МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Дослідження та проектування інформаційної системи математичного моделювання динамічних процесів

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 122 "Комп'ютерні науки та інформаційні технології" (освітня програма - "Інформаційні технології проектування")

Науковий керівник роботи:	_____ (підпис)	С.О.Сафонова (ініціали, прізвище)
Консультант з охорони праці:	_____ (підпис)	Я.О.Критська (ініціали, прізвище)
Студент:	_____ (підпис)	Г.І.Асєєва (ініціали, прізвище)
Група:		ІТП-16дм

Сєвєродонецьк 2018

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень магістр
Напрям підготовки _____
(шифр і назва)
Спеціальність 122 "Комп'ютерні науки та інформаційні технології" (освітня програма-
(шифр і назва)
"Інформаційні технології проектування")

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Асєєвій Ганні Ігорівні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та проектування інформаційної системи математичного моделювання динамічних процесів

керівник проекту (роботи) Сафонова Світлана Олександрівна, к.т.н., доц.
(прізвище, м.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «18» 10 2018 р. № 208/48
2. Строк подання студентом роботи 21.01.2018
3. Вихідні дані до роботи Матеріали науково-дослідної практики, засоби інтерактивного моделювання складних хіміко- технологічних процесів та графічного інтерфейсу користувача, модель теплообмінника,

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Організація моделювання динамічних процесів, аналіз та вибір програмно-технічних засобів для реалізації системи моделювання, розробка програмного забезпечення моделей, тестування системи та аналіз отриманих результатів, охорона праці та безпека в надзвичайних ситуаціях, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та безпека в надзвичайних ситуаціях	Критська Я.О. ст. викл. кафедри КНІ		

7. Дата видачі завдання 18.10.2017

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Розробка технічного завдання	10.09.2017-15.09.2017	
2	Розробка алгоритмів взаємодії компонентів системи	16.09.2017-22.09.2017	
3	Розробка інтерфейсу користувача системи	23.09.2017-25.09.2017	
4	Розробка програми	26.09.2017-06.10.2017	
5	Дослідження працездатності системи	07.10.2017-25.11.2017	
6	Розробка частини проекту "Охорона праці та безпеки в надзвичайних ситуаціях"	26.11.2017-1.12.2017	
7	Оформлення пояснювальної записки, автореферату та презентації	2.12.2017-13.01.2018	

Студент _____ Асєєва Г.І.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Сафонова С.О.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Асєєва Г.І. Дослідження та проектування інформаційної системи математичного моделювання динамічних процесів.

Магістерська робота присвячена проектуванню симуляційної системи з графічним користувальницьким інтерфейсом для дослідження технологічних процесів, зокрема для дослідження впливу моделі складності на швидкість моделювання. Проведено тестування розробленого модуля теплообмінника. Виявлено та означено середні швидкості виконання моделювання системи, яка складається с декількох модулей. Визначено вплив часу перемальовування графічного інтерфейсу користувача на загальний час моделювання.

Ключові слова: модель, процес, графічний інтерфейс, програма, емулятор.

THE ABSTRACT

Aseeva H.I. Research and design of information system of mathematical modeling of dynamic processes.

The master's work is devoted to the design of a simulation system with a graphical user interface for the study of technological processes, in particular, to study the influence of the complexity model on the speed of modeling. Testing of the developed module of heat exchanger has been carried out. The average simulation speed of a system, which consists of several modules, is revealed and indicated. The influence of the graphing time of the graphical user interface on the total simulation time is determined.

Keywords: model, process, graphical interface, program, emulator.

АННОТАЦИЯ

Асеева И. Исследование и проектирование информационной системы математического моделирования динамических процессов.

Магистерская работа посвящена проектированию симуляционной системы с графическим пользовательским интерфейсом для исследования технологических процессов, в частности для исследования влияния модели сложности на скорость моделирования. Разработка системы базируется на технологии Проведено тестирование разработанного модуля теплообменника. Выявлены и отмечены средние скорости выполнения моделирования системы, состоящей с нескольких модулей. Определено влияние времени перерисовки графического интерфейса пользователя на общее время моделирования.

Ключевые слова: модель, процесс, графический интерфейс, программа, эмулятор.

ЗМІСТ

ВСТУП

Моделювання систем складних технологічних процесів і установок відіграє важливу роль при проектуванні й розробці АСУТП, при роботі систем спостереження за технологічним процесом і прогнозуванні зміни параметрів технологічного процесу з метою запобігання небезпечних ситуацій на виробництві. Моделювання використовуються на етапах проектування комп'ютерних тренажерів, предназначених для підготовки персоналу технологічного обладнення небезпечних виробництвах та діях у позаштатних ситуаціях.

Моделювання – це метод дослідження складних систем, заснований на тому, що розглянута система заміняється на модель та проводиться дослідження моделі з метою одержання інформації про досліджувану систему. Під моделлю системи, розуміється деяка інша система, яка поводиться з погляду цілей дослідження аналогічно поведінці системи. Звичайно модель простіше та доступніше для дослідження та аналізу у порівнянні з реальною технологічною системою.

Моделювання застосовується при розробці комп'ютерних тренажерів. Необхідність їх застосування обумовлена великою складністю обладнання. В технологічних циклах промислових хімічних виробництв у великій кількості використовуються небезпечні хімічні речовини. Ступінь небезпеки дуже різноманітна, але в кожному разі помилки людини, що контролює технологічні процеси на виробництві, можуть обійтися дуже дорого. Принцип, на якому заснована більшість комп'ютерних тренажерів - моделювання реальності. Це означає створення деякої подоби реального обладнання, яке при впливі на нього поводиться так само як відповідне технологічне обладнання. Від ступеня подібності моделі свому реальному прототипу, та подібності у її поведінкі до реальностям, напяму залежить якість тренажеру. Людина практикується в операціях, що максимально відповідають реальним, маючи справу всього лише з їх електронним аналогом.

Під час роботи перерахованих вище систем виникає необхідність емуляції певних процесів. Існують різні системи моделювання: чисельного моделювання, імітаційного моделювання, системи, засновані на роботі SCADA. Але існуючі системі важкі, складні у використанні, недостатньо досконалі, мають вузьку область застосування та недостатньо універсальні, тобто придатні для розв'язку тільки вузького кола завдань. Для обслуговування таких систем потрібна постійна присутність фахівця, бо звичайний користувач технолог не в змозі самостійно обслуговувати подібні системи. Тому необхідно розробити таке середовище моделювання, яке є одночасно зручним (з погляду

користувача), дешевим, не надмірно складним, та універсальним – яке підходить для вирішення широкого кола завдань.

Метою магістерської роботи є поліпшення етапів процесу моделювання технологічних процесів та об’єктів за рахунок розробки гнучкої, розширюваної системи моделювання – середовища конструювання моделей, що дозволяє будувати складні моделі технологічних процесів і установок у простому й зручному виді, досліджувати характеристики цих моделей і обробляти отримані результати. Система моделювання не повинна мати підвищену складність, припускати можливість обслуговування користувачем, що не є програмістом і мати зручний інтерфейс користувача.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати стан розвитку систем моделювання технологічних процесів та об’єктів;

– спроектувати систему моделювання, котра поєднає засоби математичного та комп’ютерного моделювання

– система моделювання повинна автоматично виявляти та підключати модулі;

– система моделювання повинна мати засоби задавати зв'язки між вхідними й вихідними параметрами модулів та виконувати основну логіку з моделювання процесів.

Об'єкт дослідження - система математичного моделювання та компонентні моделі технологічних установок..

Предмет дослідження – методи та засоби проектування систем моделювання.

Практична значимість роботи - розроблена симуляційна система з простою графічним користувальницьким інтерфейсом для вивчення та моделювання технологічних процесів.

Основні результати магістерської роботи **доповідались** на Міжнародній науково-практичній конференції «Майбутній науковець – 2017», та на Всеукраїнській науково-практичній конференції «Електронні апарати та системи. Проблеми створення. Перспективи розвитку».

Магістерська робота **складається** зі вступу, 5 розділів, висновків, переліку джерел посилань, додатку. Загальний обсяг роботи становить 88 сторінок, 4 таблиці, 14 рисунок.

1 ОРГАНІЗАЦІЯ МОДЕЛЮВАННЯ ДИНАМІЧНИХ ПРОЦЕСІВ

Розглягялнемо існуючі системи моделювання, виконується їхній аналіз, визначаються їхні гідності та недоліки, області застосування. Далі розглядаються відомі методи та засоби моделювання. На підставі результатів аналізу робляться висновки та визначається завдання на магістерську роботу.

1.1 SCADA яка засоби моделювання технологічних динамічних процесів

В даний час на ринку існує безліч систем SCADA, але але мова піде про більш популярних. Перша така система - це SIMATIC WinCC.

1.1.1 SIMATIC WinCC

SCADA система SIMATIC WinCC (Windows Control Center) - це комп'ютерна система людино-машинного інтерфейсу, що працює під управлінням операційних систем Windows 2000/XP і надає широкі функціональні можливості для побудови систем керування різного призначення:

- просте побудова конфігурацій клієнт-сервер;
- підтримка резервованих структур систем автоматизації;
- необмежене розширення функціональних можливостей завдяки використання елементів ActiveX;
- відкритий OPC-інтерфейс (OLE for Process Control) інтерфейс для реалізації функцій обміну даними;
- просте і швидке конфігурування системи в поєднанні з пакетом STEP 7.

Базова конфігурація системи включає в свій склад набір функцій, які дозволяють виконувати дієво керовану сигналізацію, архівування результатів вимірювань, реєструвати технологічні дані та налаштування конфігурації, функції управління та візуалізації. Цілий ряд функцій може бути реалізований за допомогою вбудованих ANSI-C компілятора і VisualBasic-script: від найпростіших операцій до повного доступу до системних функцій SIMATIC WinCC. Крім того, базова система може доповнюватися опціональними пакетами WinCC і WinCC Add-ons.

На основі WinCC можуть створюватися як прості системи людино-машинного інтерфейсу з одного станцією оператора, так і потужні багатокористувацькі системи, включають до свого складу десятки станцій. Підтримка стандартних інтерфейсів OLE, ODBC, OLE і SQL забезпечує універсальність і відкритість WinCC, дозволяє використовувати її в поєднанні з будь-яким іншим програмним забезпеченням. WinCC легко інтегрується у внутрішню інформаційну мережу компанії. Це не тільки знижує витрати на її впровадження, але і підвищує гнучкість інформаційної системи.

1.1.2 TRACE MODE

Так само на ринку популярна SCADA-системи TRACE MODE (рис.1.1).

TRACE MODE призначена для розробки великих розподілених АСУТП (автоматизованих систем керування технологічним процесом) широкого призначення. TRACE MODE створена в 1992 році фірмою AdAstra Research Group Ltd (Росія), і до теперішнього часу має понад 7000 інсталяцій. Системи, розроблені на базі TRACE MODE, працюють в хімічної, газової, нафтової галузях, атомній енергетиці, металургії та інших галузях промисловості і в комунальному господарстві Росії. За кількістю впроваджень в Україні TRACE MODE значно випереджає закордонні пакети подібного класу.

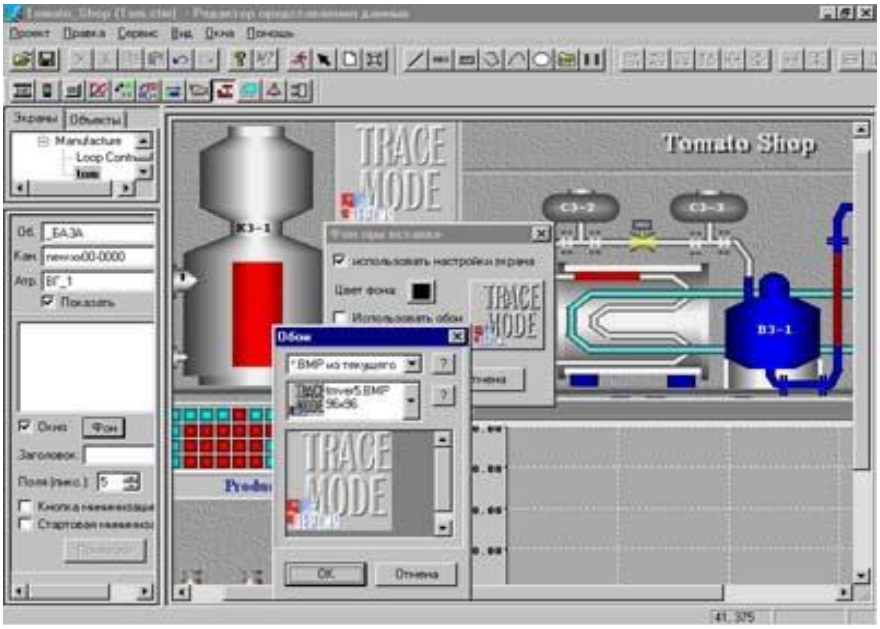


Рисунок 1.1 - Графічна мнемосхема процесу, створена в SCADA-системи TRACE MODE

Серед передових інновацій TRACE MODE можна зазначити наступне: розробка розподіленої АСУТП як єдиного проекту, автопостроение, оригінальні алгоритми обробки сигналів і управління, об'ємна векторна графіка мнемосхем, єдине мережеве час, унікальна технологія playback - графічного перегляду архівів на робочих місцях керівників. TRACE MODE - це перша інтегрована SCADA - і softlogic-система, що підтримує наскрізне програмування операторських станцій і контролерів за допомогою єдиного інструменту. Розробка графічного інтерфейсу операторських станцій проекту здійснюється в об'єктно-орієнтованому редакторі представлення даних.

Графічні зображення створюються у векторному форматі DBG, проте можливо використовувати і растрові зображення у форматі BMP. Розмір графічного поля і кількість екранів не обмежені. Редактор містить бібліотеки об'ємних зображень мнемосхем технологічних об'єктів, що включають баки, Ємності, клапани, засувки, а також їх різні перетину і сполучення. Форми динамізації містять всі необхідні елементи, в т. ч. гістограми, графічні, кольорові і звукові сигналізатори, тренди, які біжать доріжки, мультиплікацію і т.д. Великий набір бібліотек технологічних об'єктів, що включає ємності, теплообмінники, електротехнічні символи та ін., а також панелі управління, введення завдань, регуляторів, приладів і т.д. Будь-яка частина зображення може бути включена в об'єкти і анімована довільним чином. Для імпорту зображень з інших додатків Windows (наприклад з AutoCAD) редактор підтримує формати WMF, EMF. Графічні мнемосхеми можливо редагувати в реальному часі.

Історичні тренди TRACE MODE дозволяють вводити необмежену кількість змінних з необмеженою глибиною. У реальному часі користувач може додавати і видаляти виведені на тренд параметри, виробляти масштабування і зсув трендів по осях X і Y. Тренди мають візир і висновок значень в точці курсору. Основні функції:

- модульна структура - від 128 до 64000x16 I/O. Кількість тегів необмежена;
- 0,001 з - мінімальний цикл системи;
- відкритий формат драйвера для зв'язку з будь-яким УСО.
- відкритість для програмування (Visual Basic, Visual C++ і т.д.);
- розробка розподіленої АСУТП як єдиного проекту;
- кошти наскрізного програмування АСУТП верхнього (АРМ) та нижньої (ПЛК) рівня;
- вбудовані бібліотека з більш ніж 150 алгоритмів обробки даних і керування в т. ч. фільтрація, PID, PDD, нечітке, адаптивне, позиційне регулювання, ШІМ, керування пристроями (клапан, засувка, привід і т.д.), статистичні функції і довільні алгоритми;

– автоматичне гаряче резервування;

– підтримка єдиного мережевого часу;

– засоби програмування контролерів і АРМ на основі міжнародного стандарту IEC 1131-3;

– більше 200 типів форм графічного відображення інформації в т. ч. тренди, мультиплікація на основі растрових і векторних зображень, ActiveX;

– перегляд архівної інформації в реальному часі у т. ч. у вигляді трендів і таблиць;

– мережа на основі Netbios, NetBEUI, IPX/SPX, TCP/IP;

– обмін з незалежними додатками з використанням OPC client/server, DDE/NetDDE client/server, SQL/ODBC, DCOM;

– автоматичне резервування архівів і автовідновлення після збою;

– моніторинг та управління через Internet;

– повністю русифікована;

– технічна підтримка українською мовою.

Вищеописані SCADA-системи є популярними на ринку SCADA-систем. Але наступна SCADA-система GENESIS32, яка буде описана докладніше в даній дипломної роботи, на мій погляд, є лідером.

1.1.3 GENESIS32

GENESIS32 є комплексом клієнтських і серверних додатків, заснованих на технології OPC (OLE for Process control - технологія зв'язування і впровадження об'єктів для промислової автоматизації), які призначені для розробки прикладного програмного забезпечення візуалізації контрольованих параметрів, збору даних і оперативного диспетчерського управління в автоматизованих системах керування технологічними процесами. GENESIS32 є комплексом 32-розрядних додатків для Windows 98, Windows NT,Windows 2000, Windows XP і Windows Vista, побудованих у відповідності зі специфікацією OPC. Комплекс призначений для створення програмного забезпечення збору даних і оперативного диспетчерського управління верхнього рівня систем промислової автоматизації. До складу GENESIS-32 також входить середовище розробки і виконання сценарних процедур VBA, що забезпечує можливість розробки частини програмного забезпечення засобами Microsoft Visual Basic for Applications 6.0 (Visual

Basic для додатків), що входить у популярний пакет MS Office 2000. Всі програмні компоненти реалізовані на базі багатопотокової моделі і підтримують технологію ActiveX.

GENESIS for Windows підтримує стандартний протокол динамічного обміну даними - Windows Dinamic Data Exchange (DDE). Модуль DDEWorX як DDE-клієнта може отримувати дані від зовнішніх програм і передавати їх додатків-клієнтам GFW з внутрішньої програмної магістралі передачі даних Talx Data Bus. Для зв'язку з іншими робочими станціями Windows в мережі використовується NetDDE. GFW підтримує визначені користувачем списки DDE-доменів, завдяки чому з'єднання здійснюється простим натисканням кнопки миші. Не менш просто використання таких інструментів, як Cut and Paste (Вирізати і Вставити); для встановлення зв'язку досить цими стандартними функціями редагування вставити ім'я змінної з зовнішнього застосування в GFW. При налагодженні системи можливий перегляд і фільтрація DDE викликів, помилок, повідомлень тощо.

Модуль GraphWorX дозволяє користувачеві легко і швидко створювати динамічні кольорові екрани-мнемосхеми. Об'єктно-орієнтована графіка дає можливість зв'язати будь-який графічний об'єкт з технологічними параметрами процесу. Наприклад, об'єкт може одночасно змінювати колір, розмір і положення на екрані у відповідності зі значеннями різних сигналів. Серед безлічі коштів анімації зображення є і мультиплікація. Весь процес створення мнемосхеми зводиться до малювання і завданням у діалозі динамічних властивостей зображення. При створенні мнемосхеми використовуються що стали вже звичними в Windows програмах останнього покоління плаваючі інструментальні лінійки, які можна розмістити в будь-якому місці екрану.

Зображення всій мнемосхеми або її окремих частин може бути скопійовано з інших Windows додатків, імпортовано з AutoCAD або GENESIS for DOS. Також є бібліотека таких часто використовуваних символів як засувки, насоси, панелі регуляторів і т.п. Будь-яке зображення на екрані можна зберегти у власній бібліотеці. Особливо зручно те, що поряд зі статичними в бібліотеках можуть зберігатися і динамічні символи. Як бібліотеки символів, так і цілі мнемосхеми можуть використовуватися без переробок у різних проектах, дозволяючи накопичувати не тільки досвід роботи з пакетом, але і готові рішення, тиражовані у всіх подальших системах.

GraphWorX забезпечує можливості конфігурування безпосередньо в режимі on-line, що забезпечує налагодження динаміки мнемосхеми на реальних даних з об'єкту.

Вікно GraphWorX, як і будь-яке вікно системи Windows, має властивості зміни розміру та переміщення по екрану, при цьому, природно, вся графіка і тексти розтягуються пропорційно зміні масштабу без втрати графічної інформації. Ця

можливість дозволяє Вам стежити за процесом під час роботи інших програм або розташувати на екрані безліч мнемосхем одночасно.

В якості елементів мнемосхеми можуть бути використані такі GFW-додатки, як TrendWorX і AlarmWorX. Під час конфігурування мнемосхеми доступні всі конфігураційні можливості цих додатків, а в режимі реального часу меню керування переглядом тренда або буфера повідомлень викликається простим натисненням клавіші миші на зображенні тренда або списку повідомлень.

1.2 Математичне моделювання технологічних динамічних процесів

1.2.1 Загальні відомості про математичне моделювання

Математична модель – це наближений опис довільного класу явищ зовнішнього світу, поданий за допомогою математичної символіки. Математичне моделювання виступає як метод пізнання зовнішнього світу, а також прогнозування і управління. Аналіз математичних моделей дозволяє проникнути в сутність досліджуваних явищ.

Математичне моделювання проходить такі етапи:

– постановка задачі, тобто прийняття рішення про необхідність моделювання і його мету. На цьому етапі слід чітко визначити і сформувати мету досліджень. З мети досліджень випливатиме сукупність властивостей об'єкта моделювання, які підлягатимуть відбиттю у моделі;

– побудова математичної моделі;

– дослідження системи на моделі, прогнозування й управління оригіналом за результатами цих досліджень.

Моделювання зводиться до дослідження властивостей певного об'єкта вивченням (дослідженням, аналізом) аналогічних властивостей іншого об'єкта, більш зручного для дослідження, який знаходиться з першим у певній відповідності. Перший об'єкт називається в цьому випадку оригіналом, а другий – моделлю. Як модель, так і оригінал можуть бути матеріальними тілами чи фізичними явищами, або описом цих тіл чи явищ за допомогою тих чи інших засобів. В ролі оригіналу може виступати, наприклад, певна проблема, моделлю якої буде задача меншого рівня складності. Скажімо, так звана обчислювальна модель є абстрактною чи конкретною задачею, яка відповідає проблемі чисельного розв'язання певного класу математичних чи прикладних задач. Якщо при переході від оригіналу до моделі використовується заміна оригіналу на матеріальне тіло чи явище, то така модель називається фізичною; якщо ж оригінал заміняється його

описом, то модель може бути вербальною, математичною або графічною, залежно від використовуваних при описі символів. Реалізована у вигляді макета чи пристрою, чи зафіксована у вигляді словесного опису, рівняння, формули, графіка, креслення, модель є системою наших уявлень про оригінал, його властивості і взаємозв'язки на певному етапі пізнання оригіналу. Вибір об'єктів і методів моделювання визначається поставленою задачею [11].

До основних характеристик математичних моделей (ММ) належать: ступінь універсальності моделі; точність моделі; адекватність моделі; економічність моделі.

Ступінь універсальності ММ характеризує повноту відображення у моделі властивостей реального об'єкта; кількісно ступінь універсальності може бути описаний співвідношенням потужності множини відображених властивостей до множини наявних властивостей системи.

Точність математичної моделі оцінюється за збіжністю значень параметрів реального об'єкта і значень тих же параметрів, отриманих за допомогою побудованої моделі; при цьому ступінь збіжності розраховують через відхилення цих параметрів.

Економічність математичної моделі характеризується витратами обчислювальних ресурсів на її реалізацію. Чим вони менші, тим модель економічніша. Останнім часом, для характеристики економічності моделі застосовують так звані комбіновані параметри: середня кількість операцій, яка виконується під час одного звертання до моделі, розмірність системи рівнянь, кількість внутрішніх параметрів моделі тощо. Адекватність ММ – це її здатність відображати задані властивості об'єкта з похибкою, не більше заданої. При цьому адекватність моделі переважно спостерігається виключно в обмеженій області зміни зовнішніх параметрів, яка називається областю адекватності (ОА) математичної моделі. Подібність моделі та оригіналу є невід'ємною умовою адекватності моделювання.

За ступенем відповідності параметрів моделі і оригіналу розрізняють подібності абсолютну і практичну (неабсолютну). Остання, в свою чергу, буває повною, неповною і наближеною. За адекватністю фізичної природи аналогічних явищ подібність поділяють на математичну і фізичну (електричну, механічну, теплову тощо). Фізична подібність досягається за однакової фізичної природи явищ, математична – за відповідності схожих параметрів процесів різної фізичної природи. І перша, і друга подібності можуть бути повною, неповною і наближеною [11-13].

При абсолютній подібності оригінал і модель структурно та фізично подібні; вони відрізняються лише значеннями параметрів, що характеризують елементи і зв'язки між ними. Процеси у моделі і оригіналі в цілому, так само як стани окремих елементів,

описуються однаковими функціональними залежностями, що пропорційно відрізняються лише значеннями аргументів. Відтворення процесу на моделі здійснюється без жодних спотворень щодо оригіналу і відрізняється від нього лише масштабом.

Слід підкреслити, що якщо з абсолютної фізичної подібності процесів випливає реальна або потенційна ідентичність математичних співвідношень, що їх описують, то зворотне ствердження у загальному випадку неправильне: ідентичність форм запису математичних рівнянь ще не означає подібності процесів, оскільки характер перебігу процесу визначається не лише видом функціональної залежності між змінними, що беруть в них участь, але і співвідношенням їх конкретних значень.

Абсолютна подібність свідчить про тотожність явищ, яка є поняттям доволі абстрактним і реалізується на практиці виключно в геометричних побудовах та в окремих видах математичної подібності. В переважній більшості випадків розв'язання конкретних задач дослідник не має змоги працювати з явищами, схожими абсолютно у всіх деталях. Тому виникає потреба введення поняття практичної подібності, в межах якої розрізняють повну, неповну і наближену подібності.

Повна подібність – це подібність перебігу у часі та просторі тих процесів, які є суттєвими для цього дослідження і з достатньою повнотою характеризують досліджуване явище стосовно конкретної постановки задачі дослідження.

Неповна подібність – це подібність перебігу процесів лише в просторі чи лише в часі (наприклад, при подібності перебігу перехідних процесів у двох електричних лініях розподіл електричного поля може бути різним внаслідок різної геометрії дроту). Наближена подібність характеризується існуванням спрощених допущень, які дозволяють вважати подібними відмінні процеси за рахунок свідомих спотворень деяких їх властивостей. Наближена подібність може бути і повною, і неповною. Так, наближеною можна вважати подібність двох генераторів, виявлену на основі їх спрощених рівнянь, що не враховують аперіодичну складову струму статора і періодичну складову струму ротора.

Стосовно фізичної природи розрізняють фізичну і математичну подібності. Фізична подібність передбачає однакову фізичну природу подібних явищ. За фізичної подібності механічним процесам у досліджуваній системі ставляться у відповідність механічні процеси у подібних їй системах, електричним – електричні тощо. Деколи виділяють кінематичну (подібність швидкостей і прискорень), матеріальну (подібність мас окремих елементів системи) і динамічну (подібність сил, що викликають рух) подібності. Системи, подібні кінематично, матеріально і динамічно, вважаються механічно подібними. Електрична подібність існує при подібності електричних і магнітних полів, напруг,

струмів і потужностей окремих елементів. Аналогічно системи тіл, у яких подібні теплові потоки і температура мають теплову подібність тощо [11-13].

Фізична подібність може встановлюватися не лише для фізичних явищ, що підпорядковуються детермінованим законам, а і для стохастичних процесів; в цих випадках говорять про статистичну подібність.

Побудову математичної моделі, тобто вивчення явища за допомогою математичної моделі, можна умовно розбити на 4 етапи (рис. 1.2): етап змістовного опису; етап формалізації опису; етап остаточної побудови моделі (ідентифікації параметрів і перевірки адекватності моделі); етап перегляду і вдосконалення моделі за результатами узагальнення емпірично накопичених даних [11].

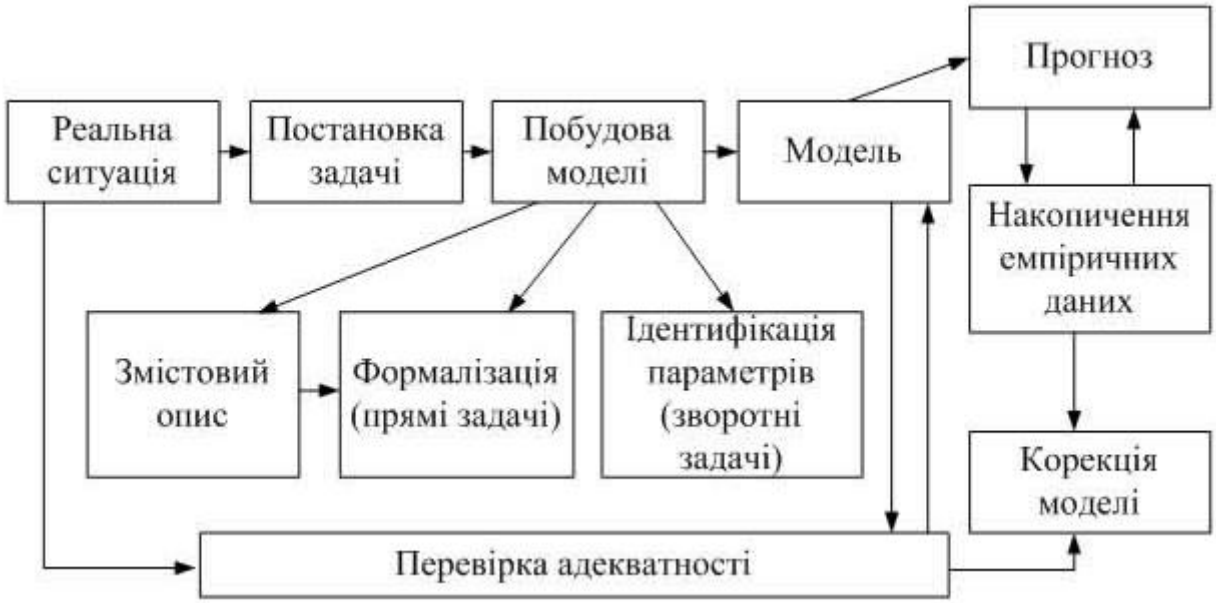


Рисунок 1.2 – Алгоритм побудови математичної моделі

1.2.2 Класифікація математичних моделей

Аналіз літературних джерел по моделюванню дозволяє класифікувати математичні моделі по наступних ознаках [14]:

1) складність об'єкту моделювання;
2) оператор моделювання (подмодель);
3) вхідні і вихідні параметри моделі;
4) цілі моделювання;
5) метод реалізації моделі.

Всі об'єкти моделювання можна розділити на дві групи: прості об'єкти і об'єкти-системи. При моделюванні простих об'єктів не розглядається внутрішньо будова об'єкту, не виділяються складові його елементи або підпроцеси. Простим об'єктом, наприклад, є матеріальна крапка в класичній механіці. Для складних систем характерна наявність великого числа взаємозв'язаних і взаємодіючих елементів. Їх поведінка багатоваріантна. При моделюванні об'єктів-систем виникають великі труднощі. Моделі об'єктів- систем, що враховують властивості і поведінку окремих елементів, а також взаємозв'язки між ними, називаються структурними моделями.

Оператор моделі визначається сукупністю рівнянь. Якщо оператор забезпечує лінійну залежність вихідних чинників від вхідних, то математична модель називається лінійною. Інакше модель називається нелінійною.

Залежно від виду використовуваної безлічі параметрів моделі діляться на якісних і кількісних, дискретних і безперервних, змішаних.

Залежно від мети моделювання виділяють дискриптивные оптимізаційні, управлінські моделі. Метою дискриптивных моделей є встановлення законів зміни параметрів моделі. Оптимізаційні моделі призначені для визначення оптимальних (якнайкращих) з погляду деякого критерію параметрів об'єкту і технологічних режимів. Управлінські моделі застосовуються для ухвалення ефективних управлінських рішень.

Залежно від методу реалізації виділяють аналітичні і алгоритмічні математичні моделі. Метод є аналітичним, якщо він дозволяє отримати вихідні чинники у вигляді аналітичних виразів. Аналітичні методи бувають алгеброю і наближеними. У алгоритмічних моделях математичні співвідношення для об'єкту дослідження замінюються алгоритмом. Алгоритмічні моделі бувають чисельними і імітаційними.

При моделюванні технічних систем і процесів класифікація математичних моделей набуває додаткових ознак [15]:

– по етапах життєвого циклу створення об'єкту виділяють моделі аналізу, моделі проектування, моделі впровадження і т. д.;

– по рівню формалізації моделі можна виділити концептуальну модель (для користувача і аналітика), формалізоване, або алгоритмічне, опис і программу-имитатор;

– по методах побудови розрізняють моделі, створені за допомогою аналітичних і статистичних методів. У основі аналітичних моделей процесів лежать фундаментальні закони тепло- і масопереносу, виражені у вигляді функціональних співвідношень (алгебрі, інтегрально-диференціальних, кінечно-різницевих і т. д.).

Тому аналітичні моделі описують і розкривають суть процесів і явищ, що протікають в досліджуваному об'єкті і визначають його властивості і поведінку. Методи

дослідження аналітичних моделей: аналітичні (отримують загальне рішення в явному вигляді і підставляють в нього значення граничних і початкових умов) і чисельні (загальні рішення в явному вигляді замінюються наближеними). Як приклад аналітичних моделей можна назвати диференціальні рівняння. У основі статистичних моделей лежать результати експериментального дослідження об'єкту. Тому ці моделі також називають емпіричними, такими, що ідентифікуються, ймовірносно-статистичними, досвідчено-статистичними.

Статистичні моделі розглядають досліджуваний об'єкт як «чорний ящик» і не розкривають суть процесів і явищ, що протікають в нім, – вони просто відображають одну з можливих залежностей вихідних змінних від вхідних, тобто носять приватний характер на відміну від аналітичних моделей, які мають більш загальний характер. Приклади емпіричних моделей – кореляційні, регресійні моделі.

1.3 Цілі математичного моделювання для технічних об'єктів і технологічних процесів

Раніше нами вже були детально викладені загальні цілі моделювання. З урахуванням специфіки технічних об'єктів і технологічних процесів машинобудівного виробництва має сенс їх конкретизувати і позначити цілі моделювання таким чином [15]:

1. Допомогти при вирішенні завдань стратегічного і тактичного управління. Існує ієрархія завдань управління технологічними процесами і комплексами. На верхньому рівні вирішуються завдання стратегічного планування і управління. На нижніх рівнях – тактичні завдання календарного планування і поточного управління. Цій ієрархії завдань відповідає ієрархія математичних моделей.

2. Замінити неприпустимі на реальному технічному об'єкті досліди експериментами на його моделі. Досліди на реальному об'єкті замінюються комп'ютерними (обчислювальними) експериментами, що дозволяє істотно підвищити якість ухвалюваних інженерних і управлінських рішень, понизити терміни і витрати на досягнення оптимальних результатів.

3. Звести дослідження реального об'єкту до рішення математичної задачі. 20 Математичне, програмне, комп'ютерне забезпечення, що є в даний час, дозволяє змоделювати і досліджувати велику кількість варіантів вирішуваного завдання, вибрати і обгрунтувати найбільш доцільне рішення.

4. Отримати ефективний інструмент дослідження складних систем і процесів. Математичне моделювання дозволяє розглянути ряд процесів, що одночасно протікають в системі, і вибрати оптимальний інструмент їх дослідження.

5. Узагальнити знання, накопичені про об'єкт. Моделі служать як би акумуляторами знань про об'єкти і виконують особливу смыслообразующую роль в системі науково-технічних знань.

1.4 Технічне завдання на проектування

Технічне завдання оформлене відповідно до вимог Єдиної системної програмної документації (ГОСТ 19.201-78).

1.4.1 Вимоги до виконуваних функцій

Функціональні можливості розроблювальної системи (прикладного програмного забезпечення) повинні враховувати всі особливості систем моделювання й забезпечувати:

- можливість керування процесом моделювання;
- уведення додаткових функцій керування параметрами процесу моделювання;
- регулювання заданих параметрів системи;
- перегляд графіків зміни параметрів системи;
- візуалізацію технологічних даних (графіки, гістограми);
- блокування “помилкових" дій користувача;
- виконання обчислення процесів швидше ходу реального часу;
- можливість створення моделей замкненої системи будь-якої складності й використання різних методів математичного моделювання.

1.4.2 Вимоги до функціонування розроблювальної системи

Розроблювальне ПЗ повинне відповідати наступним вимогам:

- система повинна надавати користувачеві зручний, сучасний і інтуїтивно зрозумілий інтерфейс для проектування моделей, їх редагування та збереження;

– система повинна забезпечувати користувача всією інформацією, необхідної для проектування;

– надавати засоби для графічного відображення стану моделі при її розрахунках і по завершенню. Мати можливість роздільного перегляду графіків стану по кожному входу / виходу;

– мати модульну й розширювану архітектуру для легшої розроби додатків;

– мати інсталяційний пакет для установки на будь-який комп'ютер.

Система, при введенні користувачем вихідних даних, повинна здійснювати перевірку значень, що вводяться. При введенні неприпустимих значень параметрів повинні видаватися повідомлення про помилки. При введенні некоректних значень повинні видаватися попередження про те, що дані не узгодяться один з одним.

14.3 Вимоги до програмного забезпечення

Для установки та коректного функціонування ПЗ необхідна наявність на комп'ютері користувача операційної системи сімейства Windows і встановленого пакета .Net Frameworkv 4.0. Для найкращого функціонування ПЗ, бажана наявність установлених останніх відновлень операційної системи та .Net Framework.

1.4.4 Вимоги до апаратного забезпечення (для установки і коректного функціонування додатка)

Для нормального функціонування ПЗ комп'ютер, на якому воно виконується, повинен відповідати наступним апаратним вимогам:

– центральний процесор: x64 або x86 з тактовою частотою 2.1 Ггц;

– обсяг оперативної пам'яті: 4 ГБ;

– ємність жорсткого диска: 40 Гб;

– графічний адаптер з підтримкою DirectX 11 і 1 Гб пам'яті.

2 АНАЛІЗ ТА ВИБІР ПРОГРАМНО-ТЕХНІЧНИХ ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ МОДЕЛЮВАННЯ

У розділі розглядаються та аналізуються засоби, необхідні для розв'язку поставленого в першому розділі завдання. Проводиться їхній короткий огляд, позначається область застосування, виявляються гідності й недоліки. Робляться висновки та приводяться аргументи на користь обраних засобів.

2.1 Огляд операційних систем

Серед всіх комп'ютерних програм, з якими працюють користувачі, особливе місце займають операційні системи. Від вибору ОС залежить продуктивність роботи, ступінь захисту даних, необхідні апаратні засоби і т.д. Однак, вибір операційної системи також залежить від технічних характеристик (конфігурації) комп'ютера. Чим більше сучасніше операційна система, тим вона не тільки надає більше можливостей і більш наочна, але також тим більше вона пред'являє вимог до комп'ютера (тактова частота процесора, оперативна і дискова пам'ять, наявність і розрядність додаткових карт і пристроїв).

ОС здійснює завантаження в оперативну пам'ять програмне забезпечення системи моделювання, передає йому керування на початку роботи, виконує різні дії за запитом програмного забезпечення і звільняє займану програмою оперативну пам'ять при її завершення.

Таким чином, вибір операційної системи дуже важливий, тому що від цього вибору залежатиме вся подальша робота ПК в цілому, та системи моделювання окремо: продуктивність, ступінь захищеності даних і т.д. Для того, щоб зробити цей вибір нам, для початку, потрібно ознайомитися з системними вимогами даного продукту, а потім підібрати той, який нам, як розробнику спеціалізованого ПЗ, більше підходить. Тому проведемо подання та порівняння з різних сторін найбільш поширених операційних систем.

ОС дозволяє абстрагуватися від деталей реалізації апаратного забезпечення, надаючи розробникам програмного забезпечення мінімально необхідний набір функцій. З точки зору обивателів, звичайних користувачів комп'ютерної техніки, ОС включає в себе і програми для користувача інтерфейсу. Основні функції (найпростіші ОС):

– завантаження додатків в оперативну пам'ять і їх виконання;

– стандартизований доступ до периферійних пристроїв (пристрої введення-виведення);

– управління оперативною пам'яттю (розподіл між процесами, віртуальна пам'ять);

– управління доступом до даних на енергонезалежних носіях за допомогою файлової системи;

– користувальницький інтерфейс;

– мережеві операції, підтримка стека протоколів

Найпоширеніші на сьогоднішній день операційні системи - це все операційні системи компанії Microsoft, Linux і Mac OS.

Всі операційні системи (Windows XP, Windows Vista, Windows Seven, Linux, Mac OS) були оцінені за кількома параметрами:

– безпеку,

– інтерфейс,

– навантаження на систему,

– ціна.

Таблиця 2.1 – Порівняння сучасних операційних систем

	Windows XP	**Windows Vista**	**Windows Seven**	**Linux**	**Mac OS**
Безпека	Без установки оновлень і патчів - сама беззахисна система для доступу в інтернет. • Найбільш ласа мета для безлічі вірусів і інших шкідливих програм в останні кілька років. • Потрібна установка Service Pack 2 і сторонніх антивірусів і фаерволлов, а також безлічі оновлень і патчів для безпечного використання в мережі.	• Покращує безпеку Windows XP за допомогою додаткових програм, але залишається головною метою для шкідливих програм. • Вбудований firewall не забезпечує зрозумілих налаштувань для вихідних з'єднань, так що потрібна установка більш простого стороннього рішення.	• Як і попередні версії Windows потребує постійного оновлення. • Вбудовані системи безпеки стали помітно краще в порівнянні з попередніми версіями Windows, але все одно не обійтися без сторонніх додатків.	• Linux більш безпечна система, ніж Windows. Наприклад, Ubuntu, замовчуванням, навіть не створює адміністраторський аккаунт, який є неодмінною метою для шкідливих програм. • У серці Unix - більш сувора система, що веде до меншої кількості дірок в безпеки в порівнянні з архітектурою Windows. • Маленька поширеність Linux призводить до того, що хакери менше звертають на неї увагу, ніж на Windows.	• Чіткий поділ системних і призначених для користувача файлів для максимальної безпеки. • Mac OS – сертифікована Unix-система з усією її надійністю і безпекою. • Можливість завантаження макінтоша в режимі зовнішнього диска допомагає при відновленні системи, але може стати причиною витоку цінної інформації, так як в даному випадку ігноруються права доступу до файлів користувачів.

Продовження табл.2.1

1	2	3	4	5	6
7Інтерфейс	• Відсутність єдності інтерфейсу. Немає чітких правил, як повинні виглядати елементи управління в різних додатках - все віддано в руки розробників додатків. • Відсутність нормальних ефектів при перемиканні між вікнами і їх згортання. • Застарілий механізм пошуку файлів. Поліпшити можна тільки сторонніми додатками, наприклад - Google Desktop Search.	• Перевантажений інтерфейс. Змінені положення деяких елементів на панелі керування. • Ефекти напівпрозорості, анімації дозволяють легше орієнтуватися в роботі і перемиканні між програмами. • Швидкий пошук файлів по всій системі. • Можливість використання Gadgets в бічній панелі на робочому столі.	• Досить приємний інтерфейс, що не викликає роздратування. • Можливість перегляду вікон в 3D, красиві і зручні панелі спрощують користування комп'ютером. • Дуже швидкий пошук допомагає краще орієнтуватися у величезній бібліотеці файлів. • Можливість використання Gadgets в бічній панелі на робочому столі.	• Інтерфейси Gnome і KDE схожі на інтерфейси Mac OS і Windows відповідно. • Вбудована можливість використання декількох віртуальних робочих столів. • Можливість включення графічного прискорення присутній, але вимагає окремого настроювання.	• Інтерфейс чіткий, неперевантаженість і логічний. Настільки хороший, що його намагаються відтворити на інших операційних системах за допомогою тем оформлення та спеціальних програм. • Напівпрозорість і ефекти анімації дуже органічні і допомагають орієнтуватися в системі. • Наявність віртуальних робочих столів з можливістю перетягування вікон між столами.

Продовження табл.2.1

1	2	3	4	5	6
Наванта ження на систему	• Прекрасно працює на не надто швидких процесорах і при невеликому обсязі оперативної пам'яті. • XP залишиться найкращим вибором для багатьох користувачів ще як мінімум найближчі пару років.	• Вимагає більше оперативної пам'яті і місця на диску, але працює повільніше Windows XP на однаковому комп'ютері. • Високі системні вимоги.	• Чималі системні вимоги ускладнює роботу даної ОС на старих процесорах. • Відмінна швидкість роботи.	• Відмінно працює навіть на дуже старих комп'ютерах через незначні системних вимог. • Підтримка нового обладнання часто відстає, тому що виробники апаратних засобів в першу чергу орієнтуються на Windows і Mac OS.	• Продуктивність на висоті, тому що програмне забезпечення дуже добре оптимізовано під конкретну платформу. • Apple не випускає повільних комп'ютерів - у всіх з них потужний процесор Intel Core 2 Duo і як мінімум 1Гб оперативної пам'яті.
Цінв	2000-4000 грн.	3000-5000 грн.	1500-6000 грн.	Безкоштовно	Входить у вартість ноутбука.

2.2 Огляд мов програмування

Microsoft Visual C++ (MSVC) — інтегроване середовище розробки засобів на мові C++, розроблена фірмою Microsoft. Постачається або як частина комплекту Microsoft Visual Studio, або окремо у вигляді безплатного функціонально обмеженого комплекту Visual C++ Express Edition.

Visual Studio дозволяють розробникам всіх рівнів швидко створювати розподілені веб–засоби й засоби з повноцінними інтерфейсами для Windows Vista, Windows 7, Windows Server 2008, Microsoft Office 2007, Microsoft Office 2010 мобільних пристроїв та мережі Інтернет.

В першу чергу Visual C++ — це компілятор C++, але це таке середовище, компоненти якого, взаємодіючи один з одним, спрощують процес розробки застосунків. Середовище Visual C++ пропонує великі можливості для програмування Windows–застосунків. Найхарактернішою його компонентою є бібліотека основних класів Microsoft (Microsoft Foundation Classes – MFC). Великий набір класів C++ складає основну частину API (Application Standart Interface) і пропонує могутню основу для написання типових програм.

Компілятор Visual C++ містить багато нових інструментальних засобів і поліпшених можливостей для створення Windows – застосунків.

Microsoft розробили бібліотеку Microsoft Foundation Classes — MFC. Використовуючи готові класи C++, можна вирішувати багато задач. Бібліотека MFC полегшує програмування в середовищі Windows. Ті, хто володіє достатнім досвідом програмування на C++, можуть допрацьовувати класи або створювати нові, похідні від існуючих. Класи бібліотеки MFC використовуються як для керування об'єктами Windows, так і для рішення певних загальносистемних задач. Наприклад, в бібліотеці є класи для керування файлами, рядками, часом, обробкою виключень і інші. По суті, в MFC представлені практично всі функції WindowsAPI. У бібліотеці є засоби обробки повідомлень, діагностики помилок і інші засоби, звичні для застосунків Windows.

MFC має такі переваги: Представлений набір функцій і класів відрізняється логічністю і повнотою. Бібліотека MFC відкриває доступ до всіх часто використовуваних функцій Windows API, включаючи функції управління вікнами застосунків, повідомленнями, елементами управління, меню, діалоговими вікнами, об'єктами GDI (Graphics Device Interface – інтерфейс графічних пристроїв), такими як шрифти, кисті, пір'я і растрові зображення, функції роботи з документами тощо.

Функції MFC легко вивчати. Фахівці Microsoft доклали всі зусилля для того, щоб імена функцій MFC і пов'язаних з ними параметрів були максимально близькі до їхніх еквівалентів з WindowsAPI. Завдяки цьому програмісти легко зможуть розібратися в їх призначенні.

Програмний код бібліотеки досить ефективний. Швидкість виконання застосунків, заснованих на MFC, буде приблизно такою ж, як і швидкість виконання застосунків, написаних з використанням стандартних функцій Windows API, а додаткові витрати оперативної пам'яті будуть досить незначними.

MFC містить засоби автоматичного керування повідомленнями. Бібліотека MFC усуває необхідність в організації циклу обробки повідомлень поширеного джерела помилок в Windows – застосунках. У MFC передбачений автоматичний контроль за появою кожного повідомлення.

MFC дозволяє організувати автоматичний контроль за виконанням функцій. Ця можливість реалізується за рахунок того, що можна записувати в окремий файл інформацію про різні об'єкти і контролювати значення змінних членів об'єкту в зручному для розуміння форматі.

MFC має чіткий механізм обробки виняткових ситуацій. Бібліотека MFC була розроблена так, щоб тримати під контролем появу таких ситуацій. Це дозволяє об'єктам MFC відновлювати роботу після появи помилок типу "outofmemory" (брак пам'яті), неправильного вибору команд меню або проблем із завантаженням файлів або ресурсів.

MFC забезпечує динамічне визначення типів об'єктів. Це надзвичайно могутній програмний засіб, що дозволяє відкласти перевірку типу динамічно створеного об'єкту до моменту виконання програми. Завдяки цьому можна вільно маніпулювати об'єктами, не піклуючись про попередній опис типу даних. Оскільки інформація про тип об'єкту повертається під час виконання програми, програміст звільняється від цілого етапу роботи, пов'язаного з типізацією об'єктів.

MFC може використовуватися спільно з підпрограмами, написаними на мові C++. Важливою особливістю бібліотеки MFC є те, що вона може "співіснувати" з застосунками, заснованими на WindowsAPI. У одній і тій же програмі програміст може використовувати класи MFC і викликати функції WindowsAPI. Така прозорість середовища досягається за рахунок узгодженості програмних позначень в двох архітектурах. Іншими словами, файли заголовків, типи і глобальні константи MFC не конфліктують з іменами з WindowsAPI. Ще одним ключовим моментом, що забезпечує таку взаємодію, є узгодженість механізмів управління пам'яттю.

Останнім часом C і C++ є найбільш використовуваними мовами для розробки комерційних і бізнес додатків. Ці мови влаштовують багато розробників, але насправді не забезпечують належної продуктивності розробки. Наприклад, процес написання додатку на C++ часто займає значно більше часу, чим розробка еквівалентного застосування, скажімо, на Visual Basic.

Зараз існують мови, що збільшують продуктивність розробки за рахунок втрати в гнучкості, яка така звична і необхідна програмістам на C/C++. Подібні рішення є вельми незручними для розробників і часто пропонуються значно менші можливості. Також ці мови не орієнтовані на взаємодію з системами, що з'являються сьогодні, і дуже часто вони не відповідають існуючій практиці програмування для Web. Багато розробників хотіли б використовувати сучасну мову, яка дозволяла б писати, читати і супроводжувати програми з простотою Visual Basic і в той же час давав би потужність і гнучкість C++, забезпечував би доступ до всіх функціональних можливостей системи, взаємодіяв би з існуючими програмами і легко працював би з виникаючими стандартами Web.

Враховуючи всі подібні побажання, Microsoft розробила нову мову - C#. У нього входить багато корисних особливостей - простота, об'єктна орієнтованість, типова захищеність, "збірка сміття", підтримка сумісності версій і багато що інше. Дані можливості дозволяють швидко і легко розробляти додатки, особливо Com+ додатку і Web сервіси. При створенні C#, його автори враховували досягнення багатьох інших мов програмування: C++, C, Java, Smalltalk, Delphi, Visual Basic і так далі Треба відмітити що унаслідок того, що C# розроблявся з чистого листа, у його авторів була можливість (якій вони явно скористалися), залишити у минулому всі незручні і неприємні особливості (що існують, як правило, для зворотної сумісності), будь-якої з попередніх йому мов. В результаті вийшов дійсно простій, зручна і сучасна мова, по потужності не поступливий C++, але що істотно підвищує продуктивність розробок.

Зважаючи на свій дуже зручний об'єктно-орієнтований дизайн, C# є хорошим вибором для швидкого конструювання різних компонентів - від високорівневої бізнес логіки до системних застосувань, що використовують низькорівневий код. Також слід зазначити, що C# є і Web орієнтованим - використовуючи прості вбудовані конструкції мови ваші компоненти можуть бути легко перетворені на Web сервіси, до яких можна буде звертатися з Internet за допомогою будь-якої мови на будь-якій операційній системі. Додаткові можливості і переваги перед іншими мовами приносить в C# використання передових технологій Web, таких як: HTML (Hypertext Markup Language), XML (Extensible Markup Language) і SOAP (Simple Object Access Protocol). Середовище розробки Web сервісів дозволяє програмістові дивитися на тих, що існують сьогодні Web додатку, як на

рідні об'єкти C#, що дає можливість розробникам співвіднести наявні сервіси Web з їх пізнаннями в об'єктно-орієнтованому програмуванні.

Дуже часто можна простежувати такий зв'язок - чим більш мова захищена і стійка до помилок, тим менше продуктивність програм, написаних на нім. Наприклад розглянемо дві крайнощі - очевидно це Assembler і Java. У першому випадку ви можете добитися фантастичної швидкості своєї програми, але вам доведеться дуже довго примушувати її працювати правильно не на вашому комп'ютері. У випадку ж з Java - ви отримуєте захищеність, незалежність від платформи, але на жаль, швидкість вашої програми навряд чи сумісна з уявленням, що склалося, про швидкість, наприклад якого-небудь окремого клієнтського застосування (звичайно існують обмовки - JIT компіляція і інше). Розглянемо C++ з цієї точки зору - на мій погляд співвідношення в швидкості і захищеності близько до бажаного результату, але на основі власного досвіду програмування я можу з упевненістю сказати, що практично завжди краще понести незначну втрату в продуктивності програми і придбати таку зручну особливість, як "збірка сміття", яка не тільки звільняє вас від утомливого обов'язку управляти пам'яттю уручну, але і допомагає уникнути вам багатьох потенційних помилок у вашому застосуванні. Насправді скоро "збірка сміття", та і будь-які кроки до усунення потенційних помилок стають відмінними рисами сучасної мови. У C#, як в поза сумнівом сучасній мові, також існують характерні особливості для обходу можливих помилок. Наприклад, окрім згаданої вище "збірки сміття", там всі змінні автоматично ініціалізувалися середовищем і володіють типовою захищеністю, що дозволяє уникнути невизначених ситуацій у випадку, якщо програміст забуде ініціалізувати змінну в об'єкті або спробує провести неприпустиме перетворення типів. Також в C# були зроблені заходи для виключення помилок при оновленні програмного забезпечення. Зміна коди, в такій ситуації, може непередбачувано змінити суть самої програми. Щоб допомогти розробникам боротися з цією проблемою C# включає підтримку сумісності версій (vesioning). Зокрема, у відмінності від C++ і Java, якщо метод класу був змінений це повинно бути спеціально обумовлено. Це дозволяє обійти помилки в коді і забезпечити гнучку сумісність версій. Також новою особливістю є native підтримка інтерфейсів і спадкоємства інтерфейсів. Дані можливості дозволяють розробляти складні системи і розвивати їх з часом.

Важливою і відмітною від C++ особливістю C# є його простота. Наприклад, чи завжди ви пам'ятаєте, коли пишіть на C++, де потрібно використовувати "->", де "::", а де "."? Навіть якщо немає, то компілятор завжди поправляє вас у разі помилки. Це говорить лише про те, що насправді можна обійтися тільки одним оператором, а компілятор сам

розпізнаватиме його значення. Так в C#, оператор "->" використовується дуже обмежено (у unsafe блоках, про які мова піде нижчим), оператор "::" взагалі не існує. Практично завжди ви використовуєте тільки оператора "." і вам більше не потрібно стояти перед вибором.

Ще один приклад. При написанні програм на C/c++ вам доводилося думати не тільки про типи даних але про їх розмір в конкретній реалізації. У C# все спрощено - тепер символ Unicode називається просто char (а не wchar_t, як в C++) і 64-бітове ціле тепер - long (а не __int64). Також в C# немає знакових і беззнакових символьних типів.

У C#, також як і в Visual Basic після кожного виразу case в блоці switch мається на увазі break. І більш не відбуватиметься дивних речей якщо ви забули поставити цей break. Проте якщо ви дійсно хочете щоб після одного виразу case програма перейшла до наступного ви можете переписати свою програму, наприклад, оператора goto.

Ще одне спрощення - в C# не існує множинного спадкоємства класів. Насправді його використання є зовсім не найпростішим завданням і часто приводить до помилок. Замість цього в C# прийнята повна підтримка концепцій Com+, які виключають необхідність використання множинного спадкоємства.

Багатьом програмістам (на той момент, напевно, майбутнім програмістам) було не так легко під час вивчення C++ повністю освоїтися з механізмом посилань і покажчиків. Багато хто плутався у використанні операторів "*" і "&". У C# (хтось зараз пригадає про Java) немає покажчиків. Насправді нетривіальність покажчиків відповідала їх корисності. Наприклад, деколи, важко собі уявити програмування без покажчиків на функції. Відповідно до цього в C# присутні Delegates - як прямий аналог покажчика на функцію, але їх відрізняє типова захищеність, безпека і повна відповідність концепціям об'єктно-орієнтованого програмування.

Архітектура .NET Framework Основою .NET являються віртуальна машина для проміжної мови (Intermediate Language — IL, іноді зустрічається скорочення Microsoft IL – MSIL, пізніше прийняли назву Common Intermediate Language – CIL), в яку транслюються усі .NET-програми, що також називається загальним середовищем виконання (Common Language Runtime — CLR), і загальна бібліотека класів (.NET Framework class library, FCL), доступна з усіх .NET-застосувань. Проміжна мова є повноцінною мовою програмування, але вона не призначена для використання людьми. Розробка у рамках .NET ведеться на одній з мов, для яких є транслятор, в проміжну мову — Visual Basic.NET, C++, C#, Java (транслятор Java в .NET називається J#, і він не забезпечує однакової роботи програм на Java, що відтрансльовані в .NET і виконуваних на JVM) і ін. Проте різні мови досить сильно відрізняються одна від одної, і щоб гарантувати

можливість з однієї мови працювати з компонентами, написаними на іншій мові, необхідно при розробці цих 2 компонентів дотримуватися загальних правил (Common Language Specifications — CLS), які визначають, якими конструкціями можна користуватися в усіх .NET-мовах без втрати можливості взаємодії між результатами. Найбільш близький до проміжної мови C# — ця мова була спеціально розроблена разом з платформою .NET.

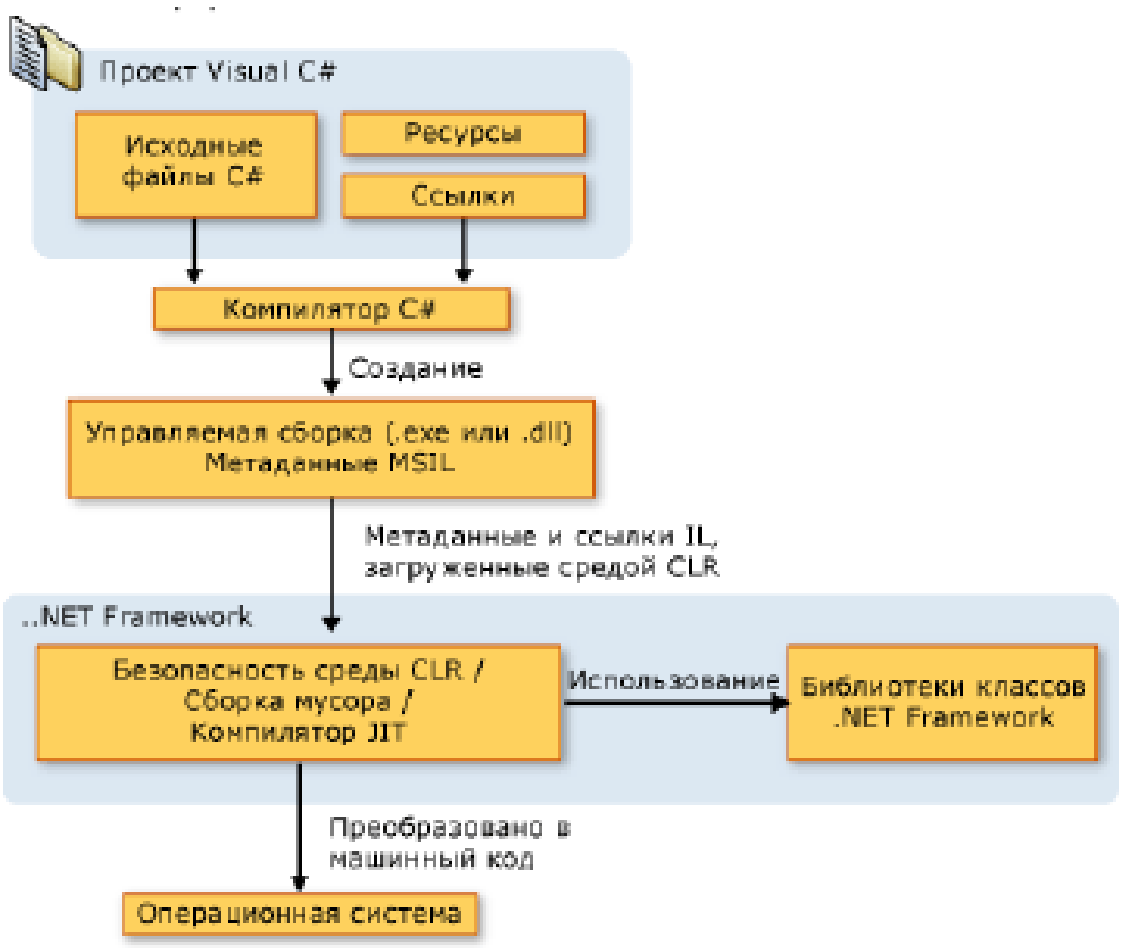


Рисунок 2.1 – Архітектура з MSDN

З точки зору написання вихідного коду платформа .NET Framework (каркас, фреймворк) здебільшого складається з велетенської бібліотеки коду FCL, який можна використати з клієнтських мов (типу C#) шляхом застосування різних прийомів об'єктно-орієнтованого програмування (Object-Oriented Programming), або ООП. Ця бібліотека поділена на модулі, які застосовуються залежно від того, які результати вимагається отримати. Наприклад, в одному модулі містяться компонувальні блоки для застосувань Windows, в іншому – для програмування мережевого обміну, в третьому – для розробки Web-застосувань. Деякі з модулів містять більше специфічні підмодулі, на

зразок модуля для розробки Web-застосувань, який містить підмодуль для написання Web-служб.

За задумом різні операційні системи повинні підтримувати деякі або усі ці модулі, залежно від їх характеристик. Пристрої PDA, наприклад, включатиме підтримку для усіх ключових функціональних можливостей .NET, але навряд чи потребуватиме в наявності більше специфічних модулів.

У одному з розділів бібліотеки .NET Framework містяться визначення ряду базових типів. Типи відповідають за спосіб представлення даних, і вказівку деяких найбільш фундаментальних з них (наприклад, типу 32-бітове ціле число зі знаком) сприяє функціональній сумісності між мовами, що використовують .NET Framework. Це носить назву системи загальних типів (Common Type System – CTS).

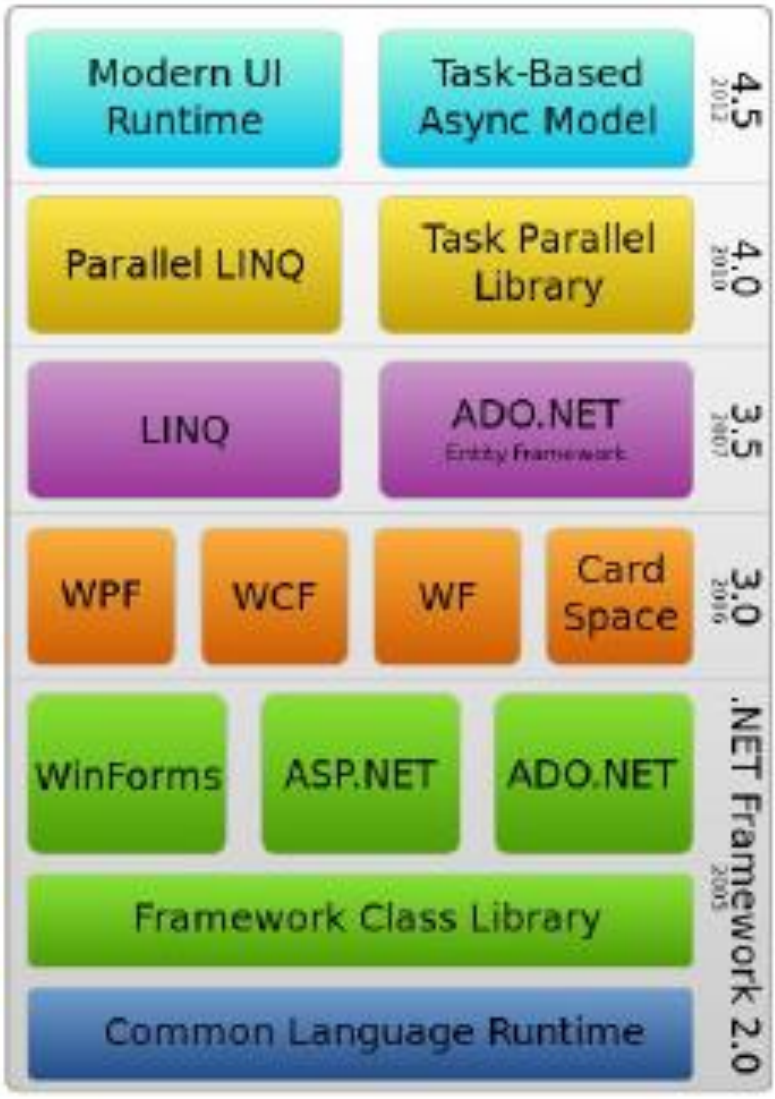


Рисунок 2.2 – Стек технологій .NET Framework

При компіляції коду, в якому використовується бібліотека .NET, він не перетворюється відразу ж в рідний код конкретної операційної системи. Замість цього він спочатку перетвориться в код MSIL/CIL. Цей код не є специфічним ні для якої-небудь операційної системи, ні для мови C#. Інші мови, наприклад, Visual Basic .NET, на першому етапі теж компілюються в код на цій мові. У разі розробки застосувань на C# такий процес компіляції виконується VS.

Очевидно, що далі для запуску застосування вимагається виконати ще деяку роботу. За це відповідає так званий JIT-компілятор (Just-in-Time compiler – оперативний компілятор), який компілює MSIL/CIL в рідний код, що відповідає вимогам конкретної операційної системи і архітектури цільового комп'ютера.

Тільки після цього етапу у операційної системи з'являється можливість запустити застосування. Абревіатура JIT в назві компілятора вказує на те, що він виконує компіляцію CIL-коду тільки при виникненні відповідної необхідності.

Однією з найбільш важливих функціональних можливостей керованого коду являється засіб збору сміття (garbage collection). Воно гарантує в .NET упевненість в повному очищенні використаної застосуванням пам'яті після завершення його експлуатації. До появи .NET про це здебільшого треба було піклуватися самим програмістам, і навіть декілька простих помилок в коді могли призводити до таємничого зникнення великих блоків пам'яті в результаті їх виділення в неправильному місці. Це зазвичай виливалося в поступове уповільнення роботи комп'ютера і зрештою завершувалося повним крахом системи.

Механізм складання сміття в .NET функціонує таким чином: він інспектує пам'ять комп'ютера час від часу і видаляє з неї все, що вже більше не треба. Ніяких встановлених часових рамок для виконання цієї операції немає; вона може відбуватися як тисячу разів в секунду, так і кожні декілька секунд або через будь-який інший проміжок часу, але в одному можна бути упевненим точно – вона обов'язково станеться.

Для програмістів це означає появу деяких наслідків. Через те, що ця операція виконується автоматично через непередбачувані проміжки часу, необхідно розробляти застосування з урахуванням цього факту. Код, що вимагає великої кількості пам'яті для виконання, повинен здійснювати очищення після себе самостійно, а не чекати того, коли її виконає механізм збору сміття, але забезпечується подібна поведінка набагато легше, ніж звучить.

Мова C# для платформи .NET ідеально підходить для цілей поставлених у роботі та є сучасною об'єкто-орієнтованою мовою програмування для широкого спектру цілей.

2.3 Огляд засобів проектування інтерфейсу користувача

Windows Forms - графічна система в складі .Net Framework. Являє собою обгортку навколо Win32 API в керованому коді. Вважається заміною графічної системи MFC, написаної під C ++ і має складну модель для розробки інтерфейсу програмного продукту.

Windows Presentation Foundation - це графічна система в складі .Net Framework 3.0 і пізніших версій. Спроектована під впливом технологій HTML і Flash і використовує апаратне прискорення.

Розробка графічного інтерфейсу для Windows-додатків грунтується на двох компонентах операційної системи:

– User32 - містить основні елементи вікна;

– GDI / GDI + - надає функціонал для малювання фігур, тексту.

Еволюція графічних систем пройшла шлях від складної системи MFC до розробки на мовах .Net і WinForms. Але хоч як мене удосконалювалися технології розробки, вони все одно спиралися на User32 і GDI / GDI +, а значить, не змогли подолати фундаментальні обмеження базових компонентів.

Для вирішення цієї проблеми була розроблена технологія WPF. Її головна відмінність від WinForms в тому, що для відображення графічних компонентів використовується не GDI / GDI +, а DirectX - технологія, що розробляється Microsoft для створення ігор в тісній співпраці з виробниками відеокарт, і з цієї причини має підтримку апаратного прискорення.

І хоч від компонента User32 не відмовилися, його використання зводиться до завдань які не стосуються відображення графічного інтерфейсу, наприклад, визначення положення вікна. Всі завдання, пов'язані з малюванням покладені на DirectX.

Перехід до DirectX являє собою величезний крок у розвитку і, апріорі, надає таке перевагу перед іншими графічними система як апаратне прискорення. Однак, WPF має і низку інших переваг, які так само підвищують його привабливість для розробників.

Розглянемо найбільш істотні зміни:

Апаратне прискорення. DirectX розроблявся у співпраці з виробниками відеокарт. Однак, просто потужної відеокарти недостатньо, необхідна програмна підтримка. Саме з цієї причини з 2004 року виробники відеокарт при написанні драйверів слідують з новими інструкціями від Microsoft.

Апаратна підтримка надається всім додаткам. У момент запуску аналізуються можливості відеокарти і на підставу отриманих результатів присвоюється рівень візуалізації від 0 до 2.

Декларований призначений для користувача інтерфейс. Ще одне нововведення в WPF це мова розмітки XAML, що є підмножиною XML. Хоча весь призначений для користувача інтерфейс можна описати на мові C # (або іншою мовою .Net), технологія WPF використовує інший підхід, завдяки якому інтерфейс описується на мові XAML, а поведінка програми записується в коді. Тепер дизайнер і розробник можуть працювати паралельно і не вникати в роботу один одного.

Стилі. В основу ідеї відтворення призначеного для користувача інтерфейсу лягла технологія HTML. Перехід до розробки на XAML дав можливість виводити властивості елементів вікон в окремі стилі, по аналоги з CSS. І тепер змінити зовнішній вигляд вікон стало її простіше, досить просто замінити файл зі стилями.

Модель малювання. Якщо в WinForms малювали пікселі, то в WPF працюють з примітивами - це вже готові базові фігури та інші графічні елементи. Крім того, є вбудована підтримка трехмерно графіки.

Анімація. У WinForm для того, щоб форма отрисовать себе, необхідно використовувати таймери. При розробці технології WPF в Microsoft пішли іншим шляхом і дали їй підтримку анімації.

Відео та аудіо. WPF працює з будь-яким форматом відео і аудіо, що підтримується програвачем Windows Media. Є можливо інтегрувати відеовміст в будь-які частини призначеного для користувача інтерфейсу. Вражаюче виглядає спільне використання тривимірної графіки і відео.

Команди. Також, як і в WinForms, в WPF присутній подієва модель, базові елементи були переписані і наділені новими можливостями, що також відбилося на події, і, тим не менш, принцип залишився тим самим. Іноді різні компоненти вікна викликають одне і теж дія, але яке викликається в різних обробниках подій. У WPF з'явився спосіб викликати реакцію елемента інтерфейсу через так звані команди. Команди підкоряються певним правилам, наприклад, назва методу закінчується на Command, яке опускається при виклику методу.

Додатки на основі сторінок. Можна будувати веб-подібні додатки на основі сторінок, що дає можливість користуватися навігацією.

Безпека. Раніше при розробці програми, в нього могли записати небезпечний код. Підвищення рівня відкритості та безпеки стали одними з причин появи мови XAML.

Незалежність від дозволу. WPF бере на себе компоновку елементів інтерфейсу підлаштовуючи його під різні дозволи.

Прив'язка даних. Прив'язати дані можна було і в WinForms, але в WPF цей механізм доведений до досконалості. Завдяки прив'язці істотно знижується кількість рядків коду і спрощує його аналіз.

Виходячи зі стратегії розвитку Microsoft, можна припустити, що технологія WPF буде розвиватися і далі. Наприклад, ця технологія використовується для створення програмного забезпечення в стилі Metro.

Якщо порівнювати графічні системи WinForms і WPF, то можна прийти до висновку, що друга система має величезну перевагу. Єдиним істотним гідністю WinForms можна назвати кроссплатформенность, такі додатки можна запустити на Mono. Про плани щодо підтримки WPF в системах сімейства Unix немає ніяких відомостей.

Але це не означає, що варто забути про технології WinForms і всі свої проекти переводити на WPF. Microsoft не повідомляла про припинення підтримки WinForms. Крім того, у неї є велика історія розвитку, а, отже, під неї написано безліч готових рішень. Проекти, в яких призначений для користувача інтерфейс стоїть не на першому місці, можуть бути написані і на WinForms.

У якості платформи для побудови інтерфейсу користувача обрана WPF. На сьогоднішній день вона є самою потужною, гнучкою платформою, що й продовжує розроблятися.

Основними перевагами WPF є:

– графічна технологія, що лежить в основі WPF - DirectX, на відміну від Windows Forms, де використовується GDI/GDI+;

– продуктивність WPF вище, чим в GDI+ за рахунок використання апаратного прискорення графіки через DirectX;

– можливість більш гнучко використовувати патерни проектування, для відділення процесу розробки інтерфейсу користувача від бізнес логіки додатка.

Одною з важливих вимог є відображення результатів моделювання у вигляді графіків, що дозволяє найбільше легко і ясно аналізувати й порівнювати результати моделювання. Існує кілька бібліотек, розроблених за допомогою WPF.

2.4 Огляд засобів інтерактивної візуалізації даних

На платформі WPF розроблено чимало засобів для відображення діаграм і графіків. Розглянемо більш докладно найбільш відомі.

Visibox Charts - ідеальний вибір для розробки додатків графічного відображення. Можливості програми:

1) Широкий вибір видів графіка. Лінії, колонки, смужки, драбинки, шматочки пирога. Повна підтримка всіх видів графіків буде додана пізніше.

2) Найвища продуктивність, зум і панорамування в реальному часі. При розробці програми одним з першорядних чинників була його продуктивність. Баланс продуктивності додатка на телефоні, де ресурси вкрай обмежені, набагато важливіше, ніж на комп'ютері або в інтернеті. Visibox Charts справляється з промальовуванням, зумом і панорамированием з легкістю.

3) Підтримка колекцій точок і видів графіків, оптимізованих для телефону. Точки графіка вибираються для телефону інакше, ніж для комп'ютера, адже натискання пальцем менш точний, ніж покажчиком миші. Обробка подій натискань в Visibox Charts оптимізована, навіть найменша точка на графіку може бути обрана користувачем при натисканні на неї або поруч з нею.

4) Легка настройка інтерфейсу програми. У додатку нова палітра і оформлення, що поєднується з оформленням платформи, кольору можуть налаштовуватися користувачем. Оформлення графіків, діаграм і ліній можна змінювати як завгодно за допомогою палітри.

5) Можливість розширення. В API додатка є базовий клас, що дозволяє легко побудувати розширення, а також інтерфейс, якщо базовий клас вам не підходить.

6) Підтримка множинних осей X і Y. Ви можете призначити ту кількість осей, яке вам потрібно, і побудувати складний графік прямо на телефоні.

7) Silverlight надає графічну систему, схожу з Windows Presentation Foundation, і об'єднує мультимедіа, графіку, анімацію і інтерактивність в одній програмній платформі. Вона була розроблена, щоб працювати з XAML і з мовами .NET. XAML використовується для розмітки сторінок, що використовують векторну графіку і анімацію. Текст, що міститься в додатках Silverlight, доступний для пошукових систем, так як він не компілюється, а доступний у вигляді XAML. Silverlight також можна використовувати для того, щоб створювати віджети для Windows Sidebar в Windows Vista.

8) Silverlight може відтворювати WMV, WMA і MP3 для всіх підтримуваних браузерів, не вимагаючи при цьому додаткових компонентів, таких як Windows Media Player. Так як Windows Media Video 9 є реалізацією стандарту SMPTE VC-1, Silverlight підтримує відео VC-1 тільки всередині контейнера ASF. Крім того, ліцензійна угода говорить, що VC-1 дозволено використовувати тільки в особистих, некомерційних цілях («personal and non-commercial use of a consumer»). Silverlight дозволяє динамічно завантажувати XML і використовувати DOM для взаємодії з ним так само, як це робиться в Ajax. Silverlight містить об'єкт Downloader, завдяки якому можна завантажувати скрипти, медіа файли і т. Д., Якщо це необхідно з додатком. Починаючи з версії 2.0, логіка програми може бути описана в будь-якому з мов .NET, включаючи динамічні мови програмування такі як Iron Ruby і Iron Python, які, в свою чергу, виконуються в DLR (Dynamic Language Runtime), а не CLR (Common Language Runtime).

Таким чином, для відображення графіків були обрані контроли Visiblox, тому що вони мають ясний і простий API, просту розмітку, не вимагають багато коду для настроювання, і мають високу продуктивність, що важливо при розробці систем моделювання реального часу.

2.5 Вибір апаратного забезпечення

Характеристики персонального комп’ютера обумовлені застосуванням операційної ситсеми Windows 7 та середовища програмування Windows 7:

1) комп'ютер з частотою процесора 2.0 ГГц або швидше;
2) оперативною пам’ятью 2 GB (32 Bit) або 4 GB (64 Bit);
3) 4Гб доступного місця на жорсткому диску
5) DirectX 11 сумісна відеокарта, що працює з 1024 x 768 або більшим дозволом дисплеєм

Висновки: у розділі розглянуті сучасні засоби розробки прикладного програмного забезпечення та обрані компоненти, придатні для реалізації поставленого в магістерській роботі завдання:

- операційна система Windows 7.
- мова програмування C# на базі платформи .Net Framework 4.0.
- фреймворк для побудови інтерфейсу користувача WPF 4.0 .
- бібліотека Visiblox – для відображення графіків.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОДЕЛЕЙ

У роботі [49] для реалізації моделювання універсальним способом технологічної системи (ТС) пропонується метод інформаційної технології, відповідно до якого структурна модель ТС складається з набору блоків, для кожного з яких задається обмежена й незмінна кількість входів і виходів. Принцип організації вхідної й вихідної інформації полягає у зв'язку між набором вхідних і вихідних параметрів.

Відповідно до поставленого технічного завдання й обраними засобами розробки, а також відповідно до методу [49] реалізоване програмне забезпечення, яке має наступні властивості:

– простий графічний інтерфейс;

– модульна архітектура (можливість швидкого підключення компонентів системи);

– повинна виконуватися у реальному часі - швидкість виконання моделювання, швидше ходу реального часу;

3.1 Розробка підсистеми моделювання

У наступному розділі розглядаються питання по динамічному виявленню й підключенню модулів без перекомпіляції програми, реалізації способу об'єднання модулів шляхом з'єднання вихідних параметрів із вхідними, реалізації основного процесу моделювання.

3.1.1 Виявлення та підключення модулів

Розглянемо реалізацію системи виявлення й підключення модулів.

Для забезпечення гнучкості при розробці й використанні зовнішніх модулів, необхідно забезпечити слабку зв'язність компонентів системи. Для цього необхідно виділити вузький, єдиний і універсальний інтерфейс для роботи з об'єктами, який є загальним для всіх модулів, тобто кожний модуль повинен реалізувати його, щоб бути виявленим при завантаженні. Таким є інтерфейс IModule:

```
public interface IModule : IDisposable
```

```
{
    #region Public Methods
    void Run();
    #endregion
}
```

Метод Run дозволяє виконати один цикл роботи модуля в одиницю часу. Усі вхідні (у тому числі й час) і вихідні параметри задаються за допомогою атрибутів, що позбавляє від громіздкості та написання зайвого коду. Час є необхідним вхідним параметром практично для всіх моделей, але в принципі реалізація модуля може й не вимагати його, як обов'язкового параметра. Нижче наведені визначення основних атрибутів:

```
[Attributeusage(Attributetargets.Property, Inherited = false, Allowmul
tiple = false)]
public sealed class Inputattribute : Attribute
{
}
[Attributeusage(Attributetargets.Property, Inherited = false, Allowmul
tiple = false)]
public sealed class Outputattribute : Attribute
{
}
[Attributeusage(Attributetargets.Property, Inherited = false, Allowmul
tiple = false)]
public sealed class Timeattribute : Attribute
{
}
```

Приклад оголошення вхідного параметра – властивості часу:

```
[Time] public Tick Time { get; set; }
```

Таким чином, можна повністю описати модуль із усіма його вхідними та вихідними параметрами.

Для підключення модуля до системи моделювання досить просто скопіювати його в папку, зазначену в конфігураційному файлі. Система сама знайде його і підключить при запуску.

Приведемо приклад секції конфігураційного файлу:

```
  <blackboxconfig>
    <custompaths>
      <add key="modules" path=".\Modules"/>
      <add key="core" path=".\Blackbox.Core.dll"/>
      <add key="service" path=".\Blackbox.Service.dll"/>
      <add key="infrastructure" path=".\Blackbox.Infrastructure.d
ll"/>
      <add key="desktop" path=".\Blackbox.Desktop.dll"/>
    </custompaths>
  </blackboxconfig>
```

У цьому випадку модулі розташовуються в підпапці Modules. Модулем є .Net збірка, тобто будь-який *.exe або *.dll файл.

3.1.2 Взаємодія та об'єднання модулів

Відповідно до [49] система моделювання повинна дозволяти зв'язувати модулі між собою, тобто з'єднувати набір вихідних параметрів із вхідними, що дозволить збирати цільову систему із блоків (модулів).

Як сказано вище, інтерфейс модуля містить тільки метод для запуску його на виконання, а сам клас модуля – набір вхідних і вихідних параметрів. Для об'єднання всіх відомостей і даних про модуль застосовується інтерфейс IBlackboxModule, який є декоратором [52] для інтерфейсу IModule і містить усередині себе посилання на нього (IBlackboxModule є внутрішнім інтерфейсом системи). Нижче показані основні властивості й методи цього інтерфейсу:

```
public interface IBlackboxModule: IDisposable
    {
        #region Properties
        Ienumerable<IConnection> InputConnections { get; }
        Ienumerable<ModulePropertyInfo> Inputs { get; }
        string Name { get; set; }
        Ienumerable<IConnection> OutputConnecitons { get; }
        Ienumerable<ModulePropertyInfo> Outputs { get; }
        #endregion
        #region Public Methods
        void Run();
        void Settime(Tick tick);
        #endregion
    }
```

Таким чином, є інтерфейс, який повністю описує модуль і звертання до самого модуля відбувається тільки через цей інтерфейс. Inputs, Outputs – це колекція входів і виходів відповідно (ті, які описуються за допомогою атрибутів). ModulePropertyInfo – клас обгортка над властивістю класу, який містить додаткову інформацію про властивість (вхід або вихід).

InputConnections, OutputConnection – колекція вхідних і вихідних зв'язків. Зв'язок (connection) – це механізм, який дозволяє поєднувати виходи із входами модулів. Нижче показаний інтерфейс зв'язків IConnection:

```
public interface IConnection
    {
```

```
        #region Properties
        bool Isdelayed { get; }
        Iblackboxmodule Sourcemodule { get; }
        Modulepropertyinfo Sourcepropinfo { get; }
        Iblackboxmodule Targetmodule { get; }
        Modulepropertyinfo Targetpropinfo { get; }
        #endregion
        #region Public Methods
        void Detach();
        void Update();
        #endregion
    }
```

Як видно інтерфейс містить інформацію, необхідну для створення зв'язку між властивостями двох модулів: Sourcemodule – модуль джерело, Sourcepropinfo – вихід модуля джерела, Targetmodule – модуль приймач, Targetpropinfo – вхід модуля приймача, а також методи для видалення зв'язку й відновлення входу одного модуля, при зміні виходу іншого.

Логіка відновлення інформації на вході може бути двох видів: пряма й зворотна. Пряма – коли інформація на вході обновляється відразу ж при відновленні інформації на виході. Зворотна – коли інформація на вході обновляється не відразу, а на наступному кроці ітерації, тобто із затримкою (такі зв'язки називаються зворотними). Ці 2 типу зв'язків представлені класами BridgeConnection і DelayedConnection.

Таким чином, створивши зв'язок між двома модулями більше немає необхідності стежити за змінами виходів, клас зв'язку сам стежить за цим і обновляє відповідні входи модулів при необхідності.

Задавати значення на входах модулів можна декількома способами. У простому випадку значення на вхід може надходити з виходу якого-небудь іншого модуля. У випадку, коли необхідно подавати яке-небудь певне значення, можна реалізувати власний модуль, наприклад який реалізує певну функцію, і потім використовувати його як звичайний модуль у системі. Інший варіант – задання значень за допомогою гістограми. Вона реалізоване так само – як окремий модуль, але в налаштуваннях містить не список входів і можливість завдання зв'язків, а таблицю, у яку заносяться пари: час/сначення. Цей спосіб є універсальним і дозволяє легко визначати вхідні дані для модулів.

3.1.3 Основна підсистема моделювання

Розглянемо основний алгоритм процесу моделювання. Можливі два підходи для розв'язку задачі процесу моделювання:

Всі модулі, готові до виконання запускати на виконання кожний в окремому потоці – асинхронно. Це дозволить уникнути затримок пов'язаних із синхронністю, тобто необхідністю очікування декількома модулями, готовими до виконання, одного модуля, який вимагає великої кількості часу для виконання.

Запускати модулі на виконання синхронно. Це дозволить уникнути накладних витрат із синхронізацією потоків, але може виникнути ситуація, коли все чекають одного.

Незважаючи на те, що на перший погляд перший підхід здається більш оптимальним, обран другий підхід - синхронний. При асинхронному алгоритмі керування часом виконання модулів, виникають неминучі витрати й витрати часу на створення й знищення потоків, що може серйозно вплинути на продуктивність системи. Наприклад, якщо в системі буде багато модулів, які самі по собі виконуються дуже швидко, то в цьому випадку кількість часу, витрачене на створення й знищення потоків, виявиться значно більше часу виконання самих модулів.

Синхронний підхід позбавлений цього недоліку, усі модулі виконуються в одному потоці по-черги. Але у випадку, коли час виконання одного модуля може бути набагато більше часу виконання інших, має сенс зробити реалізацію самого модуля багатопотоковою. Таким чином, буде досягнута максимальна швидкодія виконання.

Уся інформація про модельовану систему: підключені модулі та їх зв'язки - знаходяться в класі Blackboxservice, він же містить логіку по реалізації процесу моделювання й валідації вхідних даних.

Нижче наведений основний алгоритм одного циклу моделювання:

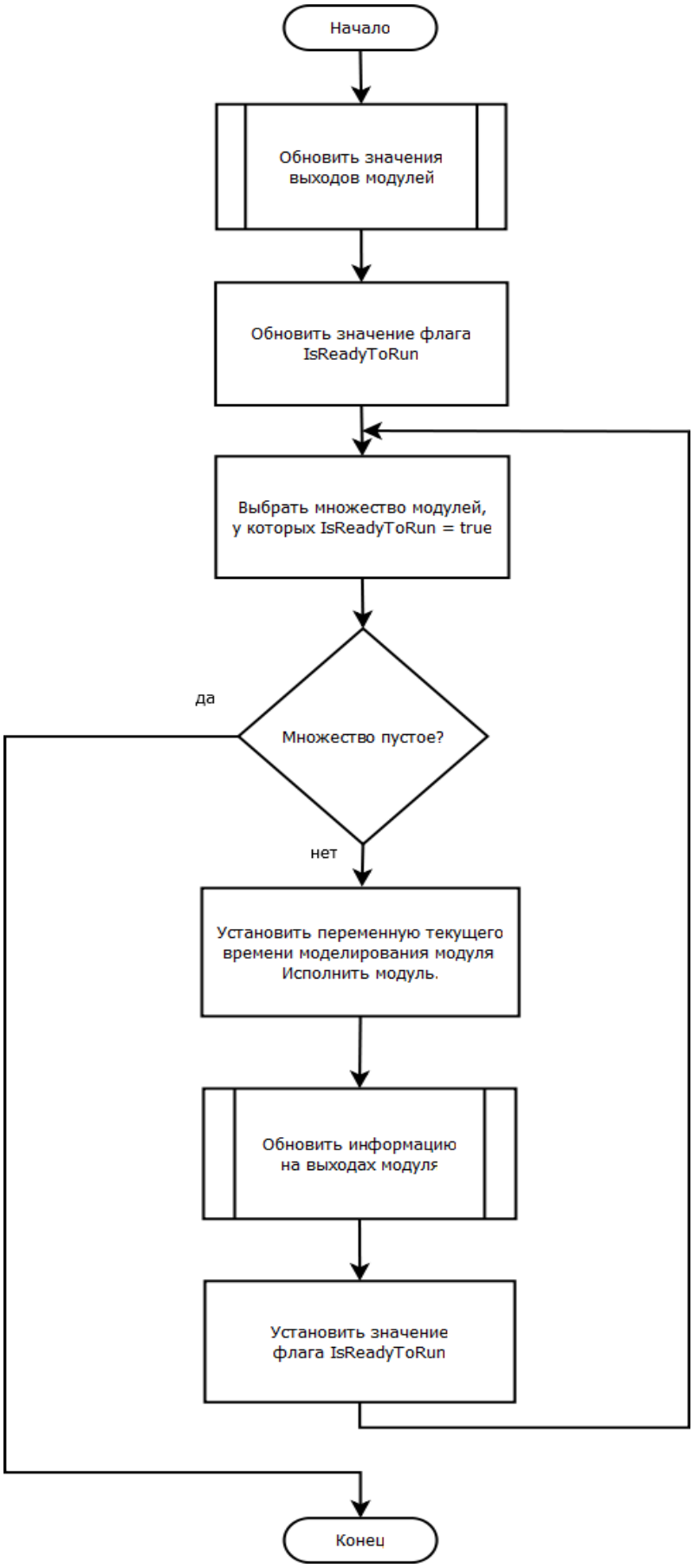


Рисунок 3.1 – Блок-схема алгоритму одного циклу моделювання

Нижче приведений основний метод, який реалізує логіку перебору модулів, запуску їх на виконання й відновлення вихідної та вхідної інформації:

```
public void Run(Tick tick)
        {
            this.modules.Foreach(m => m.Reset());
            var workset = this.modules.Tolist();
            while (workset.Count > 0)
            {
                var readytorunmodules = workset.Where(m => m.Isreadyto
run).Tolist();
                readytorunmodules.Foreach(m =>
                    {
                        m.Settime(tick);
                        m.Run();
                    });
                workset = workset.Except(readytorunmodules).Tolist();
            }
        }
```

Спочатку для всіх модулів викликається метод Reset, який виконує відновлення вихідних зв'язків (це необхідно для спрацьовування зворотних зв'язків) і визначає, чи готовий модуль до виконання. Ознакою готовності є оновлена інформація на всіх входах модуля. Потім у модулів, готових до виконання викликається метод Run, після якого, обновляється інформація на виходах модуля. Так триває доти, поки всі модулі не будуть виконані. Таким чином, відбувається виконання одного циклу моделювання.

Одним з важливих моментів є принцип відновлення інтерфейсу користувача під час моделювання. Проблема в тому, що і графічний інтерфейс, і моделювання виконуються в одному потоці. Тому, якщо виконувати процес моделювання відразу повністю, не даючи при цьому інтерфейсу обновити себе, то додаток буде як би висіти й не реагувати на натискання. Для цього застосовується підхід, схожий на той, який застосовується в комп'ютерних іграх. Зміст полягає в тому, що виконувати частину процесу моделювання необхідно по спрацьовуванню таймера. Тобто таймер спрацьовує з певною періодичністю (позначимо як timespan). І по спрацьовуванню таймера запускається процес моделювання на певну кількість циклів (позначимо як tickperiod). У підсумку, у той період часу, поки таймер не спрацював, у потоці буде виконуватися логіка по обробці подій і відновленню інтерфейсу користувача. Після спрацьовування таймера виконується частина часу моделювання й потім знову запускається таймер. Таким чином, інтерфейс користувача буде періодично обробляти події й обновлятися й зникне ефект завислого додатка.

Дві величини: timespan і timeperiod задаються в конфігураційному файлі:

```
<configuration>
      <blackboxconfig>
            <tickperiod value="50"/>
            <timespan value="50"/>
      </blackboxconfig>
</configuration>
```

Для різних ситуацій моделювання можна підібрати такі значення, щоб і інтерфейс залишався чуйним і процес моделювання йшов досить швидко. Через специфіку різних технологічних процесів і швидкості виконання модулів значення по-умовчанню не завжди можуть забезпечити прийнятні результати.

У перспективі дана система може бути реалізована як розподілена. Наприклад, можна забезпечити можливість установки кілька копій програм на різних комп'ютерах і потім об'єднати їх у єдину систему. Даний підхід дозволить підвищити швидкодію за рахунок використання більших обчислювальних ресурсів.

Висновки: Відповідно до поставленого завдання розроблені алгоритми та засоби, які забезпечують модульність та швидкодію системи моделювання. Розглянуто перспективи подальшої розробки.

Розроблена підсистема має наступні властивості:

Дозволяє автоматично виявляти й підключати модулі.

Дозволяє задавати зв'язки між вхідними й вихідними параметрами модулів та виконувати основну логіку з моделювання процесів.

3.2 Розробка інтерфейсу користувача

Інтерфейс користувача являє собою сукупність засобів і методів, за допомогою яких користувач взаємодіє з безліччю різних, найчастіше складних, елементів, машин і обладнань. У розділі розглядається метод побудови й реалізації інтерфейсу користувача, а також проводитися огляд інтерфейсу розробленої системи моделювання.

Враховуючи, що інтерфейс користувача представляє систему та повинен задовольняти непередбаченим стилістичним вимогам користувачів, він може бути найбільш непостійною частиною багатьох додатків. А тому що основною технологією в роботі для розробки інтерфейсу користувача є WPF, то він спроектований з використанням шаблону проектування Model-view-viewmodel (MVVM). Розглянемо цей шаблон докладніше [37].

3.2.1 Шаблон проектування Model-view-viewmodel

Популярні шаблони проектування спрощували людям життя з перших кроків створення інтерфейсу користувача програм. Наприклад, шаблон модель-вистава-презентатор (MVP) була популярна на різних платформах програмування інтерфейсу користувача. MVP — це різновид шаблону модель-вистава-контролер, якому вже кілька десятків років. Те, що видне на екрані, — це вистава; дані, які там відображені — це модель, а презентатор поєднує їх разом. Вистава потребує презентатор для заповнення даними моделі, реакції на введення користувача, надання перевірки введення (у тому числі за рахунок передачі цієї функції моделі) і інших подібних завдань.

В 2004 Мартін Фаулер (Martin Fowler) опублікував статтю про шаблон за назвою Модель презентації (PM). Шаблон «модель презентації» схожий на MVP у тому плані. що він відокремлює виставу від його поведінки й стану. Цікава частина шаблону PM у тому, що створюється абстракція вистави, яка називається моделлю презентації. Вистава, таким чином, стає просто результатом обробки моделі презентації. Згідно Фаулеру, модель презентації постійно обновляє своя вистава, тому вони залишаються синхронізованими один з одним. Ця логіка синхронізації існує у вигляді коду в класах моделі презентації.

В 2005 році Джон Госсман (John Gossman), зараз один з архітекторів WPF і Silverlight у корпорації Microsoft, розповів у своєму блогу про шаблон модель-вистава-модель вистави (MVVM). MVVM збігається з моделлю презентації Фаулера в тому плані. що обоє шаблону містять абстракцію вистави, що містить стан і поведінка вистави. Фаулер увів модель презентації як спосіб створення незалежного від платформи інтерфейсу користувача абстракції вистави, а Госсман запропонував MVVM як стандартизований спосіб використовувати основні функції WPF для спрощення створення інтерфейсу користувача. У цьому змісті MVVM можна вважати приватним варіантом більш загального шаблону PM, пристосованим для платформ WPF і Silverlight.

На відміну від презентатора в MVP, модель вистави не має потреби в посиланні на виставу. Вистава прив'язується до властивостей моделі вистави, яка, у свою чергу, представляє дані в об'єктах моделі й інших станах, потрібних для цієї вистави. Прив'язки між виставою й моделлю вистави створювати легко, тому що об'єкт моделі вистави встановлюється як контекст DataContext вистави. Якщо змінюються значення в моделі вистави, ці нові значення автоматично переходять у виставу через прив'язку даних. Коли користувач натискає кнопку у виставі, для добутку потрібної дії виконується команда в моделі вистави. Усі зміни даних моделі завжди робить модель вистави, а не вистава.

Класи вистави не знають про існування класів моделі, а модель вистави й модель не знають про виставу. Модель насправді взагалі не має вистави про те, що існують модель вистави й вистава. Це дуже слабко зв'язана конструкція, і це дає ряд переваг, про яких я незабаром розповім.

Якщо розроблювач звик до WPF і MVVM, ці останні стає важко розрізнити. MVVM — своєрідна загальноприйнята мова розроблювачів WPF, тому що він добре пристосований для платформи WPF, а WPF створювався для спрощення складання додатків за допомогою шаблону MVVM (і інших). У корпорації Майкрософт MVVM використовувався для внутрішніх цілей при розробці додатків WPF, наприклад Microsoft Expression Blend, поки основна платформа WPF ще тільки створювалася. Багато частин WPF, наприклад модель контролю без перегляду й шаблони даних, використовують значний поділ показу від стану й поведінки, застосовуване в MVVM.

Найважливіший момент WPF, що робить MVVM дуже зручним шаблоном, – це інфраструктура прив'язки даних. За рахунок прив'язки властивостей вистави до моделі вистави виходить слабке зв'язування цих компонентів, що повністю звільняє розроблювача від необхідності писати в моделі вистави код, що безпосередньо обновляє вистава. Система прив'язки даних підтримує також перевірку допустимості введення, що забезпечує стандартний шлях передачі помилок перевірки допустимості виставі.

Ще дві функції WPF, що роблять цей шаблон таким корисним, – це шаблони даних і система ресурсів. Шаблони даних застосовують вистави до об'єктів моделі вистави, показаним в інтерфейсі користувача. Можна оголосити шаблони в коді XAML і дозволити системі ресурсів за вас автоматично знаходити й застосовувати ці шаблони під час виконання. Якби в WPF не було підтримки команд, шаблон MVVM не був би таким універсальним. Крім функцій WPF (і Silverlight 2), за рахунок яких MVVM стає природнім способом структурування додатка, цей шаблон популярний ще й тому, що класи моделі вистави легко піддаються модульному тестуванню. Якщо логіка взаємодії додатка перебуває в наборі класів моделі вистави, легко написати тестуючий її код. У якімсь змісті вистави й модульні тести — різні типи споживачів моделі вистави. Набір тестів для моделі вистави додатка забезпечує вільне й швидке регресійне тестування, що зменшує вартість підтримки додатка в майбутньому.

Крім зручності створення автоматичних регресійних тестів, тестованость класів моделі вистави може допомогти правильно проектувати інтерфейси користувача, для яких легко робити теми оформлення. Проектуючи додаток, часто можна розв'язати, чи помістити щось у виставу або в модель вистави, представивши собі, чи потрібне буде писати модульний тест, що споживає модель вистави. Якщо ви можете написати модульні

тести для моделі вистави, не створюючи об'єкти інтерфейсу користувача, можна також повністю укласти модель вистави в тему оформлення, тому що в неї немає залежностей від певних візуальних елементів. Нарешті, для розроблювачів, що співробітничають із проектувальниками візуальних форм, використання MVVM спрощує створення безперервного потоку робіт проектувальника й розроблювача. Тому що вистава — не більш ніж необов'язковий споживач моделі вистави, неважко забрати одна вистава й замінити його для відображення моделі вистави новою.

3.2.2 Огляд інтерфейсу програми

Розглянемо складові графічного інтерфейсу розробленої системи моделювання. Інтерфейс складається з 3-х основних вікон:

– головне вікно. На ньому розташовуються доступні модулі, додані модулі й вікно таймера;

– вікно настроювань модуля. На ньому розташовується редактор зв'язків або редактор компонента «гістограма», яка дозволяє задавати значення входів модулів у ручному режимі;

– вікно, де відображаються значення виходів модулів у графічному виді (тобто графіки). Відразу надається можливість відключити деякі виходи, щоб інформація з них не виводилася.

Тепер розглянемо кожне вікно докладніше.

Головне вікно емулятора.

Головне вікно зображено на рисунку 3.2.

Угорі праворуч виводитися список усіх завантажених модулів, які можуть бути додані до загальної моделі. Додати модуль можна подвійним кліком або вибравши модуль і нажавши кнопку Add.

Ліворуч розташовується грид, у якому знаходяться додані модулі. У рядку з обраним модулем розташовуються кнопки для видалення й виклику меню редагування модуля. Наприкінці розташовується checkbox, який відкриває вікно для відображення графіків.

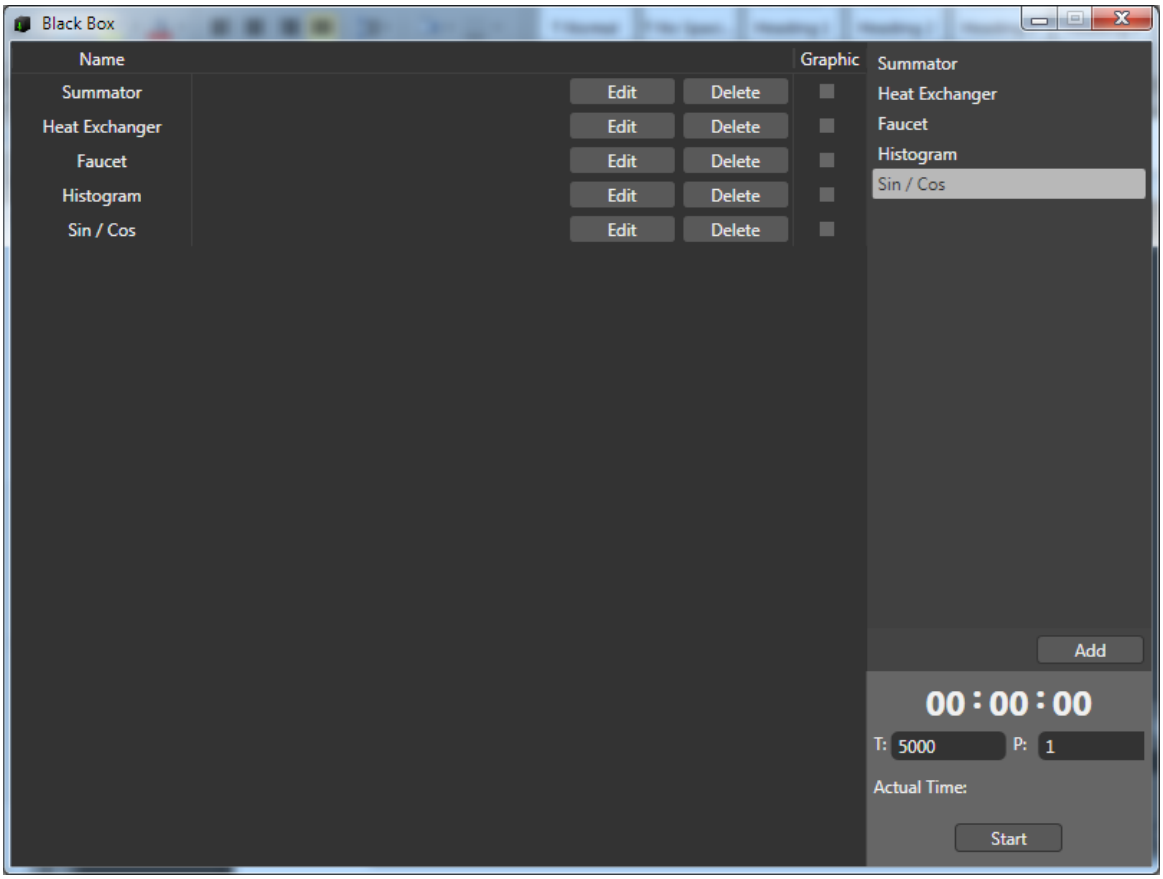


Рисунок 3.2 - Головне вікно програми.

Унизу розташовується таймер. На ньому відображається загальний час моделювання, кількість тиків (T: Ticks – дозволяє задати час моделювання), точність таймера (P: Precision) і актуальний час (Actualtime) – кількість реального часу, який був реально витрачен на моделювання. Таким чином, можна довідатися, як швидко працює модель.

Вікно з налаштуваннями модуля.

По натисканню на кнопку Edit – з'являється вікно, що дозволяє редагувати модуль, тобто задавати зв'язки з іншими модулями або у випадку з гістограмою – задавати їй значення у вигляді таблиці. Вікно настроювань наведено на рисунках 3.3 і 3.4.

Вікно налаштувань містить наступні елементи:

Textbox зверху – назва модуля.

Таблиця з редактором зв'язків. У кожному рядку таблиці знаходяться налаштування вхідних параметрів модуля.

Selectmodule – список з доступними модулями, з якими можна з'єднати поточний.

Selectoutput – список з вихідними параметрами обраного модуля.

Delay – опція, що дозволяє вибрати тип зв'язку: прямий або зворотній.

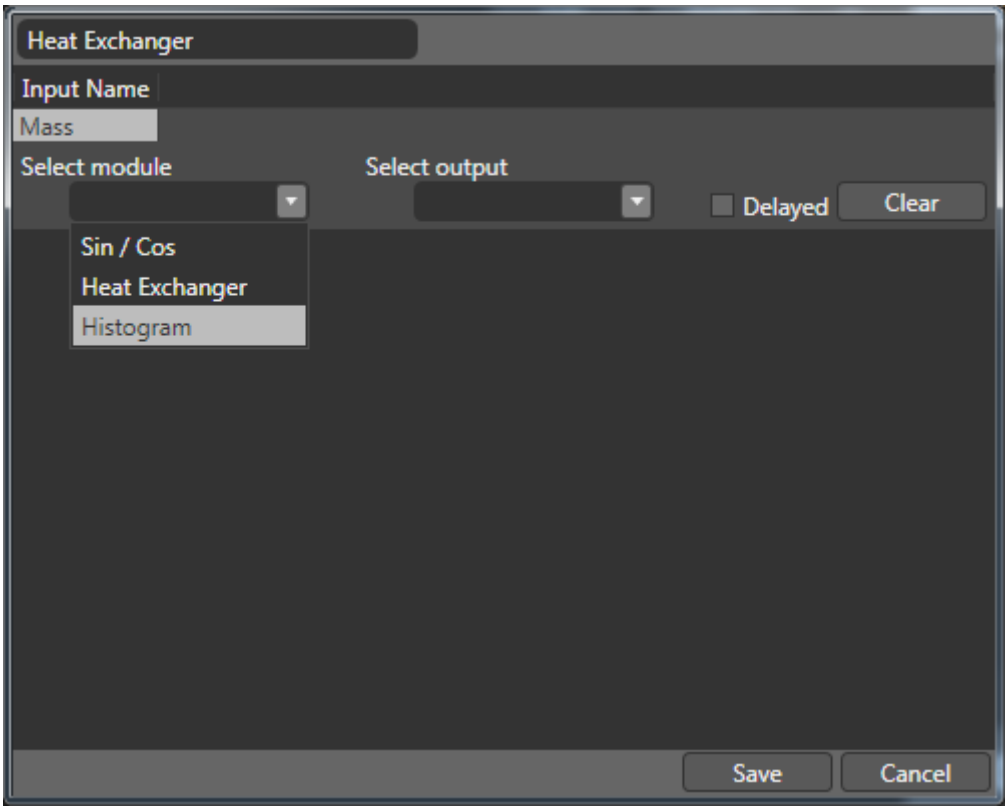


Рисунок 3.3 – Вікно налаштування модуля

Clear – очищає описані поля, тобто видаляє зв'язок між вхідними/вихідними параметрами модулів.

Таким чином, можна гнучко задавати різні типи зв'язків між модулями в табличному вигляді.

Вікно настроювань гістограми дозволяє задавати значення в табличному виді у формі: Час/сначення. Гістограми корисно використовувати для завдання вхідних параметрів модулів і імітувати зміну вхідних значень із часом.

Вікно відображення графіків.

Вікно із зображеннями графіків зображене на рис. 3.5.

Зверху втримується набір опцій для включення або відключення можливості стежити за змінами вихідних параметрів модуля (Sin, Cos).

Масштабування можливе за допомогою миші й коліщати.

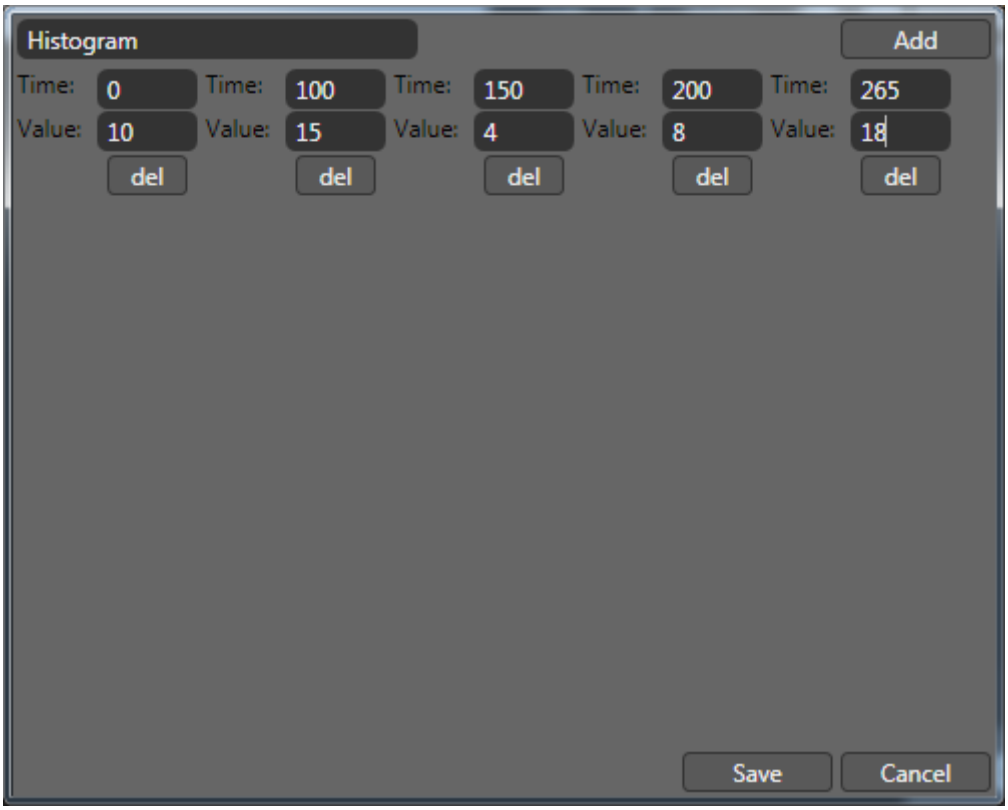


Рисунок 3.4 – Вікно установки модуля «Гістограма»

Висновок: Для системи моделювання розроблений простий і зрозумілий графічний інтерфейс, що дозволяє виконати вимоги, поставлені в технічному завданні:

Простота, відсутність надлишкової складності та функцій.

Організація й робота з модулями в табличній формі.

Зручні засоби для відображення й маніпулювання графіками.

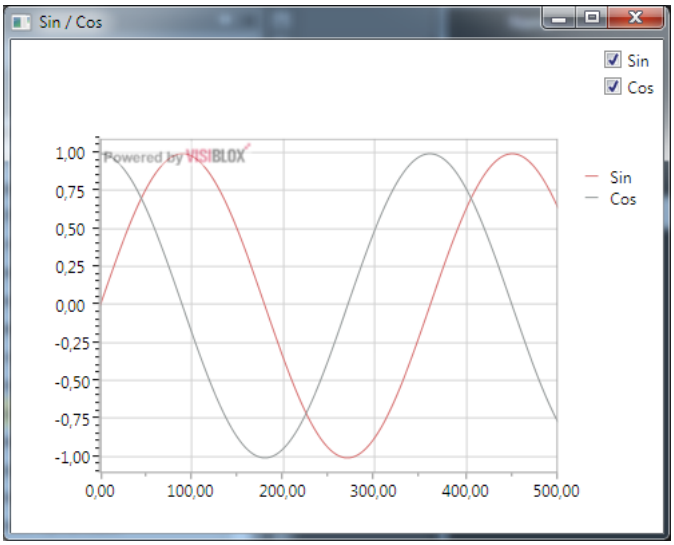


Рисунок 3.5 – Вікно для виведення графіків

4 ТЕСТУВАННЯ СИСТЕМИ ТА АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

У даному розділі проводиться аналіз, тестування різних конфігурацій системи й огляд отриманих результатів. Оцінюється середній час виконання процесу моделювання. Робляться висновки про придатність системи.

4.1 Перевірка працездатності системи.

Перевіримо адекватність системи моделювання на прикладі моделі теплообмінника та порівняємо результати з даними, отриманими в системі mathcad. Результати моделювання зображено на рисунку 4.1, а результати, отримані в Mathcad – на рисунку 4.2.

Вхідні дані: витрата – 1.324кг/с, час – 50с.

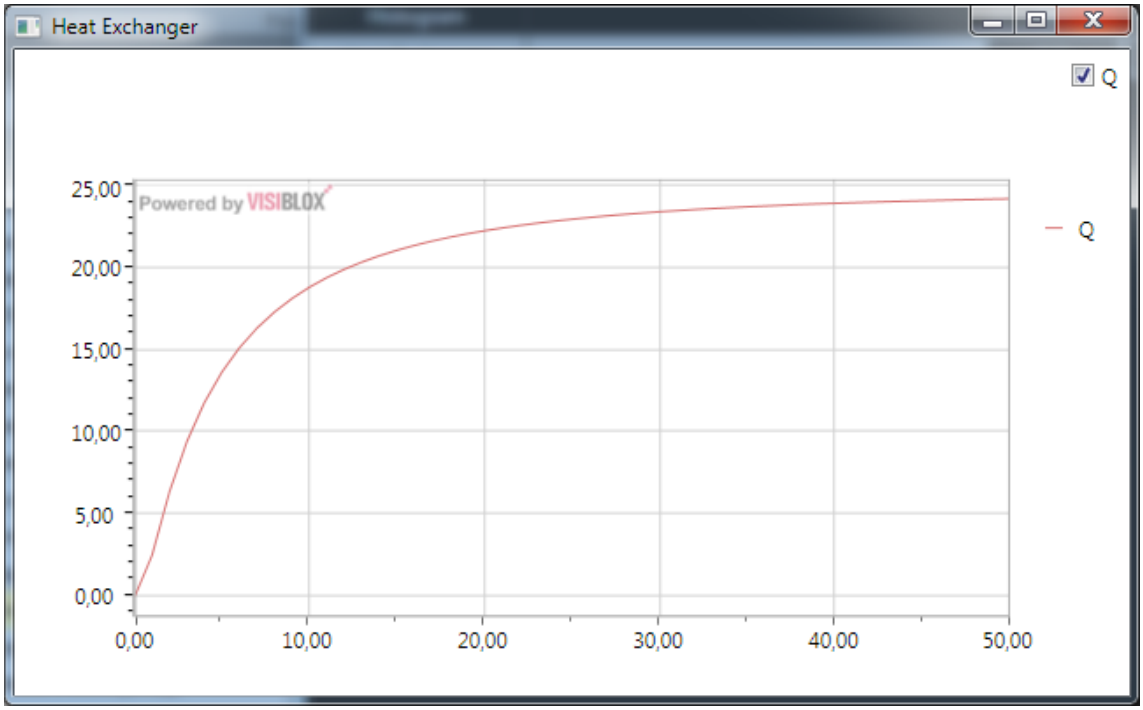


Рисунок 4.1 – Графік процесу теплообміну в теплообміннику

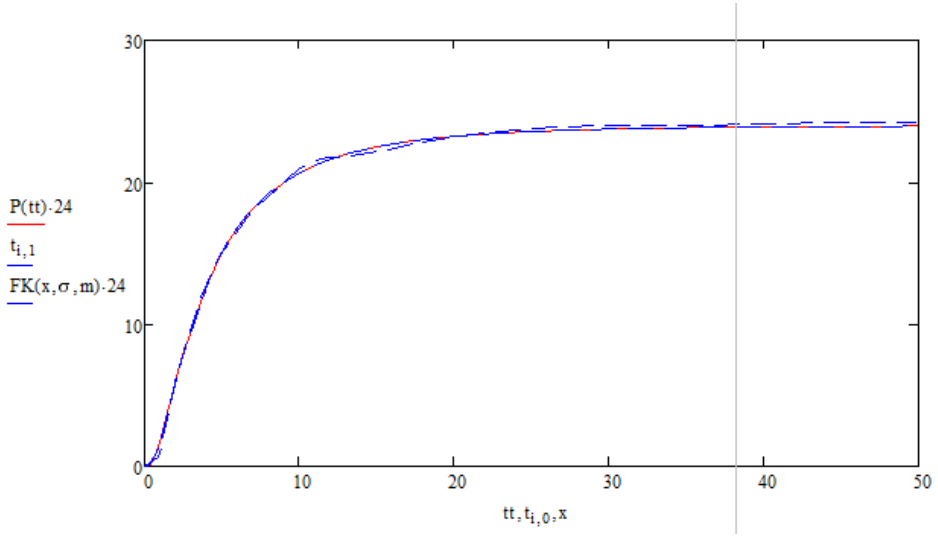


Рисунок 4.2–Графік процесу теплообміну в теплообміннику на прикладі Mathcad

Із графіків видно, що система показує ідентичні результати в обох випадках, що говорить про вірну реалізацію модуля теплообмінника.

Перевірка правильності зміни поведінки перехідного процесу при підвищенні витрати речовини.

Нехай витрата зростає під час T = 20с від m=1.324кг до m=1.45кг (рис 4.3)

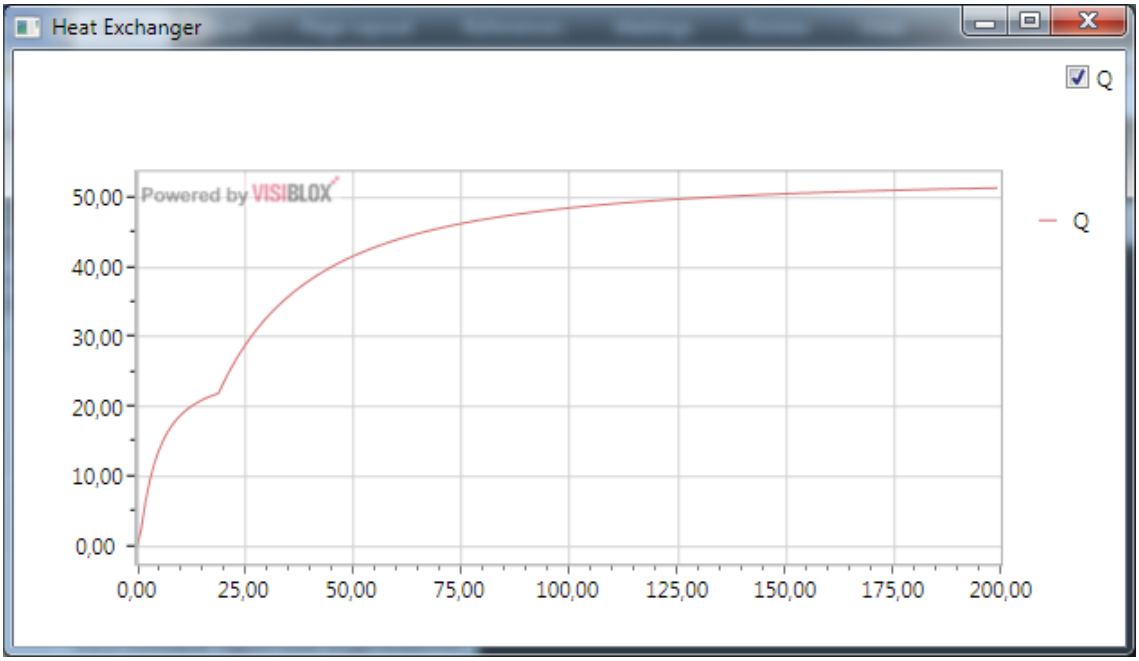


Рисунок 4.3 – Графік процесу теплообміну при зростанні витрати речовини.

Як видно відбувся стрибок і значення Q, при якому процес теплообміну вирівнюється, збільшилося (через збільшення витрати).

Оцінимо швидкість моделювання й ступінь впливу підсистеми зв'язки модулів і передачі значень між входами й виходами на загальний час моделювання.

Необхідно додати кілька модулів у систему. Наприклад приєднаємо два теплообмінники паралельно та об’єднаємо значення їх виходів через суматор (рис.4.4)

m1 = 1.324кг, m2 = 1.45кг, час моделювання – 200 секунд.

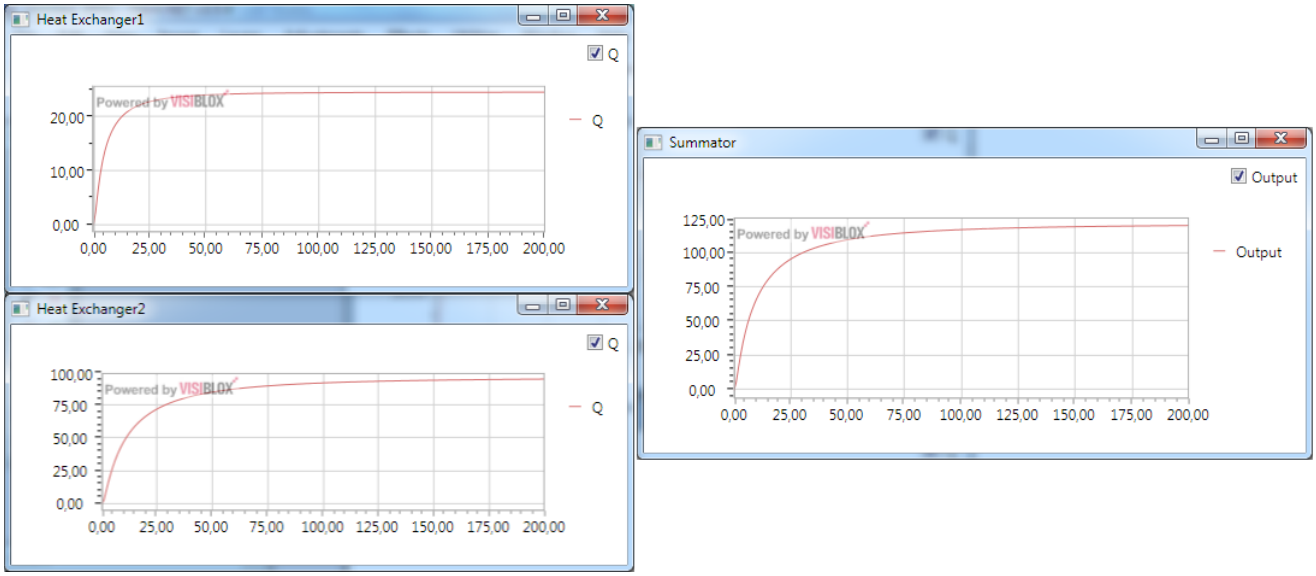


Рис 4.4 – Графіки вихідних значень двох теплообмінників і суматора.

Проведемо кілька тестів та виберемо середнє значення по них. Результати представлено в таблиці 4.1.

Таблиця 4.1 – Середній час виконання моделі із двома теплообмінниками і суматором

N тесту	Час (мс)	
1	296	
2	281	
3	267	
4	286	
5	292	
	Середній час	284

Таким чином, 200 секунд модельного часу в середньому рівні 284 мілісекундам реального часу. А це значить що система працює швидше реального ходу часу.

Оцінимо роботу системи в режимі без виводу графіків.

Таблиця 4.2 – Середній час виконання моделі із двома теплообмінниками й суматором у режимі без виводу графіків

N тесту	Час (мс)	
1	47	
2	57	
3	69	
4	48	
5	62	
	Середній час	56

В результаті одержали час, який у 5 разів менше часу моделювання з виводом графіків на екран. Із цього можна зробити висновок, що більшу частину часу все-таки займає перемальовування й рендеринг графіків на інтерфейсі користувача. Тому, якщо потрібна максимальна продуктивність, можливо має сенс провести моделювання в режимі без виводу графіків, а потім вивести отримані результати наприклад у файл. Це дозволить прискорити загальний час моделювання.

Висновки: провели тестування розробленої системи. Визначили що система коректно знаходить, завантажує та підключає модулі, дозволяє з’єднувати модулі між собою (вхідні та вихідні параметри). Провели тестування розробленого модуля теплообмінника. Виявили та означили середні швидкості виконання моделювання системи, яка складається с декількох модулей.

Таким чином, розроблена система повністю задовольняє вимогам дипломного проекту та має достатню швидкодію. Час, витрачений на обробку модулів і вибудовування їх у чергу на виконання багато менше, часу виконання самих модулів і тому практично не позначається на загальному часі моделювання.

4.2 Короткий посібник користувача

Нижче наведений посібник користувача з експлуатації та установки розробленого програмного забезпечення.

4.2.1 Посібник з експлуатації

Для додавання нових модулів потрібно скопіювати їх у папку й указати шлях до папки в конфігураційному файлі додатка. Або додати їх у вже створену папку Modules яка знаходиться в корені папки з *.exe файлом додатка.

При запуску додатка відображається головне вікно (рис 3.2). На головному вікні відображаються доступні виявлені модулі. Щоб додати модуль до моделі треба виділити його у списку і нажати кнопку Add або двічі кликнути на рядок з модулем у списку. Доданий модуль з’явиться в основній таблиці. У кожному рядку таблиці знаходяться кнопки керування модулем.

На головному вікні також розташовується компонент таймер. Він містить 2 поля: Ticks і Precision. Вони дозволяють задати кількість циклів (тиків) моделювання й роздільність (точність). Наприклад, якщо прийняти один тик за одну секунду, а роздільність вибрати 60, то фактично моделювання буде протікати по хвилинах. Для запуску розрахунків моделі необхідно нажати кнопку Start. Після виконання розрахунків моделі у полі “Actualtime” буде виведений реальний час, витрачене на розрахунки моделі у форматі mm/ss/ms.

Щоб вилучити модуль із моделі необхідно нажати кнопку Delete, при цьому будуть вилучені всі вхідні й вихідні зв'язки модуля.

Для виклику вікна налаштувань модуля натисніть кнопку Edit. При цьому відкриється одне з вікон, зображених на рисунках 3.3 – 3.4.

Вікно налаштування модуля дозволяє змінити його ім'я в моделі й задати вхідні зв'язки. Для зміни імені модуля необхідно ввести нове ім'я в поле зверху вікна налаштування. Нижче розташовується таблиця, яка дозволяє задати вхідні зв'язки модуля. Кожний рядок таблиці дозволяє настроїти окремий вхід модуля. Для того, щоб створити зв'язок, потрібно вибрати модуль у списку Selectmodule і вибрати вихід обраного модуля зі списку Selectoutput. Для видалення зв'язку натисніть кнопку Clear. Для того, щоб указати що зв'язок є зворотним, виберіть Delayed.

Налаштування системного модуля Гістограма має іншу структуру. На рис. 3.4 зображений зовнішній вигляд вікна налаштування такого модуля. Параметри часу і значення задаються в полях Time і Value відповідно. Для додавання нового значення натисніть Add. Для видалення поточного - натисніть Del.

Для виклику вікна для відображення графіків виберіть галочку Graphics у головній таблиці. Вікно графіка (рис. 3.5) являє собою модальне вікно, яке дозволяє вибрати ті виходи, значення яких будуть відображені на графіку. Для включення або відключення

необхідного виходу модуля необхідно поставити галочку напроти його назви. Для зміни й масштабування графіків використовується миша.

3.4.2 Посібник з установки

Для успішної установки й роботи програми, необхідна наявність .NET Framework останньої версії (4.0). Якщо вона не встановлена, необхідно її скачати із центру завантаження Microsoft і встановити.

Для установки системи моделювання запустіть файлsetup.exe, який знаходитьяс на диску. Дотримуючись його інструкцій, виберіть папку для установки. Після установки скопіюйте модулі в окрему папку й укажіть її в конфігураційному файлі додатка. Додаток готовий до роботи.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

5.1 Аналіз потенційних небезпечних і шкідливих виробничих чинників проектованого об’єкту, що мають вплив на персонал

У даному дипломному проекті розробляється програмне забезпечення. Розроблене програмне забезпечення орієнтоване на роботу з персональним комп'ютером. Експлуатовані для вирішення внутрішньовиробничих завдань ПЕОМ типу IBM PC мають наступні характеристики:

споживана потужність	220 Вт;
робоча напруга	220 В;
напруга джерел живлення	+12 В; - 12 В; +5 В;
робоча частота	50 Гц.

Виходячи з приведених характеристик, вочевидь, що для людини існує небезпека поразки електричним струмом, унаслідок недбалого поводження з комп'ютером і порушення правил експлуатації, залишення частин ПЕОМ, що знаходяться під напругою, відкритими або знятих для ремонту вузлів.

Відповідно до [51] до легкої фізичної роботи відносяться всі види діяльності, виконувані сидячи і ті, що не потребують фізичної напруги. Робота користувача ПК відноситься до категорії 1а.

При роботі на ПЕОМ користувач піддається ряду потенційних небезпек. Унаслідок недотримання правил техніки безпеки при роботі з машиною (невиконання огляду відкритих частин ПЕОМ, що знаходяться під напругою або знятих для ремонту вузлів) для користувача існує небезпека поразки електричним струмом.

Джерелами підвищеної небезпеки можуть служити наступні елементи:

- розподільний щит;
- джерела живлення;
- блоки ПЕОМ і друку, що знаходяться в ремонті.

Ще одна проблема полягає у тому, що спектр випромінювання комп'ютерного монітора включає рентгенівську, ультрафіолетову і інфрачервону області, а також широкий діапазон хвиль інших частот. Небезпека рентгенівського проміння мала, оскільки цей вид випромінювання поглинається речовиною екрану. Проте велику увагу

слід приділяти біологічним ефектам низькочастотних електромагнітних полів (аж до порушення ДНК).

Відповідно до [52], при обслуговуванні ПЕОМ мають місце фізичні і психофізичні небезпечні, а також шкідливі виробничі чинники:

– підвищене значення напруги в електричному ланцюзі, замикання якої може відбутися через тіло людини;
– підвищений рівень статичної електрики;
– підвищений рівень електромагнітних випромінювань;
– підвищена або знижена температура повітря робочої зони;
– підвищений або знижений рух повітря;
– підвищена або знижена вологість повітря;
– відсутність або недостатність природного світла;
– підвищена пульсація світлового потоку;
– недостатня освітленість робочого місця;
– підвищений рівень шуму на робочому місці;
– розумове перенапруження;
– емоційні навантаження;
– монотонність праці.

5.2 Заходи щодо техніки безпеки

Основним небезпечним чинником при роботі з ЕОМ є небезпека поразки людини електричним струмом, яка посилюється тим, що органи чуття людини не можуть на відстані знайти наявності електричної напруги на устаткуванні.

Проходячи через тіло людини, електричний струм чинить на нього складну дію, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (роздратування і збудження нервових волокон і інших органів тканин організму) дій.

Тяжкість поразки людини електричним струмом залежить від цілого ряду чинників:

– значення сили струму;
– електричного опору тіла людини і тривалості протікання через нього струму;
– роду і частоти струму;

– індивідуальних властивостей людини і навколишнього середовища.

Розроблений дипломний проект передбачає наступні технічні способи і засоби, що застерігають людину від ураження електричним струмом:

– заземлення електроустановок;

– занулення;

– захисне відключення;

– електричне розділення мережі;

– використовування малої напруги;

– ізоляція частин, що проводять струм;

– огорожа електроустановок.

Занулення зменшує напругу дотику і обмежує години, протягом яких людина, ткнувшись до корпусу, може потрапити під дію напруги.

Струм однофазного короткого замикання визначається по наближеній формулі:

$$I_K = \frac{\text{Uф}}{\text{Zп} + \frac{\text{Zт}}{3}}\ , \qquad (5.1)$$

де Uф - номінальна фазна напруга мережі, В;

Zп - повний опір петлі, створене фазними і нульовими дротами, Ом;

Zт - повний опір струму короткого замикання на корпус, Ом.

Згідно таблиці 4 [53]: Zт /3 = 0,1 Ом.

Для провідників і жил кабелю для розрахунку повного опору петлі використовуємо формулу(5.2.) :

$$Z\text{п} = \sqrt{\text{Rп}^2 + \text{Xп}^2}\ , \qquad (5.2)$$

де Rп = Rф + Ro - сумарний активний опір фазного Rф і нульового Ro дротів, Ом;

Xп - індуктивний опір паяння дротів, Ом.

Перетин 1 км мідного дроту S = 2.5 мм, тоді згідно таблицям 5 і 6 [53], має такий опір:

Xп = 0,11 Ом;

Rф = 7,55 Ом;

Ro = 7,55 Ом.

Отже, Rп = 7,55 + 7,55 = 15,1 Ом.

Тоді по формулі (5.2) знаходимо повний опір петлі :

$$Zп = \sqrt{15{,}1^2 + 0{,}11^2} \approx 15{,}1 \text{ (Ом).}$$

Струм однофазного короткого замикання рівний:

$$Iк = \frac{220}{15{,}1 + 0{,}1} = 14{,}47 \text{ (А).}$$

Дія плавкої вставки на ПЕОМ забезпечується, якщо виконується співвідношення:

$$Iк \geq k * Iн, \quad (5.3)$$

де Iн - номінальний струм спрацьовування плавкої вставки, А;

k - коефіцієнт кратності нелінійного струму Iн, А.

Коефіцієнт кратності нелінійного струму Iн розраховується по формулі (5.4.) :

$$Iн = P / U, \quad (5.4)$$

де P = 220 Вт - споживана потужність;

U = 220 В - робоча напруга;

k = 3 А - для плавких вставок.

Отже, Iн = 220 / 220 = 1 А.

Підставивши значення у вираз (5.3), одержимо:

$$14{,}47 > 3*1.$$

Таким чином, доведено, що апарат забезпечить спрацьовування(і захист) при підвищенні номінального струму.

4.3 Заходи, що забезпечують виробничу санітарію і гігієну праці

Вимоги до виробничих приміщень встановлюються [61], ДБН, відповідними ГОСТами і ОСТамі з урахуванням небезпечних і шкідливих чинників, що утворюються в процесі експлуатації електроустаткування.

Підвищення працездатності людини і збереження її здоров'я забезпечується стабільними метеорологічними умовами.

Мікроклімат виробничих приміщень визначається діючими на організм людини поєднаннями температури, вологості і швидкості руху повітря, а також температури навколишніх поверхонь. Значне коливання параметрів мікроклімату приводити до порушення систем кровообігу, нервової і потовидільної, що може викликати підвищення або пониження температури тіла, слабкість, запаморочення і навіть непритомність.

Відповідно до [51] встановлюють оптимальну і допустиму температуру, відносну вологість і швидкість руху повітря в робочій зоні . За відсутності надмірного тепла, вологи, шкідливих речовин в приміщенні досить природної вентиляції.

У приміщенні для виконання робіт операторського типу(категорія 1а), пов'язаних з нервово-емоційною напругою, проектом передбачається дотримання наступних нормованих величин параметрів мікроклімату (табл. 5.1).

Таблиця 4.1 - Санітарні норми мікроклімату робочої зони приміщень для робіт категорії 1а.

Пора року	Температура, С	Відносна вологість, %	Швидкість руху повітря, м/с
Холодна	22…24	40…60	0,1
Тепло	23…25	40…60	0,1

У приміщенні, де знаходиться ПЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (з пристроєм вентиляційних каналів в перекриттях будівлі і вертикальних шахт) й установленого промислового кондиціонера фірми Mitsubishi, який дозволяє вирішити переважну більшість завдань по створінню та підтримці необхідних параметрів повітряного середовища. Цей метод забезпечує приток потрібної кількості свіжого повітря, визначеного в ДБН (30 $м^3$ в годину на одного працівника).

Шум на виробництві має шкідливу дію на організм людини. Стомлення операторів через шум збільшує число помилок при роботі, призводить до виникнення травм. Для

оператора ПЕОМ джерелом шуму є робота принтера. Щоб усунути це джерело шуму, використовують наступні методи. При покупці принтера слід вибирати найбільш шумозахисні матричні принтери або з великою швидкістю роботи(струменеві, лазерні). Рекомендується принтер поміщати в найбільш віддалене місце від персоналу, або застосувати звукоізоляцію та звукопоглинання(під принтер підкладають демпфуючі підкладки з пористих звукопоглинальних матеріалів з листів тонкої повсті, поролону, пеноплену).

При роботі на ПЕОМ, проектом передбачені наступні методи захисту від електромагнітного випромінювання : обмеження часом, відстанню, властивостями екрану.

Обмеження години роботи на ПЕОМ складає 3,5-4,5 години. Захист відстанню передбачає розміщення монітора на відстані 0,4-0,5 м від оператора. Передбачений монітор 20" TFT, Samsung 2043BW відповідає вимогам стандарту TCO'03.

TCO'03 пред’являє жорсткі вимоги в таких областях: ергономіка (фізична, візуальна і зручність користування), енергія, випромінювання (електричних і магнітних полів), навколишнє середовище і екологія, а також пожежна та електрична безпека, які відповідають всім вимогам [54].

Для зниження стомлюваності та підвищення продуктивності праці обслуговуючого персоналу в колірній композиції інтер'єру приміщень для ПЕОМ дипломним проектом пропонується використовувати спокійні колірні поєднання і покриття, що не дають відблисків.

У проекті передбачається використання сумісного освітлення. У світлий час доби приміщення освітлюватиметься через віконні отвори, в решту часу використовуватиметься штучне освітлення.

Як штучне освітлення необхідно використовувати штучне робоче загальне освітлення. Для загального освітлення необхідно використовувати люмінесцентні лампи. Вони володіють наступними перевагами: високою світловою віддачею, тривалим терміном служби, хоча мають і недоліки: високу пульсацію світлового потоку.

При експлуатації ПЕОМ виробляється зорова робота. Відповідно до [58] ця робота відноситься до розряду 5а. При цьому нормоване освітлення на робочому місці(Ен) при загальному освітленні рівна 200 лк.

Приміщення завдовжки 12 м, шириною 10 м, заввишки 4 м обладнується світильниками типу ЛП02П, оснащеними лампами типу ЛБ зі світловим потоком 3120 лм кожна.

Виконаємо розрахунок кількості світильників в робочому приміщенні завдовжки a=12 м, шириною b=10 м, заввишки з=4 м, використовуючи формулу (5.5) розрахунку штучного освітлення при горизонтальній робочій поверхні методом світлового потоку:

$$n = (E \cdot S \cdot Z \cdot k)/(F \cdot U \cdot M), \quad (5.5)$$

де F - світловий потік = 3120 лм;

E - максимально допустима освітленість робочих поверхонь = 200 лк;

S - площа підлоги = 120 м2;

Z - поправочний коефіцієнт світильника = 1,2;

k - коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації світильників = 1,5;

n - кількість світильників;

U - коефіцієнт використовування освітлювальної установки = 0,6;

M - кількість ламп у світильнику =2.

З формули (5.5) виразимо n (5.6) і визначимо кількість світильників для даного приміщення:

$$n = (E \cdot S \cdot Z \cdot k)/(F \cdot U \cdot M), \quad (5.6)$$

Отже, $n = (200 \cdot 120 \cdot 1{,}2 \cdot 1{,}5)/(3120 \cdot 0{,}6 \cdot 2) = 12$.

Виходячи з цього, рекомендується використовувати 12 світильників. Світильники слід розміщувати рядами, бажано паралельно стіні з вікнами. Схема розташування світильників зображена на рис. 5.1.

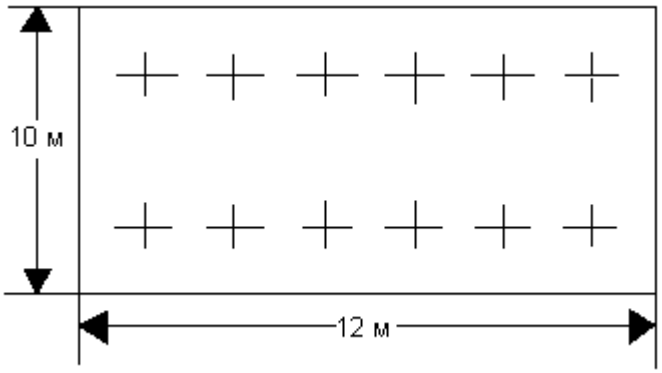


Рисунок 5.1 - Схема розташування світильників

5.4 Рекомендації по пожежній безпеці

Пожежі в приміщеннях, де встановлена обчислювальна техніка, представляють небезпеку для життя людини. Пожежі також пов'язані як з матеріальними втратами, так і з відмовою засобів обчислювальної техніки, що у свою чергу спричиняє за собою порушення ходу технологічного процесу.

Пожежа може виникнути при наявності горючої речовини та внесення джерела запалювання в горюче середовище. Пальними матеріалами в приміщеннях, де розташовані ПЕОМ, є:

– поліамід - матеріал корпусу мікросхеми, горюча речовина, температура самозаймання аерогелю 420 °С ;

– полівінілхлорид - ізоляційний матеріал, горюча речовина, температура запалювання 335 °С, температура самозаймання 530 °С, кількість енергії, що виділяється при згоранні - 18000 - 20700 кДж/кг;

– стеклотекстоліт ДЦ - матеріал друкарських плат, важкозаймистий матеріал, показник горючості 1.74, не схильний до температурного самозаймання;

– пластика кабельний №489 - матеріал ізоляції кабелю, горючий матеріал, показник горючості більш 2.1;

– деревина - будівельний і обробний матеріал, матеріал з якого виготовлені меблі, горючий матеріал, показник горючості більше 2.1, теплота згорання 18731 - 20853 кДж/кг, температура запалювання 399 °С, схильна до самозаймання.

Згідно [60] приміщення відносяться до категорії В (пожежовибухонебезпечним) і згідно правилам побудови електроустановок простір усередині приміщення відноситься до вогненебезпечної зони класу П - IIа (зони, розташовані в приміщеннях, в яких зберігаються тверді горючі речовини).

Потенційними джерелами запалення при роботі ПЕОМ є:

- іскри при замиканні і розмиканні ланцюгів;
- іскри і дуги коротких замикань;
- перегріви від тривалого перевантаження і наявності перехідного опору.

Продуктами згорання, що виділяються при пожежі, є : оксид вуглецю, сірчистий газ, оксид азоту, синильна кислота, акропеін, фосген, хлор та ін. При горінні пластмас, окрім звичайних продуктів згорання, виділяються різні продукти термічного розкладання: хлорангідрідні кислоти, формальдегіди, хлористий водень, фосген, синильна кислота, аміак, фенол, ацетон, стирол та ін., що шкідливо впливають на організм людини.

Для захисту персоналу від дії небезпечних і шкідливих чинників пожежі проектом передбачається застосування промислового протигаза з коробкою марки В(жовта).

Пожежна безпека об’єктів народного господарства регламентується [55] і забезпечується системами запобігання пожежам і протипожежному захисту. Для успішного гасіння пожеж вирішальне значення має швидке виявлення пожежі і своєчасний виклик пожежних підрозділів до місця пожежі.

Зменшити горюче навантаження не представляється можливим, тому проектом передбачається застосувати наступні способи і їх комбінації для запобігання утворенню(внесення) джерел запалення :

– застосування устаткування, що задовольняє вимогам електростатичної безпеки;

– застосування в конструкції швидкодіючих засобів захисного відключення можливих джерел запалення;

– виключення можливості появи іскрового заряду статичної електрики в горючому середовищі з енергією, рівної і вище мінімальної енергії запалення;

– підтримка температури нагріву поверхні машин, механізмів, устаткування, пристроїв, речовин і матеріалів, які можуть увійти до контакту з пальним середовищем, нижче гранично допустимої, становить 80% якнайменшої температури самозаймання пального.

– заміна небезпечних технологічних операцій більш безпечними;

– ізольоване розташування небезпечних технологічних установок і устаткування;

– зменшення кількості пальних і вибухонебезпечних речовин, що знаходяться у виробничих приміщеннях;

– запобігання можливості утворення пальних сумішей на лінії, вентиляційних системах і ін.;

– механізація, автоматизація та справність(потокова) виробництва;

– суворе дотримання стандартів і точне виконання встановленого технологічного режиму;

– запобігання можливості появи в небезпечних місцях джерел запалення;

– запобігання розповсюдженню пожеж і вибухів;

– використовування устаткування і пристроїв, при роботі яких не виникає джерел запалення;

– виконання вимог сумісного зберігання речовин і матеріалів;

– наявність громовідводу;

– ліквідація можливості самозаймання речовин і матеріалів .

Для запобігання пожежі в обчислювальних центрах проектом пропонується виконання наступних вимог :

– електроживлення ЕОМ повинно мати автоматичне блокування відключення електроенергії на випадок зупинки системи охолоджування і кондиціонування;

– система вентиляції обчислювальних центрів повинна бути обладнана блокуючими пристроями, що забезпечують її відключення на випадок пожежі;

– робочі місця повинні бути оснащені пожежними щитами, сигналізацією, засобами для сповіщення про пожежну небезпеку (телефонами), медичними аптечками для надання першої медичної допомоги, розробленим планом евакуації.

Для зниження пожежної небезпеки в приміщеннях використовуються первинні засоби гасіння пожеж, а також система автоматичної пожежної сигналізації, яка дозволяє знайти початкову стадію загоряння, швидко і точно оповістити службу пожежної охорони про час і місце виникнення пожежі.

Відповідно до правил пожежної безпеки для промислових підприємств приміщення категорії В підлягають устаткуванню системами автоматичної пожежної сигналізації. Проектом передбачається застосування датчика типу ІДФ - 1(димовий фотоелектричний датчик), оскільки специфікою пожеж обчислювальної техніки і радіоапаратури є, в першу чергу, виділення диму, а потім - підвищення температури.

При виникненні пожежі в робочому приміщенні обслуговуючий персонал зобов'язаний негайно вжити заходи по ліквідації пожежі. Для ліквідації пожежі використовують вогнегасники (хімічно-пінні, пінні для повітря ОП-5, ОП-6, ОП-9, вуглекислотні ОУ-5), пісок, пожежний інвентар (сокири, ломи, багри, шерстяну або азбестову ковдри). Як засіб індивідуального захисту проектом передбачається використовування промислового протигаза з маскою, фільтруючої коробки В.

В якості організаційно-технічних заходів рекомендується проводити навчання робочого персоналу правилам пожежної безпеки.

5.5 Вплив на навколишнє середовища

В даний час зростає кількість комп'ютерної техніки в усіх галузях діяльності людини. Багато користувачів і виробників помиляються, вважаючи, що зі зменшенням і удосконаленням комп'ютерів, зменшиться їх негативний вплив на навколишнє середовище.

На даний момент найбільш суворим з існуючих світових стандартів екологічності для комп'ютерної техніки є стандарт TCO-99. У порівнянні з попередніми він містить додаткові обмеження по частині екології, ергономіки, енергоспоживання і емісії пристроїв.

Організація по захисту навколишнього середовища Greenpeace з 2006 року оцінює виробників електроніки за кількістю важких металів і отруйних речовин, наприклад інгібіторів горіння, використовуваних ними при виробництві (інгібітор - речовина, присутність якого в невеликих кількостях призводить до запобігання або уповільнення процесів горіння або корозії; інгібітори знижують швидкість хімічних реакцій або пригнічують їх). Однак навіть оцінки такої організації, як Greenpeace, не можуть претендувати на об'єктивність. Адже в одних випадках вона використовує перевірену інформацію, що стосується, наприклад, заходів щодо утилізації відходів, а в інших спирається тільки на дані виробника. А якщо компанія не повідомляє ніяких відомостей, то автоматично опиняється на нижніх рядках рейтингу. Крім того, енергетичні витрати на виробництво і перевезення продукції також необхідно враховувати при оцінці екологічної ефективності. Адже часи, коли техніка виготовлялася тільки на одному заводі, давно пройшли. Сьогодні окремі комплектуючі закуповуються на різних підприємствах по всьому світу, після чого здійснюється складання пристроїв. Тому найчастіше навіть самі компанії не можуть знати, які шкідливі речовини потрапляють в атмосферу при виготовленні їх продукції і які саме метали або токсини в ній містяться.

ЖК-екрани - один з джерел парникових газів, які набагато шкідливіше діоксиду вуглецю. Рідкокристалічні монітори швидко знайшли популярність, прийшовши на зміну громіздким ЕПТ-моделям. І це не дивно, адже вони мають тонкі корпуса і споживають значно менше електроенергії. За іншим аспектам екологічної безпеки дисплеї на основі рідких кристалів також вважалися проривом, тому що в них не використовувався газ, що містить свинець. Досить довго ніхто не звертав уваги на застосовуваний для чищення РК-панелей трехфторістий азот (NF3), і тільки в середині 2008 року вченими було доведено наявність даної хімічної речовини в атмосфері. Відкриття було вражаючим: порівняно з

діоксидом вуглецю (CO2) NF3 має в 17 000 разів більше активного парникового газу, а його атмосферний час напіврозпаду може складати від 550 до 740 світлових років (у CO2 - від 30 до 40 років). Закону, який обмежував би рівень викиду NF3, поки не існує.

Виявлення енерговитрат є таким же проблематичним процесом, як і визначення кількості матеріалів, придатних для вторинної переробки, і важких металів, що містяться в пристроях. Таким чином, надійним показником екологічності залишається тільки рівень енергоспоживання.

Полівінілхлорид, що позначається зазвичай абревіатурою ПВХ, - це різновид пластику, що застосовується в самих різних цілях. З нього зроблена зовнішня оболонка кабелів, якими з'єднуються пристрої, він оточує електричний провід портативного комп'ютера. Це дешевий, міцний і вельми поширений матеріал. Разом з тим, за словами ІТ- аналітика «Грінпіс» Кейсі Харрелл, «ПВХ - найгірший з пластиків». Він є причиною виникнення гормонального дисбалансу, проблем в репродуктивної сфері та різних форм раку. Полівінілхлорид практично неможливо правильно утилізувати. Внаслідок старий матеріал виявляється зазвичай на звалищі з відходами або, того гірше, спалюється з метою вилучення мідних жил і інших цінних компонентів. При його згорянні утворюється вкрай шкідливий канцерогенний діоксин. Звалища і хімічні поховання забруднюють джерела води. Єдиний спосіб правильно утилізувати ПВХ полягає в тому, щоб відправити його в центр небезпечних відходів.

Залишається лише сподіватися, що настане час, коли технології будуть допомагати людині, не завдаючи незворотної шкоди здоров'ю навколишнього середовища.

ВИСНОВКИ

Метою дипломного проекту є поліпшення етапів процесу моделювання технологічних процесів та об’єктів за рахунок розробки гнучкої, розширюваної системи моделювання – середовища конструювання моделей, що дозволяє будувати складні моделі технологічних процесів і установок у простому й зручному виді, досліджувати характеристики цих моделей і обробляти отримані результати. Система моделювання не повинна мати підвищену складність, припускати можливість обслуговування користувачем, що не є програмістом і мати зручний інтерфейс користувача.

У зв'язку з тим, що існуючі системи мають ряд недоліків, то вони не можуть бути використані в чистому виді для розв'язку поставленої задачі. Тому виникає необхідність у розробці програмних засобів, які:

– дозволять універсальним способом будувати й досліджувати моделі технологічних процесів;

– мають зручний інтерфейс для створення, настроювання та дослідження програмних моделей;

– прості в освоєнні;не вимагають кваліфікованої підготовки користувача;

– не повинні бути орієнтовані на специфіку певного технологічного процесу.

Для вирішення поставленої задачі використані наступні методи:

– математичне моделювання – для реалізації складових частин (модулів) розроблювальної системи (це дозволить описувати й розробляти модулі будь-якої складності й забезпечувати максимальну швидкодію);

– комп'ютерне моделювання – необхідне для реалізації процесу моделювання з можливістю динамічно, гнучко поєднувати отримані модулі в єдину систему, і надавати засоби для наочного відображення результатів моделювання з метою їх аналізу.

Розроблено алгоритми та засоби, які забезпечують модульність та швидкодію системи моделювання. Розглянуто перспективи подальшої розробки. Розроблена система має наступні властивості:

– дозволяє автоматично виявляти та підключати модулі.

– дозволяє задавати зв'язки між вхідними й вихідними параметрами модулів та виконувати основну логіку з моделювання процесів.

Для системи моделювання розроблений простий і зрозумілий графічний інтерфейс користувача, що дозволяє виконати вимоги, встановлені в технічному завданні.

Проведено тестування розробленої системи. Визначено що система коректно знаходить, завантажує та підключає модулі, дозволяє з’єднувати модулі між собою (тобто з’єднувати вхідні та вихідні параметри). Проведено тестування розробленого модуля теплообмінника. Виявлено та означено середні швидкості виконання моделювання системи, яка складається с декількох модулей. Визначено вплив часу перемальовування графічного інтерфейсу користувача на загальний час моделювання.

Розроблена система повністю задовольняє вимогам дипломного проекту. Вона має достатню швидкодію. Час, витрачений на обробку модулів і вибудовування їх у чергу на виконання багато менше, часу виконання самих модулів і тому практично не позначається на загальному часі моделювання. Має простий та зрозумілий графічний інтерфейс користувача та засоби для виведення вихідної інформації у вигляді графіків.

У розділі «Охорона праці» виконано аналіз потенційних небезпек при роботі із засобами обчислювальної техніки і механізмами, розроблені заходи щодо техніки безпеки, заходи, які забезпечують виробничу санітарію і гігієну праці, розраховане штучне освітлення, виконані рекомендації по пожежній безпеці, розглянутий можливий вплив на навколишнє середовище.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Ричард Дирксен (Citect, Нидерланды) "Citect — новая SCADA-система на российском рынке и новые возможности", Мир компьютерной автоматизации, 3, 1999

2) Пьявченко Т.А. Проектирование АСУТП в SCADA-системе TRACE MODE. Таганрог, 2007г.

3) ftp://oscada.org.ua/OpenSCADA/0.6.3/doc/openscada_ru.pdf - OpenSCADA v. 0.6.3.

4) http://oscada.org.ua/oscadaArch/7conferOSDN/OSDN7_Thes.pdf-OpenSCADA - проникновение в PLC.

5) Техническая документация на графическую инструментальную систему для разработки АСУ ТРЕЙС МОУД. Версия 4.20. Издание второе. М., AdAstra Research Group, Ltd., 1997.

6) Анизимиров Л., Айзин В., Фридлянд А. Новая версия Trace Mode для Windows NT/ Л. Анизимиров, В. Айзин, А. Фридлянд// Современные технологии автоматизации.— 1998.— 3.—с 56;

7) Анизимиров Л. Windows- компоненты Trace Mode 4.20/ Л. Анизимиров// Современные технологии автоматизации.— 1996.— 1.—с 102;

8) Кузнецов А. SACADA- системы: программистом можешь ты не быть…/А. Волобуев// Современные технологии автоматизации.— 1996.— 1.—с 32;

9) Андреев Е. Б., Куцевич Н. А. Синенко О. В. SCADA-системы: взгляд изнутри/ Е. Б. Андреев, Н. А. Куцевич, О. В. Синенко.— М.: здательство РТСофт, 2004.— 176с;

10) Букреев В. Г., Цхе А. В. Основы инструментальной системы разработки АСУ/ В. Г. Букреев, А. В. Цхе.— Томск: издательство ТПУ, 2003.—127;

11) Локотов А. Что должна уметь система SCADA/ А. Локотов// Современные технологии автоматизации.— 1998.— 3.—с 44;

12) Самарский А.А., Михайлов А.П. Математическое моделирование: Идеи. Методы. Примеры. — М: Наука, 1997. — 320 с. — ISBN 5-9221-0120-X.

13) Муха В. С. Вычислительные методы и компьютерная алгебра: учеб.-метод. пособие. — 2-е изд., испр. и доп. — Минск: БГУИР, 2010.- 148 с.: ил, ISBN 978-985-488-522-3, УДК 519.6 (075.8), ББК 22.19я73, М92

14) Советов Б. Я., Яковлев С. А. Моделирование систем: Учеб. для вузов — 3-е изд., перераб. и доп. — М.: Высш. шк., 2001. — 343 с. — ISBN 5-06-003860-2

15) Хемди А. Таха Глава 18. Имитационное моделирование // Введение в исследование операций = OperationsResearch: AnIntroduction. — 7-е изд. — М.: «Вильямс», 2007. — С. 697-737. — ISBN 0-13-032374-8

16) Строгалев В. П., Толкачева И. О. Имитационное моделирование. — МГТУ им. Баумана, 2008. — С. 697-737. — ISBN 978-5-7038-3021-5

17) Самарский А.А., Михайлов А.П. Математическое моделирование: Идеи. Методы. Примеры. М.:Физматлит, 2001. 320 с.

18) Дворецкий С.И., Егоров А.Ф., Дворецкий Д.С.Д243 Компьютерное моделирование и оптимизация технологических процессов и оборудования: Учеб.пособие. Тамбов: Изд-во Тамб. гос. техн. ун-та, 2003. 224 с.ISBN 5-8265-0213-4

19) Вероятностные разделы математики / Под ред. Ю. Д. Максимова. — Спб.: «Иван Фёдоров», 2001. — С. 400. — 592 с. — ISBN 5-81940-050-X

20) Шеннон Р. Имитационное моделирование — искусство и наука М.: Мир, 1978

21) Окольнишников В. В.,Представление времени в имитационном моделировании Новосибирск.: СО РАН, Вычислительные технологии, том 10, №5, 2005

22) Кобелев Н. Б. Введение в общую теорию имитационного моделирования М.: Принт-Сервис, 2007

23) Колесов Ю. Б., Сениченков Ю. БМоделирование систем. Объектно-ориентированный подход СПб.: БХВ-Петербург, 2006

24) Кафаров В. В., Глебов М.Б. Математическое моделирование основных процессов химическихпроизводств: Учеб. пособие для вузов. М.: Высш. шк., 1991. 400 с.

25) Э. Таненбаум Современные операционные системы.ISBN 978-5-49807-306-4 Издательство Питер, 2010 г. Твердый переплет, 1120 стр.

26) Jeffrey Richter - CLR via C# 3rd Edition, Ms Press 2010, 896 стр.

27) Эндрю Троелсен. Язык программирования C# 2010 и платформа .NET 4.0 = Pro C# 2010 and the .NET 4.0 Platform. — 5-еизд. — М.: Вильямс, 2010. — С. 1392. — ISBN 978-5-8459-1682-2

28) Герберт Шилдт C# 4.0: полное руководство = C# 4.0 The Complete Reference. — М.: «Вильямс», 2010. — С. 1056. — ISBN 978-5-8459-1684-6

29) Кристиан Нейгел, Карли Уотсон и др. Visual C# 2010: полныйкурс = Beginning Microsoft Visual C# 2010. — М.: Диалектика, 2010. — ISBN 978-5-8459-1699-0

30) Трей Нэш C# 2010: ускоренный курс для профессионалов = Accelerated C# 2010. — М.: Вильямс, 2010. — С. 592. — ISBN 978-5-8459-1638-9

31) Мэтью Мак-Дональд WPF: Windows Presentation Foundation в .NET 3.5 с примерами на C# 2008 для профессионалов = Pro WPF in C# 2008: Windows Presentation Foundation with .NET 3.5. — 2-ое. — М.: «Вильямс», 2008. — 928 с. — ISBN 978-5-8459-1429-3

32) Андерсон, Крис Основы Windows Presentation Foundation. — СПб.: БХВ-Петербург, 2008. — 432 с. — ISBN 978-5-9775-0265-8

33) Daniel M. Solis Illustrated WPF. — United States of America: Apress, 2009. — 508 с. — ISBN 978-1-4302-1910-1

34) http://msdn.microsoft.com/ru-ru/magazine/dd419663.aspx - Приложения WPF с шаблоном проектирования модель-представление-модель представления.

35) Брайан Керниган, Деннис Ритчи. Язык программирования Си. — Санкт-Петербург: Невский диалект, 2001. — 352 с. — (Библиотека программиста). — ISBN 5794000457

36) Б. Страуструп. Язык программирования C++; The C++ Programming Language / Пер. с англ. — 3-е изд. — СПб.; М.: Невский диалект — Бином, 1999. — 991 с. — 3000 экз. — ISBN 5-7940-0031-7 (Невский диалект), ISBN 5-7989-0127-0 (Бином), ISBN 0-201-88954-4 (англ.)

37) Страуструп Б. Язык программирования C++. Специальноеиздание = The C++ programming language. Special edition. — М.: Бином-Пресс, 2007. — 1104 с. — ISBN 5-7989-0223-4

38) Герберт Шилдт. Полный справочник по C++ = C++: The Complete Reference. — 4-е изд. — М.: Вильямс, 2006. — 800 с. — ISBN 0-07-222680-3

39) Джесс Либерти, Дэвид Хорват. Освой самостоятельно C++ за 24 часа = Sams Teach Yourself C++ in 24 Hours, Complete Starter Kit. — 4-е изд. — М.: Вильямс, 2007. — 448 с. — ISBN 0-672-32681-7

40) Стефенс Д. Р. C++. Сборник рецептов. — КУДИЦ-ПРЕСС, 2007. — 624 с. — ISBN 5-91136-030-6

41) Джеф Просиз Программирование для Microsoft .NET = Programming Microsoft .NET. — М.: Русская редакция, 2003. — С. 704. — ISBN 5-7502-0217-8

42) Кей С. Хорстманн, Гари Корнелл. Java 2. Библиотека профессионала, том 1. Основы = Core Java 2, Volume I — Fundamentals. — 8-е изд. — М.: Вильямс, 2008. — 816 с. — ISBN 978-5-8459-1378-4, ISBN 978-0-13-235476-9

43) Кей С. Хорстманн, Гари Корнелл. Java 2. Библиотека профессионала, том 2. Тонкости программирования = Core Java 2, Volume II — Advanced Features. — 8-е изд. — М.: Вильямс, 2008, 4 кв. — 992 с. — ISBN 978-5-8459-1482-8, ISBN 978-0-13-235479-0

44) Монахов Вадим. Язык программирования Java и среда NetBeans. — 3-е изд. — СПб.: БХВ-Петербург, 2011. — 704 с. — ISBN 978-5-9775-0671-7

45) Джошуа Блох. Java. Эффективное программирование = Effective Java. — М.: Лори, 2002. — 224 с. — ISBN 5-85582-169-2

46) Брюс Эккель. Философия Java = Thinking in Java. — 3-е изд. — СПб.: Питер, 2003. — 976 с. — ISBN 5-88782-105-1

47) Герберт Шилдт, Джеймс Холмс. Искусство программирования на Java = The Art of Java. — М.: Диалектика, 2005. — 336 с. — ISBN 0-07-222971-3

48) Любош Бруга. Java по-быстрому: Практический экспресс-курс = Luboš Brůha. Java Hotová řešení. — М.: Наука и техника, 2006. — 369 с. — ISBN 5-94387-282-5

49) Лыфарь В.А. Моделирование сложных технологических процессов // Вісник Східноукраїнського національного університету імені Володимира Даля. 2008, № 12 (130), Частина 1, Луганськ, с. 31-37.

50) Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного проектирования. Паттерны проектирования. — ISBN 978-5-469-01136-1, ISBN 5-272-00355-1, ISBN 0-201-63361-2, ISBN 5-469-01136-4.

51) ГОСТ 12.1.005-88. Міждержавний стандарт. Система стандартів безпеки праці. Загальні санітарно-гігієнічні вимоги до повітря робочої зони

52) ГОСТ 12.0.003-74 Небезпечні і шкідливі виробничі фактори. Класифікація

53) ДСТУ 7237:2011 Національний стандарт України. Система стандартів безпеки праці. Електробезпека. Загальні вимоги та номенклатура видів захисту

54) ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми. Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.

55) ГОСТ 12.1.004-91. Пожежна безпека. Загальні вимоги .

56) ДБН В.2.5-67. Опалення вентиляція та кондиціонування.

57) ГОСТ 12.1.006-84. Електромагнітні поля радіочастот. Допустимі рівні на робочих місцях і вимоги до проведення контролю

58) ДБН В.2.5-28-2006. Природне і штучне освітлення.

59) ГОСТ 12.4.009-83. Пожежна техніка для захисту об'єктів. Основні види. Розміщення і обслуговування.

60) ДСТУ Б В.1.1-36-2016. Визначення категорії приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною безпекою.

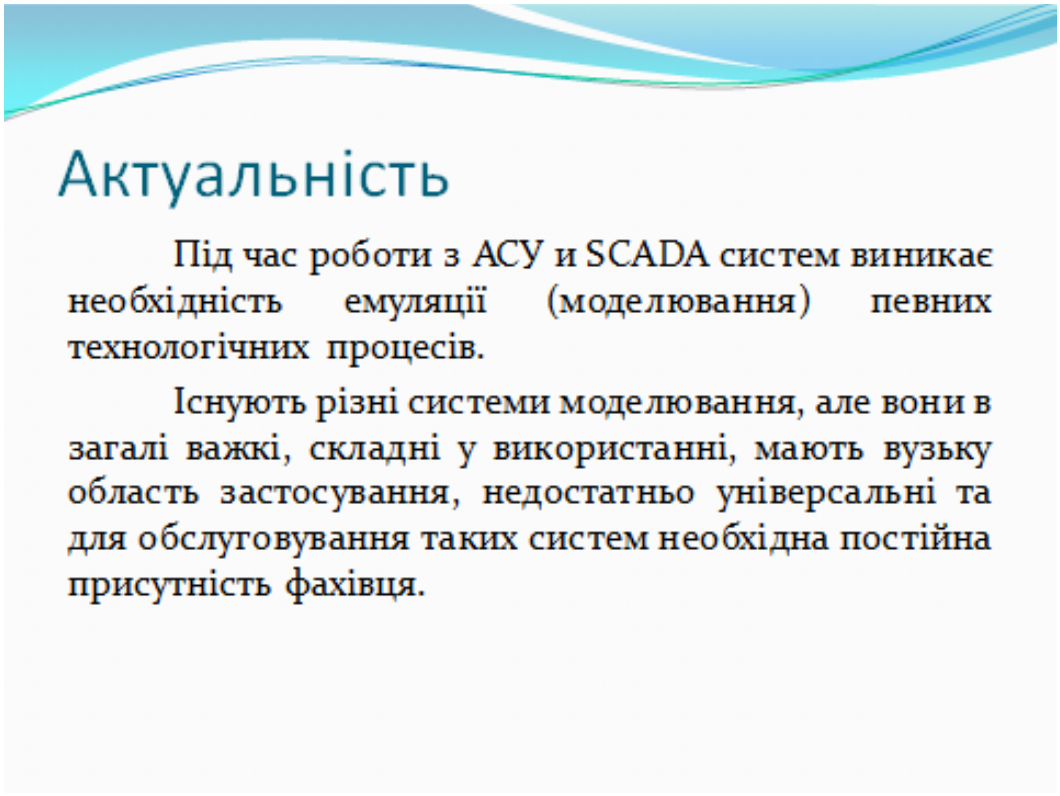
61) ДСП 173-96. Державні санітарні правила планування та забудови населених пунктів

62) Симметрон. Электронные компоненты. Каталог 2002, 2002г. – 192с.

63) Симметрон. Электронные компоненты. Каталог 2004, 2004г. – 192с.

ДОДАТОК А.

ЕЛЕКТРОННІ ПЛАКАТИ



Мета

Розробити гнучку, розширювану систему моделювання – середовища конструювання моделей, що:

- дозволяє будувати складні моделі технологічних процесів та установок;
- дозволяє досліджувати характеристики цих моделей ;
- повинна припускати можливість обслуговування користувачем, що не є програмістом ;
- мати легкий та зручний інтерфейс користувача.

Існуючі системи

1) SCADA-система Citect **Переваги:** • **клієнт-серверна архітектура;** • **масштабованість;** **Недоліки:** • **для використання усіх можливостей необхідна кваліфікована підготовка розробників;**	**2) SCADA-система TRACE MODE** **Переваги:** • **модульність;** • **клієнт-серверна архітектура;** **Недоліки:** • **містить багато нестабільних функцій;** • **складність в освоєнні архітектури системи;**
3) OpenSCADA **Переваги:** • **відкритість та бескоштовність– можливість доробити систему;** **Недоліки:** • **великі системні вимоги;** • **слабкий функціонал;**	**4) SCADA система КОНТУР** **Переваги:** • **вітчизняний розробник– підтримка рідною мовою;** **Недоліки:** • **закритість;** • **застаріла архітектура;** • **недостатня надійність;**

Термін моделювання

Моделювання – це метод дослідження складних систем, заснований на тому, що розглянута система заміняється на модель та проводиться дослідження моделі з метою одержання інформації про досліджувану систему.

Методи моделюванняя, застосовувані у роботі:

- математичне моделювання;
- комп'ютерне моделювання;
- імітаційне моделювання.

Выбір программних та технічних засобів розробки системи

Операційна система

- Microsoft Windows;
- Mac OS X;
- GNU Linux;

Мова програмування

- Мова C# и платформа .NET Framework;
- Мова C++;
- Мова Java;

Засіб розробки графічного інтерфейсу користувача

- Windows Presentation Foundation;
- Windows Forms;

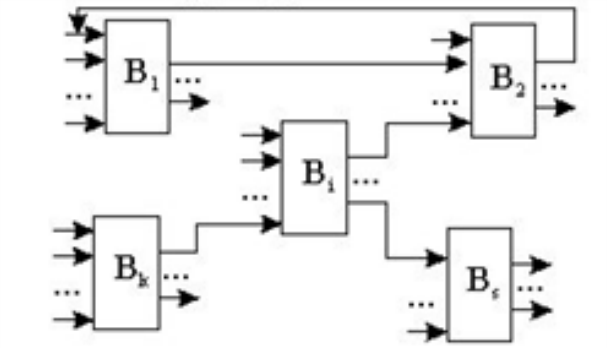
Засіб відібраження графіків

- Visiblox charts;
- Visifire;
- Dynamic Data Display;
- Silverlight Toolkit;

Розробка та дослідження системи. Аналіз отриманих результатів.

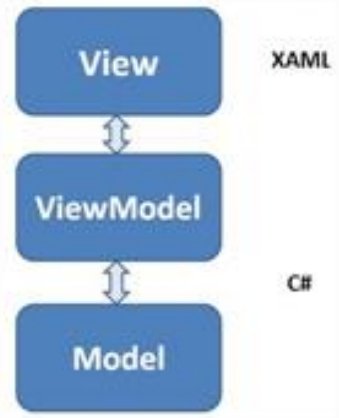
Розробка системи моделювання

- для реалізації моделювання універсальним способом створюється модель, яка має структуру, зображену на малюнку:

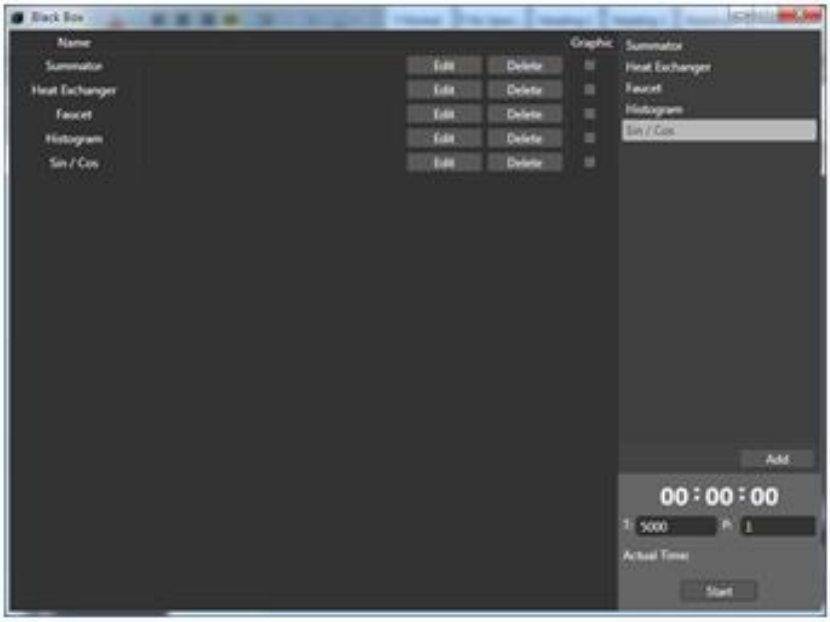


Разробка графічного интерфейсу користувача

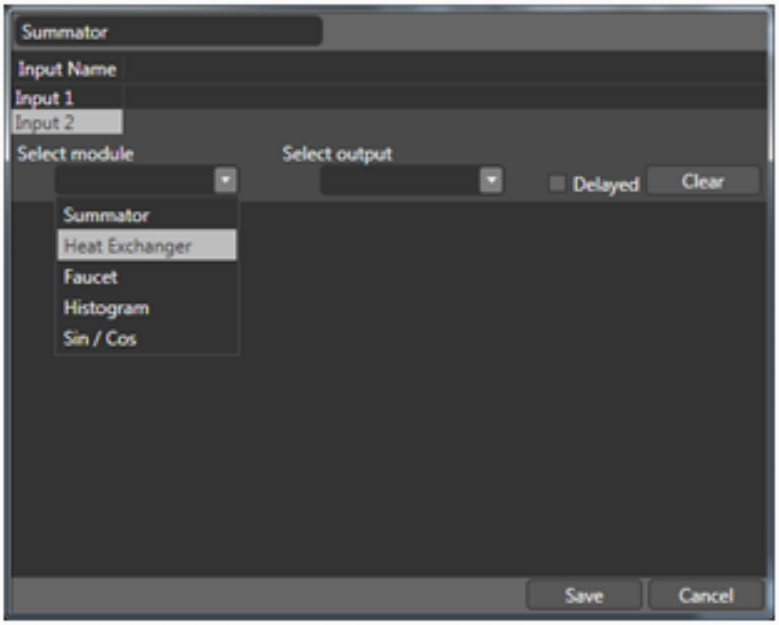
Графічний інтерфейс розроблений з використаннєм шаблона проектування MVVM (Model-View-ViewModel). Структура цього шаблона зображена на діаграмі:



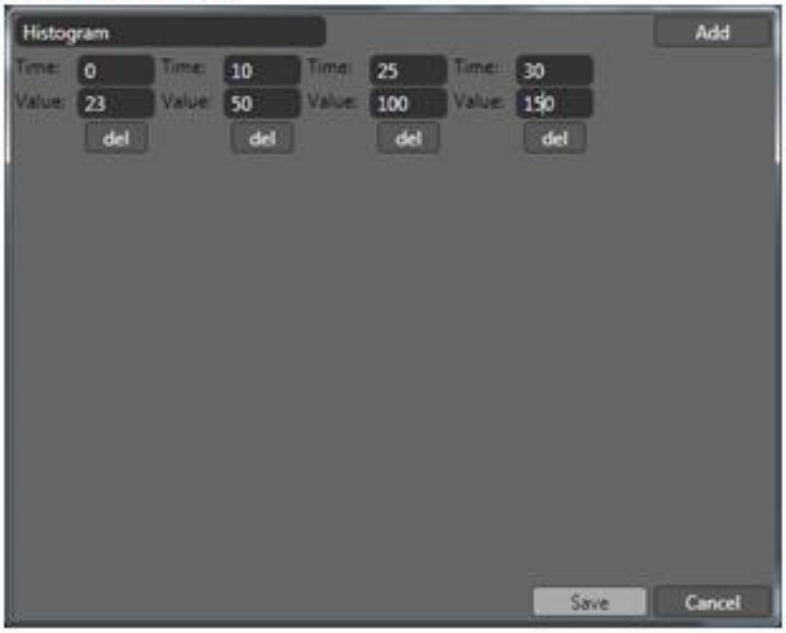
Головне вікно програми



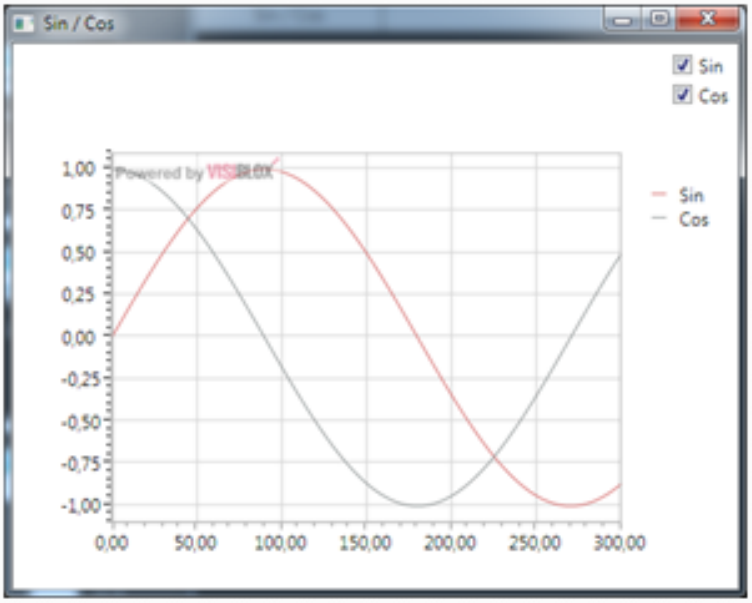
Діалог налаштування модуля



Діалог налаштування модуля «гістограма»

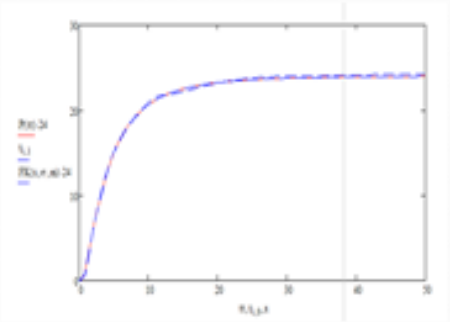


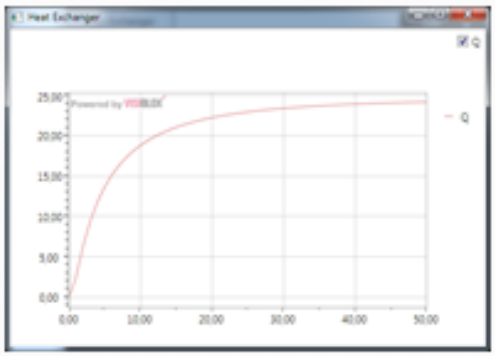
Вікно для виведення графіків



Тест 1: тестування системи на достовірність результатів

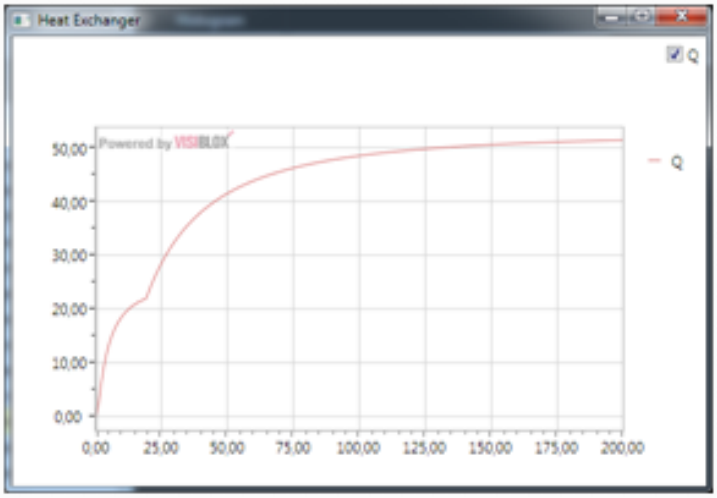
Проведемо тестування системи на прикладі теплообмінник, порівняємо результати з даними, отриманими в MathCAD (m=1.324кг/с, час 50с).





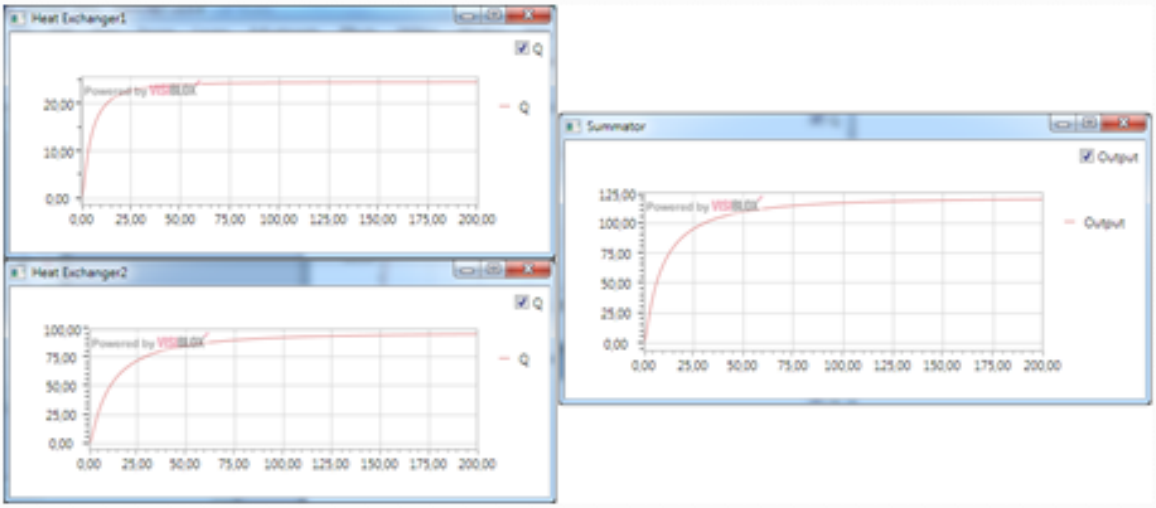
Тест 2: реакція системи на зміну вхідних впливів

Змінемо витрату речовини у момент часу T=20с з 1.324 кг/с до 1.45 кг/с.



Тест 3: модель, яка складається з декількох модулей

Приєднаємо два теплообмінника паралельно та об'єднаємо їх виходи через суматор(m1=1.324кг/с, m2=1.42кг/с, T=200с).



Тест 4: зміна швидкості моделювання

Виміряємо середню швидкість моделювання і оцінимо вплив інтерфейсу користувача на час моделювання.

N теста	Время(мс)
1	296
2	281
3	267
4	286
5	292
Среднее время	284

N теста	Время(мс)
1	47
2	57
3	69
4	48
5	62
Среднее время	56

З результатів видно, що відображення графіків в реальному часі в 5 разів уповільнює швидкість моделювання.

Висновки

Розроблена система має наступні характеристики:

1) Простий інтерфейс користувача.
2) Модульна архітектура.
3) Реалізує універсальний спосіб моделювання процесів.
4) Виконання моделювання відбувається в реальному часі.