoOnAfterNeuronCreated(ALayerIndex, ANeuronIndex: integer);
virtual;
    procedure DoOnAfterTeach; virtual;
    procedure DoOnBeforeInit; virtual;
    procedure DoOnBeforeTeach; virtual;
    procedure DoOnEpochPassed; virtual;
```

```
    // Ініціалізація ваг мережі псевдовипадковими значеннями
    procedure InitWeights; virtual;
    // Пред'явлення мережі вхідних значень приклада
    procedure LoadPatternsInput(APatternIndex :integer); virtual;
    // Пред'явлення мережі вхідних значень приклада
    procedure LoadPatternsOutput(APatternIndex :integer); virtual;
    // Поширює сигнал у прямому напрямку
    procedure Propagate; virtual;
    // Установка значень за замовчуванням
    procedure SetDefaultProperties; virtual;
    procedure SetPatternCount(const Value: integer); override;
    // Струс мережі
    procedure ShakeUp; virtual;
    property TeachStopped: boolean read FTeachStopped write FTeachStopped;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure AddLayer(ANeurons: integer); override;
    procedure Compute(AVector: TVectorFloat); virtual;
    procedure DeleteLayer(Index: integer); override;
    procedure Init; reintroduce; overload;
    procedure ResetLayers; virtual;
    procedure TeachOffLine; virtual;
    property DesiredOut[Index: integer]: double read GetDesiredOut write
SetDesiredOut;
    property EpochCurrent: integer read FEpochCurrent;
    property IdentError: double read FIdentError write FIdentError;
    property LayersBP[Index: integer]: TLayerBP read GetLayersBP write
SetLayersBP;
    property LayerCount: integer read GetLayerCount write SetLayerCount;
    property Output[Index: integer]: double read GetOutput;
    property StopTeach: boolean read FStopTeach write FStopTeach;
    property TeachError: double read FTeachError;
    property MaxTeachResidual: double read FMaxTeachResidual;
    property MaxTestResidual: double read FMaxTestResidual;
    property MidTeachResidual: double read FMidTeachResidual;
    property MidTestResidual: double read FMidTestResidual;
    property RecognizedTeachCount: integer read FRecognizedTeachCount;
    property RecognizedTestCount: integer read FRecognizedTestCount;
    property TestSetPatternCount: integer read FTestSetPatternCount write
SetTestSetPatternCount;
    property TestSetPatterns[InputIndex: integer; PatternIndex: integer]: double
read GetTestSetPatterns write SetTestSetPatterns;
    property TestSetPatternsOut[InputIndex: integer; PatternIndex: integer]:
double read GetTestSetPatternsOut write SetTestSetPatternsOut;
  published
    property Alpha: double read FAlpha write SetAlpha;
    property AutoInit: boolean read FAutoInit write FAutoInit;
    property ContinueTeach: boolean read FContinueTeach write FContinueTeach;
    property Epoch: boolean read FEpoch write FEpoch;
    property EpochCount: integer read FEpochCount write SetEpochCount;
    property Momentum: double read FMomentum write SetMomentum;
    property NeuronsInLayer: TStrings read FNeuronsInLayer write
FNeuronsInLayer;
    property OnAfterInit: TNotifyEvent read FOnAfterInit write FOnAfterInit;
    property OnAfterNeuronCreated: TNotifyEvent read FOnAfterNeuronCreated write
FOnAfterNeuronCreated;
    property OnAfterTeach: TNotifyEvent read FOnAfterTeach write FOnAfterTeach;
    property OnBeforeInit: TNotifyEvent read FOnBeforeInit write FOnBeforeInit;
    property OnBeforeTeach: TNotifyEvent read FOnBeforeTeach write
FOnBeforeTeach;
    property OnEpochPassed: TNotifyEvent read FOnEpochPassed write
FOnEpochPassed;
    property PatternCount: integer read FPatternCount write SetPatternCount;
    property TeachRate: double read FTeachRate write SetTeachRate;
  end;

  // Клас мережі back-propagation TNeuralNetExtended }
  TNeuralNetExtended = class(TNeuralNetBP)
```

```
private
  // Файл даних
  FNeuroDataSource: TNeuroDataSource;
  // Ім'я файлу даних *.txt
  FSourceFileName: TFileName;
  // Ім'я конфігураційного файлу *.nnw
  FFileName: TFileName;
  // Конфігураційний файл
  FNnwFile: TIniFile;
  // Поля
  FFields: TNeuroFields;
  // Кількість доступних полів
  FAvailableFieldsCount: integer;
  FMaxTeachError: boolean;
  FMaxTeachErrorValue: double;
  FMaxTestError: boolean;
  FMaxTestErrorValue: double;
  FMidTeachError: boolean;
  FMidTeachErrorValue: double;
  FMidTestError: boolean;
  FMidTestErrorValue: double;
  FOptions: string;
  FSettingsLoaded: boolean;
  FTestAsValid: boolean;
  FTeachIdent: boolean;
  FTeachIdentCount: integer;
  FTestIdent: boolean;
  FTestIdentCount: integer;
  FUseForTeach: integer;
  FIdentError: double;
  FRealOutputIndex: TVectorInt;
  FRealInputIndex: TVectorInt;
  function GetFields(Index: integer): TNeuroField;
  function GetInputFieldCount: integer;
  function GetOutputFieldCount: integer;
  function GetRealInputIndex(Index: integer): integer;
  function GetRealOutputIndex(Index: integer): integer;
  procedure SetFields(Index: integer; Value: TNeuroField);
  procedure SetFileName(Value: TFilename);
  procedure SetAvailableFieldsCount(Value: integer);
  procedure SetUseForTeach(const Value: integer);
  procedure SetTeachIdentCount(const Value: integer);
  procedure SetRealOutputIndex(Index: integer; const Value: integer);
  procedure SetRealOutputIndexCount(const Value: integer);
  procedure SetRealInputIndex(Index: integer; const Value: integer);
  procedure SetRealInputIndexCount(const Value: integer);
protected
  function GetOutput(Index: integer): double; override;
  procedure DoOnBeforeTeach; override;
  procedure DoOnEpochPassed; override;
  procedure SetDefaultProperties; override;
public
  constructor Create(AOwner: TComponent); override;
  destructor Destroy; override;
  procedure ComputeUnPrepData(AVector: TVectorFloat);
  // Завантажує дані з текстового файлу
  procedure LoadDataFrom;
  // Завантажує настроювання мережі
  procedure LoadNetwork;
  // Завантажує настроювання мережі
  procedure LoadPhase1;
  // Завантажує настроювання мережі
  procedure LoadPhase2;
  // Завантажує настроювання мережі
  procedure LoadPhase4;
  // Нормалізує набір даних
  procedure NormalizeData;
  // Зберігає настроювання мережі
  procedure SaveNetwork;
```

```
    // Зберігає настроювання мережі
    procedure SavePhase1;
    // Зберігає настроювання мережі
    procedure SavePhase2;
    // Зберігає настроювання мережі
    procedure SavePhase4;
    // Навчання нейронної мережі
    procedure Train;
    property AvailableFieldsCount: integer read FAvailableFieldsCount write
SetAvailableFieldsCount;
    property Fields[Index: integer]: TNeuroField read GetFields write SetFields;
    property InputFieldCount: integer read GetInputFieldCount;
    property OutputFieldCount: integer read GetOutputFieldCount;
    property SettingsLoaded: boolean read FSettingsLoaded write FSettingsLoaded;
    property RealOutputIndex[Index: integer]: integer read GetRealOutputIndex
write SetRealOutputIndex;
    property RealOutputIndexCount: integer write SetRealOutputIndexCount;
    property RealInputIndex[Index: integer]: integer read GetRealInputIndex
write SetRealInputIndex;
    property RealInputIndexCount: integer write SetRealInputIndexCount;
    property NnwFile: TIniFile read FNnwFile write FNnwFile;
  published
    property FileName: TFileName read FFileName write SetFileName;
    property IdentError: double read FIdentError write FIdentError;
    property MaxTeachError: boolean read FMaxTeachError write FMaxTeachError;
    property MaxTeachErrorValue: double read FMaxTeachErrorValue write
FMaxTeachErrorValue;
    property MaxTestError: boolean read FMaxTestError write FMaxTestError;
    property MaxTestErrorValue: double read FMaxTestErrorValue write
FMaxTestErrorValue;
    property MidTeachError: boolean read FMidTeachError write FMidTeachError;
    property MidTeachErrorValue: double read FMidTeachErrorValue write
FMidTeachErrorValue;
    property MidTestError: boolean read FMidTestError write FMidTestError;
    property MidTestErrorValue: double read FMidTestErrorValue write
FMidTestErrorValue;
    property Options: string read FOptions write FOptions;
    property SourceFileName: TFileName read FSourceFileName write
FSourceFileName;
    property TestAsValid: boolean read FTestAsValid write FTestAsValid;
    property TeachIdent: boolean read FTeachIdent write FTeachIdent;
    property TeachIdentCount: integer read FTeachIdentCount write
SetTeachIdentCount;
    property TestIdent: boolean read FTestIdent write FTestIdent;
    property TestIdentCount: integer read FTestIdentCount write FTestIdentCount;
    property UseForTeach: integer read FUseForTeach write SetUseForTeach;
  end;


procedure Register;

implementation

{$R *.RES}

{ TNeuron }

constructor TNeuron.Create(ALayer: TLayer);
begin
  inherited Create;
  // покажчик на шар у якому перебуває нейрон
  Layer := ALayer;
end;

destructor TNeuron.Destroy;
begin
  WeightCount := 0;
  FWeights := nil;
  Layer := nil;
```

```
  inherited;
end;

procedure TNeuron.ComputeOut(const AInputs: TVectorFloat);
var
  i: integer;
begin
  FOutput := 0;
  // Підраховується зважена сума нейрона
  for i := Low(AInputs) to High(AInputs) do
    FOutput := FOutput + FWeights[i] * AInputs[i];
end;

function TNeuron.GetWeights(Index: integer): double;
begin
  try
    Result := FWeights[Index];
  except
    on E: ERangeError do
      raise E.CreateFmt(SWeightRangeIndex, [Index])
  end;
end;

procedure TNeuron.InitWeights;
var
  i: integer;
begin
  // Ініціалізація ваг нейрона
  for i := Low(FWeights) to High(FWeights) do
    FWeights[i] := Random
end;

procedure TNeuron.SetWeightCount(const Value: integer);
begin
  SetLength(FWeights, Value);
end;

procedure TNeuron.SetWeights(Index: integer; Value: double);
begin
  try
    FWeights[Index] := Value;
  except
    on E: ERangeError do
      raise E.CreateFmt(SWeightRangeIndex, [Index])
  end;
end;

{ Кінець опису TNeuron }

{ TNeuronHopf }

procedure TNeuronHopf.ComputeOut(const AInputs: TVectorFloat);
begin
  inherited;
  // гранична функція
  if FOutput >= 0 then
    FOutput := 1
  else
    FOutput := -1
end;

{ Кінець опису TNeuronHopf}

{ TNeuronBP }

destructor TNeuronBP.Destroy;
begin
  FOnActivation := nil;
  FOnActivation := nil;
```

```
  PrevUpdateCount := 0;
  FPrevUpdate := nil;
  inherited;
end;

function TNeuronBP.GetLearningRate(Index: integer): double;
begin
  Result := FLearningRate[Index];
end;

function TNeuronBP.GetPrevDerivative(Index: integer): double;
begin
  Result := FPrevDerivative[Index];
end;

function TNeuronBP.GetPrevUpdateCount: integer;
begin
  Result := High(FPrevUpdate) + 1;
end;

function TNeuronBP.GetPrevUpdate(Index: integer): double;
begin
  Result := FPrevUpdate[Index];
end;

procedure TNeuronBP.ComputeOut(const AInputs: TVectorFloat);
begin
  inherited;
  // Задає зсув нейрона
  FOutput := FOutput + Weights[High(AInputs) + 1];
  FOutput := OnActivation(FOutput);
end;

procedure TNeuronBP.SetDelta(Value: double);
begin
  FDelta := Value;
end;

procedure TNeuronBP.SetLearningRate(Index: integer; const Value: double);
begin
  FLearningRate[Index] := Value;
end;

procedure TNeuronBP.SetLearningRateCount(const Value: integer);
begin
  SetLength(FLearningRate, Value)
end;

procedure TNeuronBP.SetPrevUpdate(Index: integer; Value: double);
begin
  FPrevUpdate[Index] := Value;
end;

procedure TNeuronBP.SetPrevUpdateCount(const Value: integer);
begin
  SetLength(FPrevUpdate, Value)
end;

procedure TNeuronBP.SetPrevDerivative(Index: integer; const Value: double);
begin
  FPrevDerivative[Index] := Value;
end;

procedure TNeuronBP.SetPrevDerivativeCount(const Value: integer);
begin
  SetLength(FPrevDerivative, Value)
end;

{ Кінець опису TNeuronBP }
```

```
{ TLayer }

procedure TLayer.Assign(Source: TPersistent);
var
  i: integer;
begin
  FNumber := (Source as TLayer).FNumber;
  NeuronCount := (Source as TLayer).NeuronCount;
  // Створюються нейрони
  for i := 0 to NeuronCount - 1 do
    FNeurons[i] := TNeuron.Create(Self);
end;

constructor TLayer.Create(ALayerNumber: integer; ANeuronCount: integer);
var
  i: integer;
begin
  inherited Create;
  FNumber := ALayerNumber;
  NeuronCount := ANeuronCount;
  for i := 0 to ANeuronCount - 1 do
    FNeurons[i] := TNeuron.Create(Self);
end;

destructor TLayer.Destroy;
var
  i: integer;
begin
  for i := 0 to NeuronCount - 1 do
    FNeurons[i].Free;
  NeuronCount := 0;
  FNeurons := nil;
  inherited;
end;

function TLayer.GetNeuronCount: integer;
begin
  Result := High(FNeurons) + 1;
end;

function TLayer.GetNeurons(Index: integer): TNeuron;
begin
  Result := FNeurons[Index];
end;

procedure TLayer.SetNeuronCount(Value: integer);
begin
  if Value <> High(FNeurons) + 1 then
    SetLength(FNeurons, Value);
end;

procedure TLayer.SetNeurons(Index: integer; Value: TNeuron);
begin
  try
    FNeurons[Index] := Value;
  except
    on E: ERangeError do
      raise E.CreateFmt(SNeuronRangeIndex, [Index])
  end;
end;

{ TLayerHopf }

constructor TLayerHopf.Create(ALayerNumber: integer; ANeuronCount: integer);
var
  i: integer;
begin
  FNumber := ALayerNumber;
```

```
  NeuronCount := ANeuronCount;
  for i := 0 to ANeuronCount - 1 do
    FNeurons[i] := TNeuronHopf.Create(Self);
end;

{ TLayerBP }

procedure TLayerBP.Assign(Source: TPersistent);
var
  i: integer;
begin
  FNumber := (Source as TLayerBP).FNumber;
  NeuronCount := (Source as TLayerBP).NeuronCount;
  for i := 0 to NeuronCount - 1 do
    FNeurons[i] := TNeuronBP.Create(Self);
end;

constructor TLayerBP.Create(ALayerNumber: integer; ANeuronCount: integer);
var
  i: integer;
begin
  FNumber := ALayerNumber;
  NeuronCount := ANeuronCount;
  for i := 0 to ANeuronCount - 1 do
    FNeurons[i] := TNeuronBP.Create(Self);
end;

destructor TLayerBP.Destroy;
begin
  inherited;
end;

function TLayerBP.GetNeuronsBP(Index: integer): TNeuronBP;
begin
  Result := FNeurons[Index] as TNeuronBP;
end;

procedure TLayerBP.SetNeuronsBP(Index: integer; Value: TNeuronBP);
begin
  FNeurons[Index] := Value as TNeuronBP;
end;

{ TNeuralNet }

destructor TNeuralNet.Destroy;
begin
  Clear;
  SetLength(FPatternsInput, 0, 0);
  FPatternsInput := nil;
  SetLength(FPatternsOutput, 0 ,0);
  FPatternsOutput := nil;
  FLayers := nil;
  inherited;
end;

procedure TNeuralNet.Clear;
var
  i, xCount: integer;
begin
  xCount := LayerCount;
  if  xCount > 0 then
  begin
    for i := 0 to xCount - 1 do
      FLayers[i].Free;
    LayerCount := 0;
  end;
end;

function TNeuralNet.GetInputNeuronCount: integer;
```

```
begin
  Result := Layers[SensorLayer].NeuronCount;
end;

function TNeuralNet.GetLayerCount: integer;
begin
  Result := High(FLayers) + 1;
end;

function TNeuralNet.GetLayers(Index: integer): TLayer;
begin
  Result := FLayers[Index];
end;

function TNeuralNet.GetOutputNeuronCount: integer;
begin
  Result := Layers[LayerCount - 1].NeuronCount;
end;

function TNeuralNet.GetPatternsInput(PatternIndex: integer; InputIndex:
integer): double;
begin
  Result := FPatternsInput[PatternIndex, InputIndex];
end;

procedure TNeuralNet.AddPattern(const AInputs: TVectorFloat; const AOutputs:
TVectorFloat);
var
  i: integer;
begin
  if InputNeuronCount <> High(AInputs) + 1 then
    raise EInOutDimensionError.Create(SInVectorCount);
  if OutputNeuronCount <> High(AOutputs) + 1 then
    raise EInOutDimensionError.Create(SOutVectorCount);
  PatternCount := PatternCount + 1;
  ResizeInputDim;
  ResizeOutputDim;
  for i := Low(AInputs) to High(AInputs) do
    PatternsInput[PatternCount - 1, i] := AInputs[i];
  for i := Low(AOutputs) to High(AOutputs) do
    PatternsOutput[PatternCount - 1, i] := AOutputs[i];
end;

procedure TNeuralNet.DeletePattern(Index: integer);
var
  i, j: integer;
begin
  try
    // видаляє вхідні значення приклада Index
    for i := Index to FPatternCount - 2 do
      for j := 0 to InputNeuronCount - 1 do
        FPatternsInput[i, j] := FPatternsInput[i + 1, j];
    // видаляє вихідні значення приклада Index
    for i := Index to FPatternCount - 2 do
      for j := 0 to OutputNeuronCount - 1 do
        FPatternsOutput[i, j] := FPatternsOutput[i + 1, j];
    Dec(FPatternCount);
    ResizeInputDim;
    ResizeOutputDim;
  except
    on E: ERangeError do
      raise E.CreateFmt(SPatternRangeIndex, [Index])
  end;
end;

procedure TNeuralNet.ResetPatterns;
begin
  FPatternCount := DefaultPatternCount;
  ResizeInputDim;
```

```
  ResizeOutputDim;
end;

procedure TNeuralNet.SetPatternCount(const Value: integer);
begin
  if Value < DefaultPatternCount then
    FPatternCount := DefaultPatternCount
  else
    FPatternCount := Value;
  ResizeInputDim;
  ResizeOutputDim;
end;

procedure TNeuralNet.SetPatternsOutput(PatternIndex: integer; InputIndex:
integer; Value: double);
begin
  FPatternsOutput[PatternIndex, InputIndex] := Value;
end;

procedure TNeuralNet.SetPatternsInput(PatternIndex: integer; InputIndex:
integer; Value: double);
begin
  FPatternsInput[PatternIndex, InputIndex] := Value;
end;

procedure TNeuralNet.Init(const ANeuronsInLayer: TVectorInt);
var
  i,j: integer;
begin
  LayerCount := High(ANeuronsInLayer) + 1;
  // FLayers[0] нульовий шар і виконує роль розподільного,
  // використовується тільки поле Output
  FLayers[0] := TLayer.Create(0, ANeuronsInLayer[0]);
  // для нульового шару не потрібні вагові коефіцієнти
  for i := 1 to LayerCount - 1 do
  begin
    FLayers[i] := TLayer.Create(i, ANeuronsInLayer[i]);
    for j := 0 to ANeuronsInLayer[i] - 1 do
    with FLayers[i].FNeurons[j] do
      // задає кількість елементів у векторі ваг нейрона j в
      //  шарі i рівним кількості виходів попереднього шару
      WeightCount := FLayers[ i-1].NeuronCount;
  end;
end;

procedure TNeuralNet.ResizeInputDim;
begin
  SetLength(FPatternsInput, FPatternCount, InputNeuronCount)
end;

procedure TNeuralNet.ResizeOutputDim;
begin
  SetLength(FPatternsOutput, FPatternCount, OutputNeuronCount)
end;

procedure TNeuralNet.SetLayerCount(Value: integer);
begin
  SetLength(FLayers, Value);
end;

procedure TNeuralNet.SetLayers(Index: integer; Value: TLayer);
begin
  try
    FLayers[Index] := Value;
  except
    on E: ERangeError do
      raise E.CreateFmt(SLayerRangeIndex, [Index])
  end;
end;
```

```
{  TNeuralNetHopf }

constructor TNeuralNetHopf.Create(AOwner: TComponent);
begin
  inherited;
  PatternCount := DefaultPatternCount;
  InputNeuronCount := DefaultNeuronCount;
  MaxIterCount := DefaultMaxIterCount;
  AutoInit := False;
end;

destructor TNeuralNetHopf.Destroy;
begin
  FOnAfterInit := nil;
  FOnBeforeInit := nil;
  FOnPatternRecognized := nil;
  SetLength(FPatterns, 0, 0);
  FPatterns := nil;
  inherited;
end;

function TNeuralNetHopf.GetInput(Index: integer): double;
begin
  Result := Layers[1].Neurons[Index].Output;
end;

function TNeuralNetHopf.GetInputNeuronCount: integer;
begin
  Result := Layers[SensorLayer].NeuronCount;
end;

function TNeuralNetHopf.GetLayerCount: integer;
begin
  Result := DefaultHopfLayerCount;
end;

function TNeuralNetHopf.GetPatterns(InputIndex: integer; PatternIndex: integer):
integer;
begin
  Result := FPatterns[InputIndex, PatternIndex];
end;

function TNeuralNetHopf.Stabled: boolean;
var
  i: integer;
begin
  // Порівнює вихідні значення попередньої
  //  ітерації зі значеннями поточної
  Result := True;
  for i := 0 to InputNeuronCount - 1  do
    if FLayers[1].FNeurons[i].FOutput <> FLayers[0].FNeurons[i].FOutput then
    begin
      Result := False;
      Exit
    end;
end;

procedure TNeuralNetHopf.AddPattern(const ANewPattern: TVectorInt);
var
  i: integer;
begin
  if InputNeuronCount <> High(ANewPattern)+ 1 then
    raise EInOutDimensionError.Create(SInNeuronCount);
  PatternCount := PatternCount + 1;
  ResizePatternsDim;
  for i := 0 to FInputNeuronCount - 1 do
    FPatterns[FPatternCount - 1, i] := ANewPattern[i];
  if AutoInit then
```

```
    InitWeights;
end;

procedure TNeuralNetHopf.Calc;
var
  i: integer;
  xCurrentIter: integer;
  xArray: TVectorFloat;
begin
  SetLength(xArray, InputNeuronCount);
  // Цикл працює поки не стабілізуються виходи
  xCurrentIter := 0;
  repeat
    for i := 0 to InputNeuronCount - 1 do
    begin
      // Запам'ятовує попередній крок ітерації, для
      //  цього використовується нульовий шар
      Layers[SensorLayer].Neurons[i].Output := Layers[1].Neurons[i].Output;
      xArray[i] := Layers[1].Neurons[i].Output;
    end;
    for i := 0 to InputNeuronCount - 1 do
      with Layers[1].Neurons[i] do
        // Розраховується новий стан нейронів і аксонів
        ComputeOut(xArray);
    Inc(xCurrentIter);
  until Stabled or (MaxIterCount = xCurrentIter);
  if Assigned(FOnAfterInit) then
    FOnAfterInit(Self);
  SetLength(xArray, 0);
  xArray := nil;
end;

procedure TNeuralNetHopf.DeletePattern(Index: integer);
var
  i, j: integer;
begin
  try
    for i := Index to FPatternCount - 2 do
      for j := 0 to FInputNeuronCount - 1 do
        FPatterns[i, j] := FPatterns[i + 1, j];
    Dec(FPatternCount);
    ResizePatternsDim;
    if AutoInit then
      InitWeights;
  except
    on E: ERangeError do
      raise E.CreateFmt(SPatternRangeIndex, [Index])
  end;
end;

procedure TNeuralNetHopf.Init;
var
  i, j: integer;
begin
  if Assigned(FOnBeforeInit) then
    FOnBeforeInit(Self);
  LayerCount := DefaultHopfLayerCount;
  for i := 0 to LayerCount - 1 do
    FLayers[i] := TLayerHopf.Create(i, FInputNeuronCount);
  // Для нульового шару не потрібні вагові коефіцієнти
  for j := 0 to FInputNeuronCount - 1 do
    with FLayers[1].FNeurons[j] do
      // задає кількість елементів у векторі
      WeightCount := FInputNeuronCount;
  if Assigned(FOnAfterInit) then
    FOnAfterInit(Self);
end;

procedure TNeuralNetHopf.InitWeights;
```

```
var
  i, j, k : integer;
begin
  // Ініціалізує вагову матрицю
  for i := 0 to InputNeuronCount - 1 do
    for j := 0 to InputNeuronCount - 1 do
    with Layers[1].Neurons[i] do
    begin
      Weights[j] := 0;
      if i <> j then
        for k := 0 to PatternCount - 1 do
          Weights[j] := Weights[j]+ Patterns[k, i] * Patterns[k, j]
    end;
end;

procedure TNeuralNetHopf.ResetPatterns;
begin
  PatternCount := DefaultPatternCount;
  if AutoInit then
    InitWeights;
end;

procedure TNeuralNetHopf.ResizePatternsDim;
begin
  SetLength(FPatterns, FPatternCount, FInputNeuronCount);
end;

procedure TNeuralNetHopf.SetInput(Index: integer; Value: double);
begin
  try
    Layers[1].Neurons[Index].Output := Value;
  except
    on E: ERangeError do
      raise E.CreateFmt(SPatternRangeIndex, [Index])
  end;
end;

procedure TNeuralNetHopf.SetInputNeuronCount(Value: integer);
begin
  if Value > DefaultNeuronCount then
    FInputNeuronCount := Value
  else
    FInputNeuronCount := DefaultNeuronCount;
  ResizePatternsDim;
  Init;
end;

procedure TNeuralNetHopf.SetPatternCount(const Value: integer);
begin
  if Value < DefaultPatternCount then
    FPatternCount := DefaultPatternCount
  else
    FPatternCount := Value;
end;

procedure TNeuralNetHopf.SetPatterns(InputIndex: integer; PatternIndex: integer;
Value: integer);
begin
  FPatterns[InputIndex, PatternIndex] := Value;
end;

{ TNeuralNetBP }

constructor TNeuralNetBP.Create(AOwner: TComponent);
var
  i: integer;
begin
  inherited;
  FNeuronsInLayer := TStringList.Create;
```

```
  for i := 0 to DefaultLayerCount do
    AddLayer(DefaultNeuronCount);
  TStringList(FNeuronsInLayer).OnChange := NeuronsInLayerChange;
  AutoInit := True;
  StopTeach := False;
  TeachStopped := False;
  NeuronsInLayerChange(Self);
  SetDefaultProperties;
end;

destructor TNeuralNetBP.Destroy;
begin
  FNeuronsInLayer.Free;
  SetLength(FRandomOrder, 0);
  FRandomOrder := nil;
  SetLength(FDesiredOut, 0);
  FDesiredOut := nil;
  SetLength(FTestSetPatterns, 0, 0);
  FTestSetPatterns := nil;
  SetLength(FTestSetPatternsOut, 0, 0);
  FTestSetPatternsOut := nil;
  FOnAfterInit := nil;
  FOnAfterTeach := nil;
  FOnBeforeInit := nil;
  FOnBeforeTeach := nil;
  FOnEpochPassed := nil;
  inherited;
end;

function TNeuralNetBP.GetLayersBP(Index: integer): TLayerBP;
begin
  Result := FLayers[Index] as TLayerBP;
end;

function TNeuralNetBP.GetLayerCount: integer;
begin
  Result := High(FLayers) + 1;
end;

function TNeuralNetBP.GetDesiredOut(Index: integer): double;
begin
  Result := FDesiredOut[Index];
end;

function TNeuralNetBP.GetOutput(Index: integer): double;
begin
  try
    Result := LayersBP[LayerCount - 1].NeuronsBP[Index].Output;
  except
    on E: ERangeError do
    raise E.CreateFmt(SNeuronRangeIndex, [Index])
  end;
end;

function TNeuralNet.GetPatternsOutput(PatternIndex: integer; OutputIndex:
integer): double;
begin
  Result := FPatternsOutput[PatternIndex, OutputIndex];
end;

function TNeuralNetBP.QuadError: double;
var
  i: integer;
begin
  // розраховує середньоквадратичну помилку
  Result := 0;
  for i := 0 to OutputNeuronCount - 1 do
    Result := Result + sqr(LayersBP[LayerCount - 1].NeuronsBP[i].Output -
DesiredOut[i]);
```

```
  Result := Result/2;
end;

function TNeuralNetBP.Activation(Value: double): double;
begin
  // Активаційна функція - сигмоїд
  Result := 1/( 1 + exp(-FAlpha * Value) )
end;

function TNeuralNetBP.Activation(Value: double): double;
begin
  // Похідна сигмоїди
  Result := FAlpha * Value * (1 - Value)
end;

function TNeuralNetBP.GetTestSetPatterns(InputIndex, PatternIndex: integer):
double;
begin
  Result := FTestSetPatterns[InputIndex, PatternIndex];
end;

function TNeuralNetBP.GetTestSetPatternsOut(InputIndex, PatternIndex: integer):
double;
begin
  Result := FTestSetPatternsOut[InputIndex, PatternIndex];
end;

procedure TNeuralNetBP.AddLayer(ANeurons: integer);
begin
  if ANeurons < DefaultNeuronCount then
    NeuronCountError
  else
    NeuronsInLayer.Add(IntToStr(ANeurons));
end;

procedure TNeuralNetBP.AdjustWeights;
var
  i, j, k: integer;
  xCurrentUpdate: double;
begin
  // Підстроювання ваг починаючи з першого шару
  for i := 1 to LayerCount - 1 do
    for j := 0 to LayersBP[i].NeuronCount - 1 do
    begin
      for k := 0 to LayersBP[ i-1].NeuronCount  do
      with LayersBP[i].NeuronsBP[j] do
      begin
        // коректує вагу з'єднуючого j-нейрона шару i
        //  з k-нейроном шару i-1:  добутком дельта j-нейрона
        //  на вихід k-нейрона шару i-1
        if k = LayersBP[ i-1].NeuronCount then
           // якщо це нейрон, що задає зсув
           xCurrentUpdate := -TeachRate * Delta + Momentum * PrevUpdate[k]
        else
           xCurrentUpdate := -TeachRate * Delta *
           LayersBP[ i-1].NeuronsBP[k].Output + Momentum * PrevUpdate[k];
        Weights[k]:= Weights[k] + xCurrentUpdate;
        PrevUpdate[k] := xCurrentUpdate;
      end;
    end
end;

procedure TNeuralNetBP.CalcLocalError;
var
  i, j, k: integer;
begin
  // Дельта-правило з останнього шару до першого
  for i := LayerCount - 1 downto 1 do
    // для останнього шару
```

```
    if i = LayerCount - 1 then
      for j := 0 to LayersBP[i].NeuronCount - 1 do
        LayersBP[i].NeuronsBP[j].Delta := (LayersBP[i].NeuronsBP[j].Output-
DesiredOut[j])
        * Activation(LayersBP[i].NeuronsBP[j].Output)
    else
      for j := 0 to LayersBP[i].NeuronCount - 1 do
        with LayersBP[i].NeuronsBP[j] do
        begin
          Delta := 0;
          // Підсумує добуток локальної помилки k-нейрона шару i+1
          //  на вагу з'єднуючий k-нейрон шару i+1 з j-нейроном шару i
          for k := 0 to LayersBP[i+1].NeuronCount - 1 do
            Delta := Delta + LayersBP[i+1].NeuronsBP[k].Delta *
                             LayersBP[i+1].NeuronsBP[k].Weights[j];
          Delta := Delta * Activation(Output)
        end;
end;

procedure TNeuralNetBP.CheckTestSet;
var
  i, j: integer;
  xArray: TVectorFloat;
  xFirstTestSample: boolean;
  xQuadError: double;
  // функція розраховує середньоквадратичну помилку
  function QuadError(APatternCount: integer): double;
  var
    i: integer;
  begin
    Result := 0;
    for i := 0 to OutputNeuronCount - 1 do
      Result := Result + sqr(LayersBP[LayerCount - 1].NeuronsBP[i].Output -
TestSetPatternsOut[APatternCount, i]);
    Result := Result/2;
  end;
begin
  SetLength(xArray, InputNeuronCount);
  xFirstTestSample := True;
  FRecognizedTestCount := 0;
  FMidTestResidual := 0;
  FMaxTestResidual := 0;
  for i := 0 to TestSetPatternCount - 1 do
  begin
    for j := 0 to InputNeuronCount - 1 do
      xArray[j] := TestSetPatterns[i, j];
    Compute(xArray);
    xQuadError := QuadError(i);
    // перевірка - чи розпізнаний приклад з тестової множини
    if xQuadError < IdentError then
      Inc(FRecognizedTestCount);
    FMidTestResidual := FMidTestResidual + xQuadError;
    // максимальна помилка на тестовій множині
    if xFirstTestSample then
    begin
      FMaxTestResidual := xQuadError;
      xFirstTestSample := False;
    end
    else
      if FMaxTestResidual < xQuadError then
        FMaxTestResidual := xQuadError;
  end;
  // середня помилка на тестовій множині
  FMidTestResidual := FMidTestResidual/TestSetPatternCount;
  SetLength(xArray, 0);
  xArray := nil;
end;

procedure TNeuralNetBP.Compute(AVector: TVectorFloat);
```

```
var
  i: integer;
begin
  if InputNeuronCount <> High(AVector)+ 1 then
    raise EInOutDimensionError.Create(SInNeuronCount);
  for i := Low(AVector) to High(AVector) do
    LayersBP[SensorLayer].NeuronsBP[i].Output :=  AVector[i];
  Propagate;
end;

procedure TNeuralNetBP.DoOnAfterInit;
begin
  if Assigned(FOnAfterInit) then
    FOnAfterInit(Self);
end;

procedure TNeuralNetBP.DoOnAfterNeuronCreated(ALayerIndex, ANeuronIndex:
integer);
var
  i: integer;
begin
  with LayersBP[ALayerIndex].NeuronsBP[ANeuronIndex] do
    for i := 0 to PrevUpdateCount - 1 do
      PrevUpdate[i] := 0;
  if Assigned(FOnAfterNeuronCreated) then
    FOnAfterNeuronCreated(Self);
end;

procedure TNeuralNetBP.DoOnAfterTeach;
begin
  if Assigned(FOnAfterTeach) then
    FOnAfterTeach(Self);
end;

procedure TNeuralNetBP.DoOnBeforeInit;
begin
  if Assigned(FOnBeforeInit) then
    FOnBeforeInit(Self);
end;

procedure TNeuralNetBP.DoOnBeforeTeach;
begin
  if Assigned(FOnBeforeTeach) then
    FOnBeforeTeach(Self);
end;

procedure TNeuralNetBP.DoOnEpochPassed;
begin
  if Assigned(FOnEpochPassed) then
    FOnEpochPassed(Self);
end;

procedure TNeuralNetBP.DeleteLayer(Index: integer);
var
  i: integer;
begin
  try
    NeuronsInLayer.Delete(Index);
    for i := Index to LayerCount - 2 do
      LayersBP[i].Assign(LayersBP[i + 1]);
    FLayers[LayerCount - 1].Free;
    LayerCount := LayerCount - 1;
  except
    on E: ERangeError do
      raise E.CreateFmt(SLayerRangeIndex, [Index])
  end;
end;

procedure TNeuralNetBP.Init;
```

```
var
  i, j: integer;
begin
  DoOnBeforeInit;
  if NeuronsInLayer.Count > 0 then
  begin
  LayerCount := NeuronsInLayer.Count;
    // FLayers[0] нульовий шар, використовується тільки поле Output
    FLayers[0] := TLayerBP.Create(0, StrToInt(NeuronsInLayer.Strings[0]));
    // для нульового шару не потрібні вагові коефіцієнти
    for i := 1 to LayerCount - 1 do
    begin
      FLayers[i] := TLayerBP.Create(i, StrToInt(NeuronsInLayer.Strings[i]));
      for j := 0 to StrToInt(NeuronsInLayer.Strings[i]) - 1 do
      with LayersBP[i].NeuronsBP[j] do
      begin
        // задає кількість елементів у векторі ваг + зсув
        WeightCount := LayersBP[ i-1].NeuronCount + BiasNeuron;
        // задає кількість  у векторі утримуючих попередню
        //  корекцію елементів  + зсув
        PrevUpdateCount := LayersBP[ i-1].NeuronCount + BiasNeuron;
        PrevDerivativeCount := LayersBP[ i-1].NeuronCount + BiasNeuron; // для
швидких алгоритмів
        LearningRateCount := LayersBP[ i-1].NeuronCount + BiasNeuron;   // для
швидких алгоритмів
        OnActivation := Activation;
        OnActivation := Activation;
        Randomize;
        DoOnAfterNeuronCreated(i, j);
      end
    end;
    // установлює розмірність масиву виходів
    //  число нейронів в останньому шарі = числу виходів
    SetLength(FDesiredOut, OutputNeuronCount);
  end;
  DoOnAfterInit;
end;

procedure TNeuralNetBP.InitWeights;
var
  i,j: integer;
begin
  Randomize;
  // Ініціалізація ваг
  for i := 1 to LayerCount - 1 do
    for j := 0 to LayersBP[i].NeuronCount - 1 do
       LayersBP[i].NeuronsBP[j].InitWeights;
end;

procedure TNeuralNetBP.LoadPatternsInput(APatternIndex :integer);
var
  i: integer;
begin
  for i := 0 to InputNeuronCount - 1 do
    LayersBP[SensorLayer].NeuronsBP[i].Output :=  PatternsInput[APatternIndex,
i];
end;

procedure TNeuralNetBP.LoadPatternsOutput(APatternIndex :integer);
var
  i: integer;
begin
  for i := 0 to OutputNeuronCount - 1 do
    DesiredOut[i] := PatternsOutput[APatternIndex, i];
end;

procedure TNeuralNetBP.NeuronsInLayerChange(Sender: TObject);
begin
  if AutoInit then
```

```
    Init;
end;

procedure TNeuralNetBP.NeuronCountError;
begin
  raise ENeuronCountError.Create(SNeuronCount)
end;

procedure TNeuralNetBP.Propagate;
var
  i, j, xIndex: integer;
  xArray: TVectorFloat;
begin
   // Поширення сигналу в прямому напрямку з першого шару
  for i := 1 to LayerCount - 1 do
  begin
    // формування масиву входів з виходів попереднього шару
    SetLength(xArray, LayersBP[ i-1].NeuronCount);
    for xIndex := 0 to LayersBP[ i-1].NeuronCount - 1 do
       xArray[xIndex] := LayersBP[ i-1].NeuronsBP[xIndex].Output;
    // обчислення виходу нейрона
    for j := 0 to LayersBP[i].NeuronCount - 1 do
      with LayersBP[i].NeuronsBP[j] do
        ComputeOut(xArray);
    for xIndex := 0 to LayersBP[ i-1].NeuronCount - 1 do
       xArray[xIndex] := 0;
  end;
  SetLength(xArray, 0);
  xArray := nil;
end;

procedure TNeuralNetBP.ResetLayers;
begin
  Clear;
  FNeuronsInLayer.Clear;
end;

procedure TNeuralNetBP.SetDesiredOut(Index: integer; Value: double);
begin
  FDesiredOut[Index] := Value;
end;

procedure TNeuralNetBP.SetLayersBP(Index: integer; Value: TLayerBP);
begin
  FLayers[Index] := Value as TLayerBP;
end;

procedure TNeuralNetBP.SetAlpha(Value: double);
begin
  if (Value > 10) or (Value < 0.01) then
    FAlpha := DefaultAlpha
  else
    FAlpha := Value;
end;

procedure TNeuralNetBP.SetTeachRate(Value: double);
begin
  if (Value > 1) or (Value <= 0) then
    FTeachRate := DefaultTeachRate
  else
    FTeachRate := Value;
end;

procedure TNeuralNetBP.SetTestSetPatterns(InputIndex, PatternIndex: integer;
const Value: double);
begin
  FTestSetPatterns[InputIndex, PatternIndex] := Value;
end;
```

```
procedure TNeuralNetBP.SetTestSetPatternsOut(InputIndex, PatternIndex: integer;
const Value: double);
begin
  FTestSetPatternsOut[InputIndex, PatternIndex] := Value;
end;

procedure TNeuralNetBP.SetTestSetPatternCount(const Value: integer);
begin
  FTestSetPatternCount := Value;
  SetLength(FTestSetPatterns, FTestSetPatternCount, InputNeuronCount);
  SetLength(FTestSetPatternsOut, FTestSetPatternCount, OutputNeuronCount);
end;

procedure TNeuralNetBP.SetMomentum(Value: double);
begin
  if (Value > 1) or (Value < 0) then
    FMomentum := DefaultMomentum
  else
    FMomentum := Value;
end;

procedure TNeuralNetBP.SetEpochCount(Value: integer);
begin
  if Value < 1 then
    FEpochCount := 1
  else
    FEpochCount := Value;
end;

procedure TNeuralNetBP.ShakeUp;
var
  i, j, k: integer;
begin
  Randomize;
  for i := 1 to LayerCount - 1 do
    for j := 0 to LayersBP[i].NeuronCount - 1 do
      for k := 0 to LayersBP[ i-1].NeuronCount  do
        with LayersBP[i].NeuronsBP[j] do
          Weights[k]:= Weights[k] + Random*0.1-0.05;
end;

procedure TNeuralNetBP.Shuffle;
var
  i, j, xNewInd, xLast: integer;
  xIsUnique : boolean;
begin
  xNewInd := 0;
  FRandomOrder[0] := Round(Random(FPatternCount));
  xLast := 0;
  for i := 1 to PatternCount - 1 do
  begin
    xIsUnique := False;
    while not xIsUnique do
    begin
      xNewInd := Round((Random(FPatternCount)));
      xIsUnique := True;
      for j := 0 to xLast do
        if xNewInd = FRandomOrder[j] then
          xIsUnique := False;
    end;
    FRandomOrder[i] := xNewInd;
    xLast := xLast +1;
  end;
end;

procedure TNeuralNetBP.TeachOffLine;
var
  j: integer;
  xQuadError: double;
```

```
  xNewEpoch: boolean;
begin
  DoOnBeforeTeach;
  if not ContinueTeach then
  begin
    // ваги ініціалізуються, якщо мережа навчається з "нуля"
    InitWeights;
    FEpochCurrent := 1;
  end;
  Randomize;
  SetLength(FRandomOrder, FPatternCount);
  TeachStopped := False;
  while (FEpochCurrent <= EpochCount) do
  begin
    FTeachError := 0;
    FMaxTeachResidual := 0;
    FRecognizedTeachCount := 0;
    xNewEpoch := True;
    Shuffle;
    for j := 0 to PatternCount - 1 do
    begin
      LoadPatternsInput(FRandomOrder[j]);
      LoadPatternsOutput(FRandomOrder[j]);
      Propagate;
      xQuadError := QuadError;
      // перевірка - чи розпізнаний приклад з навчальної множини
      if xQuadError < IdentError then
        Inc(FRecognizedTeachCount);
      FTeachError := FTeachError + xQuadError;
      // максимальна помилка на навчальній множині
      if xNewEpoch then
      begin
        FMaxTeachResidual := xQuadError;
        xNewEpoch := False;
      end
      else
        if MaxTeachResidual < xQuadError then
          FMaxTeachResidual := xQuadError;
      CalcLocalError;
      AdjustWeights;
    end;
    // середня помилка на навчальній множині
    FMidTeachResidual := TeachError/PatternCount;
    // перевірка мережі на узагальнення
    if TestSetPatternCount > 0 then
      CheckTestSet;
    DoOnEpochPassed;
    if StopTeach then
    begin
      TeachStopped := True;
      Exit;
    end;
    Inc(FEpochCurrent);
  end;
  DoOnAfterTeach;
end;

procedure TNeuralNetBP.SetPatternCount(const Value: integer);
begin
  FPatternCount := Value;
  inherited;
end;

procedure TNeuralNetBP.SetDefaultProperties;
begin
  // параметри встановлювані за замовчуванням
  Alpha := DefaultAlpha;
  ContinueTeach := False;
  Epoch := True;
```

```
  EpochCount := DefaultEpochCount;
  Momentum := DefaultMomentum;
  TeachRate := DefaultTeachRate;
  ResizeInputDim;
  ResizeOutputDim;
end;

{ TNeuralNetExtended }

constructor TNeuralNetExtended.Create(AOwner: TComponent);
begin
  inherited;
  SetDefaultProperties;
end;

destructor TNeuralNetExtended.Destroy;
var
  i: integer;
begin
  if Assigned(FNnwFile) then
    FNnwFile.Free;
  FNeuroDataSource.Free;
  for i := 0 to FAvailableFieldsCount - 1 do
    FFields[i].Free;
  inherited;
end;

function TNeuralNetExtended.GetFields(Index: integer): TNeuroField;
begin
  Result := FFields[Index];
end;

function TNeuralNetExtended.GetInputFieldCount: integer;
var
  i: integer;
begin
  Result := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if Fields[i].KindName = fdInput then
      Inc(Result);
end;

function TNeuralNetExtended.GetOutputFieldCount: integer;
var
  i: integer;
begin
  Result := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if Fields[i].KindName = fdOutput then
      Inc(Result);
end;

function TNeuralNetExtended.GetOutput(Index: integer): double;
var
  xTmp: double;
begin
  with Fields[RealOutputIndex[Index]] do
    case NormTypeName of
      nrmAuto: begin
                 xTmp := -ln(1/LayersBP[LayerCount - 1].NeuronsBP[Index].Output -
1);
                 LayersBP[LayerCount - 1].NeuronsBP[Index].Output := xTmp *
Dispersion + ValueMid;
               end;
      nrmLinear:  Result := (LayersBP[LayerCount - 1].NeuronsBP[Index].Output +
1)*(ValueMax - ValueMin)/2 + ValueMin;
      nrmLinearOut: Result := LayersBP[LayerCount -
1].NeuronsBP[Index].Output*(ValueMax - ValueMin) + ValueMin;
```

```
      nrmSigmoid:  Result := - Ln(1/LayersBP[LayerCount -
1].NeuronsBP[Index].Output - 1)/Alpha;
    end;
end;

function TNeuralNetExtended.GetRealInputIndex(Index: integer): integer;
begin
  Result := FRealInputIndex[Index];
end;

function TNeuralNetExtended.GetRealOutputIndex(Index: integer): integer;
begin
  Result := FRealOutputIndex[Index];
end;

procedure TNeuralNetExtended.ComputeUnPrepData(AVector: TVectorFloat);
var
  i: integer;
  xTmp: double;
begin
  if InputNeuronCount <> High(AVector)+ 1 then
    raise EInOutDimensionError.Create(SInNeuronCount);
  for i := Low(AVector) to High(AVector) do
  with FFields[RealInputIndex[i]] do
    case NormTypeName of
      nrmAuto: begin
                 xTmp := (LayersBP[SensorLayer].NeuronsBP[i].Output -
ValueMid)/Dispersion;
                 LayersBP[SensorLayer].NeuronsBP[i].Output := 1/(1 + exp(-
xTmp));
               end;
      nrmLinear: LayersBP[SensorLayer].NeuronsBP[i].Output := 2*(AVector[i] -
ValueMin)/(ValueMax - ValueMin) - 1;
      nrmLinearOut: LayersBP[SensorLayer].NeuronsBP[i].Output := (AVector[i] -
ValueMin)/(ValueMax - ValueMin);
      nrmSigmoid: LayersBP[SensorLayer].NeuronsBP[i].Output := 1/( 1 + exp(-
Alpha * AVector[i]));
    end;
  Propagate;
end;

procedure TNeuralNetExtended.DoOnBeforeTeach;
begin
  if InputNeuronCount <> InputFieldCount then
    raise ENeuronNotEqualFieldError.Create(SInFieldCount);
  if OutputNeuronCount <> OutputFieldCount then
    raise ENeuronNotEqualFieldError.Create(SOutFieldCount);
  if InputNeuronCount < 0 then
    raise EInOutDimensionError.Create(SInNeuronCount);
  if OutputNeuronCount < 0 then
    raise EInOutDimensionError.Create(SOutNeuronCount);
  if (not FMaxTeachError) and (not FMaxTestError) and
    (not FMidTeachError) and (not FMidTestError) and (not FEpoch) then
    raise EBPStopCondition.Create(SBPStopCondition);
  inherited DoOnBeforeTeach;
end;

procedure TNeuralNetExtended.DoOnEpochPassed;
begin
  if MaxTeachError and (MaxTeachResidual < MaxTeachErrorValue) then
    StopTeach := True;
  if MidTeachError and (MidTeachResidual < MidTeachErrorValue) then
    StopTeach := True;
  if MaxTestError and (MaxTestResidual < MaxTestErrorValue) then
    StopTeach := True;
  if MidTestError and (MidTestResidual < MidTestErrorValue) then
    StopTeach := True;
  if TeachIdent and (Round((FRecognizedTeachCount * 100)/PatternCount) <
TeachIdentCount) then
```

```
    StopTeach := True;
  if TestIdent and (Round((FRecognizedTestCount * 100)/TestSetPatternCount) <
TestIdentCount) then
    StopTeach := True;
  inherited DoOnEpochPassed;
end;

procedure TNeuralNetExtended.LoadDataFrom;
var
  xTempStream: TFileStream;
  i, j: integer;
  xFieldCount: integer;
  xArray: TVectorFloat;
  xPatternsList: TStringList;
begin
  // створюється потік
  xTempStream := TFileStream.Create(FSourceFileName, fmOpenRead);
  // створюється список
  xPatternsList := TStringList.Create;
  xPatternsList.LoadFromStream(xTempStream);
  try
    if SettingsLoaded then
    begin
      xFieldCount := FNeuroDataSource.FieldCount(xPatternsList.Strings[0]);
      if AvailableFieldsCount <> xFieldCount then
        if MessageDlg('Кількість полів у файлі даних не відповідає значенню
AvailableFieldsCount'+ #13 + 'Установити нове значення AvailableFieldsCount = '+
IntToStr(xFieldCount),
                      mtConfirmation, [mbYes, mbNo], 0) = mrYes then
          AvailableFieldsCount := xFieldCount;
    end
    else
      AvailableFieldsCount :=
FNeuroDataSource.FieldCount(xPatternsList.Strings[0]);
    FNeuroDataSource.ExtractHeaders(FFields, xPatternsList.Strings[0]);
    // установлюється розмірність часового масиву
    SetLength(xArray, FAvailableFieldsCount);
    // установлюється розмірність масиву даних
    if FUseForTeach = 100 then
      PatternCount := xPatternsList.Count - 1
    else
    begin
      PatternCount := Round((xPatternsList.Count - 1) * FUseForTeach / 100);
      TestSetPatternCount := xPatternsList.Count - PatternCount - 1;
    end;
    for i := 0 to FAvailableFieldsCount - 1 do
      FFields[i].DataInCount := xPatternsList.Count - 1;
    for j := 0 to xPatternsList.Count - 2 do
    begin
      FNeuroDataSource.ExtractValues(xArray, xPatternsList.Strings[j + 1]);
      for i := 0 to FAvailableFieldsCount - 1 do
        FFields[i].DataIn[j] := xArray[i]
    end;
  finally
    xTempStream.Free;
    xPatternsList.Free;
    SetLength(xArray, 0);
    xArray := nil;
  end;
end;

procedure TNeuralNetExtended.LoadPhase1;
begin
  FSourceFileName := FNnwFile.ReadString('Phase1', 'LearnSampleFileName', '');
  FNeuroDataSource.Name := FSourceFileName;
end;

procedure TNeuralNetExtended.LoadPhase2;
var
```

```
  i: integer;
begin
  AvailableFieldsCount := FNnwFile.ReadInteger('Phase2', 'AvailableFieldsCount',
1);
  for i := 0 to AvailableFieldsCount - 1 do
  with FFields[i] do
  begin
    Name := FNnwFile.ReadString('Phase2', 'FieldName_'+IntToStr(i), '');
    Kind := FNnwFile.ReadInteger('Phase2', 'FieldType_'+IntToStr(i), 0);
    NormType := FNnwFile.ReadInteger('Phase2', 'NormType_'+IntToStr(i), 0);
    ValueMax := FNnwFile.ReadFloat('Phase2', 'Max_'+IntToStr(i), 0);
    ValueMin := FNnwFile.ReadFloat('Phase2', 'Min_'+IntToStr(i), 0);
    ValueMid := FNnwFile.ReadFloat('Phase2', 'Mid_'+IntToStr(i), 0);
    Dispersion := FNnwFile.ReadFloat('Phase2', 'Disp_'+IntToStr(i), 0);
    Alpha := FNnwFile.ReadFloat('Phase2', 'Alpha_'+IntToStr(i), 0);
    Ind := FNnwFile.ReadBool('Phase2', 'Ind_'+IntToStr(i), False);
  end;
  SettingsLoaded := True;
end;

procedure TNeuralNetExtended.LoadPhase4;
begin
  UseForTeach := FNnwFile.ReadInteger('Phase4', 'UseForTeach',
DefaultUseForTeach);
  IdentError:= FNnwFile.ReadFloat('Phase4', 'IdentErr', DefaultErrorValue);
  TestAsValid := FNnwFile.ReadBool('Phase4', 'TestAsValid', False);
  Epoch:= FNnwFile.ReadBool('Phase4', 'Epoch', False);
  EpochCount:= FNnwFile.ReadInteger('Phase4', 'Epoch', DefaultEpochCount);
  MaxTeachError:= FNnwFile.ReadBool('Phase4', 'MaxTeachErr', False);
  MaxTeachErrorValue:= FNnwFile.ReadFloat('Phase4', 'MaxTeachErr',
DefaultErrorValue);
  MidTeachError:= FNnwFile.ReadBool('Phase4', 'MidTeachErr', False);
  MidTeachErrorValue:= FNnwFile.ReadFloat('Phase4', 'MidTeachErr',
DefaultErrorValue);
  TeachIdent:= FNnwFile.ReadBool('Phase4', 'TeachIdent', False);
  TeachIdentCount:= FNnwFile.ReadInteger('Phase4', 'TeachIdent',
DefaultTeachIdentCount);
  MaxTestError:= FNnwFile.ReadBool('Phase4', 'MaxTestErr', False);
  MaxTestErrorValue:= FNnwFile.ReadFloat('Phase4', 'MaxTestErr',
DefaultErrorValue);
  MidTestError:= FNnwFile.ReadBool('Phase4', 'MidTestErr', False);
  MidTestErrorValue:= FNnwFile.ReadFloat('Phase4', 'MidTestErr',
DefaultErrorValue);
  TestIdent:= FNnwFile.ReadBool('Phase4', 'TestIdent', False);
  TestIdentCount:= FNnwFile.ReadInteger('Phase4', 'TestIdent',
DefaultTestIdentCount);
end;

procedure TNeuralNetExtended.LoadNetwork;
var
  i,j,k: integer;
  xLayerCount: integer;
begin
  // знищується поточна конфігурація нейромережі
  ResetLayers;
  Alpha := FNnwFile.ReadFloat('Network', 'Alpha', DefaultAlpha);
  Momentum := FNnwFile.ReadFloat('Network', 'Miu', DefaultMomentum);
  TeachRate := FNnwFile.ReadFloat('Network', 'TeachSpeed', DefaultTeachRate);
  EpochCount := FNnwFile.ReadInteger('Network', 'Epoch', DefaultEpochCount);
  xLayerCount := FNnwFile.ReadInteger('Network','CountLayers',
DefaultLayerCount);
  // задається кількість нейронів у шарах
  AutoInit := False;
  for i := 0 to xLayerCount - 1 do
    AddLayer(FNnwFile.ReadInteger('Network','Layer_'+IntToStr(i),
DefaultNeuronCount));
  AutoInit := True;
  // ініціалізація нової конфігурації нейромережі
  Init;
```

```
  // завантаження вагових коефіцієнтів і зсуву
  for i:= 1 to LayerCount - 1 do
    for j := 0 to LayersBP[i].NeuronCount - 1 do
    begin
      for k := 0 to LayersBP[ i-1].NeuronCount - 1 do
        LayersBP[i].NeuronsBP[j].Weights[k] := FNnwFile.ReadFloat('Network',
                              'W_'+IntToStr( i-
1)+'_'+IntToStr(k)+'_'+IntToStr(j), 0);
      LayersBP[i].NeuronsBP[j].Weights[LayersBP[ i-1].NeuronCount] :=
FNnwFile.ReadFloat('Network',
       'WT_'+IntToStr( i-1)+'_'+IntToStr(j), 0);
    end;
end;

procedure TNeuralNetExtended.SavePhase1;
begin
  FNnwFile.WriteString('Phase1', 'LearnSampleFileName', FNeuroDataSource.Name);
end;

procedure TNeuralNetExtended.SavePhase2;
var
  i: integer;
begin
  FNnwFile.WriteInteger('Phase2', 'AvailableFieldsCount',
FAvailableFieldsCount);
  for i := 0 to AvailableFieldsCount - 1 do
  with Fields[i] do
  begin
    FNnwFile.WriteString('Phase2', 'FieldName_'+IntToStr(i), Name);
    FNnwFile.WriteInteger('Phase2', 'FieldType_'+IntToStr(i), Kind);
    FNnwFile.WriteInteger('Phase2', 'NormType_'+IntToStr(i), NormType);
    FNnwFile.WriteFloat('Phase2', 'Max_'+IntToStr(i), ValueMax);
    FNnwFile.WriteFloat('Phase2', 'Min_'+IntToStr(i), ValueMin);
    FNnwFile.WriteFloat('Phase2', 'Mid_'+IntToStr(i), ValueMid);
    FNnwFile.WriteFloat('Phase2', 'Disp_'+IntToStr(i), Dispersion);
    FNnwFile.WriteFloat('Phase2', 'Alpha_'+IntToStr(i), Alpha);
    FNnwFile.WriteBool('Phase2', 'Ind_'+IntToStr(i), Ind);
  end;
end;

procedure TNeuralNetExtended.SavePhase4;
begin
  FNnwFile.WriteBool('Phase4', 'Epoch', Epoch);
  FNnwFile.WriteInteger('Phase4', 'Epoch', EpochCount);
  FNnwFile.WriteFloat('Phase4', 'IdentErr', IdentError);
  FNnwFile.WriteBool('Phase4', 'MaxTeachErr', MaxTeachError);
  FNnwFile.WriteFloat('Phase4', 'MaxTeachErr', MaxTeachErrorValue);
  FNnwFile.WriteBool('Phase4', 'MaxTestErr', MaxTestError);
  FNnwFile.WriteFloat('Phase4', 'MaxTestErr', MaxTestErrorValue);
  FNnwFile.WriteBool('Phase4', 'MidTeachErr', MidTeachError);
  FNnwFile.WriteFloat('Phase4', 'MidTeachErr', MidTeachErrorValue);
  FNnwFile.WriteBool('Phase4', 'MidTestErr', MidTestError);
  FNnwFile.WriteFloat('Phase4', 'MidTestErr', MidTestErrorValue);
  FNnwFile.WriteFloat('Phase4', 'Miu', Momentum);
  FNnwFile.WriteBool('Phase4', 'TeachIdent', TeachIdent);
  FNnwFile.WriteFloat('Phase4', 'TeachSpeed', TeachRate);
  FNnwFile.WriteInteger('Phase4', 'TeachIdent', TeachIdentCount);
  FNnwFile.WriteBool('Phase4', 'TestAsValid', TestAsValid);
  FNnwFile.WriteBool('Phase4', 'TestIdent', TestIdent);
  FNnwFile.WriteInteger('Phase4', 'TestIdent', TestIdentCount);
  FNnwFile.WriteInteger('Phase4', 'UseForTeach', UseForTeach);
end;

procedure TNeuralNetExtended.SaveNetwork;
var
  i, j, k: integer;
begin
  FNnwFile.WriteFloat('Network', 'TeachSpeed', TeachRate);
  FNnwFile.WriteFloat('Network', 'Miu', Momentum);
```

```
  FNnwFile.WriteFloat('Network', 'Alpha', Alpha);
  FNnwFile.WriteInteger('Network', 'Epoch', EpochCount);
  FNnwFile.WriteInteger('Network','CountLayers', LayerCount);
  // задається кількість нейронів у шарах
  for i := 0 to LayerCount - 1 do
    FNnwFile.WriteInteger('Network','Layer_'+IntToStr(i),
StrToInt(NeuronsInLayer[i]));
  // завантаження вагових коефіцієнтів і зсуву
  for i:= 1 to LayerCount - 1 do
    for j := 0 to StrToInt(NeuronsInLayer[i]) - 1 do
    begin
      for k := 0 to StrToInt(NeuronsInLayer[ i-1]) do
        FNnwFile.WriteFloat('Network','W_'+IntToStr( i-1)+'_'+IntToStr(k)+
                            '_'+IntToStr(j),
LayersBP[i].NeuronsBP[j].Weights[k]);
      FNnwFile.WriteFloat('Network','WT_'+IntToStr( i-1)+'_'+IntToStr(j),
         LayersBP[i].NeuronsBP[j].Weights[StrToInt(NeuronsInLayer[j])]);
   end;
end;

procedure TNeuralNetExtended.NormalizeData;
var
  i: integer;
begin
  // нормалізація вхідних і вихідних значень
  for i := 0 to FAvailableFieldsCount - 1 do
  begin
    FFields[i].FindMinMax;
    FFields[i].Normalize;
  end;
end;

procedure TNeuralNetExtended.Train;
var
  i, j, k: integer;
begin
  if FUseForTeach = 100 then
  begin
    PatternCount := FFields[0].DataInCount;
    TestSetPatternCount := 0;
  end
  else
  begin
    PatternCount := Round((FFields[0].DataInCount - 1) * FUseForTeach / 100);
    TestSetPatternCount := FFields[0].DataInCount - PatternCount;
  end;
  if not TeachStopped then
    NormalizeData;
  // формування вхідних значень навчальної множини
  RealOutputIndexCount := OutputFieldCount;
  RealInputIndexCount := InputFieldCount;
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdInput then
    begin
      for j := 0 to PatternCount - 1 do
        FPatternsInput[j, k] := FFields[i].DataIn[j];
      // запам'ятовує індекс поля
      RealInputIndex[k] := i;
      Inc(k);
    end;
  // формування вихідних значень навчальної множини
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdOutput then
    begin
      for j := 0 to PatternCount - 1 do
        FPatternsOutput[j, k] := FFields[i].DataIn[j];
      // запам'ятовує індекс поля
```

```
      RealOutputIndex[k] := i;
      Inc(k);
    end;
  // формування вхідних значень тестової множини
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdInput then
    begin
      for j := PatternCount to FFields[i].DataInCount - 1 do
        FTestSetPatterns[j - PatternCount, k] := FFields[i].DataIn[j];
      Inc(k);
    end;
  // формування вихідних значень тестової множини
  k := 0;
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdOutput then
    begin
      for j := PatternCount to FFields[i].DataInCount - 1 do
        FTestSetPatternsOut[j - PatternCount, k] := FFields[i].DataIn[j];
      Inc(k);
    end;
  // навчання або донавчання мережі
  TeachOffLine;
end;

procedure TNeuralNetExtended.SetAvailableFieldsCount(Value : integer);
var
  i: integer;
begin
  FAvailableFieldsCount := Value;
  // встановлюється кількість полів
  SetLength(FFields, Value);
  for i := 0 to FAvailableFieldsCount - 1 do
    FFields[i] := TNeuroField.Create;
end;

procedure TNeuralNetExtended.SetFields(Index: integer; Value: TNeuroField);
begin
  try
    FFields[Index] := Value;
  except
    on E: ERangeError do
      raise E.CreateFmt(SFieldIndexRange, [Index])
  end;
end;

procedure TNeuralNetExtended.SetDefaultProperties;
begin
  // параметри встановлювані за замовчуванням
  Epoch := False;
  IdentError:= DefaultErrorValue;
  MaxTeachError := False;
  MaxTeachErrorValue := DefaultErrorValue;
  MaxTestError:= False;
  MaxTestErrorValue:= DefaultErrorValue;
  MidTestError:= False;
  MidTestErrorValue:= DefaultErrorValue;
  MidTeachError := False;
  MidTeachErrorValue :=  DefaultErrorValue;
  SettingsLoaded := False;
  TeachIdent := False;
  TeachIdentCount:= DefaultTeachIdentCount;
  TestAsValid := False;
  TestIdent:= False;
  TestIdentCount:= DefaultTestIdentCount;
  UseForTeach := DefaultUseForTeach;
end;

procedure TNeuralNetExtended.SetFileName(Value: TFilename);
```

```
begin
  if Assigned(FNnwFile) then
    FNnwFile.Free;
  try
    FNnwFile := TIniFile.Create(Value);
    FFileName := Value;
  except
    on E: EInOutError do
      raise E.CreateFmt(SWrongFileName, [Value]);
  end;
  FNeuroDataSource := TNeuroDataSource.Create;
  LoadPhase1;
  LoadPhase2;
  LoadPhase4;
  LoadNetwork;
  LoadDataFrom;
end;

procedure TNeuralNetExtended.SetTeachIdentCount(const Value: integer);
begin
  if (Value <= 0) or (Value > 100) then
    FTeachIdentCount := DefaultTeachIdentCount
  else
    FTeachIdentCount := Value;
end;

procedure TNeuralNetExtended.SetUseForTeach(const Value: integer);
begin
  if (Value <= 0) or (Value > 100) then
    FUseForTeach := DefaultUseForTeach
  else
    FUseForTeach := Value;
end;

procedure TNeuralNetExtended.SetRealOutputIndex(Index: integer; const Value:
integer);
begin
  FRealOutputIndex[Index] := Value;
end;

procedure TNeuralNetExtended.SetRealOutputIndexCount(const Value: integer);
begin
  SetLength(FRealOutputIndex, Value)
end;
procedure TNeuralNetExtended.SetRealInputIndex(Index: integer; const Value:
integer);
begin
  FRealInputIndex[Index] := Value;
end;
procedure TNeuralNetExtended.SetRealInputIndexCount(const Value: integer);
begin
  SetLength(FRealInputIndex, Value)
end;
procedure Register;
begin
  RegisterComponents('NeuralBase', [TNeuralNetHopf, TNeuralNetBP,
TNeuralNetExtended]);
end;
end.
```

Pumpdata.pas – формування бази знань

```
unit PumpData;

interface

uses
  SysUtils, IniFiles, Classes, NeuralBaseTypes;

type

  EFieldNormError = class(Exception);
  EFieldKindError = class(Exception);

  TNeuroField = class;
  TNeuroFields = array of TNeuroField;

  TNeuroField = class(TObject)
  private
    FAlpha: double;
    FDataIn: TVectorFloat;
    FDispersion: double;
    FInd: boolean;
    FKind: byte;
    FName: string;
    FNormType: byte;
    FValueMax: double;
    FValueMid: double;
    FValueMin: double;
    function GetDataIn(Index: integer): double;
    function GetKindName: TNeuroFieldType;
    function GetNormTypeName: TNormalize;
    function GetDataInCount: integer;
    procedure SetDataIn(Index: integer; Value: double);
    procedure SetKind(Value: byte);
    procedure SetNormType(Value: byte);
    procedure SetDataInCount(Value: integer);
  public
    procedure FindMinMax;
    procedure CalcMid;
    procedure CalcDispersion;
    procedure Normalize;
    procedure DeNormalize;
    property Alpha: double read FAlpha write FAlpha;
    property DataIn[Index: integer]: double read GetDataIn write SetDataIn;
    property DataInCount: integer read GetDataInCount write SetDataInCount;
    property Dispersion: double read FDispersion write FDispersion;
    property Ind: boolean read FInd write FInd;
    property Kind: byte read FKind write SetKind;
    property KindName: TNeuroFieldType read GetKindName;
    property Name: string read FName write FName;
    property NormType: byte read FNormType write SetNormType;
    property NormTypeName: TNormalize read GetNormTypeName;
    property ValueMax: double read FValueMax write FValueMax;
    property ValueMin: double read FValueMin write FValueMin;
    property ValueMid: double read FValueMid write FValueMid;
  end;

  TNeuroDataSource = class(TObject)
  private
    FName: TFileName;
    function IsHeaderChar(AValue: char): boolean;
  public
    function FieldCount(AHeader: string): integer;
    procedure ExtractHeaders(const AFields: TNeuroFields; AHeader: string);
    procedure ExtractValues(const AVector:TVectorFloat; AHeader: string);
    property Name: TFileName read FName write FName;
  end;
```

```
implementation

{ Клас TNeuroField }

function TNeuroField.GetDataIn(Index: integer): double;
begin
  Result := FDataIn[Index];
end;

function TNeuroField.GetDataInCount: integer;
begin
  Result := High(FDataIn) + 1;
end;

function TNeuroField.GetKindName: TNeuroFieldType;
begin
  case FKind of
    0 : Result := fdInput;
    1 : Result := fdOutput;
    2 : Result := fdNone;
  end;
end;

function TNeuroField.GetNormTypeName: TNormalize;
begin
  case FNormType of
    0 : if KindName = fdInput then
          Result := nrmLinear
        else if KindName = fdOutput then
          Result := nrmLinearOut;
    1 : Result := nrmSigmoid;
    2 : Result := nrmAuto;
    3 : Result := nrmNone;
  end;
end;

procedure TNeuroField.CalcMid;
var
  i: integer;
begin
  FValueMid := 0;
  for i := Low(FDataIn) to High(FDataIn) do
    FValueMid := FValueMid + FDataIn[i];
  FValueMid := FValueMid/(High(FDataIn) + 1);
end;

procedure TNeuroField.CalcDispersion;
var
  i: integer;
begin
  if High(FDataIn) > 1 then
  begin
    FDispersion := 0;
    for i := Low(FDataIn) to High(FDataIn) do
      FDispersion := FDispersion + sqr(FDataIn[i] - ValueMid);
    FDispersion := sqrt(FDispersion/High(FDataIn));
  end
  else
    FDispersion := 0;
end;

(*procedure TNeuroField.DeNormalize;
var
  i: integer;
  xTmp: double;
begin
  case NormTypeName of
    nrmLinear: for i := Low(FDataIn) to High(FDataIn) do
```

```
                FDataIn[i] := (FDataIn[i] + 1)*(FValueMax - FValueMin)/2 +
FValueMin;
    nrmLinearOut: for i := Low(FDataIn) to High(FDataIn) do
                FDataIn[i] := FDataIn[i]*(FValueMax - FValueMin) + FValueMin;
    nrmSigmoid: for i := Low(FDataIn) to High(FDataIn) do
                FDataIn[i] := - Ln(1/FDataIn[i] - 1)/Alpha;
  end;
end;*)

procedure TNeuroField.FindMinMax;
var
  i: integer;
begin
  FValueMax:= FDataIn[0];
  FValueMin:= FDataIn[0];
  for i := 1 to High(FDataIn) do
  begin
    if FValueMin > FDataIn[i] then
      FValueMin := FDataIn[i];
    if FValueMax < FDataIn[i] then
      FValueMax := FDataIn[i]
  end;
end;

procedure TNeuroField.Normalize;
var
  i: integer;
  xTmp: double;
begin
  case NormTypeName of
    nrmAuto:  begin
                CalcMid;
                CalcDispersion;
                for i := Low(FDataIn) to High(FDataIn) do
                begin
                  xTmp := (FDataIn[i] - FValueMid)/FDispersion;
                  FDataIn[i] := 1/(1 + exp(-xTmp));
                end;
              end;
    nrmLinear: for i := Low(FDataIn) to High(FDataIn) do
               FDataIn[i] := 2*(FDataIn[i] - FValueMin)/(FValueMax - FValueMin)-
1;
    nrmLinearOut: for i := Low(FDataIn) to High(FDataIn) do
               FDataIn[i] := (FDataIn[i] - FValueMin)/(FValueMax - FValueMin);
    nrmSigmoid: for i := Low(FDataIn) to High(FDataIn) do
               FDataIn[i] := 1/( 1 + exp(-Alpha * FDataIn[i]));
  end;
end;

procedure TNeuroField.SetNormType(Value: byte);
begin
  if (Value < 0) or (Value > 3) then
    raise EFieldNormError.CreateFmt(SFieldNorm, [Value])
  else
    FNormType := Value;
end;

procedure TNeuroField.SetKind(Value: byte);
begin
  if (Value < 0) or (Value > 2) then
    raise Exception.CreateFmt(SFieldKind, [Value])
  else
    FKind := Value;
end;

procedure TNeuroField.SetDataIn(Index: integer; Value: double);
begin
  FDataIn[Index] := Value;
end;
```

```
procedure TNeuroField.SetDataInCount(Value: integer);
begin
  SetLength(FDataIn, Value)
end;

{ Клас TNeuroDataSource }

function TNeuroDataSource.IsHeaderChar(AValue: char): boolean;
begin
  if (AValue in Letters) or (AValue in Capitals) or (AValue in DigitChars) then
    Result := True
  else
    Result := False;
end;

procedure TNeuroDataSource.ExtractValues(const AVector:TVectorFloat; AHeader:
string);
var
  s: string;
  i, xCurPos: integer;
begin
  i := 0;
  AHeader := Trim(AHeader);
  xCurPos := Pos(SpaceChar, AHeader);
  try
    while xCurPos > 0 do
    begin
      s := Copy(AHeader, 1, xCurPos - 1);
      AVector[i] := StrToFloat(s);
      Inc(i);
      Delete(AHeader, 1, xCurPos - 1);
      AHeader := Trim(AHeader);
      xCurPos := Pos(SpaceChar, AHeader);
    end;
    s := AHeader;
    AVector[i] := StrToFloat(s);
  except
    on EConvertError do
      EConvertError.CreateFmt(SCannotBeNumber, [s])
  end;
end;

procedure TNeuroDataSource.ExtractHeaders(const AFields: TNeuroFields; AHeader:
string);
var
  s: string;
  xFieldCount, j, xCurPos: integer;
begin
  { виділяє заголовки з файлу }
  xFieldCount := 0;
  AHeader := Trim(AHeader);
  xCurPos := Pos(SpaceChar, AHeader);
  while xCurPos > 0 do
  begin
    s := Copy(AHeader, 1, xCurPos - 1);
    AFields[xFieldCount].FName := '';
    for j := 1 to Length(s) do
      if isHeaderChar(s[j]) then
        AFields[xFieldCount].FName := AFields[xFieldCount].FName + s[j];
    Inc(xFieldCount);
    Delete(AHeader, 1, xCurPos - 1);
    AHeader := Trim(AHeader);
    xCurPos := Pos(SpaceChar, AHeader);
  end;
  AFields[xFieldCount].FName := '';
  for j := 1 to Length(AHeader) do
    if isHeaderChar(AHeader[j]) then
      AFields[xFieldCount].FName := AFields[xFieldCount].FName + AHeader[j];
```

```
  { повертає кількість полів }
end;

function TNeuroDataSource.FieldCount(AHeader: string): integer;
var
  xFieldCount, xCurPos: integer;
begin
  { виділяє заголовки з файлу }
  xFieldCount := 0;
  AHeader := Trim(AHeader);
  xCurPos := Pos(SpaceChar, AHeader);
  while xCurPos > 0 do
  begin
    Inc(xFieldCount);
    Delete(AHeader, 1, xCurPos - 1);
    AHeader := Trim(AHeader);
    xCurPos := Pos(SpaceChar, AHeader);
  end;
  { повертає кількість полів }
  Result := xFieldCount + 1;
end;


end.
```

NeuralNetExtend.pas – навчання мережі

```
unit NeuralNetExtend;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ComCtrls, NeuralBaseComp, ExtCtrls, StdCtrls, Spin, Grids, NeuralBaseTypes,
  IniFiles;

const
  FormCaption = 'Навчання мережі';

type
  TfrmNeuralNetExtend = class(TForm)
    PageControl: TPageControl;
    pnlNavigation: TPanel;
    Tab1: TTabSheet;
    btnBack: TButton;
    rgrFileType: TRadioGroup;
    btnNext: TButton;
    btnCancel: TButton;
    Tab2: TTabSheet;
    lblFileName: TLabel;
    btnOpenFile: TButton;
    edtFileName: TEdit;
    OpenDialog: TOpenDialog;
    Tab3: TTabSheet;
    ltbFieldName: TListBox;
    Label2: TLabel;
    rdgFieldType: TRadioGroup;
    rdgNormType: TRadioGroup;
    GroupBox1: TGroupBox;
    Label3: TLabel;
    edtMin: TEdit;
    Label4: TLabel;
    edtMax: TEdit;
    Label5: TLabel;
    edt: TEdit;
    Tab4: TTabSheet;
    speLayers: TSpinEdit;
    Label6: TLabel;
    stgNeuronsInLayer: TStringGrid;
    Label7: TLabel;
    Tab5: TTabSheet;
    Label8: TLabel;
    tbrAlpha: TTrackBar;
    sttAlpha: TStaticText;
    Label9: TLabel;
    edtMomentum: TEdit;
    Label10: TLabel;
    edtTeachRate: TEdit;
    Tab6: TTabSheet;
    Label11: TLabel;
    Label12: TLabel;
    btnContinueTeach: TButton;
    sttMaxTeachError: TStaticText;
    sttEpochCount: TStaticText;
    GroupBox2: TGroupBox;
    Label13: TLabel;
    edtIdentError: TEdit;
    speEpochCount: TSpinEdit;
    cbxEpoch: TCheckBox;
    cbxMaxTeachError: TCheckBox;
    edtMaxTeachErrorValue: TEdit;
    cbxMidTeachError: TCheckBox;
```

```
    edtMidTeachErrorValue: TEdit;
    cbxTeachIdent: TCheckBox;
    speTeachIdentValue: TSpinEdit;
    btnBeginTeach: TButton;
    cbxMaxTestError: TCheckBox;
    cbxMidTestError: TCheckBox;
    cbxTestIdent: TCheckBox;
    edtMaxTestErrorValue: TEdit;
    edtMidTestErrorValue: TEdit;
    speTestIdentValue: TSpinEdit;
    Tab7: TTabSheet;
    Label14: TLabel;
    stgInput: TStringGrid;
    Label15: TLabel;
    stgOutput: TStringGrid;
    btnCompute: TButton;
    Memo1: TMemo;
    Label16: TLabel;
    Label17: TLabel;
    Memo2: TMemo;
    Bevel1: TBevel;
    Memo3: TMemo;
    Label1: TLabel;
    sttMaxTestError: TStaticText;
    Label18: TLabel;
    sttMidTeachError: TStaticText;
    Label19: TLabel;
    sttMidTestError: TStaticText;
    SaveDialog: TSaveDialog;
    edtUseForTeach: TEdit;
    Label20: TLabel;
    btnSave: TButton;
    procedure btnCancelClick(Sender: TObject);
    procedure btnNextClick(Sender: TObject);
    procedure btnBackClick(Sender: TObject);
    procedure btnOpenFileClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure ltbFieldNameClick(Sender: TObject);
    procedure rdgFieldTypeClick(Sender: TObject);
    procedure rdgNormTypeClick(Sender: TObject);
    procedure edtAChange(Sender: TObject);
    procedure tbrAlphaChange(Sender: TObject);
    procedure edtMomentumChange(Sender: TObject);
    procedure edtTeachRateChange(Sender: TObject);
    procedure NeuralNetExtendedEpochPassed(Sender: TObject);
    procedure btnContinueTeachClick(Sender: TObject);
    procedure edtIdentErrorChange(Sender: TObject);
    procedure cbxEpochClick(Sender: TObject);
    procedure speEpochCountChange(Sender: TObject);
    procedure cbxMaxTeachErrorClick(Sender: TObject);
    procedure edtMaxTeachErrorValueChange(Sender: TObject);
    procedure cbxMidTeachErrorClick(Sender: TObject);
    procedure edtMidTeachErrorValueChange(Sender: TObject);
    procedure cbxTeachIdentClick(Sender: TObject);
    procedure speTeachIdentValueChange(Sender: TObject);
    procedure cbxMaxTestErrorClick(Sender: TObject);
    procedure edtMaxTestErrorValueChange(Sender: TObject);
    procedure cbxMidTestErrorClick(Sender: TObject);
    procedure edtMidTestErrorValueChange(Sender: TObject);
    procedure cbxTestIdentClick(Sender: TObject);
    procedure speTestIdentValueChange(Sender: TObject);
    procedure NeuralNetExtendedAfterTeach(Sender: TObject);
    procedure btnBeginTeachClick(Sender: TObject);
    procedure btnComputeClick(Sender: TObject);
    procedure speLayersChange(Sender: TObject);
    procedure btnSaveClick(Sender: TObject);
    procedure edtUseForTeachChange(Sender: TObject);
  private
    { Private declarations }
```

```
    Teach: boolean;
    NotSaved: boolean;
    procedure ChangePage;
    procedure OpenFile;
    procedure LoadFile;
    procedure Tune;
    procedure CreateNet;
    procedure RunTeach;
    procedure TestNet;
  public
    { Public declarations }
  end;

var
  frmNeuralNetExtend: TfrmNeuralNetExtend;
implementation

{$R *.DFM}

// Запуск програми
procedure TfrmNeuralNetExtend.FormActivate(Sender: TObject);
begin
  Caption := FormCaption;
  PageControl.ActivePage := PageControl.Pages[0];
  Teach := false;
  NotSaved := false;
end;

// Вихід із програми
procedure TfrmNeuralNetExtend.btnCancelClick(Sender: TObject);
begin
  if NotSaved then
    if MessageDlg('Є незбережені дані. Зберегти?',mtConfirmation, [mbYes, mbNo],
0) = mrYes then
      btnSave.Click;
  Close;
end;

// Натискання кнопки "Вперед"
procedure TfrmNeuralNetExtend.btnNextClick(Sender: TObject);
begin
  with PageControl do
  begin
    ActivePage := FindNextPage(ActivePage, true, false);
    ChangePage;
    if ActivePage.PageIndex = PageCount - 1 then
      btnNext.Enabled := false
    else
      btnNext.Enabled := true;
    btnBack.Enabled := true;
  end;
end;

// Натискання кнопки "Назад"
procedure TfrmNeuralNetExtend.btnBackClick(Sender: TObject);
begin
  with PageControl do
  begin
    ActivePage := FindNextPage(ActivePage, false, false);
    if ActivePage.PageIndex = 0 then
      btnBack.Enabled := false
    else
      btnBack.Enabled := true;
    btnNext.Enabled := true;
  end;
end;

// Дія на зміну сторінки
procedure TfrmNeuralNetExtend.ChangePage;
```

```
begin
  case PageControl.ActivePage.PageIndex of
    1: OpenFile;
    2: LoadFile;
    3: Tune;
    4: CreateNet;
    6: TestNet;
  end;
end;

// Вибрати файл - джерело даних
procedure TfrmNeuralNetExtend.OpenFile;
begin
  if rgrFileType.ItemIndex = 0 then
  begin
    OpenDialog.Filter := 'NNW files (*.nnw)|*.nnw';
    lblFileName.Caption := 'Виберіть nnw-файл';
  end
  else
  begin
    OpenDialog.Filter := 'Text files (*.txt)|*.txt';
    lblFileName.Caption := 'Виберіть txt-файл';
  end;
end;

// Вибір файлу в діалоговому вікні
procedure TfrmNeuralNetExtend.btnOpenFileClick(Sender: TObject);
begin
  OpenDialog.Execute;
  Caption := FormCaption + ' - ' + ExtractFileName(OpenDialog.FileName);
  edtFileName.Text := OpenDialog.FileName;
end;

// Завантажити в компонент обраний файл
procedure TfrmNeuralNetExtend.LoadFile;
var
  i: integer;
begin
  try
    if rgrFileType.ItemIndex = 0 then
      NeuralNetExtended.FileName := edtFileName.Text   // nnw-файл
    else
    begin
      NeuralNetExtended.SourceFileName := edtFileName.Text; // текстовий файл
      NeuralNetExtended.LoadDataFrom; // завантажує дані з текстового файлу
      // конфігурація нейронної мережі за замовчуванням
      NeuralNetExtended.AddLayer(2);
      NeuralNetExtended.AddLayer(3);
      NeuralNetExtended.AddLayer(1);
    end;
  except
    raise Exception.Create('Помилка при відкритті файлу');
  end;
  NeuralNetExtended.Init;  // Ініціалізація мережі
  // Формування списку полів для StringList-A
  ltbFieldName.Clear;
  for i := 0 to NeuralNetExtended.AvailableFieldsCount - 1 do
    ltbFieldName.Items.Add(NeuralNetExtended.Fields[i].Name);
  ltbFieldName.ItemIndex := 0;
  ltbFieldNameClick(Self);
end;

// Настроювання параметрів мережі
procedure TfrmNeuralNetExtend.Tune;
var
  i: integer;
begin
  speLayers.Value := NeuralNetExtended.LayerCount - 2;
  stgNeuronsInLayer.RowCount := speLayers.Value + 1;
```

```
  stgNeuronsInLayer.Cells[0, 0] := '№ шаруючи';
  stgNeuronsInLayer.Cells[1, 0] := 'Нейронів';
  for i := 0 to speLayers.Value - 1 do
  begin
    stgNeuronsInLayer.Cells[0, i + 1] := IntToStr(i);
    stgNeuronsInLayer.Cells[1, i + 1] := IntToStr(NeuralNetExtended.Layers[i +
1].NeuronCount);
  end;

  tbrAlpha.Position := trunc(NeuralNetExtended.Alpha * 100);
  edtMomentum.Text := FloatToStr(NeuralNetExtended.Momentum);
  edtTeachRate.Text := FloatToStr(NeuralNetExtended.TeachRate);
  edtIdentError.Text := FloatToStr(NeuralNetExtended.IdentError);
  edtUseForTeach.Text := FloatToStr(NeuralNetExtended.UseForTeach);

  cbxEpoch.Checked := NeuralNetExtended.Epoch;
  speEpochCount.Text := IntToStr(NeuralNetExtended.EpochCount);

  cbxMaxTeachError.Checked := NeuralNetExtended.MaxTeachError;
  edtMaxTeachErrorValue.Text :=
FloatToStr(NeuralNetExtended.MaxTeachErrorValue);

  cbxMidTeachError.Checked := NeuralNetExtended.MidTeachError;
  edtMidTeachErrorValue.Text :=
FloatToStr(NeuralNetExtended.MidTeachErrorValue);

  cbxTeachIdent.Checked := NeuralNetExtended.TeachIdent;
  speTeachIdentValue.Value := NeuralNetExtended.TeachIdentCount;

  cbxMaxTestError.Checked := NeuralNetExtended.MaxTestError;
  edtMaxTestErrorValue.Text := FloatToStr(NeuralNetExtended.MaxTestErrorValue);

  cbxMidTestError.Checked := NeuralNetExtended.MidTestError;
  edtMidTestErrorValue.Text := FloatToStr(NeuralNetExtended.MidTestErrorValue);

  cbxTestIdent.Checked := NeuralNetExtended.TestIdent;
  speTestIdentValue.Value := NeuralNetExtended.TeachIdentCount;
end;

// Відображення інформації про обране поле (тип поля, нормалізація та інше)
procedure TfrmNeuralNetExtend.ltbFieldNameClick(Sender: TObject);
begin
  // Тип поля - вхідне, вихідне, не використовувати
  rdgFieldType.ItemIndex :=
NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Kind;
  // Тип нормалізації
  rdgNormType.ItemIndex :=
NeuralNetExtended.Fields[ltbFieldName.ItemIndex].NormType;
  // Параметр нормалізації
  edt.Text :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Alpha);
  // Мінінум та максимум (для лінійної нормалізації)
  edtMin.Text :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].ValueMin);
  edtMax.Text :=
FloatToStr(NeuralNetExtended.Fields[ltbFieldName.ItemIndex].ValueMax);
end;

// Змінити тип поля
procedure TfrmNeuralNetExtend.rdgFieldTypeClick(Sender: TObject);
begin
  NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Kind :=
rdgFieldType.ItemIndex
end;

// Змінити тип нормалізації
procedure TfrmNeuralNetExtend.rdgNormTypeClick(Sender: TObject);
begin
```

```
  NeuralNetExtended.Fields[ltbFieldName.ItemIndex].NormType :=
rdgNormType.ItemIndex
end;

// Змінити параметр нормалізації
procedure TfrmNeuralNetExtend.edtAChange(Sender: TObject);
begin
 NeuralNetExtended.Fields[ltbFieldName.ItemIndex].Alpha := StrToFloat(edt.Text);
end;

// Змінити параметр Alpha мережі
procedure TfrmNeuralNetExtend.tbrAlphaChange(Sender: TObject);
begin
  sttAlpha.Caption := FloatToStr(tbrAlpha.Position / 100);
end;

// Зміна кількості схованих шарів
procedure TfrmNeuralNetExtend.speLayersChange(Sender: TObject);
begin
  stgNeuronsInLayer.RowCount := speLayers.Value + 1;
end;

// Створення мережі обраної топології
procedure TfrmNeuralNetExtend.CreateNet;
var
  i: integer;
  xInput, xOutput: integer;
begin
  // Змінюється тільки кількість нейронів у схованих шарах,
  // Кількість нейронів у вхідному й вихідному шарі залежить від
  // типів полів
  with NeuralNetExtended do
  begin
    xInput := InputFieldCount;
    xOutput := OutputFieldCount;
    ResetLayers;
    AddLayer(xInput);
    for i := 0 to speLayers.Value - 1 do
      AddLayer(StrToInt(stgNeuronsInLayer.Cells[1, i + 1]));
    AddLayer(xOutput);
  end;
end;

// Змінити момент мережі
procedure TfrmNeuralNetExtend.edtMomentumChange(Sender: TObject);
begin
  NeuralNetExtended.Momentum := StrToFloat(edtMomentum.Text);
end;

// Змінити швидкість навчання мережі
procedure TfrmNeuralNetExtend.edtTeachRateChange(Sender: TObject);
begin
  NeuralNetExtended.TeachRate := StrToFloat(edtTeachRate.Text);
end;

// Дія по проходженню однієї епохи навчання
procedure TfrmNeuralNetExtend.NeuralNetExtendedEpochPassed(Sender: TObject);
begin
  sttMaxTestError.Caption := '';
  sttMidTestError.Caption := '';
  with NeuralNetExtended do
  begin
    sttMaxTeachError.Caption := FloatToStr(MaxTeachResidual); // Показати макс.
помилку на навчальній множині
    sttMidTeachError.Caption := FloatToStr(MidTeachResidual); // Показати серед.
помилку на навчальній множині
    if NeuralNetExtended.UseForTeach <> 100 then
    begin
```

```
      sttMaxTestError.Caption := FloatToStr(MaxTestResidual); // Показати макс.
помилку на тестовій множині
      sttMidTestError.Caption := FloatToStr(MidTestResidual); // Показати сред.
помилку на тестовій множині
    end;
    sttEpochCount.Caption := FloatToStr(EpochCurrent); // Показати номер
поточної епохи
  end;
  Application.ProcessMessages; // Дати можливість Windows перемалювати форму
end;

// Натискання кнопки "Продовжити навчання"
procedure TfrmNeuralNetExtend.btnContinueTeachClick(Sender: TObject);
begin
  NeuralNetExtended.ContinueTeach := true; // Скинути прапор - "Продовжити
навчання"
  RunTeach;  // Запуск
end;

// Натискання кнопки "Навчити"
procedure TfrmNeuralNetExtend.btnBeginTeachClick(Sender: TObject);
begin
  NeuralNetExtended.ContinueTeach := false; // Скинути прапор - "Почати навчання
знову"
  RunTeach;
end;

procedure TfrmNeuralNetExtend.edtIdentErrorChange(Sender: TObject);
begin
  NeuralNetExtended.IdentError := StrToFloat(edtIdentError.Text);
end;

procedure TfrmNeuralNetExtend.edtUseForTeachChange(Sender: TObject);
begin
  NeuralNetExtended.UseForTeach := StrToInt(edtUseForTeach.Text);
end;

procedure TfrmNeuralNetExtend.cbxEpochClick(Sender: TObject);
begin
  NeuralNetExtended.Epoch := cbxEpoch.Checked;
end;

procedure TfrmNeuralNetExtend.speEpochCountChange(Sender: TObject);
begin
  NeuralNetExtended.EpochCount := StrToInt(speEpochCount.Text);
end;

procedure TfrmNeuralNetExtend.cbxMaxTeachErrorClick(Sender: TObject);
begin
  NeuralNetExtended.MaxTeachError := cbxMaxTeachError.Checked;
end;

procedure TfrmNeuralNetExtend.edtMaxTeachErrorValueChange(Sender: TObject);
begin
  NeuralNetExtended.MaxTeachErrorValue :=
StrToFloat(edtMaxTeachErrorValue.Text);
end;

procedure TfrmNeuralNetExtend.cbxMidTeachErrorClick(Sender: TObject);
begin
  NeuralNetExtended.MidTeachError := cbxMidTeachError.Checked;
end;

procedure TfrmNeuralNetExtend.edtMidTeachErrorValueChange(Sender: TObject);
begin
  NeuralNetExtended.MidTeachErrorValue :=
StrToFloat(edtMidTeachErrorValue.Text);
end;
```

```
procedure TfrmNeuralNetExtend.cbxTeachIdentClick(Sender: TObject);
begin
  NeuralNetExtended.TeachIdent := cbxTeachIdent.Checked;
end;

procedure TfrmNeuralNetExtend.speTeachIdentValueChange(Sender: TObject);
begin
  NeuralNetExtended.TeachIdentCount := speTeachIdentValue.Value;
end;

procedure TfrmNeuralNetExtend.cbxMaxTestErrorClick(Sender: TObject);
begin
  NeuralNetExtended.MaxTestError := cbxMaxTestError.Checked;
end;

procedure TfrmNeuralNetExtend.edtMaxTestErrorValueChange(Sender: TObject);
begin
  NeuralNetExtended.MaxTestErrorValue := StrToFloat(edtMaxTestErrorValue.Text);
end;

procedure TfrmNeuralNetExtend.cbxMidTestErrorClick(Sender: TObject);
begin
  NeuralNetExtended.MidTestError := cbxMidTestError.Checked;
end;

procedure TfrmNeuralNetExtend.edtMidTestErrorValueChange(Sender: TObject);
begin
  NeuralNetExtended.MidTestErrorValue := StrToFloat(edtMidTestErrorValue.Text);
end;

procedure TfrmNeuralNetExtend.cbxTestIdentClick(Sender: TObject);
begin
  NeuralNetExtended.TestIdent := cbxTestIdent.Checked;
end;

procedure TfrmNeuralNetExtend.speTestIdentValueChange(Sender: TObject);
begin
  NeuralNetExtended.TeachIdentCount := speTestIdentValue.Value;
end;

// Зупинка навчання
procedure TfrmNeuralNetExtend.NeuralNetExtendedAfterTeach(Sender: TObject);
begin
  btnBack.Enabled := true;
  btnNext.Enabled := true;
  btnCancel.Enabled := true;
  btnContinueTeach.Caption := 'Навчити';
  NeuralNetExtended.StopTeach := true;
end;

procedure TfrmNeuralNetExtend.RunTeach;
begin
  Teach := not Teach;  // Перемикач стану "вчимося/не вчимося"
  if Teach then
  begin
    btnBeginTeach.Enabled := false;
    btnBack.Enabled := false;
    btnNext.Enabled := false;
    btnCancel.Enabled := false;
    NeuralNetExtended.StopTeach := false;
    btnContinueTeach.Caption := 'Зупинити навчання';
    NotSaved := true;
    NeuralNetExtended.Train;  // Запуск нейромережі на навчання
    btnCompute.Enabled := true;
    btnSave.Enabled := true;
  end
  else
  begin
    btnBeginTeach.Enabled := true;
```

```
    btnBack.Enabled := true;
    btnNext.Enabled := true;
    btnCancel.Enabled := true;
    btnContinueTeach.Caption := 'Продовжити навчання';
    NeuralNetExtended.StopTeach := true; // Зупинити навчання
  end;
end;

// Відкриття сторінки - тестування навченої нейромережі
procedure TfrmNeuralNetExtend.TestNet;
var
  i, j: integer;
begin
  stgInput.RowCount := NeuralNetExtended.InputFieldCount + 1;
  stgInput.Cells[0, 0] := 'Поле';
  stgInput.Cells[1, 0] := 'Значення';

  // Проставити імена вхідних полів
  j := 0;
  for i := 0 to NeuralNetExtended.AvailableFieldsCount - 1 do
    if (NeuralNetExtended.Fields[i].KindName = fdInput) then // Ознака того, що
поле вхідне
    begin
      Inc(j);
      stgInput.Cells[0, j] := NeuralNetExtended.Fields[i].Name;
    end;
  stgOutput.RowCount := NeuralNetExtended.OutputFieldCount + 1;
  stgOutput.Cells[0, 0] := 'Поле';
  stgOutput.Cells[1, 0] := 'Значення';

  // Проставити імена вихідних полів
  j := 0;
  for i := 0 to NeuralNetExtended.AvailableFieldsCount - 1 do
    if (NeuralNetExtended.Fields[i].KindName = fdOutput) then  // Ознака того,
що поле вихідне
    begin
      Inc(j);
      stgOutput.Cells[0, j] := NeuralNetExtended.Fields[i].Name;
    end;
end;

// Натискання кнопки "Обчислити"
procedure TfrmNeuralNetExtend.btnComputeClick(Sender: TObject);
var
  xVectorFloat: TVectorFloat;
  i: integer;
begin
  // Створити вектор, що будемо подавати на вхід
  // довжиною, рівною кількості нейронів на вхідному шарі
  SetLength(xVectorFloat, NeuralNetExtended.InputFieldCount);
  // Заповнити значення елементів вектора
  for i := 0 to NeuralNetExtended.InputFieldCount - 1 do
    xVectorFloat[i] := StrToFloat(stgInput.Cells[1, i + 1]);
  // Подати на вхід нейромережі. Результати будуть у вихідному шарі нейромережі
  NeuralNetExtended.ComputeUnPrepData(xVectorFloat);
  // Відобразити отримані результати
  for i := 0 to NeuralNetExtended.OutputFieldCount - 1 do
    stgOutput.Cells[1, i + 1] := FloatToStr(NeuralNetExtended.Output[i]);
  // Знищити вектор
  SetLength(xVectorFloat, 0);
  xVectorFloat := nil;
end;

// Зберегти навчену нейромережу
procedure TfrmNeuralNetExtend.btnSaveClick(Sender: TObject);
begin
  SaveDialog.InitialDir := ExtractFilePath(NeuralNetExtended.FileName);
  SaveDialog.FileName := ExtractFileName(NeuralNetExtended.FileName);
  if SaveDialog.Execute then
```

```
  begin
    NeuralNetExtended.NnwFile := TIniFile.Create(SaveDialog.FileName);
    NeuralNetExtended.SavePhase1;
    NeuralNetExtended.SavePhase2;
    NeuralNetExtended.SavePhase4;
    NeuralNetExtended.SaveNetwork;
    NotSaved := false;
  end;
end;

end.
```

Hopf.pas - Блок формування та розпізнавання образів

```
unit Hopf;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Grids, NeuralBaseComp, NeuralBaseTypes, Db, DBTables, ExtCtrls, DBCtrls,
StdCtrls,
  ToolWin, ComCtrls;

type
  TForm1 = class(TForm)
    Table: TTable;
    btnExecute: TButton;
    DBNavigator: TDBNavigator;
    DataSource: TDataSource;
    btnEdit: TButton;
    stgDatabase: TStringGrid;
    stgInput: TStringGrid;
    stgOutput: TStringGrid;
    NeuralNetHopf: TNeuralNetHopf;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    Bevel: TBevel;
    TableLETTERS: TStringField;
    dbMain: TDatabase;
    procedure DataSourceDataChange(Sender: TObject; Field: TField);
    procedure FormActivate(Sender: TObject);
    procedure GridClick(Sender: TObject);
    procedure btnEditClick(Sender: TObject);
    procedure btnExecuteClick(Sender: TObject);
    procedure GridDrawCell(Sender: TObject; ACol, ARow: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure FormCreate(Sender: TObject);
  public
    { Public declarations }
    procedure AddPattern(Value: string);
    procedure Clear(Grid: TStringGrid);
    procedure Init;
    procedure ShowMatrix(Grid: TStringGrid; Value: string);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

// Показати вектор у вигляді сітки
procedure TForm1.ShowMatrix(Grid: TStringGrid; Value: string);
var
  i, j: integer;
begin
  Clear(Grid);
  for i := 0 to Grid.ColCount - 1 do
    for j := 0 to Grid.RowCount - 1 do
    begin
      try
        if Value[i * Grid.RowCount + j + 1] = '1' then
          Grid.Cells[i, j] := '1'
        else
          Grid.Cells[i, j] := ' '
      except
        Grid.Cells[i, j] := ' '
```

```
      end;
    end;
end;

// Очистити сітку
procedure TForm1.Clear(Grid: TStringGrid);
var
  i, j: integer;
begin
  for i := 0 to Grid.ColCount - 1 do
    for j := 0 to Grid.RowCount - 1 do
      Grid.Cells[i, j] := ' ';
end;

// Показати символ з таблиці
procedure TForm1.DataSourceDataChange(Sender: TObject; Field: TField);
begin
  ShowMatrix(stgDatabase, TableLETTERS.Value);
end;

// Ініціалізація мережі значеннями з таблиці
procedure TForm1.Init;
begin
  // Очистити мережу від зразків
  NeuralNetHopf.ResetPatterns;

  // Додати зразки з таблиці до мережі
  Table.First;
  while not Table.Eof do
  begin
    AddPattern(TableLETTERS.AsString);
    Table.Next;
  end;
  Table.First;

  // Ініціалізувати ваги
  NeuralNetHopf.InitWeights;
  // Мережа підготовлена до розпізнавання
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
  Clear(stgDatabase);
  Clear(stgInput);
  Clear(stgOutput);
  Init;
end;

procedure TForm1.GridClick(Sender: TObject);
begin
  with Sender as TStringGrid do
    if Cells[Col, Row] = '1' then
      Cells[Col, Row] := ' '
    else
      Cells[Col, Row] := '1'
end;

// Додавання нового образа до мережі
procedure TForm1.AddPattern(Value: string);
var
  i: integer;
  xVector: TVectorInt;
begin
  SetLength(xVector, stgDatabase.RowCount * stgDatabase.ColCount);

  // Перетворення символьного рядка у вектор
  for i := 1 to stgDatabase.RowCount * stgDatabase.ColCount do
    try
      if TableLETTERS.AsString[i] = '1' then
```

```
        xVector[i - 1] := 1
      else
        xVector[i - 1] := -1;
    except
      xVector[i - 1] := -1;
    end;


  NeuralNetHopf.AddPattern(xVector);
end;

procedure TForm1.btnEditClick(Sender: TObject);
var
  i, j: integer;
  xString: string;
begin
  xString := '';
  for i := 0 to stgDatabase.ColCount - 1 do
    for j := 0 to stgDatabase.RowCount - 1 do
      if stgDatabase.Cells[i, j] = '1' then
        xString := xString + '1'
      else
        xString := xString + ' ';
  Table.Edit;
  TableLETTERS.AsString := xString;
  Table.Post;
end;

// Розпізнавання символу
procedure TForm1.btnExecuteClick(Sender: TObject);
var
  i, j: integer;
  xString: string;
begin
  // Подаємо сигнали на вихід мережі
  for i := 0 to stgInput.ColCount - 1 do
    for j := 0 to stgInput.RowCount - 1 do
      if stgInput.Cells[i, j] = '1' then
        NeuralNetHopf.Layers[1].Neurons[i * stgInput.RowCount + j].Output := 1
      else
        NeuralNetHopf.Layers[1].Neurons[i * stgInput.RowCount + j].Output := -1;

  // Запуск процесу розпізнавання
  NeuralNetHopf.Calc;

  // Перетворення виходів мережі до рядка
  xString := '';
  for i := 1 to stgOutput.RowCount * stgOutput.ColCount do
    if NeuralNetHopf.Layers[1].Neurons[i - 1].Output = 1 then
      xString := xString + '1'
    else
      xString := xString + ' ';

  // Відобразити результат
  ShowMatrix(stgOutput, xString);
end;

procedure TForm1.GridDrawCell(Sender: TObject; ACol, ARow: Integer;
  Rect: TRect; State: TGridDrawState);
begin
  with Sender as TStringGrid do
  begin
    Canvas.Brush.Color := clBlack;
    if Cells[ACol,ARow] <> ' ' then
      Canvas.FillRect(Rect)
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
  dbMain.Params.Values['Path'] := ExtractFilePath(Application.ExeName);
  dbMain.Open;
  Table.Open;

end;

end.
```