

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2021 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему: Дослідження та застосування методів імітаційного моделювання для розробки інформаційної системи проектування комп'ютерних мереж

Виконав: студент 6141 м групи

_____ ***Борис А.В.***
(підпис)

Керівник роботи:

к.ф.-м.н., доцент ,
(посада, науковий ступінь вчене звання)

_____ ***Коваль С.С.***
(підпис)

м. Миколаїв – 2021 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

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

"ЗАТВЕРДЖУЮ"
Гарант освітньої програми

к.т.н, доц. Ажищев В.Ф.
" ____ " _____ 2021 р.

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

Студенту Борису Артуру Віталійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження та застосування методів імітаційного моделювання для розробки інформаційної системи проектування комп'ютерних мереж

Керівник роботи к.ф.-м.н., доцент Коваль С.С.

Затверджені наказом ректора № 1232-уч. від "12" жовтня 2021 року.

2. Термін подання роботи: 21 грудня 2021 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація для розробки web-сервісів, матеріали наукового стажування

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

- Титульний аркуш, завдання на дипломну роботу, анотація (українською, російською, англійською), зміст, перелік умовних означень, символів, одиниць та термінів (при необхідності).
- Вступ.
- Дослідження методів імітаційного моделювання в інформаційних системах проектування.
- Формування вимог до інформаційної системи, розробка її концепції і постановка задачі.
- Розробка проектних рішень по інформаційній системі.
- Реалізація проекту інформаційної системи (технічна документація по проекту).
- Розділ з економіки.
- Розділ з охорони праці.
- Розділ з охорони навколишнього середовища.
- Висновки.
- Список використаної літератури.
- Додатки: текст програми.

5. Перелік презентаційних матеріалів _____

5.1 Мета та завдання роботи.

5.2 Переваги методу імітаційного моделювання.

5.3 Схема організації процесу імітаційного моделювання.

5.4 Математична модель стану мережі.

5.5 Діаграма прецедентів.

5.6 Діаграма послідовності.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	К.т.н., доцент Коваль С.С.		
2	К.т.н., доцент Коваль С.С.		
3	К.т.н., доцент Коваль С.С.		
4			
5	Ст. викл. Гурець Н.В.		
6	Ст. викл. Гурець Н.В.		

7. Дата видачі завдання 04 вересня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Вибір теми, визначення мети, завдань, об'єкта та предмета дослідження	01 жовтня 2020 р.	
2	Підбір та опрацювання літератури	01 лютого 2021 р.	
3	Дослідження технологій інформаційного забезпечення	01 травня 2021 р.	
4	Розробка концепції інформаційного забезпечення	01 вересня 2021 р.	
5	Розробка та реалізація проєкту	01 жовтня 2021 р.	
6	Розробка додаткових розділів, оформлення роботи, підготовка презентаційних матеріалів	01 грудня 2021 р.	
7	Представлення до попереднього захисту	01 грудня 2021 р.	
8	Отримання відгуку керівника	13 грудня 2021 р.	
9	Отримання рецензії	15 грудня 2021 р.	
10	Захист кваліфікаційної роботи	21 грудня 2021 р.	

Студент _____

(підпис)

Борис А.В.
(ПІБ)

Керівник роботи _____

Коваль С.С.

Анотація

Дослідження та використання методів імітаційного моделювання в системі автоматизованого проектування комп'ютерних мереж. Аналіз мереж, та рішення задач їх проектування. Імітаційне моделювання систем автоматизованого проектування.

Дипломна робота складається з __ сторінок, містить __ рисунків, __ таблиць і список використаних джерел з __ найменувань.

Ключові слова: сучасні інформаційні технології, імітаційне моделювання, модель, система автоматизованого проектування, класи, об'єкти, діаграма, потоки даних, комп'ютерна мережа.

Аннотация

Исследования и использования методов имитационного моделирования в системе автоматизированного проектирования компьютерных сетей. Анализ сетей, и решение задач их проектирования. Имитационное моделирование систем автоматизированного проектирования.

Дипломная работа состоит из __ страниц, содержит __ рисунков, __ таблиц и список использованных источников из __ наименований.

Ключевые слова: современные информационные технологии, имитационное моделирование, модель, система автоматизированного проектирования, классы, объекты, диаграмма, потоки данных, компьютерная сеть.

Abstract

Investigation and use of simulation methods in the computer-aided design of computer networks. Network analysis, and decision of tasks of their designing. Simulation of automated design systems.

Thesis consists of __ pages, contains __ figures, __ tables and a list of used sources of __ titles.

Key words: modern information technologies, simulation modeling, model, automated design system, classes, objects, diagram, data flows, computer network.

ЗМІСТ

ЗМІСТ	
ВСТУП	
РОЗДІЛ 1 ДОСЛІДЖЕННЯ МЕТОДІВ МОДЕЛЮВАННЯ КОМП'ЮТЕРНИХ МЕРЕЖ.....	
1.1 Поняття та цілі моделювання комп'ютерних мереж.....	
1.2 Особливості проектування комп'ютерної мережі	
1.3 Характеристики комп'ютерної мережі.....	
1.4 Вплив на мережеві характеристики програмного і апаратного складу комп'ютерних мереж	
1.5 Аналіз готових рішень.....	
1.6 Імітаційне моделювання	
1.7 Основні методи імітаційного моделювання.....	
1.8 Імітаційне моделювання в автоматизованому проектуванні	
1.8 Використання моделювання для оптимізації продуктивності мережі.....	
РОЗДІЛ 2 РОЗРОБКА КОНЦЕПЦІЇ СИСТЕМИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ МЕРЕЖІ	
2.1 Постановка завдання аналізу характеристик комп'ютерної мережі.....	
2.2 Імітаційне моделювання комп'ютерної мережі.....	
РОЗДІЛ 3 РОЗРОБКА ПРОЕКТУ СИСТЕМИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ КОМП'ЮТЕРНИХ МЕРЕЖ	
3.1 Загальносистемні рішення.....	
3.2 Рішення з інформаційного забезпечення	
3.3 Модель послідовностей.....	
3.4 Модель переходів станів	
3.5 Діаграма класів програмного забезпечення	
3.6 Вибір мови програмування.....	57
РОЗДІЛ 4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ.....	
4.1 Вступ.....	

4.2 Цілі і завдання, які вирішуються в економічній частині.....	
4.3 Розрахунок амортизаційних відрахувань	
4.4 Розрахунок витрат на енергоспоживання	
4.5 Розрахунок заробітної плати програміста.....	
4.6 Розрахунок загальних витрат на створення та експлуатацію програми....	
4.7 Визначення відпускної ціни програми	
4.8 Економічна ефективність розробки і впровадження системи	
РОЗДІЛ 5 ОХОРОНА ПРАЦІ	
5.1 Вступ та теоретичні відомості охорони праці	
5.2 Загальні вимоги до приміщення.....	
5.3 Вимоги до організації робочого місця	
5.4 Аналіз параметрів робочого місця та приміщення	
5.5. Вимоги електробезпеки.....	
5.6 Пожежна безпека	
РОЗДІЛ 6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА	
6.1 Електромагнітне забруднення навколишнього середовища і шляхи захисту від нього	
6.2 Висновки.....	
ВИСНОВКИ.....	
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	
ДОДАТОК А.....	

ВСТУП

Завдання проектування комп'ютерних мереж залишається актуальною і в даний час. Розроблено велику кількість програмних продуктів для проектування комп'ютерних мереж (COMNET, OPNET, NetMaker та ін.) Вони не здатні вирішити всі завдання, оскільки одні дозволяють аналізувати мережі, інші орієнтовані на рішення задач проектування. Існуючі продукти не можуть також працювати з постійно з'являються новими технологіями і новими типами мереж. Дана робота присвячена розробляється в даний час системі автоматизованого проектування комп'ютерних мереж. В якості методу дослідження комп'ютерних мереж використовується метод імітаційного моделювання. За допомогою графічного редактора системи користувач може швидко спроектувати структуру комп'ютерної мережі з різних вбудованих елементів. Система дозволяє знаходити різні характеристики структури мережі, моделювати і аналізувати її роботу.

Дуже часто проектування описують як окремий етап розробки проекту між аналізом і розробкою. Проте насправді чіткого поділу етапів розробки проекту немає - проектування, як правило, не має явно вираженого початку і закінчення і часто продовжується на етапах тестування і реалізації. Говорячи про етап тестування, також слід зазначити, що і етап аналізу, і етап проектування містять елементи роботи тестерів, наприклад для отримання експериментального обґрунтування вибору того чи іншого рішення, а також для оцінки критеріїв якості одержуваної системи. На етапі експлуатації доречний розмову і про супровід системи.

Актуальність теми. У більшості випадків імітаційні моделі комп'ютерних систем і мереж використовують на етапі їх проектування для оцінювання прогнозованих характеристик. Під час створення імітаційної моделі мережі головним завданням проектувальника є ретельний аналіз роботи мережі, що складається з великої кількості окремих компонентів: комп'ютерів, комутаторів, маршрутизаторів, пристроїв введення-виведення, ліній зв'язку, і прийняття рішення стосовно вибору типу моделі та методів дослідження. Незважаючи на те, що імітаційне моделювання комп'ютерних мереж враховує багато факторів та забезпечує розв'язання низки складних завдань, таких як визначення типів і

параметрів активного і пасивного обладнання мережі; оцінювання необхідного запасу продуктивності основних пристроїв мережі; вибір архітектури мережі на базі критерію вартості обладнання; оцінювання середнього часу затримки передачі файлів у мережі; керування перевантаженням у мережі та ін., вони залишаються актуальними. Розвиток методів і технологій ймовірнісного моделювання процесів росту та структуризації комп'ютерних мереж стимулює інтерес дослідників до вивчення теорії складних мереж (Complex Networks), в рамках якої пропонуються підходи до розв'язання обчислювально складних задач, які є характерними для сучасних систем.

У відповідності з поставленою метою дипломна робота включає розв'язання таких задач:

- проведення аналізу основних типів мереж, їх структури та методів дослідження;
- здійснення системного аналізу методів та засобів імітаційного моделювання комп'ютерних мереж для прогнозування процесів їх росту, обґрунтування вимог до них, виходячи з огляду існуючих математичних моделей, розроблення на цій основі переліку актуальних та недосліджених задач з метою подальшого удосконалення та розробки нових методів їх науково обґрунтованих рішень;
- аналіз відомих підходів та розроблення моделі розвитку структури комп'ютерної мережі;
- застосування розробленого методу моделювання для аналізу, оцінювання і розвитку процесів стійкості комп'ютерних мереж до спрямованих хакерських атак та розповсюдження комп'ютерних вірусів у них.

Об'єкт дослідження – комп'ютерні мережі та процеси їх розвитку.

Предметом дослідження є імітаційні моделі, числові методи та програмні комплекси для вирішення задач аналізу становлення та розвитку комп'ютерних мереж.

Наукова новизна одержаних результатів. Наукова новизна роботи полягає у розв'язанні наукової задачі імітаційного моделювання реальних комп'ютерних мереж з урахуванням їх ймовірнісних характеристик.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ МЕТОДІВ МОДЕЛЮВАННЯ КОМП'ЮТЕРНИХ МЕРЕЖ

1.1 Поняття та цілі моделювання комп'ютерних мереж

Інформація та зв'язок є одними з найбільш важливих стратегічних питань для успіху кожного підприємства. У той час, як сьогодні майже кожна організація використовує значну кількість комп'ютерів і засобів зв'язку (наприклад телефон або факс), які і раніше часто ізольовані один від одного. Хоча робітники сьогодні мають можливість використовувати такі додатки, як тестові редактори або електронні таблиці, які здебільшого не мають інструментів для пошуку інформації або для зв'язку з іншими департаментами. Щоб подолати ці перешкоди для ефективного використання інформаційних технологій, необхідні комп'ютерні мережі.

Комп'ютерна мережа – це новий вид організації комп'ютерних систем, які створюються з необхідністю об'єднати комп'ютери та комунікації. Комп'ютерні мережі прибирають перешкоди між інформацією, яка раніше могла зберігатись лише на декількох комп'ютерних системах окремо. Тільки за допомогою комп'ютерних мереж можна побудувати систему без обмежень в доступі до інформації.

Комп'ютерні мережі дозволяють користувачеві отримувати доступ до віддалених програми і віддалених баз даних в рамках однієї організації або з інших підприємств або державних джерел. Комп'ютерні мережі забезпечують комунікаційні можливості швидше, ніж інші об'єкти. Через ці оптимальні інформаційні та комунікаційні можливості, комп'ютерні мережі можуть підвищити швидкість роботи з інформацією.

Комп'ютерні мережі мають ряд переваг:

1. Зниження витрат шляхом обміну апаратно-програмними ресурсами.
2. Висока надійність, маючи кілька джерел живлення.
3. Зниження витрат шляхом розподілення обчислень між комп'ютерами ніж придбання дорогого мейнфрейму.

4. Більша гнучкість через можливість підключення до мережі пристроїв різних виробників.

Завдяки координації всіх даних і додатків через єдину мережу, можливо створювати резервні копії даних до яких матимуть доступ всі комп'ютери мережі. Додаткове програмне забезпечення для всіх машин в мережі може бути встановлене за допомогою одного комп'ютера. Централізована система спрощує аспекти управління мережею. При вдало вибраному програмного забезпеченні, інформаційна безпека також може бути реалізована більш ефективно в мережі, ніж безпека інформації яка розподілена між багатьма окремими жорсткими дисками [1].

Основу інформаційної системи складає обчислювальна система, що включає такі компоненти, як кабельна мережа і активне мережеве обладнання, комп'ютерне та периферійне устаткування, обладнання зберігання даних (бібліотеки), системне програмне забезпечення (операційні системи, системи управління базами даних), спеціальне ПЗ (системи моніторингу та управління мережами) і в деяких випадках прикладне ПЗ.

Основу моделювання складають моделі обладнання та процесів (технологій, програмного забезпечення), використовуваних при роботі об'єкта, що цікавить. При моделюванні на комп'ютері відтворюються реальні процеси в обстежуваному об'єкті, досліджуються особливі випадки, відтворюються реальні і гіпотетичні критичні ситуації. Основною перевагою моделювання є можливість проведення різноманітних експериментів з досліджуванним об'єктом, не вдаючись до фізичної реалізації, що дозволяє передбачити і запобігти велике число несподіваних ситуацій у процесі експлуатації, які могли б призвести до невиправданих витрат, а може, і до псування обладнання.

У разі моделювання обчислювальних систем таким об'єктом є інформаційна система, що визначає способи отримання, зберігання, обробки та використання різної корпоративної та зовнішньої інформації.

У процесі моделювання можливо наступне:

- визначення мінімально необхідного, але забезпечуючого потреби передачі, обробки та зберігання інформації обладнання (навіть, що не має реальних аналогів) в даний час;

- оцінка необхідного запасу продуктивності обладнання, що забезпечує можливе збільшення виробничих потреб найближчим часом (один-два роки);
- вибір декількох варіантів обладнання з урахуванням поточних потреб, перспективи розвитку на підставі критерію вартості обладнання;
- проведення перевірки роботи обчислювальної системи, складеної з рекомендованого обладнання.

Якщо для задання інформації про топологію мережі не потрібно мати реальну мережу, то для збору вихідних даних про інтенсивност джерел мережного трафіку можуть знадобитися виміри на пілотних мережах, що представляють собою натурну модель проектованої мережі. Ці вимірювання можуть бути виконані різними засобами, в тому числі і за допомогою аналізаторів протоколів.

Крім отримання вихідних даних для імітаційного моделювання пілотна мережа може використовуватися для вирішення інших важливих завдань. Вона може дати відповіді на питання, які стосуються принципової працездатності того чи іншого технічного рішення або сумісності обладнання. Натурні експерименти можуть викликати значні матеріальні витрати, але вони компенсуються високою достовірністю отриманих результатів.

Пілотна мережа повинна бути схожа на ту мережу, яка проектується, для вибору параметрів якої і створюється пілотна мережа. Для цього необхідно в першу чергу виділити ті особливості створюваної мережі, які можуть найбільше вплинути на її працездатність і продуктивність.

Якщо є сумніви в сумісності продуктів різних виробників, наприклад, комутаторів, що підтримують віртуальні мережі, то в пілотній мережі повинні перевірятися на сумісність саме ці пристрої і саме в тих режимах, які викликають найбільші сумніви [2].

1.1. Стандартизація у комп'ютерних мереж

Комп'ютерна мережа – система зв'язку між двома чи більше комп'ютерами. У ширшому розумінні комп'ютерна мережа (КМ) – це система зв'язку через кабельне чи повітряне середовище, самі комп'ютери різного функціонального

призначення і мережеве обладнання. Для передачі інформації можуть бути використані різні фізичні явища, як правило – різні види електричних сигналів чи електромагнітного випромінювання. Середовищами передавання у комп'ютерних мережах можуть бути телефонні кабелі, та спеціальні мережеві кабелі: коаксіальні кабелі, виті пари, волоконно-оптичні кабелі, радіохвилі, світлові сигнали.

Комп'ютерна мережа забезпечує:

1. Колективне опрацювання даних користувачами.
2. Обмінювання файлами та іншими даними між користувачами
3. Спільне використовувати програми
4. Спільне використання принтерів, модемів та ін. [1, с. 34 – 35].

Деякі з найбільш визначальних досягнень сталося в середині 1970-х років впровадженням IBM архітектури мережевих систем, власної набору високостабільних протоколів для мереж мейнфреймів і систем середнього радіусу дії, були досягнуто декілька важливих всій галузі стандартів, що розчистили шлях для широкого впровадження комп'ютерних мереж. Першим з них був дебют Інституту інженерів з електротехніки та радіоелектроніки »(IEEE) 802.x серії стандартів, яка передбачала технічні характеристики для різних типів обміну даними в мережі. Стандарт IEEE, які періодично оновлюються і розширюються, які використовуються і сьогодні. Далі, загальна модель архітектури яку називають взаємозв'язком відкритих систем (OSI) був прийнятий Міжнародною організацією зі стандартизації (ISO). Хоча OSI широка модель, це дало змогу розробникам мереж використовувати міжнародну прийняту класифікацію різних функцій і процесів в мережі і як вони повинні працювати разом. OSI і IEEE стандарти доповнюють один одного.

Принципово можна виокремити два види обробки даних засобами мережевих технологій: локальні комп'ютерні мережі ЛКМ (LAN - Local Area Networks) та глобальні комп'ютерні мережі ГКМ (WAN - Wide Area Networks).

Одною з визначальних ознак локальної мережі є наявність високошвидкісного каналу зв'язку, яка на порядок перевищує швидкість передачі даних до периферійних пристроїв. Ця швидкість наближається до швидкості передачі даних системною шиною комп'ютера. В таких мережах

швидкість обробки даних залежить в основному від швидкодії комп'ютерів, а не від швидкості передачі даних у каналі зв'язку. Якщо трактувати локальну мережу як один великий комп'ютер, який складається з багатьох окремих комп'ютерів, то цю систему можна з певними припущеннями назвати розподіленою інформаційною системою (РІС).

Розрізняють три ступені використання РІС:

1. розподілення ресурсів (пам'ять, дисковий простір, принтери). Таке використання РІС зараз найпоширеніше;
2. розподіл навантаження. Задачі, які потрапляють у систему, надходять на вільні комп'ютери;
3. розподіл опрацювання даних. Йдеться про сукупність елементів обробки даних під логічно та фізично децентралізованим управлінням ресурсами з метою спільного використання прикладного програмного забезпечення.

Найдорожчими в локальних мережах є пристрої обробки інформації (комп'ютери), а не комунікаційні пристрої. Ефективність локальних мереж визначається ефективністю використання прикладного програмного забезпечення [3, с. 119 - 120]

В ГКМ найдорожчий елемент - обладнання та лінії зв'язку, які визначають ефективність та швидкість функціонування таких мереж. Швидкість обробки даних в комп'ютері тут практично несуттєва. Різницю між можливостями локальних та глобальних мереж з погляду швидкості обробки даних в майбутньому можливо вдасться ліквідувати за допомогою технології АТМ (Asynchronous Transfer Mode - режим асинхронного передавання) [2].

Останнім часом в зв'язку зі швидким розвитком мережевих технологій та розширенням сфери їх застосування в літературі можна зустріти випадки виділення в окрему групу регіональних та корпоративних комп'ютерних мереж.

Регіональні комп'ютерні мережі (РКМ) (MAN - Metropolitan Area Networks) - це мережі масштабів району, міста, області, регіону. В залежності від конкретної реалізації ці мережі можуть створюватись і функціонувати на засадах локальних чи глобальних мереж. Корпоративні комп'ютерні мережі (ККМ) - це

об'єднання кількох ЛКМ одної організації чи підприємства у єдину мережу за допомогою ліній зв'язку ГKM.

Створення, розвиток та поширення використання комп'ютерних мереж були б неможливими без використання однакових правил передачі та обробки даних - стандартів.

Основними організаціями, які розробляють та сприяють впровадженню стандартів в галузі комп'ютерних мереж є:

1. Міжнародний телекомунікаційний союз (МТС) (ITU - International Telecommunication Union). Стандарти МТС - 10 - поділяються на серії. Стандарти кожної серії присвячені одній тематиці і позначаються великою літерою латинського алфавіту. Після літери ставиться крапка і номер стандарту. Так, наприклад, літерою V позначаються стандарти щодо передавання даних телефонними каналами, літерою X - стандарти щодо мереж передавання даних, літерою Q - стандарти щодо телефонної комутації та сигналізації.

2. Технічний комітет 97 - комітет міжнародної організації зі стандартизації (ISO - International Standard Organization). Технічний комітет розробляє стандарти щодо опрацювання інформації за допомогою ЕОМ. Стандарти цієї організації позначаються чотиризначним числом та суфіксом ISO. Так, наприклад, стандарт, який стосується протокольного стеку TCP/IP має позначення 7498 ISO.

3. Комісія з питань діяльності Internet (IAB - Internet Activities Board) - розробляє стандарти щодо діяльності Internet. Дотримання стандартів, які розроблені цими організаціями, обов'язкове під час проектування, створення та експлуатації будь-яких комп'ютерних мереж [3, с. 112 - 113].

1.2 Особливості проектування комп'ютерної мережі

Процес проектування об'єкта - комп'ютерної мережі носить ітераційний характер. Ітерації можуть включати в себе більш ніж один рівень проектування. Тобто, в процесі проектування доводиться багаторазово виконувати процедуру аналізу об'єкта. Тому очевидно прагнення зменшити трудомісткість кожного варіанту аналізу без шкоди для якості остаточного проекту. У цих умовах

доцільно на початкових стадіях проектування, коли високою точності результатів не потрібно, використовувати найбільш прості і економічні моделі [1].

Створити проект мережі означає вибрати структуру мережі, визначити значення всіх його параметрів і представити результати у встановленій формі. Результати (проектна документація) можуть бути виражені у вигляді схем, пояснювальних записок, програм та інших документів на папері або на машинних носіях інформації.

Розробка (або вибір) структури мережі є проектна процедура структурного синтезу, а розрахунок (або вибір) значень параметрів елементів процедура параметричного синтезу.

Завдання структурного синтезу полягає у визначенні мети, безлічі можливих рішень і обмежуючих умов. Завдання полягає в синтезі (або корекції) структури, визначенні типів серверів (програмно-апаратних засобів), розподілі функцій по серверам таким чином, щоб досягався екстремум цільової функції при виконанні заданих обмежень.

Наступна після синтезу група проектних процедур – процедури аналізу. Мета аналізу – отримання інформації про характер функціонування і значеннях вихідних параметрів при заданих структурі об'єкта, відомостях про зовнішніх параметрах і параметрах елементів.

Для аналізу комп'ютерної мережі широко використовуються математичні методи і моделі масового обслуговування. Спрощені моделі масового обслуговування дозволяють знаходити явний вид цільової функції, в якості якої використовується характеристика мережі, така як час затримки повідомлень (пакетів).

Моделювання життєвого циклу комп'ютерної мережі проводиться за ітеративної моделі – це виконання робіт паралельно з безперервним аналізом отриманих результатів і коригуванням попередніх етапів роботи. Комп'ютерна мережа при цьому підході в кожній стадії проходить повторюваний цикл: планування - реалізація - перевірка – оцінка (англ. plan-do-check-act cycle) [4, с. 56 - 59]

В автоматизованих проектних процедурах замість проектованого ще не існуючого об'єкта оперують моделлю, яка відображає деякі цікаві для дослідника властивості об'єкта.

Математичні функціональні моделі в загальному випадку являє собою алгоритм обчислення вектора вихідних параметрів при заданих векторах параметрів елементів і зовнішніх параметрів.

Закон функціонування мережі може бути представлений в наступному вигляді:

$$H(t) = f_c(S, F, Y, X, t),$$

де f_c – функція, функціонал, логічні умови, алгоритм, методика, таблиця або словесний опис, визначальне правило (закон) перетворення вхідних величин (параметрів) у вихідні величини (характеристики);

$H(t)$ - вектор характеристик, що залежить від поточного моменту часу t ($t \geq 0$):

$$H = \{V, T, N, C, Z\}$$

Параметри – первинні дані мережі:

S - структурні,

F - функціональні,

Y - навантажувальні,

X - зовнішнього середовища.

Характеристики - вторинні дані мережі:

V – потужність ,

T - тимчасові,

N - надійнісні,

C - економічні,

Z – інші [5, с. 101 – 104].

1.3 Характеристики комп'ютерної мережі

Поняття характеристики функціонування мережі в роботі включає вторинні властивості комп'ютерної мережі, які визначаються в процесі вирішення завдань аналізу, як функція параметрів. Параметри комп'ютерної

мережі описують первинні властивості мережі і є вихідними даними при вирішенні задач аналізу.

Характеристики комп'ютерних мереж - це сукупність показників ефективності (якості) мережі. Характеристики комп'ютерних мереж можна розділити на якісні та кількісні [4].

Кількісні характеристики комп'ютерних мереж можна розділити на дві групи:

1. Глобальні, що визначають найбільш важливі властивості мережі як цілісного об'єкта;
2. Локальні, що визначають властивості окремих пристроїв або частин мережі і дозволяють отримати більш детальне уявлення про ефективності мережі.

До глобальних належать характеристики продуктивності, оперативності, надійності, вартісні, інші (енергоспоживання, масогабаритні і т.п.).

Продуктивність комп'ютерної мережі - міра потужності мережі, визначальна кількість роботи, що виконується мережею в одиницю часу. Поняття продуктивності охоплює широку номенклатуру показників ефективності комп'ютерної мережі, що визначають якість функціонування як мережі в цілому, так і окремих її підсистем і елементів - технічних і програмних засобів [6, с. 220].

Продуктивність мережі залежить, в першу чергу, від продуктивності окремих її елементів, званої швидкістю роботи або швидкодією пристроїв, наприклад, швидкість передачі даних по каналах зв'язку, вимірювана об'ємом даних, переданих за одиницю часу, швидкодія ЕОМ або, точніше, процесора, вимірюваний числом команд, виконуваних в одиницю часу, і т.п. для оцінки продуктивності комп'ютерної мережі в цілому використовується наступна сукупність показників:

- Продуктивність СТК (мережі передачі даних), яка вимірюється числом повідомлень (пакетів, кадрів, біт) переданих по мережі за одиницю часу;

- продуктивність СВТ (засобів обробки даних), представляє собою сумарну продуктивність всіх засобів ВТ (ЕОМ і систем), що входять до складу мережі.

Продуктивність СТК (комунікаційна потужність) може бути задана наступними показниками:

- Максимальна або гранична продуктивність, звана пропускною здатністю мережі передачі даних і вимірювана кількістю пакетів (кадрів), переданих в мережі за одиницю часу;

- Реальна чи фактична продуктивність мережі передачі даних, яка може бути задана як середнє значення на деякому інтервалі часу або як миттєве значення в конкретний момент часу.

Продуктивність СВТ (обчислювальна потужність) в цілому складається з производительностей ВС, що виконують обробку даних у мережі. Найбільш важливим показником продуктивності ВС, як сукупності технічних і програмних засобів, є системна продуктивність λ_0 , вимірювана числом завдань, що виконуються системою за одиницю часу.

Характеристики оперативності описують затримки, що виникають при передачі і обробці даних у мережі. Для оцінки оперативності мережі в цілому використовуються наступні показники: час доставки пакетів (повідомлень), час відгуку (відповіді) [7, с. 89 – 90].

Час доставки (час затримки) пакетів характеризує ефективність організації передачі даних в обчислювальній мережі і являє собою інтервал часу, вимірюваний від моменту надходження пакета або повідомлення в мережу до моменту отримання пакета адресатом. У загальному випадку, час затримки - величина випадкова, що обумовлено випадковим характером процесів надходження і передачі даних у мережі.

У комп'ютерних мережах зазвичай час доставки задається середнім значенням T , на яке може накладатися обмеження $T < T^*$ в залежності від типу переданих даних.

При передачі мультимедійних даних крім середнього значення часу доставки пакетів важливою характеристикою є варіація або джиттер затримки, що представляє собою среднеквадратическое відхилення часу затримки різних пакетів.

Час відгуку (відповіді) - інтервал часу від моменту надходження запиту (повідомлення, транзакції) в мережу до моменту завершення його обслуговування, пов'язаного з виконанням деякої прикладної або обслуговуючої програми, зі зверненням до бази даних і т.п.

Час відповіді являє собою час перебування запиту в мережі і характеризує ефективність як телекомунікаційних, так і обчислювальних засобів комп'ютерної мережі.

Час відгуку, як і час затримки, - величина випадкова і може задаватися середнім значенням U або у вигляді ймовірності $P(t_u \leq U^*)$ неперевикнення деякого заданого значення U^* .

У мережах реального часу замість терміна "час відповіді" часто використовують термін "час реакції".

В якості характеристик надійності зазвичай використовуються наступні показники:

1. Ймовірність безвідмовної роботи мережі $P(t)$ - ймовірність того, що протягом часу t не відбудеться відмови;
2. Інтенсивність відмов λ_0 - середнє число відмов за одиницю часу;
3. Час напрацювання на відмову - проміжок часу між двома суміжними відмовами - величина випадкова, а її середнє значення T_0 називається середньою напрацюванням на відмову $T_0 = 1 / \lambda_0$;
4. Час відновлення - інтервал часу від моменту настання відмови до моменту відновлення працездатності системи - величина випадкова і зазвичай задається середнім значенням t_v , званім середнім часом відновлення;
5. Коефіцієнт готовності K_g - частка часу, протягом якого мережа працездатна: $K_g = T_0 / (T_0 + T_v)$.

Величина K_g може трактуватися як ймовірність того, що в будь момент часу мережа працездатна. Аналогічно, значення $(1-K_g)$ визначає ймовірність того, що мережа знаходиться в стані відновлення (непрацездатна) [8, с. 200 – 207]

В якості вартісних (економічних) характеристик комп'ютерної мережі можуть використовуватися такі показники:

1. Повна вартість володіння (Total cost of ownership, TCO) - витрати, розраховуються на всіх етапах життєвого циклу мережі і включають вартість технічних, інформаційних та програмних засобів (прямі витрати) і витрати на експлуатацію мережі (непрямі витрати);

2. Вартість (ціна) передачі даних та обробки даних у мережі, обумовлена обсягом і вартістю використовуваних ресурсів мережі відповідно при передачі і обробці даних.

В якості локальних характеристик комп'ютерних мереж можуть використовуватися в залежності від цілей дослідження самі різноманітні показники ефективності.

Локальні характеристики описують ефективність функціонування:

1. Вузлів і каналів зв'язку;
2. Окремих сегментів мережі;
3. Вузлів обробки даних: обчислювальної системи та її підсистем.

Локальні характеристики можуть бути розбиті на дві групи: тимчасові; безрозмірні. До тимчасових характеристик відносяться:

1. Час доставки (затримки) пакетів при передачі між сусідніми вузлами мережі;
2. Час очікування передачі даних у вузлах мережі або звільнення ресурсів ВС (сервера);
3. Час перебування даних у різних вузлах, пристроях або підсистемах.

До безрозмірним характеристик відносяться:

1. Число пакетів, що знаходяться в буферній пам'яті вузлів (маршрутизаторів, комутаторів);
2. Коефіцієнти завантажень вузлів, каналів зв'язку і пристроїв ПС і т.д.

Коефіцієнт завантаження або просто завантаження ρ пристрою це – частка часу, протягом якого пристрій працює: $\rho = \lim_{T \rightarrow \infty} \frac{t}{T}$, де t - час, протягом якого пристрій працював; T – час спостереження.

Завантаження ρ характеризує ступінь використання пристрою і часто називається коефіцієнтом використання пристрою. Оскільки $0 \leq \rho \leq 1$, то завантаження може трактуватися як ймовірність того, що в будь-який момент часу пристрій працює. Величина $\eta = 1 - \rho$ називається коефіцієнтом простою пристрою і характеризує частку часу, в протягом якого пристрій не працює (простоює) [9, с. 310 – 313]

1.4 Вплив на мережеві характеристики програмного і апаратного складу комп'ютерних мереж

Сучасні досягнення виробників програмного і апаратного складу засобів обчислювальної техніки та мережевого обладнання суттєво впливають на можливості зміни структур комп'ютерної мережі (КС).

Виробники комутаційного обладнання створюють нові технології, що істотно змінюють можливості апаратно програмних засобів побудови комп'ютерної мережі і тим самим що значно збільшує кількість і характер можливих для реалізації структур комп'ютерної мережі.

Можна виділити компоненти, що мають найбільшу вагу в процесі впливу на мережеві характеристики:

1. Розробки фірм-виробників апаратної частини комутаційного обладнання (керовані і некеровані switches L2, L2 +, L3, L3 + рівнів),
2. Розробки засобів віртуалізації фірм – виробників прикладного програмного забезпечення (VMWare, HyperVisor),
3. Розробки програмного забезпечення фірм-виробників комутаційного обладнання (реалізації технологій VLAN, Link Aggregation, Spanning Tree, RSTP, MSTP, Loop Guard, ARP Spoofing, DHCP Snooping, VRRP, ...) [10, с. 238]

Зміна структури КС, в свою чергу, неоднозначно впливають на мережеві характеристики. Набір апаратних засобів і ресурсів комп'ютерної мережі, відповідних першому, другому, третьому рівням семиуровневої моделі міжмережевого взаємодії ISO / OSI, включає в себе мережеві інтерфейси, структуровану кабельну систему, різне комутаційне обладнання, що дозволяє направляти потоки даних між користувачами. З наявного набору коштів є можливість створити певну кількість структур комп'ютерної мережі: $S_1, S_2, S_3, \dots S_k$..., де:

S_k - структура з k кількістю вузлів і конкретної фізичної топологією структури S_i розрізняються кількістю вузлів і способом взаємодії між ними, утворюючи певну фізичну топологію $S_i, i = 1, k$.

Фірми-виробники комутаційного обладнання пропонують рішення на базі апаратно-програмних можливостей своїх продуктів, які можна назвати типовими структурами комп'ютерної мережі: $S_i, i = 1, n; n < k$.

Крім типових фірмових структур, можуть бути запропоновані структури на основі аналітичного та імітаційного моделювання та методики аналізу характеристик мережі цілеспрямованою зміною структури мережі, запропонованої в даній роботі: $S_i, i = (n + 1, k); n < k$.

Виробники програмного забезпечення комутаційного обладнання пропонують кошти з використанням технології віртуальних локальних мереж VLAN, стандартів функцій і протоколів фірм-виробників, які також при налаштуванні помітно змінюють потоки в комп'ютерній мережі і характеристики мережі. Зокрема додавання однієї або декількох VLAN кардинально перерозподіляють потоки даних у мережі з однією і тією ж фізичною топологією. З'являється необхідність додаткових досліджень конкретного варіанту мережі з раціонального використання програмного забезпечення комутаційного обладнання. Функції, які можуть бути налаштовані на комутаційному обладнанні або в мережі пропонуються фірмами-виробниками для підвищення продуктивності і надійності комп'ютерної мережі. До них відносяться: Технології Віртуальних Мереж VLAN, Підтримка протоколу Spanning Tree, Об'єднання (агрегування) портів, Підтримка SNMP - управління, Port Security або прив'язка MAC-адреси до певного порту, Підтримка 802.1x, Протоколи групового мовлення, управління потоком, Обмеження смуги пропускання, Налаштування IP-доменів на комутаторах, Використання класифікатора і політики, застосування OSPF та ін. Технології та функції, перераховані вище, збільшують число і характер структур комп'ютерної мережі S_i .

$S_i, i = (k, f); k < f$.

Виробники прикладного програмного забезпечення пропонують засоби віртуалізації, такі як додатки VMWare, створені на її основі віртуальні (гостьові) робочі станції і сервери, віртуальні мережі сервери і світчі. Віртуальні (гостьові) операційні системи, віртуальні сервери і світчі мають можливість взаємодіяти між собою і з материнськими (хостової) операційними системами і фізичними

комп'ютерними мережами за допомогою вбудованих засобів комутації та маршрутизації.

Використання різних засобів прикладної віртуалізації значно змінює кількість і характер можливих для реалізації структур комп'ютерної мережі S_i .

Таким чином, при додатковій налаштуванні або адмініструванні однієї і тієї ж структури S_k комп'ютерної мережі з однаковими постійними вкладеннями можна отримати кілька варіантів S_k структур, з іншими характеристиками мережі, одна з яких є найкращою за заданим критерієм S_k

Зміна структурно-функціональної організації комп'ютерної мережі призводить до поліпшення одних показників ефективності функціонування мережі і до погіршення інших показників мережі, що суттєво ускладнює вибір найкращого варіанта структурно функціональної організації проектованої комп'ютерної мережі, тому показники ефективності мережі є суперечливими.

При некоректному застосуванні технології VLAN, використанні різних засобів прикладної віртуалізації, додаткового налаштування або адмініструванні однієї і тієї ж структури S_k комп'ютерної мережі з однаковими постійними вкладеннями можна погіршити працездатність мережі. З'являється необхідність додаткових досліджень конкретного варіанти мережі по раціональному використанню програмного забезпечення комутаційного обладнання.

Отже, виникає завдання зміни структури мережі, підвищення характеристик продуктивності та оперативності, що не погіршуючи надійності мережі.

Вкладення дають можливість урізноманітнити структури фізично і логічно. Наприклад, функція вкладень створює безліч структур S_i : $f(c) = \{S1_{\text{фізичне}}, \dots, S1_{\text{логічне}} \dots\}$. Нові структури дадуть деякі характеристики $H = f(S1_{\text{фізичне}}, \dots, S1_{\text{логічне}} \dots)$. потрібно знайти раціональний шлях аналізу характеристик функціонування структури мережі, при обмеженні - забезпеченні надійності не нижче необхідної [11, с. 70 – 79].

Фізичний захист даних комп'ютерних мереж полягає в наступному.

Найкращим способом позбавити себе проблем з приводу неправильної прокладки кабелю є використання набули широкого поширення останнім часом так званих структурованих кабельних систем, що використовують однакові

кабелі для передачі даних у локальній обчислювальній мережі, локальної телефонної мережі, передачі відеоінформації чи сигналів від датчиків пожежної безпеки або охоронних систем. До структурованих кабельних систем відносяться, наприклад, SYSTIMAX SCS фірми AT & T, OPEN DEConnect компанії Digital, кабельна система корпорації IBM.

Поняття “структурованість” означає, що кабельну систему будівлі можна розділити на декілька рівнів в залежності від призначення і місцезнаходження компонентів кабельної системи. Наприклад, кабельна система SYSTIMAX SCS складається з:

- Некоректне використання програмного і апаратного забезпечення, що веде до знищення чи зміни даних.

- Зовнішньої підсистеми (campus subsystem)

- Апаратних (equipment room)

- Адміністративної підсистеми (administrative subsystem)

- Магістралі (backbone cabling)

- Горизонтальної підсистеми (horizontal subsystem) [12, с. 120 - 122]

Зовнішня підсистема складається з мідного і оптоволоконного кабелю, пристроїв електричного захисту та заземлення і пов'язує комунікаційну та обробну апаратуру в будівлі (або комплексі будівель). Крім того, в цю підсистему входять пристрої сполучення зовнішніх кабельних ліній з внутрішніми.

Апаратні служать для розміщення різного комунікаційного обладнання, призначеного для забезпечення роботи адміністративної підсистеми.

Адміністративна підсистема призначена для швидкого і легкого управління кабельною системою SYSTIMAX SCS при зміні планів розміщення персоналу і відділів. До її складу входять кабельна система (неекранована вита пара і оптоволоконно), пристрої комутації і сполучення магістралі і горизонтальної підсистеми, з'єднувальні шнури, маркувальні засоби і т.д.

Магістраль складається з мідного кабелю або комбінації мідного і оптоволоконного кабелю і допоміжного обладнання. Вона пов'язує між собою поверхи будівлі або великі площі одного і того ж поверху.

Горизонтальна система на базі крученого мідного кабелю розширює основну магістраль від вхідних точок адміністративної системи поверху до розеток на робочому місці.

І, нарешті, обладнання робочих місць включає в себе з'єднувальні шнури, адаптери, пристрою сполучення і забезпечує механічне та електричне з'єднання між обладнанням робочого місця і горизонтальної кабельної підсистемою.

Найкращим способом захисту кабелю від фізичних (а іноді і температурних і хімічних впливів, наприклад, у виробничих цехах) є прокладання кабелів з використанням у різного ступеня захищених коробів. При прокладці мережевого кабелю поблизу джерел електромагнітного випромінювання необхідно виконувати наступні вимоги:

а) неекранована вита пара повинна відстояти мінімум на 15-30 см від електричного кабелю, розеток, трансформаторів і т. д.

б) вимоги до коаксіального кабелю менш жорсткі – відстань до електричної лінії або електроприладів повинно бути не менше 10-15 см.

Інша важлива проблема правильної інсталяції та безвідмовної роботи кабельної системи – відповідність всіх її компонентів вимогам міжнародних стандартів [13, с. 13 - 19].

Найбільшого поширення в даний час отримали такі стандарти кабельних систем:

Специфікації корпорації IBM, які передбачають дев'ять різних типів кабелів. Найбільш поширеним серед них є кабель IBM type 1 – екранована вита пара (STP) для мереж Token Ring.

Система категорій Underwriters Labs (UL) представлена цією лабораторією спільно з корпорацією Anixter. Система включає п'ять рівнів кабелів. В даний час система UL приведена у відповідність з системою категорій EIA / TIA.

Стандарт EIA / TIA 568 був розроблений спільними зусиллями UL, American National Standards Institute (ANSI) і Electronic Industry Association / Telecommunications Industry Association, підгрупою TR41.8.1 для кабельних систем на кручений парі (UTP).

На додаток до стандарту EIA / TIA 568 існує документ DIS 1 180i, розроблений International Standard Organisation (ISO) і International

Electrotechnical Commission (IEC). Даний стандарт використовує термін “категорія” для окремих кабелів і термін “клас” для кабельних систем.

Необхідно також зазначити, що вимоги стандарту EIA/TIA 568 відносяться тільки до мережевого кабелю. Але реальні системи, крім кабелю, включають також з'єднувальні роз'єми, розетки, розподільні панелі і інші елементи. Використання тільки кабелю категорії 5 не гарантує створення кабельної системи цієї категорії. У зв'язку з цим все перераховане вище обладнання повинно бути також сертифіковане на відповідність даної категорії кабельної системи.

Організація надійної та ефективної системи архівації даних є однією з найважливіших завдань щодо забезпечення збереження інформації в мережі. У невеликих мережах, де встановлені один-два сервери, найчастіше застосовується установка системи архівації безпосередньо у вільні слоти серверів. У великих корпоративних мережах найбільш переважно організувати виділений спеціалізований архіваційний сервер.

Такий сервер автоматично виробляє архівування інформації з жорстких дисків серверів і робочих станцій у вказане адміністратором локальної обчислювальної мережі час, видаючи звіт про проведене резервну копіювання. При цьому забезпечується управління всім процесом архівації з консолі адміністратора, наприклад, можна вказати конкретні томи, каталоги або окремі файли, які необхідно архівувати. Можлива також організація автоматичного архівування по настанні тієї чи іншої події (“event driven backup”), наприклад, при отриманні інформації про те, що на жорсткому диску сервера або робочої станції залишилося мало вільного місця, або при виході з ладу одного з “дзеркальних” дисків на файловому сервері. Серед найбільш поширених моделей архіваційних серверів можна виділити Storage Express System корпорації Intel, ARCserve for Windows, виробництва фірми Cheyenne і ряд інших.

Зберігання архівної інформації, що представляє особливу цінність, має бути організовано у спеціальному приміщенні, що охороняється. Фахівці рекомендують зберігати дублікати архівів найбільш цінних даних в іншому будинку, на випадок пожежі або стихійного лиха. Для забезпечення відновлення даних при збої магнітних дисків останнім часом найчастіше застосовуються системи дискових масивів – групи дисків, що працюють як єдиний пристрій,

відповідних стандартам RAID (Redundant Arrays of Inexpensive Disks). Ці масиви забезпечують найбільш високу швидкість запису / зчитування даних, можливість повного відновлення даних і заміни вийшли з ладу дисків в “гарячому” режимі (без відключення інших дисків масиву).

Організація дискових масивів передбачає різні технічні рішення, реалізовані на декількох рівнях.

Рівень 0 передбачає просте розділення потоку даних між двома або кількома дисками. Перевага подібного рішення полягає у збільшенні швидкості введення / виводу пропорційно кількості задіяних у масиві дисків. У той же час таке рішення не дозволяє відновити інформацію при виході з ладу одного з дисків масиву.

RAID рівня 1 полягає в організації так званих “дзеркальних” дисків. Під час запису даних інформація основного диска системи дублюється на дзеркальному диску, а при виході з ладу основного диска в роботу відразу включається “дзеркальний”.

Рівні 2 і 3 передбачають створення так званих паралельних дискових масивів, при записі на які дані розподіляються по дисках на бітовому рівні. Спеціальний диск виділяється для збереження надлишкової інформації, яка використовується для відновлення даних при виході з ладу будь-якого з дисків масивів.

Рівні 4 і 5 є модифікацією нульового рівня, при якому потік даних розподіляється по дисках масиву. Відмінність полягає в тому, що на рівні 4 виділяється спеціальний диск для зберігання надлишкової інформації, а на рівні 5 надлишкова інформація розподіляється по всіх дисках масиву. Організація дискових масивів у відповідності зі стандартом 5 рівня забезпечує високу швидкість читання / запису інформації і дозволяє відновлювати дані при збої будь-якого диска без відключення усього дискового масиву.

Серед всіх перерахованих вище рівнів дискових масивів рівні 3 та 5 є найбільш переважними і припускають менші в порівнянні з організацією “дзеркальних” дисків матеріальні витрати при тому ж рівні надійності.

Підвищення надійності і захист даних в мережі, заснована на використанні надлишкової інформації, реалізуються не тільки на рівні окремих елементів

мережі, наприклад дискових масивів, але і на рівні мережевих ОС. Так, протягом останніх десяти років компанія Novell реалізує відмовостійкі версії операційної системи Netware – SFT (System Fault Tolerance), що передбачають три основних рівня:

- SFT Level I. Перший рівень передбачає, зокрема, створення додаткових копій FAT і Directory Entries Tables, негайну верифікацію кожного знову записаного на файловий сервер блоку даних, а також резервування на кожному жорсткому диску близько 2% від обсягу диска. При виявленні збою дані перенаправляються в зарезервовану область диска, а зіпсований блок позначається як “поганий” і надалі не використовується.

- SFT Level II містила додатково можливості створення “дзеркальних” дисків, а також дублювання дискових контролерів, джерел живлення і інтерфейсних кабелів.

- Версія SFT Level III дозволяє використовувати в локальній мережі дубльовані сервери, один з яких є “головним”, а другий, що містить копію всієї інформації, вступає в роботу у разі виходу “головного” сервера з ладу [14, с. 43 – 51]

1.5 Аналіз готових рішень

Існують спеціальні, орієнтовані на моделювання обчислювальних мереж програмні системи, в яких процес створення моделі спрощений. Такі програмні системи самі генерують модель мережі на основі вихідних даних про її топології і використовуваних протоколах, про інтенсивностях потоків запитів між комп'ютерами мережі, протяжності ліній зв'язку, про типи використовуваного обладнання та додатків. Програмні системи моделювання можуть бути вузько спеціалізованими і досить універсальними, дозволяють імітувати мережі найрізноманітніших типів. Якість результатів моделювання в значній мірі залежить від точності вихідних даних про мережі, переданих в систему імітаційного моделювання.

Програмні системи моделювання мереж - інструмент, який може стати в нагоді будь-якому адміністратору корпоративної мережі, особливо при

проектуванні нової мережі або внесенні кардинальних змін у вже існуючу. Продукти даної категорії дозволяють перевірити наслідки впровадження тих чи інших рішень ще до оплати придбаного обладнання. Звичайно, більшість з цих програмних пакетів коштують досить дорого, але і можлива економія може бути теж вельми відчутною.

Програми імітаційного моделювання мережі використовують у своїй роботі інформацію про просторове розташування мережі, зокрема вузлів, конфігурації зв'язків, швидкостях передачі даних, протоколи, що використовуються і типи обладнання, а також про додатки, що виконуються в мережі.

Зазвичай імітаційна модель будується не з нуля. Існують готові імітаційні моделі основних елементів мереж: найбільш поширених типів маршрутизаторів, каналів зв'язку, методів доступу, протоколів і т.п. Ці моделі окремих елементів мережі створюються на підставі різних даних: результатів тестових випробувань реальних пристроїв, аналізу принципів їх роботи, аналітичних співвідношень. У результаті створюється бібліотека типових елементів мережі, які можна налаштовувати за допомогою заздалегідь передбачених у моделях параметрів.

Системи імітаційного моделювання зазвичай включають також набір засобів для підготовки вихідних даних про досліджувану мережі - попередньої обробки даних про топологію мережі і измеренном трафіку. Ці кошти можуть бути корисні, якщо моделювана мережа являє собою варіант існуючої мережі і є можливість провести в ній вимірювання трафіку та інших параметрів, потрібних для моделювання. Крім того, система забезпечується коштами для статистичної обробки отриманих результатів моделювання.

За функціональністю системи моделювання, що використовуються при дослідженні обчислювальних систем, можуть бути розбиті на два основні класи:

- системи, що моделюють окремі елементи (компоненти) системи;
- системи, що моделюють обчислювальну систему цілком.

У наступній таблиці (табл. 1.1) наведено характеристики декількох популярних систем імітаційного моделювання різного класу - від простих програм, призначених для установки на персональному комп'ютері, до потужних

систем, що включають бібліотеки більшості наявних на ринку комунікаційних пристроїв і дозволяють в значній мірі автоматизувати дослідження досліджуваної мережі [15, с. 65]

Таблиця 1.1.

Характеристики систем імітаційного моделювання

Компанія та продукт	Вартість (дол.)	Тип мережі	Необхідні ресурси	Примітки
American HYTech, Prophesy	1495	ЛС	8МбОП, 6 Мбдиск, DOS, Windows, OS/2	Оцінювання продуктивності при роботі з текстовими і графічними даними по окремих сегментах і мережі в цілому
CACI Product, COMNET III	34500-39500	ЛС, ГС	32 МбОП, 100 Мбдиск, Windows, Windows NT, OS/2, Unix	Моделює мережі X.25, ATM, Frame Relay, зв'язку LAN-WAN, SNA, DECnet, протоколи OSPF, RIP. Доступ CSMA / CD і токеном доступ, FDDI та ін. Вбудована бібліотека маршрутизаторів 3COM, Cisco, DEC, HP, Wellfleet.
Make System, NetMaker XA	6995-14995	ЛС, ГС	128 МбОП, 2000 Мбдиск, AIX, Sun OS, Sun Solaris	Перевірка даних про топологію мережі; імпорт інформації про трафік, що одержується в реальному часі.
NetMagic System, StressMagik	2995	ЛС	2 МбОП, 8 Мбдиск, Windows	Підтримка стандартних тестів виміру продуктивності; імітація пікового навантаження на файл-сервер
Network Analysis Center, MIND	9400-70000	ГС	8 МбОП, 65 Мбдиск, DOS, Windows	Засіб проектування, оптимізації мережі, містить дані про вартість типових конфігурацій з можливістю точного оцінювання продуктивності.
Network Design and Analysis Group, AutoNet/ Designer	25000	ГС	8 МбОП, 40 Мбдиск, Windows, OS/2	Визначення оптимального розташування концентратора в ГС, можливість оцінки економії коштів за рахунок зниження тарифної плати, зміни постачальника послуг і оновлення обладнання; порівняння варіантів зв'язку через найближчу і

				оптимальну точку доступу, а також через міст і місцеву телефонну мережу
Network Design and Analysis Group, AutoNet/ MeshNET	30000	ГС	8 МбОП, 40 Мбдиск, Windows, OS/2	Моделювання смуги пропускання і оптимізація витрат на організацію ГС шляхом імітації пошкоджених ліній, підтримка тарифної сітки компаній AT & T, Sprint, WiTel, Bell
Network Design and Analysis Group, AutoNet/ Performance-1	4000	ГС	8 МбОП, 1 Мбдиск, Windows, OS/2	Моделювання продуктивності ієрархічних мереж шляхом аналізу чутливості до тривалості затримки, часу відповіді, а також вузьких місць в структурі мережі
Network Design and Analysis Group, AutoNet/ Performance-3	6000	ГС	8 МбОП, 3 Мбдиск, Windows, OS/2	Моделювання продуктивності багатопрокольних об'єднань локальних і глобальних мереж; оцінювання затримок в чергах, прогнозування часу відповіді, а також вузьких місць в структурі мережі; облік реальних даних про трафік, що надходять від мережевих аналізаторів.
System& Networks, BONES	20000-40000	ЛС, ГС	32 МбОП, 80 Мбдиск, Sun OS, Sun Solaris, HP-UX	Аналіз впливу додатків клієнт-сервер і нових технологій на роботу мережі
MIL3, Opnet	16000-40000	ЛС	16 МбОП, 100 Мбдиск, DEC AXP, Sun OS, Sun Solaris, HP-UX	Має бібліотеку різних мережевих пристроїв, підтримує анімацію, генерує карту мережі, моделює смугу пропускання.

1.6 Імітаційне моделювання

Імітаційне моделювання – метод прикладного системного аналізу, який є потужним інструментом дослідження складних систем та процесів, у тому числі і таких, управління якими пов'язане з ухваленням рішень в умовах невизначеності. В порівнянні з іншими методами імітаційне моделювання дозволяє розглядати велике число альтернатив, покращувати якість управлінських рішень і точніше прогнозувати їх наслідки.

Імітаційне моделювання засноване на відтворенні за допомогою ЕОМ розгорнутого в часі процесу функціонування системи з врахуванням взаємодії із зовнішнім середовищем. Основою всякої імітаційної моделі (ІМ) є:

1. Розробка моделі системи на основі часткових імітаційних моделей (модулів) підсистем, об'єднаних своїми взаємодіями в єдине ціле;
2. Вибір інформативних (інтеграційних) характеристик об'єкта, способів їх здобуття і аналізу;
3. Побудова моделі впливу зовнішнього середовища на систему у вигляді сукупності імітаційних моделей зовнішніх впливаючих чинників;
4. Вибір способу дослідження імітаційної моделі відповідно до методів планування імітаційних експериментів (ІЕ) [17, с. 325 - 328]

Умовно імітаційну модель можна представити у вигляді програмно (абоапаратно) реалізованих функціональних блоків.

На рис. 1.1 показана структура імітаційної моделі. Блок імітації зовнішніх дій (БІЗВ) формує реалізації випадкових або детермінованих процесів, що імітують дії зовнішнього середовища на об'єкт. Блок обробки результатів (БОР) призначений для отримання інформативних характеристик досліджуваного об'єкта. Необхідна для цього інформація надходить з блоку математичних моделей об'єкта (БМО). Блок управління моделями (БУМ) реалізує спосіб дослідження імітаційної моделі, основне його призначення – автоматизація процесу проведення ІЕ.

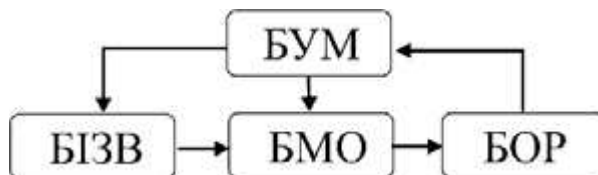


Рис. 1.1. Загальна структура імітаційної моделі

Метою імітаційного моделювання є створення імітаційної моделі об'єкта і проведення імітаційного експерименту над нею для вивчення закону функціонування і поведінки з врахуванням заданих обмежень і цільових функцій

в умовах імітації і взаємодії із зовнішнім середовищем. До переваг методу імітаційного моделювання можуть бути віднесені:

1. Проведення ІЕ над ММ системи, для якої натурний експеримент неможливо здійснити з етичних міркувань або у випадках, коли експеримент пов'язаний з небезпекою для життя, або має значну вартість, або внаслідок того, що експеримент не можна провести з минулим;
2. Вирішення завдань, аналітичні методи для яких непридатні, наприклад, в разі безперервно-дискретних чинників, випадкових дій, нелінійних характеристик елементів системи тощо;
3. Є можливість аналізу загальносистемних ситуацій і ухвалення рішення за допомогою ЕОМ, у тому числі для таких складних систем, вибір критерію порівняння стратегій поведінки яких на рівні проектування не можна здійснити;
4. Скорочення термінів і пошук проектних рішень, які є оптимальними за деякими критеріями оцінки ефективності;
5. Проведення аналізу варіантів структури великих систем, різних алгоритмів управління, вивчення впливу змін параметрів системи на її характеристики і так далі.

Процес функціонування складної системи можна розглядати як зміну її станів, що описуються її фазовими змінними $Z_1(t)$, $Z_2(t)$. $Z_n(t)$ в n -мірному просторі. Завданням імітаційного моделювання є визначення траєкторії руху даної системи в n -мірному просторі ($Z_1, Z_2, \dots, Z_n$), а також обчислення деяких показників, що залежать від вихідних сигналів системи і характеризують її властивості. В даному випадку під “рухом” системи розуміється в загальному сенсі будь-яка зміна, що відбувається в ній [18, с. 215]

1.7 Основні методи імітаційного моделювання

Основними методами імітаційного моделювання є: аналітичний метод, метод статичного моделювання і комбінований метод (аналітико-статистичний) метод.

Аналітичний метод в більшості випадків використовується для імітації процесів для малих і простих систем, коли відсутній чинник випадковості. Наприклад, коли процес їх функціонування описаний диференційними або інтегродиференційними рівняннями. Метод названий умовно, оскільки він об'єднує можливості імітації процесу, модель якого отримана у вигляді аналітично замкнутого рішення, або рішення, отриманого методами обчислювальної математики. Метод статистичного моделювання спочатку розвивався як метод статистичних випробувань (Монте-Карло). Це – чисельний метод, що полягає в здобутті оцінок імовірнісних характеристик, які співпадають з аналітичних завдань (наприклад, з вирішенням і обчисленням визначеного інтеграла). Надалі цей метод став застосовуватися для імітації процесів, що відбуваються в системах, усередині яких є джерело випадковості або які здатні до випадкових дій. Метод отримав назву методу статистичного моделювання.

Комбінований метод (аналітико-статистичний) дозволяє об'єднати переваги аналітичного і статистичного методів моделювання. Він застосовується в разі розробки моделі, що складається з різних модулів, які представляють набір статистичних та аналітичних моделей, і взаємодіють як єдине ціле. Метою імітаційного моделювання на ЕОМ є відтворення різних сигналів, математичні моделі яких описуються випадковими процесами. Слід мати на увазі, що відтворення на ЕОМ процесів з безперервним часом неможливе, зважаючи на дискретну природу ЕОМ. Завдання моделювання випадкових процесів надалі розуміється як завдання відшукування алгоритму, що дозволяє формувати на ЕОМ реалізації таких процесів. У загальному випадку проведення імітаційного експерименту можна розбити на наступні етапи.

1. Встановлення взаємозв'язку між вихідними і вхідними показниками у вигляді математичного рівняння або нерівності.
2. Визначення законів розподілу значень для ключових параметрів моделі.
3. Проведення комп'ютерної імітації значень ключових параметрів моделі.
4. Розрахунок основних характеристик розподілів вихідних і вихідних показників.

5. Проведення аналізу отриманих результатів і прийняття рішення.

Результати імітаційного експерименту можуть бути доповнені статистичним аналізом, а також використовуватися для побудови прогнозних моделей і сценаріїв.

Таким чином, імітаційне моделювання дозволяє врахувати максимально можливе число чинників при ухваленні рішень при проектуванні і є найбільш потужним засобом аналізу складних систем і процесів [19, с.43 – 54]

1.8 Імітаційне моделювання в автоматизованому проектуванні

Складність і різноманіття процесів функціонування проєктованих систем не завжди дозволяють одержати для них окремі математичні моделі традиційного вигляду (аналітичні моделі: цільова функція й обмеження подані в аналітичному вигляді). Виникаючи тут проблеми математичного моделювання в багатьох випадках можуть бути успішно вирішені за рахунок використання методики імітаційного моделювання (ІМ). Суть ІМ у тому, що досліджувана динамічна система замінюється її імітатором і з ним проводяться експерименти. Призначення імітаційних експериментів – одержати інформацію про досліджувану систему. Найбільш ефективним є підхід, пов'язаний з реалізацією імітатора на ЕОМ. Для імітатора складається програма, що є, в свою чергу, програмою функціонування проєктованого об'єкта. У процесі ІМ ЕОМ відтворює явища, описувані математичною моделлю, зі збереженням їхньої логічної структури і послідовності чергування в часі. Рівень деталізації імітаційної моделі може бути різним (залежно від поставлених цілей), що спрямовано на одержання потрібних характеристик. Ці характеристики виводяться до друку і використовуються як прямі або непрямі результати проєктування.

Основний принцип побудови імітаційних моделей полягає в тому, що всі їхні компоненти діють послідовно. Щоб зробити в моделі одночасність декількох подій, що відбуваються в різних частинах реальної системи, необхідно побудувати механізм задання часу в моделях [20, с. 269 – 270]

Ще до побудови моделюючого алгоритму повинні бути вирішені питання вибору математичного апарата дослідження. Для імітації процесів функціонування окремих елементів об'єкта і всього об'єкта в цілому мають бути обрані основні оператори, що погоджуються між собою відповідно до формалізованої системи досліджуваного процесу. До основних операторів відносяться обчислювальні (арифметичні) й логічні оператори, оператори формування реалізацій випадкових процесів і не випадкових величин, оператори рахунку. Охарактеризуємо перераховані вище оператори.

Обчислювальні оператори призначені для реалізації будь-яких обчислювальних функцій за допомогою системи арифметичних операцій, властивій системі команд моделюючої ЕОМ. До моменту початку роботи оператора треба забезпечити наявність усіх необхідних даних, одержуваних від інших операторів алгоритму [6, с 201 – 203].

З метою імітації дії різних випадкових факторів на процес, що моделюється, використовують оператори формування реалізацій випадкових процесів (датчики випадкових чисел). Формування різних констант і не випадкових функцій часу, власне кажучи, не відрізняється від формування звичайних обчислювальних операцій.

Оператори рахунку (лічильники) служать для підрахунку числа різних об'єктів, що володіють заданими властивостями. Результати, видавані лічильниками, як правило, служать вихідними даними для логічних операторів, що забезпечують синхронізацію моделюючого алгоритму.

Логічні оператори служать для перевірки справедливості заданих умов і вироблення ознак, що позначають результат перевірки.

Процес імітації містить у собі велике число операцій, пов'язаних з формуванням, перетворенням і використанням реалізації випадкових подій, величин і процесів, тому результати моделювання теж мають випадковий характер. Вони відбивають випадкові сполучення діючих факторів, що складаються у процесі моделювання. Якщо процес, що моделюється, нестационарний, то шукані величини при імітаційному моделюванні визначають у результаті статистичної обробки сукупності даних деякого числа реалізацій процесу моделювання. Сукупність реалізацій виступає в ролі статистичного

матеріалу при машинному експерименті, а оцінка параметрів – у ролі експериментальних даних.

При автоматизованому проектуванні імітаційні моделі призначені для вивчення особливостей функціонування проєктованих структур, що складаються з різноманітних елементів (дискретних і безупинних, детермінованих і стохастичних).

Імітаційні програми доцільно будувати за модульним принципом, при якому всі елементи системи описуються одноманітно, у вигляді деякої стандартної математичної схеми – модуля. Схеми й оператори сполучення моделей один з одним дозволяють будувати універсальні програми імітації. Ці програми повинні здійснювати введення і формування масиву вихідних даних для моделювання, перетворення елементів системи і схем сполучення до стандартного вигляду, імітацію модуля і взаємодії елементів системи, обробку й аналіз результатів моделювання, а також керування моделлю.

Модель проєктованого об'єкта повинна бути деталізованою, однак не настільки, щоб стати громіздкою в поводженні. Вибір мови програмування визначається постановкою задачі, де враховуються характеристики об'єкта моделювання, тип розроблюваної імітаційної моделі, умови проведення експерименту.

До мов висувають наступні вимоги:

1. Система мови має бути такою, щоб скласти модель процесу, що імітується, тобто у користувача повинні бути можливості опису процесу моделювання відповідно до предметного мислення.

2. Мова повинна бути плановим засобом імітації, коли всі роботи на реалізованій моделі (включаючи створення імітатора) автоматизовані настільки, що від користувача не потрібні спеціальні знання.

3. Мова повинна мати зручні синтаксичні засоби для швидкого і простого опису системи, що моделюється.

Схема організації процесу імітаційного моделювання при автоматизованому проектуванні має вигляд, наведений на рис. 1.2 [7, с. 55 – 56]

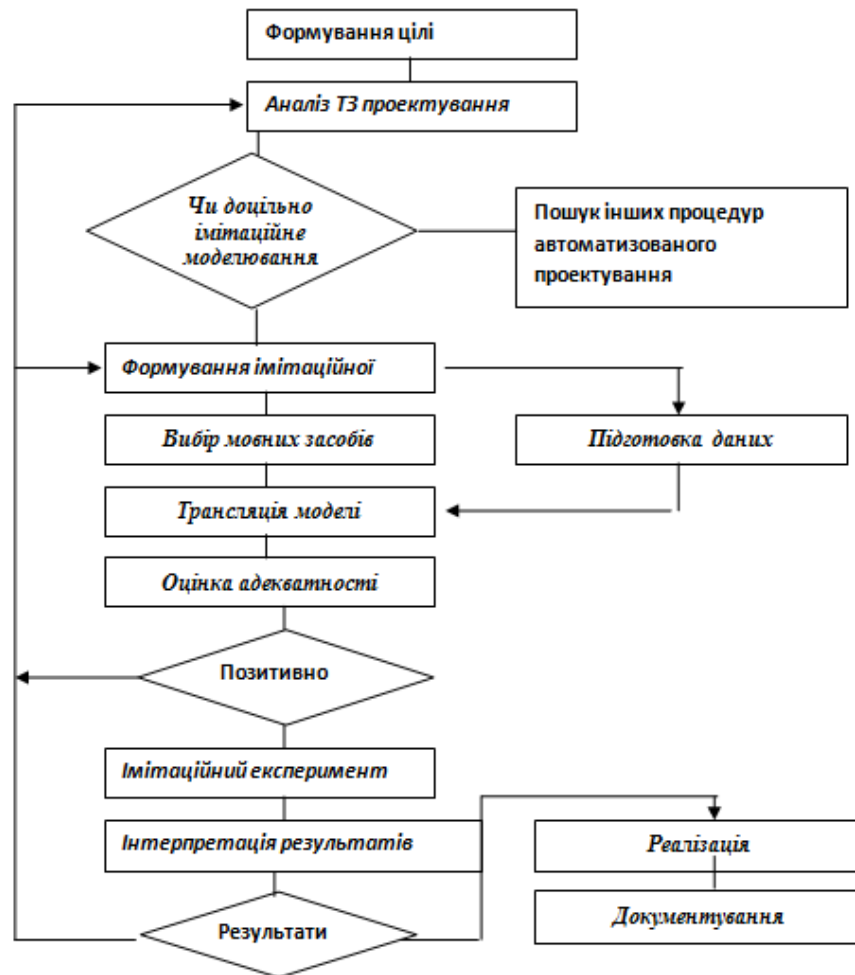


Рис. 1.2. Схема організації процесу імітаційного моделювання

1.8 Використання моделювання для оптимізації продуктивності мережі

Аналізатори протоколів незамінні для дослідження реальних мереж, але вони не дозволяють отримувати кількісні оцінки характеристик для ще не існуючих мереж, що знаходяться в стадії проектування. У цих випадках проектувальники можуть використовувати засоби моделювання, за допомогою яких розробляються моделі, що відтворюють інформаційні процеси, які існують в мережах.

Існують спеціальні, орієнтовані на моделювання обчислювальних мереж програмні системи, в яких процес створення моделі спрощений. Такі програмні системи самі генерують модель мережі на основі вихідних даних про її топологію і протоколах що використовуються, інтенсивність потоків запитів між комп'ютерами мережі, протяжності ліній зв'язку, про типи використовуваного устаткування і додатків. Програмні системи моделювання можуть бути вузько

спеціалізованими і достатньо універсальними, що дозволяють імітувати мережі найрізноманітніших типів. Якість результатів моделювання в значній мірі залежить від точності вихідних даних про мережу, переданих в систему імітаційного моделювання .

Програмні системи моделювання мереж - інструмент, який може стати в нагоді будь-якому адміністратору корпоративної мережі, особливо при проектуванні нової мережі або внесенні кардинальних змін у вже існуючу. Продукти даної категорії дозволяють перевірити наслідки впровадження тих чи інших рішень ще до оплати придбаного обладнання [7, с. 60]

Програми імітаційного моделювання мережі використовують у своїй роботі інформацію про просторове розташування мережі, числі вузлів, конфігурації зв'язків, швидкостях передачі даних, використовуваних протоколах і типі обладнання, а також про виконувані в мережі додатках.

Зазвичай імітаційна модель будується не з нуля. Існують готові імітаційні моделі основних елементів мереж: найбільш поширених типів маршрутизаторів, каналів зв'язку, методів доступу, протоколів і т.п. Ці моделі окремих елементів мережі створюються на підставі різних даних: результатів тестових випробувань реальних пристроїв, аналізу принципів їх роботи, аналітичних співвідношень. У результаті створюється бібліотека типових елементів мережі, які можна налаштовувати за допомогою заздалегідь передбачених в моделях параметрів.

Системи імітаційного моделювання зазвичай включають також набір засобів для підготовки вихідних даних про досліджувану мережі - попередньої обробки даних про топологію мережі [5, с. 89].

РОЗДІЛ 2 РОЗРОБКА КОНЦЕПЦІЇ СИСТЕМИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ КОМП'ЮТЕРНОЇ МЕРЕЖІ

2.1 Постановка завдання аналізу характеристик комп'ютерної мережі

Бурхливе зростання складності комплектуючих комп'ютерних мереж вимагає підвищення ефективності їх застосування та вдосконалення методів управління та планування мереж.

Теорія масового обслуговування забезпечує можливість розрахунку кількісних характеристик функціонування мереж, включаючи оцінку ймовірно-часових характеристик вузлів комутації, але не дозволяє розрахувати надійність мережі. Спрощений підхід ТМО вимагає подальшого уточнення характеристик мережі за допомогою більш реальних моделей, що призводить до ітераційної процедури проектування комп'ютерних мереж [3].

В даний час технічні та програмні засоби комп'ютерної мережі та умови їх роботи стають все більш складними. Кількість елементів в окремих видах пристроїв комп'ютерної мережі обчислюється сотнями тисяч. Імовірність виникнення хоча б одного відмови сучасного складного пристрою стає досить великий, отже, необхідні спеціальні заходи, забезпечують доступність ресурсів і послуг. Доступність – такий стан комп'ютерної мережі, при якому мережу відповідає всім вимогам, що пред'являються до її функціонування, тобто це властивість зберігати працездатний стан протягом деякого напрацювання. Для оцінки доступності застосовується кількісний показник: коефіцієнт оперативної готовності R_s .

Методи теорії надійності використовуються для забезпечення заданих вимог до функціонування комп'ютерної мережі та підвищення доступності на етапах проектування, виготовлення, випробування і експлуатації мережі. Методи теорії ймовірностей, як математичного апарату теорії надійності, використовуються для розрахунку показника ймовірності безвідмовної роботи комп'ютерної мережі [42, с. 178 – 183]

При проектуванні мережі зміна її структури з метою поліпшення характеристик або поновлення з'являється небезпека зниження готовності, тому необхідно розраховувати коефіцієнт готовності проекту мережі на кожному кроці ітерації проектування.

Використання запропонованої методики аналізу проекту КС призводить до цілеспрямованого зміни структури мережі і обчисленню, у тому числі і надійностних характеристик функціонування.

Характеристики функціонування мережі в цьому випадку пропонується представити як функцію структурно-функціональної організації та коефіцієнта оперативної готовності, не нижче заданого.

$H(t) = f(SF, R_s \geq R_z, C)$ де:

1. SF - структурно - функціональні параметри комп'ютерної мережі;
2. R_s, R_z - коефіцієнти оперативної готовності;
3. умова $R_s \geq R_z$ обмежує ймовірність безвідмовної роботи мережі,
4. C - вкладення в комп'ютерну мережу.

Основною вимогою до систем моделювання обчислювальної систем є відсутність необхідності програмування; можливість імпорту інформації з існуючих систем управління мережами та засобів моніторингу, наявність розширюваної бібліотеки об'єктів, інтуїтивно-зрозумілий інтерфейс; просте налаштування на об'єкти реального світу, гнучка система побудови сценаріїв моделювання, зручне представлення результатів моделювання; анімація процесу моделювання; автоматичний контроль моделі на внутрішню несуперечливість.

2.2 Імітаційне моделювання комп'ютерної мережі

Є дослідження, які стверджують, що Internet і Ethernet трафіки, є самоподібними (self-similar)). Додаток або робоча станція формує Ethernet-трафік сплесками з періодами бездіяльності між ними і також є самоподібним. Само подібна структура трафіку містить зменшені копії самої себе на всіх рівнях масштабування. Пропонується моделювати само подібний трафік розподілом

Парето. Потрібно оцінити вплив само подібності на продуктивність мережі. Порівняння проводиться на моделях систем масового обслуговування. Для цього аналізуються для моделей $M / M / 1$ і $G / M / 1$ з самоподібним вхідним потоком середня і максимальна довжина черги, середній час очікування і середній час перебування в черзі.

Інтервали часу між прибулими з частотою по розподілу Пуассона пакетами, розподілені експоненціально з функцією розподілу:

$$F(x) = 1 - (e^{-\lambda x})$$

Розподіл Парето має функцію розподілу ймовірностей:

$$F(x) = 1 - (\lambda / x)^\alpha \quad (x > k; \alpha > 0)$$

Розподіл Парето можна спостерігати в широкому спектрі соціальних і фізичних явищ, а також в області передачі даних. Хвіст розподілу Парето спадає значно повільніше, ніж експоненціального. Наявність важкого хвоста у певних мережових змінних (наприклад, розмірів файлів і тривалості з'єднання) є основною причиною довгостроковій залежності і само подібності [43, с. 226 - 231].

Для відображення процесів в системі при наявності неекспоненціальних зовнішніх впливів розроблені імітаційні. Час очікування, перебування, довжини черг вимірюються залежно від завантаження системи: $\rho = \lambda / \mu$, де λ - середня інтенсивність вхідного потоку; μ - середня інтенсивність обслуговування заявки.

Розподіл Парето - повільно загасаюче розподіл з параметрами, що визначають мінімальне значення випадкової величини і середнього значення і дисперсії випадкової величини.

Параметри розподілу:

λ - визначає мінімальне значення, яке може приймати випадкова величина x ,

α - визначає середнє значення і дисперсію випадкової величини. Якщо параметр α приймає значення від 1 до 2, то випадкова величина має кінцеве середнє і нескінченну дисперсію.

Параметр λ підбирався таким, щоб забезпечити завантаження, рівну завантаженні моделі $M / M / 1$. Результати моделювання представлені в таблиці 2.7. і рис.2.1 і рис.2.2.

Аналіз результатів моделювання систем $P / M / 1$ і $M / M / 1$ показує наступне:

1. Всі основні характеристики (середня довжина черги, максимальна довжина черги, середній час очікування в черги, середній час перебування в системі) для системи $P / M / 1$ мають великі значення в порівнянні з системою $M / M / 1$ з аналогічними параметрами.
2. При завантаженні системи $\rho \leq 0,1$ різниця не велика для характеристик систем.
3. При завантаженні системи $0,1 \leq \rho \leq 0,3$ різниця збільшується до декількох разів.
4. При завантаженні системи $0,3 \leq \rho \leq 0,7$, значення характеристик для систем $P / M / 1$ і $M / M / 1$ різняться в рази, і з підвищенням навантаження на систему зростають до десятків разів.
5. Коли ρ перевищує значення 0,7, різниця в значеннях між двома системами продовжує зростати до декількох десятків разів.
6. Довжина черги передбачалася нескінченною, і не використовувалися пріоритети, але довжина черги має значення і тому вплив самоподібного вхідного потоку на продуктивність системи більш помітно і може призводити до нестабільності в її роботі або більших затримок.
7. Вплив самоподібності і довгостроковій залежності необхідно враховувати при проектуванні та аналізі буферів і мереж

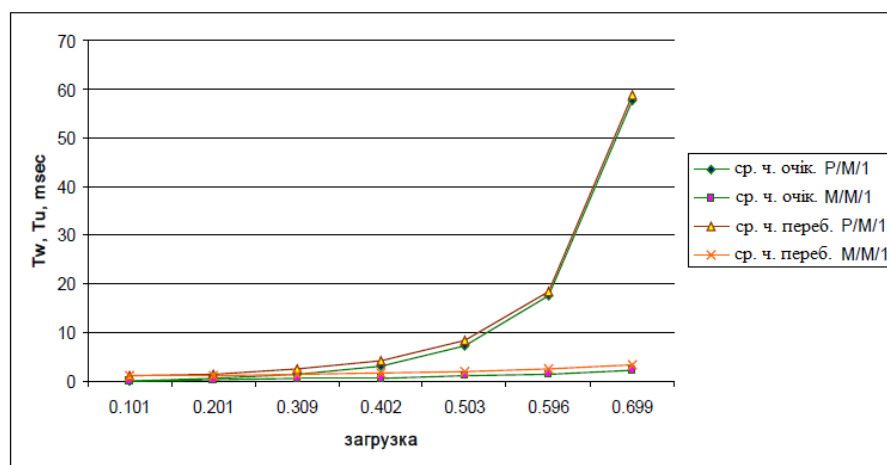


Рис. 2.1 Залежність середніх часів очікування і перебування від завантаження для випадків експоненціального і самоподібного потоків при завантаженні від 0,1 до 0,7

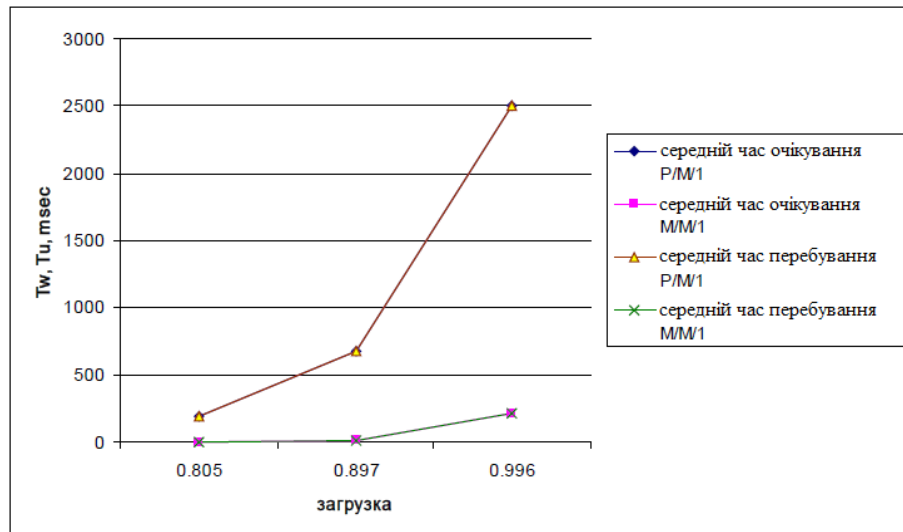


Рис.2.2 Залежність середніх часів очікування і перебування від завантаження для випадків експоненціального і самоподібного потоків при завантаженні від 0,8 до 0,996

Для моделі з експоненціально розподіленими тимчасовими інтервалами надходження заявок і експоненціально розподіленої тривалістю обслуговування (ММ / 1) залежність необхідного розміру буфера q від середнього коефіцієнта використання ρ приймає вид: $q = \rho / (1 - \rho)$

Таблиця 2.1. Результати експериментів для порівняння поведінки моделей М / М / 1 (з експоненціальним вхідним потоком) і G / М / 1 з самоподібним вхідним потоком для максимальної і середньої довжини черги, середнього часу очікування в черзі і перебування заявок в мережі.

Завантаження	0,101	0,201	0,309	0,402	0,503	0,596	0,699	0,805	0,897	0,996
Максимальна довжина черги P/M/1	5	10	17	33	65	103	387	1114	2171	6106
Максимальна довжина черги M/M/1	5	7	10	13	14	17	23	37	58	402
Середня довжина черги P/M/1	0,009	0,102	0,45	1,291	3,697	10,437	40,537	157,82	604,84	2495,99
Середня довжина черги M/M/1	0,011	0,05	0,13	0,272	0,513	0,888	1,644	3,375	7,76	210,207

Середній час очікування Р/М/1	0,093	0,503	1,45	3,203	7,334	17,473	57,809	195,68	673,58	2508,33
Середній час очікування М/М/1	0,111	0,25	0,44	0,674	1,016	1,485	2,347	4,18	8,631	210,82
Середній час перебування Р/М/1	1,09	1,501	2,451	4,2	8,331	18,471	58,806	196,68	674,58	2509,33
Середній час перебування М/М/1	1,109	1,247	1,446	1,672	2,013	2,482	3,344	5,178	9,629	211,818

При довгостроковій залежності високого ступеня потреби в буфері ростуть стрімко зі зростанням коефіцієнт використання ρ . При певних припущеннях залежність необхідного розміру буфера q від середнього коефіцієнта використання ρ підкоряється закону:

$$q = \frac{\rho^{\frac{1}{2(1-H)}}}{(1-\rho)^{\frac{H}{(1-H)}}},$$

де H - параметр, який приймає значення $0.5 < H < 1$.

Залежність необхідного розміру буфера показана на рис.2.3.

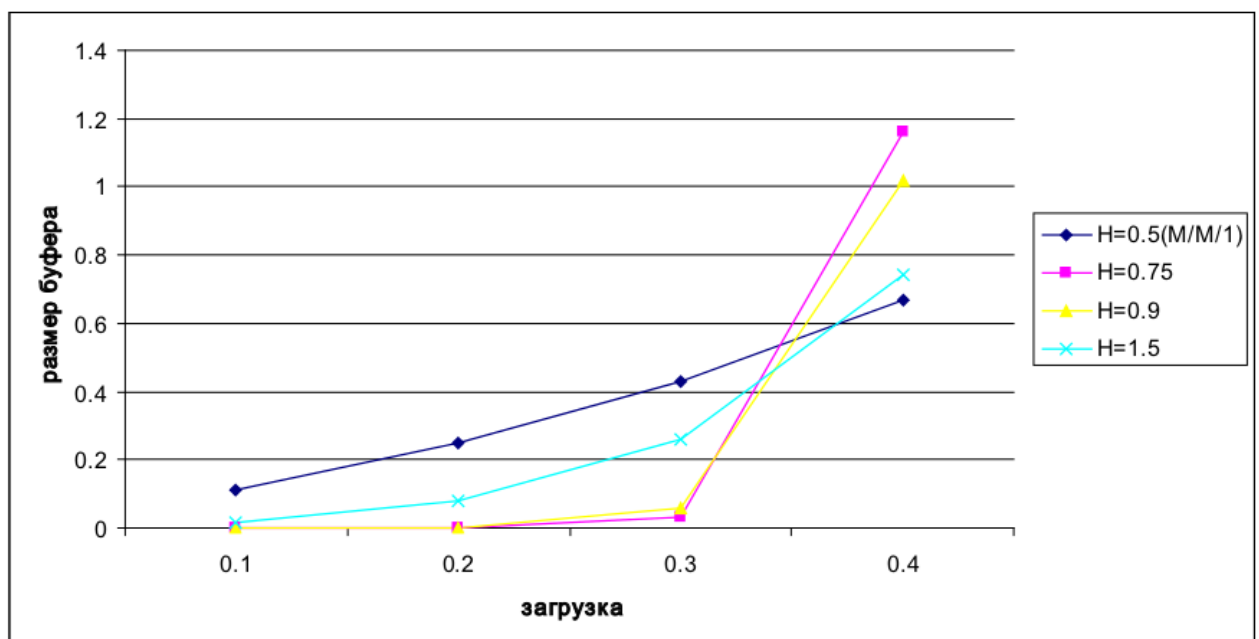


Рис.2.3. Залежність необхідного розміру буфера q від середнього коефіцієнта використання ρ

Оцінка середнього часу перебування вимог в системі задається формулою:

$$\mu T = \frac{\rho^{\frac{H-1}{2}}}{(1-\rho)^{\frac{H}{1-H}}}$$

Самоподібний трафік прикладного та мережевого рівнів розрізняється. На прикладному рівні самоподоба притаманне джерела потоку даних без взаємодії з мережею. Оскільки поведінка само подібного трафіку прикладного рівня незалежно від поточного стану мережі, цим трафіком рекомендується керувати в контексті управління доступом і виділення ресурсів для гарантування необхідного класу обслуговування.

Самоподібний трафік мережевого рівня проявляє властивості самоподібності в широкому діапазоні тимчасової шкали в результаті численних взаємодій з мережею або об'єднаної мережею. Прикладом є ТСП - трафік. Самоподібний трафік мережевого рівня змінює свою поведінку залежно від навантаження, схеми повторної передачі, кількості конкуруючих користувачів, розмірів запитуваних в Павутині або по FTP файлів і т.д. і ця відмінність є предметом досліджень.

В результаті роботи моделі виходить інформація про:

- середні затримках (часах доставки) повідомлень різних типів;
- гістограмах і функціях розподілу затримки (часу доставки) повідомлень;
- гістограмах щільності і функцій розподілу зайнятої пам'яті по вузлах комутації;
- кількостях повідомлень різних типів, що дійшли до адресата;
- кількостях відмов в доставці повідомлень з різних причин (брак пам'яті, перевищення допустимого часу перебування в мережі і т.д.);

РОЗДІЛ 3 РОЗРОБКА ПРОЕКТУ СИСТЕМИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ КОМП'ЮТЕРНИХ МЕРЕЖ

3.1 Загальносистемні рішення

З метою опису потоків даних та сховищ даних системи та на основі результатів проведеного аналізу предметної області, розробимо модель потоків даних системи. З урахуванням визначених акторів предметної галузі та інформації, якою вони обмінюються між собою – було збудовано діаграму варіантів використання. На рисунку 3.1 зображено діаграму прецедентів між користувачем та графічним інтерфейсом користувача, на рисунку 3.2 зображено взаємодію графічного інтерфейсу користувача з системою моделювання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів.

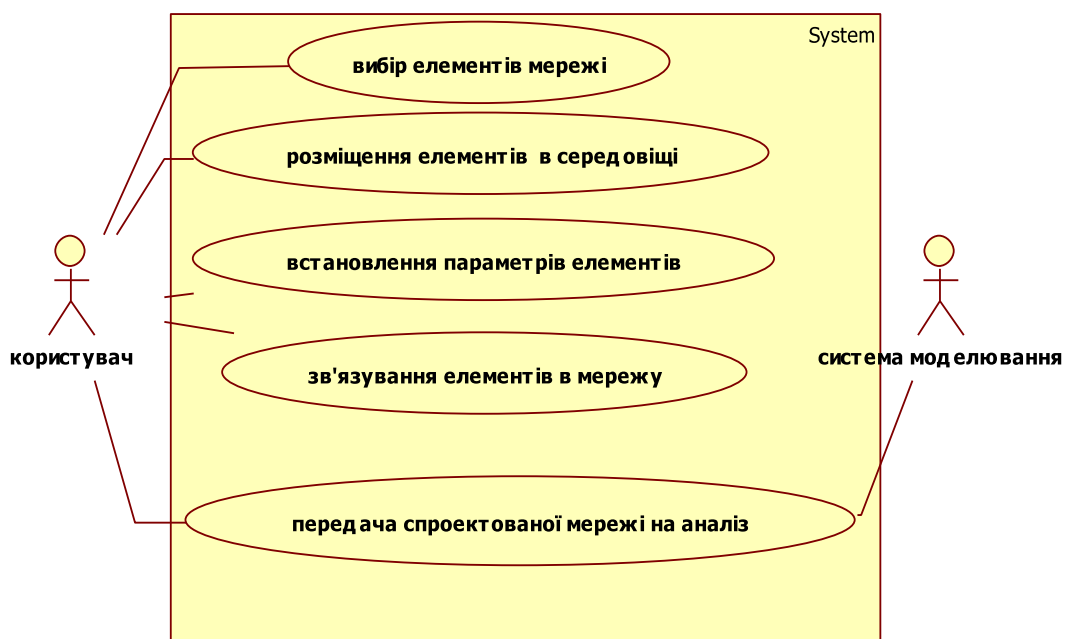


Рис. 3.1 Діаграма прецедентів. Користувач, графічний інтерфейс

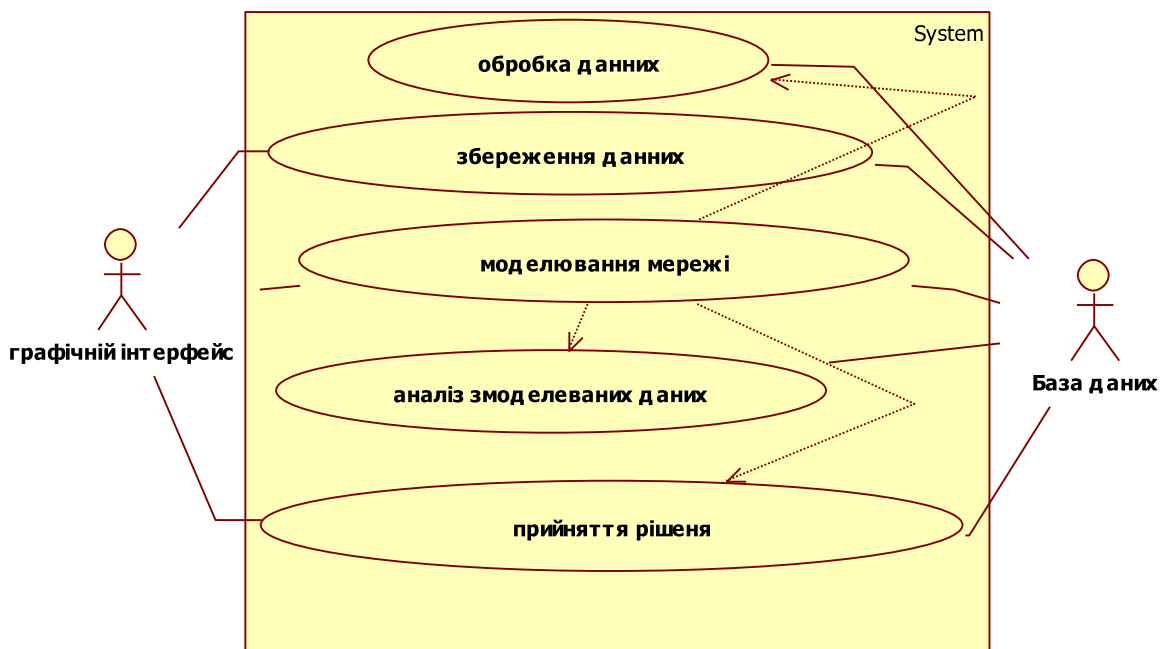


Рис. 3.2 Діаграма прецедентів. Графічний інтерфейс, система моделювання комп'ютерної мережі

Як видно на рис. 3.1 зовнішніми сутностями розроблюваного програмного забезпечення графічного інтерфейсу є користувач та система моделювання роботи комп'ютерної мережі. Користувач вибирає бажані елементи для проектованої мережі та розміщує їх в робочому просторі графічного інтерфейсу. Від користувача надходить необхідна інформація про характеристики роботи вибраних елементів, які згодом він з'єднує між собою елементами, які імітують мережеві кабелі, будучи замкнутою системою для передачі даних між її вузлами. Після чого за згоди користувача надходить звернення до підсистеми з моделювання та аналізу спроектованої мережі.

На рис. 3.2 зображено розроблювану підсистему моделювання та аналізу роботи комп'ютерної мережі, зовнішніми сутностями є графічний інтерфейс та база даних яка зберігає інформацію про сутності які створюються в системі. За згоди користувача зберігаються характеристики сутностей системи в базу даних, після чого виконується сам процес моделювання мережі, при якому в першу чергу виконується обробка даних які являються характеристиками сутностей мережі, моделюються зв'язки між ними, аналіз роботи і прийняття рішення що до оптимальності передачі даних в замкнутій мережі.

3.2 Рішення з інформаційного забезпечення

З метою опису сутностей та структури даних системи та на основі проведеного аналізу предметної галузі розробимо модель даних системи. З урахуванням постановленої задачі діаграма має включати сутності та зв'язки між ними.

У класичній теорії баз даних, модель даних є формальна теорія подання та обробки даних в системі управління базами даних (СКБД), яка включає, щонайменше, три аспекти:

- 1) аспект структури: методи опису типів і логічних структур даних в базі даних;
- 2) аспект маніпуляції: методи маніпулювання даними;
- 3) аспект цілісності: методи опису та підтримки цілісності бази даних.

Аспект структури визначає, що з себе логічно представляє база даних, аспект маніпуляції визначає способи переходу між станами бази даних (тобто способи модифікації даних) і способи отримання даних з бази даних, аспект цілісності визначає засоби описів коректних станів бази даних.

Модель даних - це абстрактне, самодостатнє, логічне визначення об'єктів, операторів і інших елементів, в сукупності складових абстрактну машину доступу до даних, з якою взаємодіє користувач. Ці об'єкти дозволяють моделювати структуру даних, а оператори - поведінка даних [2, с 167 – 169].

Діаграма моделі даних мовою UML зображена на рисунку 3.3 [4]

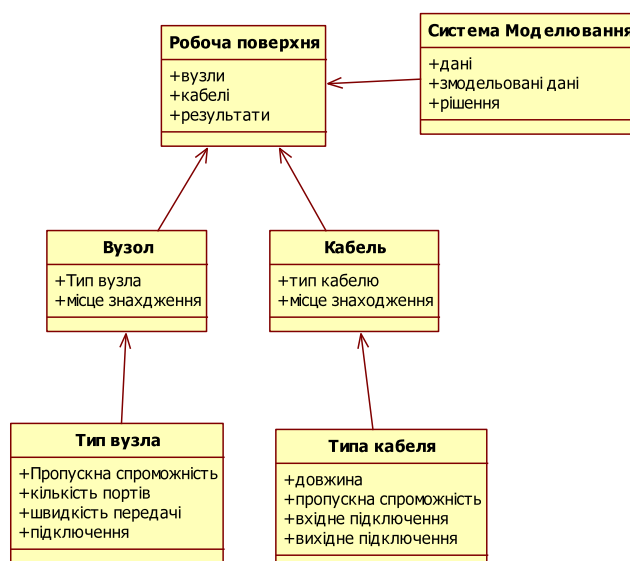


Рис. 3.3 Модель даних

На рисунку 3.3 зображено модель даних системи. Основною одиницею є робоча поверхня яка відповідає за управління і розміщення проекрованої комп'ютерної мережі.

Робоча область зберігає в собі такі об'єкти як вузли та кабелі, які мають інформацію про тип робочої одиниці проекрованої системи та координати розміщення об'єкта в робочому просторі [8, с 205].

Тип вузла зберігає в собі основні характеристики вузла такі як: пропускна спроможність, швидкість передачі інформації та кількість підключень кабелів до вузла.

Тип кабель свою чергу зберігає інформацію про основні характеристики кабелю: довжина, пропускна спроможність, тобто з якою максимальною швидкістю може проходити інформація від одного вузла до іншого через кабель, вхідне і вихідне підключення описують які вузли цей кабель з'єднує між собою.

Система моделювання в свою чергу отримує від робочої області інформацію про основні характеристики робочих одиниць проекрованої комп'ютерної мережі, після роботи підсистеми моделювання формуються змодельовані данні роботи проекрованої системи а також рішення що до оптимізації мережі, які потім відображаються в інтерфейсі користувача в робочій області [3, с 345].

3.3 Модель послідовностей

З метою опису послідовності взаємодії модулів системи на основі розробленої моделі класів системи було розроблено діаграму взаємодії.

Діаграма послідовності (діаграма послідовності англ.) - Діаграма, на якій показано взаємодію об'єктів (обмін між ними сигналами і повідомленнями), впорядковане за часом, з відображенням тривалості обробки і послідовності її прояви, використовується в UML.

Основними елементами діаграми послідовності є позначення об'єктів (прямокутники з назвами об'єктів), вертикальні "лінії життя" (англ. виживання), що відображають протягом часу, прямокутники, що відображають діяльність об'єкта або виконання ним певної функції (прямокутники на

пунктирною "лінії життя"), і стрілки, що показують обмін сигналами або повідомленнями між об'єктами [7].

На даній діаграмі, рисунок 3.4, об'єкти розташовуються зліва направо.

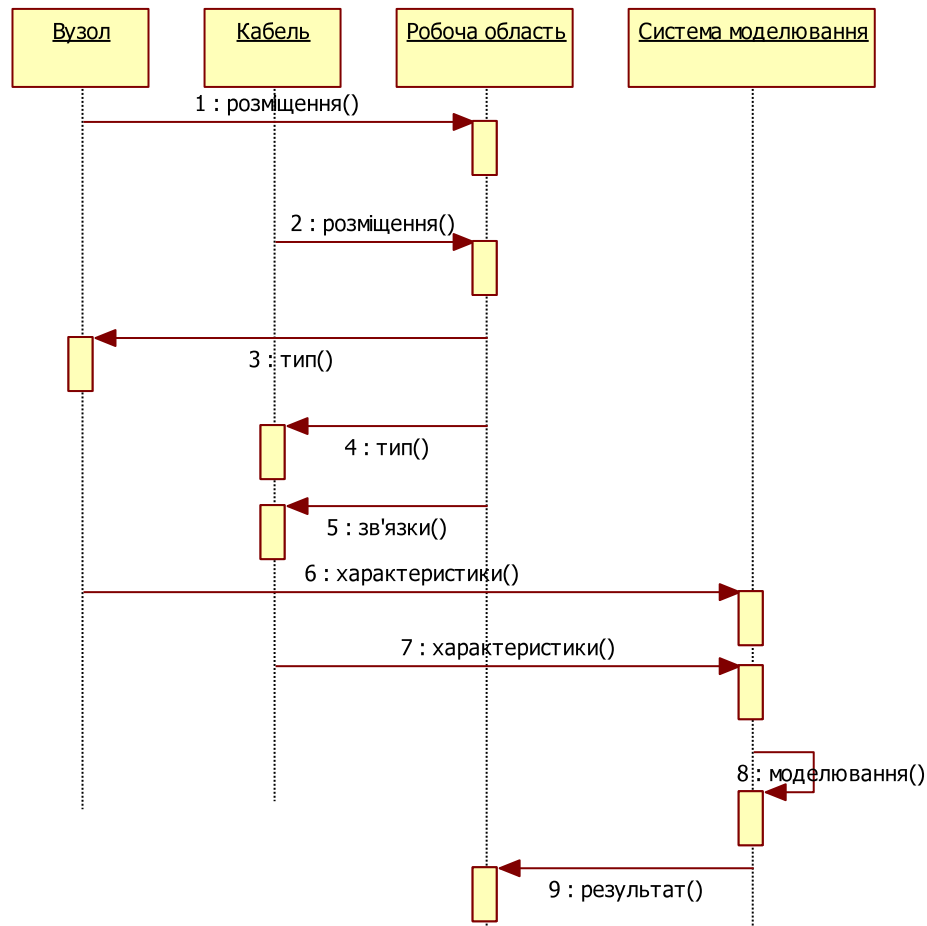


Рис. 3.4 Діаграма послідовностей

На рисунку 3.4 зображено діаграму послідовностей яка зображує діяльність об'єктів в системі та передачу повідомлень між ними.

Вузли та кабелі передають інформацію про своє розміщення в робочій області проектування. Після чого користувач указує тип та характеристики робочих елементів системи в робочій області які зберігаються в об'єктах вузол та кабель. Потім елементи зв'язуються в закриту систему яка являється проектованою комп'ютерною мережею.

Характеристики вузлів та кабелів передаються в систему моделювання в якій проходить імітація роботи мережі, моделюються вихідні данні для аналізу оптимальності роботи системи і формуються результати які вертаються до робочої області для відображення в графічному інтерфейсі користувача.

3.4 Модель переходів станів

З метою опису послідовності зміни станів системи під час діалогу з користувачем, було розроблено модель станів системи (Рис. 3.5).

Діаграма станів - це, по суті, діаграма станів з теорії автоматів зі стандартизованими умовними позначеннями, яка може визначати безліч систем від комп'ютерних програм до бізнес-процесів. Використовуються такі умовні позначення:

Коло, що позначає початковий стан.

Округність з маленьким колом усередині, що позначає кінцевий стани (якщо є).

Округлений прямокутник, що означає стан. Верхівка прямокутника містить назву стану. У середині може бути горизонтальна лінія, під якою записуються активності, що відбуваються в даному стані.

Стрілка, що позначає перехід. Назва події (якщо є), що викликає перехід, відзначається поряд із стрілкою. Охороняє вираз може бути додано перед "/" і укладено у квадратні дужки], що означає, що цей вираз має бути істинним, щоб перехід мав місце. Товста горизонтальна лінія з або безліччю вхідних ліній і однієї виходить, або однієї вхідної лінією і безліччю виходять. Це позначає об'єднання і розгалуження відповідно [6].

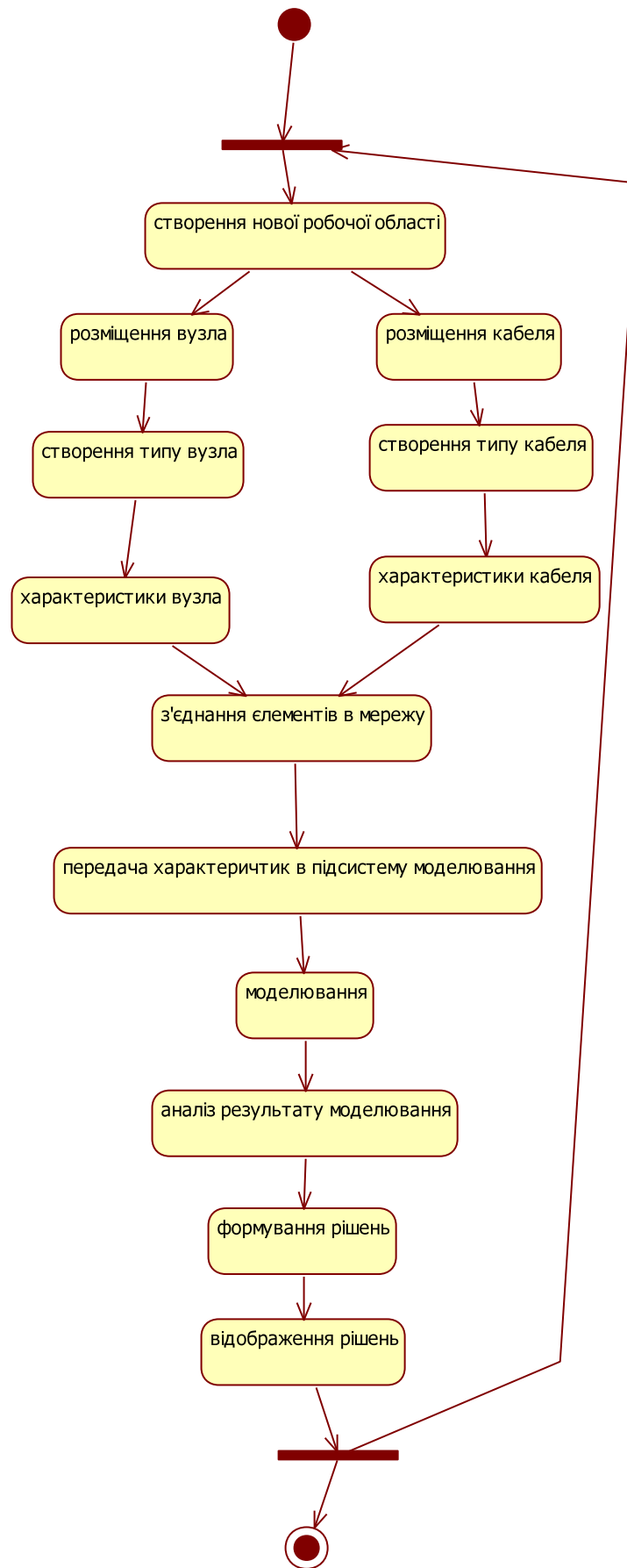


Рис. 3.5 Діаграма станів

На рисунку 3.5 зображено діаграму яка відображає порядок зміни станів проектованої системи. Після входу в систему автоматизованого проектування комп'ютерної мережі система переходить стан створення нового проекту для проектування нової мережі. Після створення проекту потрібно почати розміщувати елементи мережі в робочій області, після розміщення кожного елемента потрібно обов'язково вказати його тип и описати потрібні характеристики елемента після чого зв'язати робочі одиниці в закриту систему.

Після проектування закритої комп'ютерної мережі система переходить в стан передачі даних елементів в підсистему моделювання, для подальшої імітації та моделювання роботи мережі. Після успішного моделювання наступає аналіз результаті та формування рішень які згодом будуть відображені в робочій області [5].

3.5 Діаграма класів програмного забезпечення

З метою опису сутностей системи та зв'язків між ними, на основі розробленої моделі даних системи було розроблено діаграму класів. З урахуванням постановлених задач діаграма має включати сутності та зв'язки між ними. Так як в розробці системи для спрощення реалізації було прийнято рішення побудови програми в декілька рівнів абстракції.

Загальна структура системи зображена на рисунку 3.6

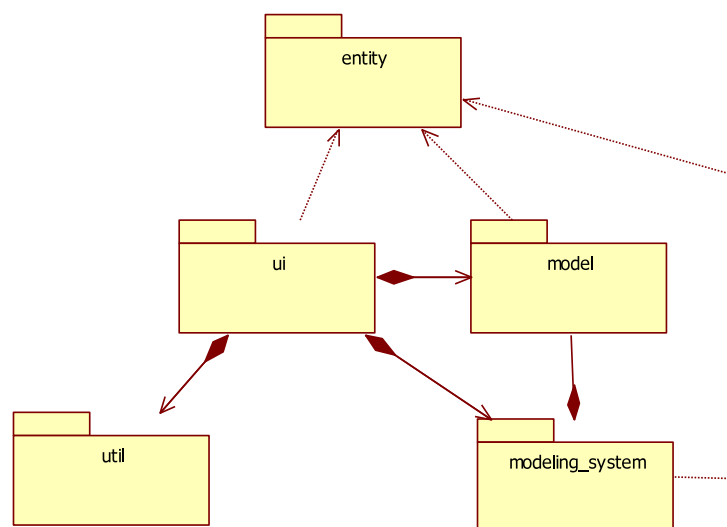


Рис. 3.6 Загальна структура системи

Пакет `entity` зберігає в собі класи які описують сутності зображені на рисунку 3.3.

Пакет `model` зберігає в собі класи які описують доступ до бази даних.

Пакет `ui` зберігає в собі класи які описують графічний інтерфейс користувача системи автоматизованого проектування комп'ютерної мережі.

Пакет `util` зберігає в собі класи які відповідають за управління елементами проекрованої комп'ютерної мережі. За надання елементам їх типів та основних характеристик.

Пакет `modeling_system` зберігає в собі класи які формують підсистему моделювання, імітації та аналізу проекрованої мережі.

В свою чергу класи з пакетів `ui`, `modeling_system`, `model` використовують об'єкти пакета `entity` які відповідають за передачу інформації між компонентами системи [1, с. 117]

3.6 Вибір мови програмування

Для реалізації системи обрано об'єктно-орієнтовану мову програмування Java а також її обчислювальну платформу JavaSE. Платформа надає API та виконавче середовище для розробки і програмного забезпечення.

Для розробки графічного інтерфейсу користувача вибрано платформу JavaFX. JavaFX включає в себе набір утиліт, за допомогою яких веб-розробники та дизайнери можуть швидко створювати та надавати розвинуті інтернет застосунки для десктопів, мобільних пристроїв, телебачення та інших платформ.

JavaFX складається з JavaFX Script і JavaFX Mobile. Починаючи з випуску JavaFX 2.0 забезпечена можливість створення JavaFX-застосунків, написаних цілком на мові Java. Для розробки застосунків доступний багатий графічний і мультимедіа API, що спрощує створення візуальних програм [2, с. 103]

РОЗДІЛ 4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

4.1 Вступ

Економічна ефективність — досягнення найбільших результатів за найменших витрат живої та уречевленої праці. Економічна ефективність — конкретна форма вияву закону економії часу, їх зв'язок здійснюється через зростання продуктивності праці її підвищення означає зростання ефективності сукупної праці, збільшення ефективності всього виробництва, зумовленої насамперед прогресом продуктивних сил.

Основні напрями зростання економічної ефективності в Україні суттєве зниження енергомісткості (витрати енергії на одиницю продукції в Україні в 90-х ХХ ст. приблизно в 5-6 разів перевищують аналогічні витрати в розвинутих державах світу), матеріаломісткості (витрати матеріалів та ресурсів в Україні на одиницю продукції в 2-2,5 рази перевищують аналогічні показники у країнах Заходу), зростання фондівддачі тощо. Досягнення таких цілей передбачає, у свою чергу, демократичне трансформування всієї економічної системи, в т. ч. її елементів продуктивних сил, техніко-економічних відносин, виробничих відносин (або відносин економічної власності), організаційно-економічних відносин та господарського механізму.

У західній економічній науці економічну ефективність розглядають як складну і багатоаспектну категорію, що складається з:

1. відношення між витратами ресурсів і обсягом товарів та послуг, які виробляють з цих ресурсів;
2. максимального обсягу виробництва товарів і послуг з використанням мінімальної вартості ресурсів;
3. виробництва товарів певної вартості за найменших витрат ресурсів;
4. отримання максимуму з доступних для людини обмежених ресурсів;
5. відношення цінності результату до цінності витрат та ін.

Різні сторони економічної ефективності виділяють і у вітчизняній економічній літературі. Проте в усіх західних і вітчизняних працях економістів не виокремлюють речовий зміст і суспільну форму категорії «економічної ефективності». Щодо речового змісту – це досягнення найбільших результатів за найменших витрат живої та уречевленої праці, щодо суспільної форми – це послаблення конфлікту, антагонізму, гострих форм суперечності між працею і власністю на результати цієї праці, тобто дотримання вимог закону єдності праці та власності.

4.2 Цілі і завдання, які вирішуються в економічній частині

В даному дипломному проекті розроблено web-додаток для замовлення авіабілетів. Результатом вирішення цього завдання буде адаптивний продукт web-оглядачів. Метою даного розділу є розрахунок витрат. В результаті розрахунку знаходиться собівартість прикладної програми.

4.3 Розрахунок амортизаційних відрахувань

У даній роботі використовуються наступні технічні засоби:

1) Комп'ютер;

2) монітор LCD Acer 19 ".

Закупівельна ціна комп'ютера – 7000 грн., Монітора - 3000 грн. Приймемо норму амортизації на технічні засоби 20%.

Загальна вартість технічних засобів, грн:

$$Ц_{ТС} = Ц_k + Ц_m, (1)$$

де $Ц_k$ - ціна комп'ютера;

$Ц_m$ - ціна монітора.

Звідси:

$$Ц_{ТС} = 7000 + 3000 = 10000.$$

Для створення мобільного додатку, що є кінцевим результатом досліджень, застосовувалося наступне програмне забезпечення:

- платформа Windows 10 – 1000грн;

- JetBrains PhpStorm - 2500 грн.

Загальна вартість програмного забезпечення становить 3500 грн.

Загальна вартість технічних засобів та програмного забезпечення, грн. .:

$$Ц_3 = Ц_{ТС} + Ц_{ПО}, (2)$$

$$Ц_3 = 10000 + 3500 = 13500.$$

Річні амортизаційні відрахування на повне відновлення технічних засобів і програмного забезпечення розраховуються за формулою, грн:

$$A_B = Ц_3 \cdot Na, (3)$$

$$A_B = 13500 \cdot 0,2 = 2700.$$

Амортизаційні відрахування за період створення програмного продукту, грн:

$$A_{\Pi} = \frac{A_B \cdot K_{\text{д}}}{K_{\text{рД}}} \quad (4)$$

де $K_{\text{д}} = 30$ дн. - кількість відпрацьованих днів;

$K_{\text{рД}} = 280$ дн. - кількість робочих днів у році.

$$A_{\Pi} = \frac{2700 \cdot 30}{280} = 289$$

4.4 Розрахунок витрат на енергоспоживання

ПЕОМ, на якій була розроблена програма, є споживачем електричної енергії мережі змінного струму, напругою 220 В. Згідно з технічною документацією, сумарна потужність, споживана комп'ютером і монітором, становить

$$P_c = 250 \text{ Вт} \cdot \text{год.}$$

Витрата грошових коштів, пов'язаний з енергоспоживанням технічних засобів можна знайти за формулою, грн:

$$V_e = K_{\text{дн}} \cdot V_{\text{роб}} \cdot P_c \cdot C_{\text{ел}}, (5)$$

де $K_{\text{дн}}$ - період написання програми, дн.,

$K_{\text{дн}} = 30$ днів;

$V_{\text{роб}}$ - тривалість робочої зміни, ч., $V_{\text{роб}} = 8$ годин;

P_c - потужність, споживана технічними засобами, кВт · год;

$C_{\text{ел}}$ - вартість електроенергії за діючими тарифами, грн., кВт · год;

$$\text{Ц}_{\text{ел}} = 1,5 \text{ грн за кВт} \cdot \text{год.}$$

Звідси:

$$\text{В}_e = 30 \cdot 8 \cdot 0,25 \cdot 1,5 \approx 90.$$

4.5 Розрахунок заробітної плати програміста

Виходячи з фактично відпрацьованого часу програміста, яке склало 30 робочих восьмигодинних днів, знайдемо кількість фактично відпрацьованого часу, ч .:

$$\text{T}_{\text{ф}} = \text{K}_{\text{дн}} \cdot \text{В}_{\text{роб}}, (6)$$

де $\text{T}_{\text{ф}}$ - фактично відпрацьований час, ч .;

$\text{K}_{\text{дн}}$ - кількість відпрацьованих днів, дн;

$\text{В}_{\text{роб}}$ - тривалість робочого дня, год.

$$\text{T}_{\text{ф}} = 30 \cdot 8 = 240.$$

Беручи годинну заробітну плату програміста в розрахунку 400 грн.,
Отримаємо основну заробітну плату, грн:

$$\text{З}_{\text{осн}} = \text{T}_{\text{ф}} \cdot \text{T}_{\text{г}}, (7)$$

де $\text{T}_{\text{г}}$ - годинна тарифна ставка програміста.

$$\text{З}_{\text{осн}} = 240 \cdot 300 = 96000.$$

Для визначення загальної суми витрат на оплату праці необхідно врахувати доплати і надбавки. Приймаємо що доплата і надбавка в розмірі 15% від основної заробітної плати, грн:

$$\text{П}_{\text{доп}} = \text{З}_{\text{осн}} \cdot 0,15, (8)$$

$$\text{П}_{\text{доп}} = 96000 \cdot 0,15 = 14400.$$

Звідси знаходимо загальні витрати на оплату праці, грн :

$$B_{\text{заг}} = Z_{\text{осн}} + \Pi_{\text{доп}}, (9)$$

$$B_{\text{заг}} = 72000 + 10800 = 110400.$$

Далі визначимо єдиний соціальний податок; відповідно до ставки він становить 22% від витрат на оплату праці, що складе, грн:

$$O_{\text{соц}} = B_{\text{заг}} \cdot 0,22 (10)$$

$$O_{\text{соц}} = 82800 \cdot 0,22 \approx 24288.$$

4.6 Розрахунок загальних витрат на створення та експлуатацію програми

Разом, витрати на створення та експлуатацію мобільного додатку, складають, грн:

$$T = A_{\text{п}} + P_{\text{е}} + B_{\text{заг}} + O_{\text{соц}}, (11)$$

$$T = 289 + 90 + 110400 + 24288 \approx 135067.$$

4.7 Визначення відпускної ціни програми

Для знаходження відпускної ціни необхідно врахувати:

- прибуток від реалізації 15%;
- податок на додану вартість 18%.

$$C_{\text{н}} = T + \Pi_{\text{р}}, (12)$$

де $\Pi_{\text{р}}$ - прибуток від реалізації продукту, грн

$$\Pi_{\text{р}} = T \cdot 0,15, (13)$$

$$C_{\text{р}} = 135067 + 135067 \cdot 0,15 \approx 155327.$$

Ціна пакету програм з урахуванням ПДВ, грн:

$$C_{\text{п}} = C_{\text{р}} + \text{ПДВ}, (14)$$

де $\text{ПДВ} = C_{\text{н}} \cdot 0,18$ - податок на додану вартість.

$$C_{\text{п}} = 116567 + 116567 \cdot 0,18 \approx 183285.$$

Отже, відпускна ціна програми становить 183285 грн.

4.8 Економічна ефективність розробки і впровадження системи

Зарплата працівника в рік складає

$$(400 \cdot 8 \cdot 264) \cdot 0,2 = 168960 \text{ грн}$$

Річний економічний ефект розраховується по формулі:

$$\mathcal{E}_{\text{год}} = \Delta C_n - E_n * K$$

ΔC_n - вивільнені кошти після впровадження програмного продукту (168960 грн.) мінус технічні засоби та програмного забезпечення (13500 грн.) = 155460 грн.

E_n - коефіцієнт ефективності (дорівнює коефіцієнту амортизації (0,2));

K - одноразові витрати на впровадження продукту (155460 грн)

$$\mathcal{E}_{\text{год}} = 155460 - 135067 * 0,2 = 128447 \text{ (грн.)}$$

Строк окупності системи розраховується по формулі:

$$T = \frac{k}{\Delta C} = \frac{135067}{155460} \cong 0.8 \text{ (року)}$$

Отже строк окупності системи складає приблизно 9,6 місяців

РОЗДІЛ 5 ОХОРОНА ПРАЦІ

5.1 Вступ та теоретичні відомості охорони праці

Охорона праці - це система законодавчих актів, соціально-економічних, організаційних, технічних, гігієнічних і лікувально-профілактичних заходів та засобів, що забезпечують безпеку, збереження здоров'я і працездатності людини в процесі праці. Техніка безпеки являє собою систему засобів і методів, що запобігають або знижують до безпечного рівня вплив небезпечних факторів. Виробнича санітарія покликана усунути або знизити до безпечного рівня вплив шкідливих факторів [9].

Повністю безпечних та нешкідливих виробничих процесів не існує. Завдання охорони праці - звести до мінімуму ймовірність ураження або захворювання працюючого з одночасним забезпеченням комфорту при максимальній продуктивності праці.

З точки зору ступеня потенційної небезпеки для здоров'я і життя людини фактори поділяються на небезпечні та шкідливі.

Небезпечним виробничим фактором є такий фактор виробничого процесу, вплив якого на працюючого приводить до травми або різкого погіршення здоров'я.

Шкідливі виробничі фактори - це несприятливі фактори трудового процесу або умов навколишнього середовища, які можуть зробити шкідливий вплив на здоров'я і працездатність людини [10]. Тривала дія на людину шкідливого виробничого фактора призводить до захворювання.

Негативні фактори трудового процесу призводять до зниження працездатності та погіршення якості продукції, що випускається. Тривалий вплив несприятливих умов праці може призвести до порушення здоров'я працюючого, розвитку професійного захворювання або інвалідності.

Завданням охорони праці є гарантування безпечних і здорових умов праці та підтримання працездатності робітників. Безпечними умовами праці вважаються такі умови, при яких вплив на працюючих шкідливих та небезпечних виробничих факторів виключено або рівні їх впливу не перевищують встановлені нормативи.

5.2 Загальні вимоги до приміщення

Обов'язки забезпечення санітарно-побутових умов праці повинні брати на себе керівники структурних підрозділів.

Площа приміщення, в якому буде розташовано персональний комп'ютер, має підпадати під норми, які визначають згідно з чинними нормативними документами з розрахунку на одне робоче місце, обладнане ПК [11]:

- Площа має бути не менше 6 кв.м.
- Об'єм – не менше 20 куб.м.
- Відстань від вікна до робочого місця - не менше 1 м.
- Відстань між бічними поверхнями комп'ютерів – не менше 1.5 м.
- Відстань між тильною поверхнею одного комп'ютера і екраном іншого – не менше 2.5 м.
- Прохід між рядами – не менше 1м.

Також існують вимоги до організації приміщення: заземлені конструкції, що знаходяться в приміщеннях (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном тощо), мають бути надійно захищені діелектричними щитками або сітками для запобігання випадкового дотику. В робочих приміщеннях повинні бути медичні аптечки першої допомоги та система автоматичної пожежної сигналізації з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками з розрахунку 2 шт. на кожні 20 кв.м площі приміщення. Засоби пожежогасіння повинні бути вільними для швидкого доступу.

5.3 Вимоги до організації робочого місця

Робоче місце працівника повинно забезпечувати оптимальну сидячу позу з такими характеристиками: ступні ніг - на підлозі або на підставці для ніг; стегна - в горизонтальній площині; передпліччя - вертикально; лікті - під кутом 70 - 90 град. до вертикальної площини; зап'ястя зігнуті під кутом не більше 20 град. відносно горизонтальної площини, нахил голови - 15 - 20 град. відносно вертикальної площини. Якщо користування ПК є основним видом діяльності, то ПК і його периферійні пристрої (принтер, сканер тощо) розміщується на

основному робочому столі з лівого боку. Висота робочої поверхні столу для ПК має бути в межах 680 - 800 мм, а ширина - забезпечувати можливість виконання операцій в зоні досяжності моторного поля. Він повинен мати простір для ніг висотою не менше 600 мм, шириною не менше 500 мм, глибиною на рівні колін не менше 450 мм, на рівні витягнутої ноги – не менше 650 мм.

Робоче крісло користувача ПК повинно мати такі елементи: сидіння, спинку стаціонарні або знімні підлокітники.

Монітор та клавіатура мають розташовуватися на такій оптимальній відстані від очей користувача, але не повинні бути ближче ніж 600 мм, з урахуванням розміру алфавітно-цифрових знаків та символів.

Хибна організація робочого місця сприяє загальній і локальній напрузі м'язів шиї, тулуба, верхніх кінцівок, скривленню хребта й розвитку остеохондрозу та інших захворювань.

5.4 Аналіз параметрів робочого місця та приміщення

5.4.1 Геометричні параметри приміщення

В даному розділі проводиться аналіз середовища в якому розроблювався програмний продукт на основі санітарних норм України. Робоче приміщення (Табл. 4.2) розташоване на першому поверсі офісного будинку [12].

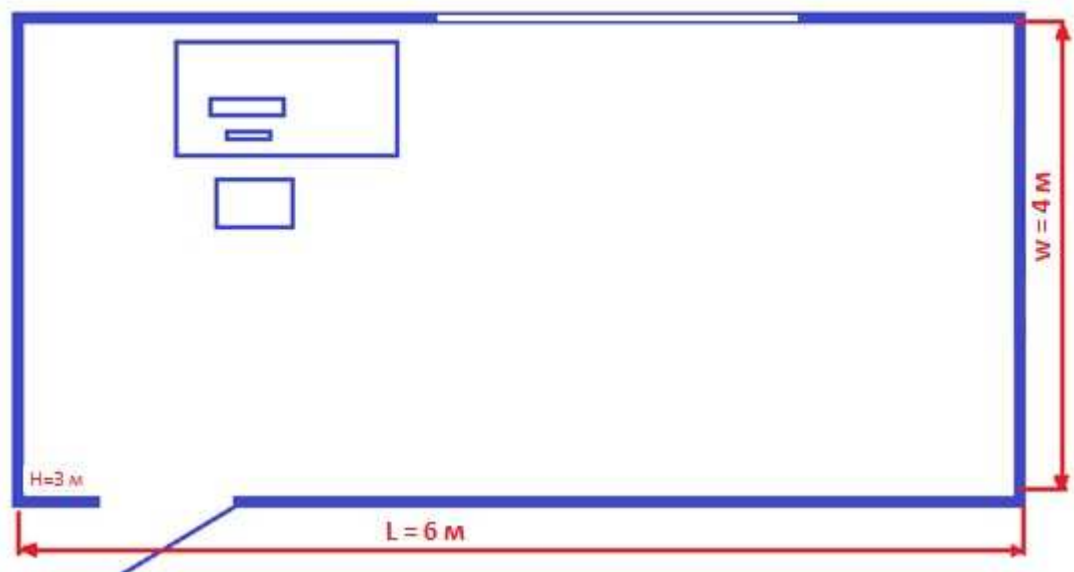


Рис.4.1 - План приміщення

Таблиця 4.1 – Норми приміщення

Параметр приміщення	Значення
Довжина, м	6
Ширина, м	4
Висота, м	3
Площа, м ²	24
Об'єм, м ³	72

Відповідно до [13], площа S' , виділена для одного робочого місця з персональною ЕОМ, повинна складати не менше 6 кв. м, а об'єм V' – не менше 20 куб. м. В нашому випадку на одного працюючого виділене все приміщення, тобто 24 кв. м. і 72 куб.м.

Бачимо, що площа та об'єм приміщення відповідає нормам згідно [15].

5.4.2 Параметри робочого місця

Таблиця 4.2 - Параметри робочого місця

Параметр	Нормативне значення	Фактичне значення
Висота робочої поверхні	680-800 мм	700 мм
Глибина робочої поверхні	800-1000мм	900 мм
Параметр	Нормативне значення	Фактичне значення
Висота сидіння над рівнем підлоги	400-500 мм	480 мм
Висота спинки стільця	300+/- 20 мм	320 мм
Регулювання нахилу спинки крісла	1-30°	1-30°
Глибина сидіння	400 мм та більше	500 мм
Висота простору для ніг	600 мм та більше	750 мм
Ширина простору для ніг	500 мм та більше	500 мм

Глибина простору для ніг	650 мм та більше	700 мм
Відстань від екрану до очей	600-700 мм	600 мм

Робочий стіл на досліджуваному місці також містить достатньо простору для ніг. Крісло, що використовується в якості робочого сидіння, є підйомноповоротним, має підлокітники і можливість регулювання за висотою і кутом нахилу спинки. Екран монітору знаходиться на відстані 0.8м, клавіатура має можливість регулювання кута нахилу 5-15°. Отже, за всіма параметрами робоче місце відповідає нормативним вимогам [14].

5.4.3 Аналіз та розрахунок стану освітлення в приміщенні

Відносно вікон робоче місце повинно бути розміщено так, щоб природне світло було збоку, переважно з лівого. Робоче місце, обладнане ПК повинно бути розташоване так, щоб уникнути попадання в очі прямого світла. Джерела штучного світла рекомендується розташувати з обох сторін від екрану паралельно напрямку зору. Вікна приміщень повинні мати регульовальні пристрої для відкривання. Як в приміщенні джерелом світла є штучне освітлення, то застосовуватися, як правило, люмінесцентні лампи [9].

Робота за дисплеєм ПЕОМ за розрядом зорових робіт відноситься до III розряду. При загальному висвітленні освітленість робочого місця повинна становити від 200 до 400 лк.

Розглянемо джерела природнього освітлення. В приміщенні знаходиться одне велике вікно з однієї сторони. Його характеристики (Табл. 4.3):

Таблиця 4.3 – Параметри приміщення

Параметр	Значення
Висота L	1.7 м.
Ширина W	2 м.
Відстань від підлоги	1 м.
Площа S	3.4 кв. м.

Вікна металопластикові, виходить на захід, можуть відкриватися та мають жалюзі.

Розглянемо джерела штучного освітлення. При штучному освітленні нормуються наступні параметри:

E (лк) - найменша припустима освітленість;

M - показник дискомфорту;

K_p (%) - коефіцієнт пульсації освітленості;

Перевіримо, чи відповідають нормам фактичні параметри штучного освітлення в приміщенні. Номінальний світловий потік лампи білого світіння ЛБ-40.

$$\Phi_{\text{л}} = 3120 \text{ лм.}$$

У приміщенні застосовуються світильники, у яких встановлені дві лампи.

Висоту підвісу світильника визначимо з формули:

$$h = H - h_c - h_p - h_n,$$

де

H - висота приміщення, м; h_c - висота світильника, м; h_n - відстань від стелі до підвісу, м; h_p - висота робочої поверхні, м.

Для розглянутого приміщення:

$$H = 2,7 \text{ м; } h_c = 0,2 \text{ м; } h_n = 0,3 \text{ м; } h_p = 0,7 \text{ м. звідси:}$$

$$h = 2,7 - 0,2 - 0,3 - 0,7 = 1,5 \text{ м.}$$

Світильники розташовані в 4 ряди. Відстань між рядами 2 метра, відстань від ряду до стіни 0,5 метра. Приміщення має наступні габарити:

$$\text{довжина } L = 6 \text{ м. ширина } W = 4 \text{ м.}$$

Визначимо освітленість у робочій точці. Для розрахунку загальної рівномірної освітленості при горизонтальній робочій поверхні використаємо метод коефіцієнта використання світлового потоку.

Розрахункова формула для світлового потоку світильника має вигляд:

$$\Phi_{\text{л}} = \frac{E \cdot K_s \cdot S \cdot Z}{N \cdot n}$$

де

N - число світильників у приміщенні, $N = 4 \cdot 2 = 8$; n - коефіцієнт використання світлового потоку;

$\Phi_{\text{л}}$ - світловий потік ламп;

K_3 - коефіцієнт запасу, $K_3 = 1.5$;

Z - коефіцієнт нерівномірності;

S - площа приміщення;

E - освітленість, створювана всіма світильниками.

Звідси одержуємо формулу для розрахунку освітленості на робочому місці:

$$E = \frac{\Phi_{\text{л}} \cdot N \cdot n}{K_3 \cdot S \cdot Z}$$

Коефіцієнт використання світлового потоку залежить від:

ККД, кривій розподілу сили світла світильника;

Коефіцієнта відбиття стелі R_c і стін R_s ;

Висоти підвісу світильників $h_{\text{п}}$;

Показника приміщення і обчислимо за формулою:

$$i = \frac{L * W}{h * (L + W)}$$

$$i = (6 * 4) / (1,5 * (6 + 4)) = 1,6.$$

Нам відомо, що стеля й стіни пофарбовані в світло-сірий і світло-бежевий кольори. Приймаємо:

$$R_{\text{п}} = 50\%, R_c = 30\%.$$

$$\text{Звідси: } n = 42\%.$$

$$E = \frac{3120 \cdot 8 \cdot 0,42}{1,5 \cdot 24,94 \cdot 1,1} = 254,75 \text{ лк}$$

Виходячи з всього вище згаданого фактична освітленість робочого місця повністю задовольняє вимогам і становить 254,75 лк. (при нормі від 200 до 400 лк.). Отже, для роботи з дисплеєм цілком достатньо існуючих джерел світла.

5.4.4 Аналіз повітряного середовища приміщення

У виробничих приміщеннях на робочих місцях забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря. Оптимальні та фактичні параметри мікроклімату для умов, що розглядаються (категорія робіт «Легка 1а») наведені в таблиці параметрів мікроклімату 4.4.

Таблиця 4.4 – Температурні норми

Період Року	Оптимальні			Фактичні		
	t, °C	W, %	V, м/с	t, °C	W, %	V, м/с
Холодний	20-23	60-40	0,1	22-23	55	0 – 0,2
Теплий	24-26	60-40	0,1	25-30	40	0 – 0,2

Спеціальні заходи з поліпшення або нормалізації цього параметру не потрібні. Умови мікроклімату у розглянутому приміщенні задовольняють вимогам встановленим у [16].

5.4.5 Шум

На робочому місці, що досліджується, джерелом шуму є ЕОМ та зовнішній кондиціонер (постійний). Допустимий рівень постійного звуку при програмуванні на ЕОМ – 50 дБА [9]. Оскільки основні механічні частини кондиціонера знаходяться зовні приміщення, шумовий тиск, що він створює, є мінімальний, та не перевищує фоновий. ПК, а саме кулери та блоки живлення, при роботі створюють мінімальний шумовий тиск. Інших джерел шуму у приміщенні немає. Отже, рівень звуку, який створюється джерелом шуму, повністю відповідає нормам [17].

5.4.6 Вимоги до пожежної та електро безпеки

5.5. Вимоги електробезпеки

Вимоги до електромережі, а також запобіжні засоби для уникнення травм від контакту з струмовідними елементами електроустаткування [16]:

- величина напруги мережі не більше за 380В та 220В (міжфазна лінійна і фазна відповідно);

- всі струмовідні елементи (в першу чергу електричні дроти) вкриті ізоляційними матеріалами;
- в джерелі безперебійного живлення персонального комп'ютера використовується механічне захисне блокування, що забезпечує вимикання напруги при його відкриванні;
- електромережа в приміщенні розведена в спеціальних каналах стін і підлоги.

В заключення можна сказати, що приміщення задовольняє вимогам електробезпеки.

5.6 Пожежна безпека

З огляду на можливість виникнення пожежі слід з'ясувати, які речовини і матеріали можуть горіти. У приміщенні, що розглядається, можуть горіти вироби з дерева, пластмас, тканини і паперу. Тому приміщення, що аналізується, відноситься, відповідно до нормативної документації, до зони П-Па [8] і до категорії пожежної небезпеки В.

Ймовірними причинами виникнення пожежу можуть бути несправність електрообладнання (кабелів, розеток), короткі замикання внаслідок виходу з ладу чи експлуатації несправного електроустаткування (ПЕОМ, периферійних пристроїв), порушення правил протипожежної безпеки тощо.

Комплекс заходів для попередження пожеж:

- - обов'язковий інструктаж персоналу з питань охорони праці,
- - зокрема, правила пожежної безпеки у приміщеннях з ЕОМ;
- - заборона використання відкритого вогню у приміщенні;
- - наявність системи автоматичної пожежної сигналізації з димовими пожежними оповіщувачами;
- - ступінь вогнестійкості будівлі, у якій розташовано приміщення – П;
- - наявність шляхів евакуації при виникненні пожежі;
- - розміщення схеми евакуації людей при пожежі і ознайомлення з нею персоналу.

Для гасіння пожежі кожна кімната обладнана ручними вуглекислотними вогнегасниками ВВК-1,4 [9]. У загальному коридорі встановлені пінні

вогнегасники ВВП. На сходах присутній спеціальний щит пожежного гідранта з відповідним рукавом. Розглянуте приміщення обладнане датчиками централізованої системи пожежної сигналізації. Призначена відповідальна особа, що відповідає за дотримання персоналом вимог пожежної безпеки.

Розроблено план евакуації персоналу і найбільш коштовного устаткування.

Отже, пожежна безпека відповідає нормам[9].

РОЗДІЛ 6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

Охорона навколишнього середовища - це комплекс заходів, призначених для обмеження негативного впливу людської діяльності на природу.

Мета охорони навколишнього середовища — протидія негативним змінам у навколишньому середовищі, які мали місце в минулому, відбуваються зараз або можуть бути.

Актуальність охорони навколишнього середовища, що перетворилася в глобальну проблему, пов'язана головним чином зі зростанням антропогенного впливу. Це зумовлено демографічним вибухом, урбанізацією, що прискорюється, і розвитком гірничих розробок і комунікацій, забрудненням навколишнього середовища відходами, надмірним навантаженням на орні землі, пасовища, ліси, водойми. У результаті гірничо-технічної діяльності у світі порушено не менше 15 — 20 млн га земель, з них 59% площі використано під різні гірничі виробки, 38% — під відвали пустої породи або відходів збагачення, 3% — місця осідання, провалів та інших порушень поверхні, пов'язаних з підземними розробками. Інколи порушення правил ведення гірничих робіт чи масштабна аварія призводить до катастрофічних незворотних наслідків.

Заходами, спрямованими на охорону довкілля можуть бути:

- Обмеження викидів в атмосферу та гідросферу з метою поліпшення загальної екологічної обстановки.
- Створення заповідників, заказників і національних парків з метою збереження природних комплексів.
- Обмеження лову риби, полювання з метою збереження певних видів.
- Обмеження несанкціонованого викидання сміття.

Використання методів екологічної логістики для тотального очищення від несанкціонованого засмічення території регіону.

Питання про вплив випромінювання мобільних телефонів на організм користувача до сих пір залишається відкритим. Численні дослідження, проведені вченими різних країн на біологічних об'єктах (в тому числі і на добровольцях) привели до неоднозначних, іноді суперечить одна одній результатами. Незаперечним залишається лише той факт, що організм людини «відгукується»

на наявність випромінювання стільникового телефону. Дійсно, мобільний телефон - це мініатюрний радіопередавач, щільно прикладається до вуха. При цьому антена (вбудована чи ні) знаходиться в безпосередній близькості від голови людини. Таким чином, електромагнітні хвилі просто не можуть не проникати в голову користувача. Фахівці рекомендують дотримуватися деяких запобіжних заходів власникам «мобільників»: - Не користуйтеся стільниковим телефоном без необхідності. - Розмовляйте безперервно не більше 3-4 хвилин. - Не допускайте, щоб МРТ користувалися діти, які не досягли підліткового віку-в доповіді Управління досліджень Європейського парламенту йдеться, що дитячий мозок особливо чутливий до впливу електромагнітного випромінювання, що призводить, зокрема, до послаблення імунітету і ряду неврологічних розладів, а також уповільнює розвиток дитини і, як мінімум, шкодить його успіхам у навчанні. - При покупці вибирайте стільниковий телефон з меншою максимальною потужністю випромінювання. - Безпосередньо в період виклику, коли «мобільник» шукає соту і вихідна потужність ЕМІ найбільша, варто відвести руку з телефоном на 15-20 см від голови. Коли зв'язок з абонентом встановлена, рівень випромінювання різко знижується і можна спокійно розмовляти. Для людей, які оточують людину, що розмовляє по мобільному радіотелефону, електромагнітне поле, створюване МРТ, не представляє ніякої небезпеки.

6.1 Електромагнітне забруднення навколишнього середовища і шляхи захисту від нього

В нашому оточуючому середовищі, окрім природних джерел електромагнітних променів (далі ЕМП), небезпеку для людини створюють і штучні джерела військового, промислового і побутового призначення. Електромагнітні джерела цивільного призначення включають в себе промислове обладнання, чисельні радіостанції і телецентри, інтерференція від хвиль яких досягає великих значень, мобільні телефони, комп'ютери та інші побутові прилади. Вони, наче електромагнітна павутина, обкутують середовище проживання людини, відбираючи у неї здоров'я. Всесвітня організація охорони здоров'я, вчені усіх розвинених країн світу в останній час зайнялися ретельним вивченням впливу ЕМП на людину. Результати таких досліджень невтішні: ми

живемо в умовах електромагнітного забруднення середовища. В останній час з'явився навіть такий термін - "електромагнітний смог". В 1990 році Управління з охорони навколишнього середовища США оприлюднило доповідь, в якій говорилося про причинний зв'язок між низькочастотними ЕМП і лейкемією, лімфомою і раком головного мозку у людей. Мобільні телефони, комп'ютери і різноманітна побутова електронна техніка – користуючись ними, люди не замислюються про невидимі оком ЕМП. Люди звикли недооцінювати небезпеку, особливо коли вона відразу не помітна, а вплив її проявляється через якийсь час. Однак ЕМП викликають масу серйозних проблем медико-біологічного характеру. Стан хронічної втоми, безсоння, відчуття тривоги, дискомфорту, пригніченість психіки, швидка утомлюваність, гул в голові – це типові функціональні розлади від дії ЕМП. Існує таке поняття "біорезонансні частоти". Більшість мобільних телефонів працюють на частоті 1800 МГц. Як відомо, люба хвиля має так звані «модуляції», тобто коливання в одну та іншу сторони відносно основної хвилі. Велика кількість модуляцій ЕМП потрапляє в резонанс з частотами живого організму людини. Незважаючи на те, що потужність мобільного телефону невелика, за рахунок резонансу в живих організмах все рівно буде проявлятися несприятливий ефект. Принцип несприятливої дії ЕМП на організм на клітковому рівні приблизно такий же, як при механічному резонансі, тобто збіг власних і примусових коливань об'єкту підсилює амплітуду коливань. Під впливом резонансу порушується нормальна робота живої клітини на інформаційному та енергетичному рівнях, руйнуються природні гідрофобні, водневі зв'язки і т.д. А оскільки нормальна робота клітини порушена, то це призведе до тієї чи іншої хвороби. Енергетичне ураження призводить до соматичних захворювань, підвищується алергічна реакція. У людини пошкоджується її енергетична захисна оболонка (біополе) і її уражають хвороби. Те, що ослаблюються нервова і імунна системи, серцево-судинна діяльність – це доведений факт. Тривалі розмови по мобільному телефону підвищують ризик виникнення пухлини слухового нерву. ЕМП здатні викликати загущення крові людини, що може призвести до утворення в ній тромбів. У відмінності від мобільного телефону комп'ютер є джерелом як надвисокочастотних полів самого комп'ютера, так і низькочастотних полів імпульсного блоку живлення

комп'ютера. Системний блок і блок живлення, на відміну від монітору, не мають захисту, тому особливо шкідливі. Навіть у разі нетривалої (45 хвилин) роботи з комп'ютером у людини під впливом електромагнітного випромінювання монітора відбуваються значні зміни гормонального стану і специфічні зміни біонапруженості головного мозку. У залежності від потужності випромінювання, розрізняють теплову і нетеплову дію випромінювання. Умовною межею між цими поняттями є величина в 10 мВт/см² опромінюваної поверхні. При такій величині потужності живі тканини можуть прогрітися на долі градуса. Відносно нешкідливими для людини впродовж тривалого часу вважаються поля, енергія яких сумірна з енергією геомагнітного поля Землі (0,04 кА/м) і з його аномаліями (0,16 кА/м). Якщо напруженість ЕМП вище, то починає відбуватися реакція на рівні організму. Значення напруженості, при яких починається виявлятися в тій чи іншій формі його дія на організм людини, наведені в таблиці.

№	п/п	Механізм дії магнітного поля	Напруженість, кА/м
1	Порушення просторової орієнтації молекул	800	
2	Зміна електропровідності рідини в живих тканинах	115	
3	Магнітні ефекти в хімічних реакціях	8...80	
4	Збільшення в'язкості води в живих тканинах	1	

Санітарно-гігієнічні і медико-профілактичні заходи із захисту людини в побуті від ЕМП мають на увазі дотримання наступних правил:

- Застосовувати при використанні мобільного телефону принцип "захист відстанню і часом" (як відомо, напруженість ЕМП прямо пропорційна напрузі і зворотно пропорційна квадрату відстані до джерела ЕМП), тобто використовувати подовжувальну гарнітуру "Вільні руки" або тримати при розмові телефон якомога далі від голови (режим роботи "Гучний зв'язок"); якомога менше носити його на поясі або на грудях, якщо є можливість - тримати його на відстані не менше 1 метра від себе; по можливості обмежувати час безперервного користування мобільним телефоном (не більше 3-х хвилин розмови, потім 15-ти хвилинна перерва); не користуватися мобільним телефоном в екранованих приміщеннях а також в місцях слабого зв'язку; під час розмови не триматися рукою за антенну телефону, бо це зменшує сигнал телефону і він автоматично підвищує передавану в антенну потужність.
- Використовувати прилади з найменшим рівнем енергоспоживання; розміщувати найбільш

небезпечні прилади на відстані не менш, ніж 1,5м від місць тривалого відпочинку та сну; не розміщувати прилади в кутах кімнат у будинках із залізобетонними конструкціями; по можливості використовувати прилади з автоматичним керуванням, що дозволяють не знаходитися поруч з ними під час роботи; не оставлять вилку у розетці при вимкненому приладі; заземлювати потужні прилади; додержуватися визначеної відстані між глядачем і екраном телевізора; користуючись комп'ютером, по можливості не перевищувати оптимальний нешкідливий час користування; систематично проводити вологе прибирання приміщень, застосовувати побутові іонізатори повітря, утримувати оптимальну вологість повітря в приміщенні. в) За 2-3 дні до магнітних бурь випивати 2-3 літри рідини щоденно у вигляді різних напоїв, страв та ін. для запобігання загущення крові, що є особливо актуальним для людей із серцево-судинними захворюваннями.

6.2 Висновки

Внаслідок проведеного аналізу санітарно-гігієнічних умов праці, умов електробезпеки і пожежної безпеки приміщення, де виконуються роботи з використанням ЕОМ, було зроблено висновок про відповідність переважної більшості чинників нормативним вимогам.

Загалом незначна кількість невідповідностей фактичних умов праці нормам дозволяє зробити висновок, що умови праці в приміщенні, що розглядалося, є задовільними. Рекомендації щодо подальшої роботи - щогодини треба робити перерву на 5-10 хвилин, а через 2 години - на 15 хвилин. Один або кілька разів у годину необхідно виконувати серію легких вправ на розтягування, що можуть зменшити напругу, що накопичується в м'язах при тривалій роботі на комп'ютері.

ВИСНОВКИ

В магістерській роботі розв'язано актуальну наукову задачу розроблення методів та засобів імітаційного моделювання процесів розвитку структур комп'ютерних мереж, що має важливе значення для підвищення ефективності функціонування цих мереж і захищеності їх елементів від спрямованих атак.

Розвиток сучасних інформаційних технологій супроводжується збільшенням ролі телекомунікаційних систем різного призначення та комп'ютерних мереж.

Особливе місце в цих завданнях займають сучасні технології комп'ютерних мереж, серед яких слід виділити локальні та глобальні мережі. Це пояснюється необхідністю використання корпоративної інформації, що міститься в корпоративних базах даних, які можуть розташовуватися як в окремих підрозділах підприємства, так й за його межами. Отже сучасні технології оброблення документів різного призначення повинні базуватися на засобах телекомунікаційного зв'язку й стандартів комп'ютерних мереж, які виступають як транспортні системи передачі даних.

Особливої актуальності набуває моделювання в області проектування комп'ютерних мереж. При проектуванні цих систем виникають численні завдання, що вимагають оцінки кількісних і якісних закономірностей процесів функціонування цих систем, проведення аналізу їх роботи. Ці системи відносяться до складних систем, що характеризується складністю структури, випадковістю зв'язків між окремими їх елементами, неоднозначністю поведінки при різних внутрішніх станах і зовнішніх впливах, великій кількості випадкових факторів, мінливим характером вихідної інформації. З урахуванням складності такої системи, основним засобом її дослідження на ранньому етапі розробки є моделювання. По суті справи моделювання є єдиною можливістю створення ефективної системи організації оптимальних режимів її експлуатації.

Використовуючи моделювання при проектуванні обчислювальної системи, ми можемо зробити наступне: оцінити пропускну здатність мережі та її компонентів, визначити вузькі місця в структурі обчислювальної системи;

порівняти різні варіанти організації обчислювальної системи; здійснити перспективний прогноз розвитку обчислювальної системи; передбачити майбутні вимоги по пропускну здатності мережі, використовуючи дані прогнозу; оцінити необхідну кількість і продуктивність серверів в мережі; порівняти різні варіанти модернізації обчислювальної системи; оцінити вплив на обчислювальну систему модернізації ПО, потужності робочих станцій або серверів, зміни мережевих протоколів.

Дослідження параметрів обчислювальної системи при різних характеристиках окремих компонентів дозволяє вибрати мережеве та обчислювальне устаткування з урахуванням продуктивності, якості обслуговування, надійності і вартості.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Комп'ютерні мережі. Загальні принципи функціонування комп'ютерних мереж. Навчальний посібник. С. В. Мінухін, С. В. Кавун, С. В. Знахур. – Харків: Вид. ХНЕУ, 2008. – с. 210
2. Основы компьютерных сетей // [Електронний ресурс] www.gatefromitam.omsk.net.
3. Мінухін С. В. Лабораторний практикум із навчальної дисципліни "Комп'ютерні мережі" / С. В. Мінухін, В. Ю. Жукареєв. – Харків: Вид. ХНЕУ, 2007. – 230 с.
4. Богуш К.Ю., Богуш Ю.П., Шиян А.И. GPSS World Моделювання телекомунікаційних систем та мереж. Посібник для дипломного проектування . – 2010. – с. 325
5. Кудрявцев Е.М. GPSS Word. Основы имитационного моделирования различных систем.- М. ДМК Пресс, 2004. – с. 119
6. Задорожный В.Н. Аналитико-имитационные исследования систем и сетей массового обслуживания: монография / В.Н. Задорожный. – Омск: Изд- во ОмГТУ. 2010. – 280 с.
7. Боев В. Д. Об адекватности систем имитационного моделирования GPSS World и AnyLogic. Часть 2 // Прикладная информатика. № 4 (34). 2011. С. 50-62.
8. Романова О. І. «Телекоммуникационные сети и управление». – К: ВПЦ «Киевский Университет», 2003, – 247 с.
9. Швиденко М.З., Матус Ю.В.. Комп'ютерні мережні технології. / Навч.- метод. посібник. – Київ. – ТОВ "Авета", - 2008. с. 350
10. Кулаков Ю.О., Максимено Є.В., Безштанько В.М. Комп'ютерні мережі //Конспект лекцій. К.: Вид-во ІСЗЗІ НТУУ «КПІ», 2009 – с. 259
11. Організація комп'ютерної локальної мережі. Василь Олексюк, Надія Балик, Анатолій Балик: Тернопіль, - 2006р. - с. 80
12. Пасічник В. В. Моделювання і візуалізація засобами PROCESSING комп'ютерних мереж / В. В. Пасічник, Н. М. Іванущак // Математичні машини і системи. – 2013. – № 1. – С. 118-126.

ДОДАТОК А

```
package leader_election_protocol;
import java.sql.Connection;
import java.sql.SQLException;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.Currency;
import java.util.Date;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;
import java.util.Locale;
import java.util.Map;
import java.util.Random;
import java.util.Set;
import java.util.SortedMap;
import java.util.SortedSet;
import java.util.Stack;
import java.util.Timer;
import java.util.TreeMap;
import java.util.TreeSet;
import java.util.Vector;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;

import static java.lang.Math.*;
import static com.xj.anylogic.engine.presentation.UtilitiesColor.*;
import static com.xj.anylogic.engine.presentation.UtilitiesDrawing.*;
import static com.xj.anylogic.engine.HyperArray.*;

import com.xj.anylogic.engine.*;
import com.xj.anylogic.engine.analysis.*;
import com.xj.anylogic.engine.connectivity.*;
import com.xj.anylogic.engine.connectivity.ResultSet;
import com.xj.anylogic.engine.connectivity.Statement;
import com.xj.anylogic.engine.presentation.*;

import java.awt.geom.Arc2D;

public class Main extends ActiveObject
{
    // Параметры

    public
    double MeanLatency;

    /**
     * Возвращает значение по умолчанию параметра <code>MeanLatency</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
    */
}
```



```

    */
    public
    double _MeanLatency_DefaultValue_xjal() {
        return
        1.0
        ;
    }

    public void set_MeanLatency(
    double MeanLatency ) {
        if (MeanLatency == this.MeanLatency) {
            return;
        }
        this.MeanLatency = MeanLatency;
        onChange_MeanLatency();
        onChange();
    }

    void onChange_MeanLatency() {
        int index;
        network.set_MeanLatency( MeanLatency );
    }

    public
    double LossRate;

    /**
     * Возвращает значение по умолчанию параметра <code>LossRate</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    public
    double _LossRate_DefaultValue_xjal() {
        return
        0.0
        ;
    }

    public void set_LossRate(
    double LossRate ) {
        if (LossRate == this.LossRate) {
            return;
        }
        this.LossRate = LossRate;
        onChange_LossRate();
        onChange();
    }

    void onChange_LossRate() {
        int index;
        network.set_LossRate( LossRate );
    }

    public
    int NoOfMachines;

    /**
     * Возвращает значение по умолчанию параметра <code>NoOfMachines</code>.
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    public
    int _NoOfMachines_DefaultValue_xjal() {

```

```

        return
19
;
    }

    public void set_NoOfMachines(
int NoOfMachines ) {
    if (NoOfMachines == this.NoOfMachines) {
        return;
    }
    this.NoOfMachines = NoOfMachines;
    onChange_NoOfMachines();
    onChange();
}

void onChange_NoOfMachines() {
    int index;
    index = 0;
    for ( Machine object : machines ) {
        object.set_NoOfMachines(NoOfMachines);
        index++;
    }

}

public
double ConflictTimeout;

/**
 * Возвращает значение по умолчанию параметра <code>ConflictTimeout</code>.
 * <i>Пользователь не должен вызывать этот метод</i>
 */
public
double _ConflictTimeout_DefaultValue_xjal() {
    return
5.0
;
}

public void set_ConflictTimeout(
double ConflictTimeout ) {
    if (ConflictTimeout == this.ConflictTimeout) {
        return;
    }
    this.ConflictTimeout = ConflictTimeout;
    onChange_ConflictTimeout();
    onChange();
}

void onChange_ConflictTimeout() {
    int index;
    index = 0;
    for ( Machine object : machines ) {
        object.set_ConflictTimeout(ConflictTimeout);
        index++;
    }

}

public
double MTTR;

/**

```

```

    * Возвращает значение по умолчанию параметра <code>MTTR</code>.
    * <i>Пользователь не должен вызывать этот метод</i>
    */
    public
    double _MTTR_DefaultValue_xjal() {
        return
100
    ;
    }

    public void set_MTTR(
double   MTTR ) {
        if (MTTR == this.MTTR) {
            return;
        }
        this.MTTR = MTTR;
        onChange_MTTR();
        onChange();
    }

    void onChange_MTTR() {
        int index;
        index = 0;
        for ( Machine object : machines ) {
            object.set_MTTR(MTTR);
            index++;
        }
    }

    public
    double   SynchroTimeout;

    /**
    * Возвращает значение по умолчанию параметра <code>SynchroTimeout</code>.
    * <i>Пользователь не должен вызывать этот метод</i>
    */
    public
    double _SynchroTimeout_DefaultValue_xjal() {
        return
3.0
    ;
    }

    public void set_SynchroTimeout(
double   SynchroTimeout ) {
        if (SynchroTimeout == this.SynchroTimeout) {
            return;
        }
        this.SynchroTimeout = SynchroTimeout;
        onChange_SynchroTimeout();
        onChange();
    }

    void onChange_SynchroTimeout() {
        int index;
        index = 0;
        for ( Machine object : machines ) {
            object.set_SynchroTimeout(SynchroTimeout);
            index++;
        }
    }
}

```

```

// События

public EventTimeout _machinesStatesCharts_autoUpdateEvent_xjal = new
EventTimeout(this);

@Override
public String getNameOf( EventTimeout _e ) {
    if ( _e == _machinesStatesCharts_autoUpdateEvent_xjal ) return
"machinesStatesCharts auto update event";
    return super.getNameOf( _e );
}

@Override
public int getModeOf( EventTimeout _e ) {
    if ( _e == _machinesStatesCharts_autoUpdateEvent_xjal ) return
EVENT_TIMEOUT_MODE_CYCLIC;
    return super.getModeOf( _e );
}

@Override
public double getFirstOccurrenceTime( EventTimeout _e ) {
    if (
        _e == _machinesStatesCharts_autoUpdateEvent_xjal
    ) return getEngine().getStartTime();
    return super.getFirstOccurrenceTime( _e );
}

@Override
public double evaluateTimeoutOf( EventTimeout _e ) {
    if ( _e == _machinesStatesCharts_autoUpdateEvent_xjal ) return
1
;
    return super.evaluateTimeoutOf( _e );
}

@Override
public void executeActionOf( EventTimeout _e ) {
    if ( _e == _machinesStatesCharts_autoUpdateEvent_xjal ) {
        machinesStatesCharts.updateData();
        return;
    }
    super.executeActionOf( _e );
}
// Вложенные объекты

public Network network;

public String getNameOf( ActiveObject ao ) {
    if ( ao == network ) return "network";
    return null;
}

public ActiveObjectArrayList<Machine> machines = new
ActiveObjectArrayList<Machine>();

public String getNameOf( ActiveObjectCollection<?> aolist ) {
    if( aolist == machines ) return "machines";
    return null;
}

/**
 * Этот метод создает и добавляет новый вложенный объект в список экземпляров
реплицированного объектов machines<br>
 * @return только что созданный вложенный объект

```

```

*/
public Machine add_machines() {
    int index = machines.size();
    Machine object = instantiate_machines_xjal( index );
    setupParameters_machines_xjal( object, index );
    create_machines_xjal( object, index );
    object.start();
    return object;
}

/**
 * Этот метод создает и добавляет новый вложенный объект в список экземпляров
 * реплицированного объектов machines<br>
 * Этот метод использует заданные значения параметров для инициализации
 * созданного вложенного объекта<br>
 * Индекс нового экземпляра реплицированного объекта может быть получен с
 * помощью метода <code>machines.size()</code> <strong>перед</strong> вызовом
 * этого метода
 * @param StartUpTimeout
 * @param ConsistencyTimeout
 * @param AcceptTimeout
 * @param CandidateTimeout
 * @param ElectionTimeoutHigh
 * @param ConflictTimeout
 * @param SynchroTimeout
 * @param ResolveTimeoutHigh
 * @param ResolveTimeoutLow
 * @param ElectionTimeoutLow
 * @param MTTR
 * @param NoMasterTimeout
 * @param MTTF
 * @param NoOfMachines
 * @param radius
 * @return только что созданный вложенный объект
 */
public Machine add_machines( double StartUpTimeout, double ConsistencyTimeout,
double AcceptTimeout, double CandidateTimeout, double ElectionTimeoutHigh,
double ConflictTimeout, double SynchroTimeout, double ResolveTimeoutHigh, double
ResolveTimeoutLow, double ElectionTimeoutLow, double MTTR, double
NoMasterTimeout, double MTTF, int NoOfMachines, int radius ) {
    int index = machines.size();
    Machine object = instantiate_machines_xjal( index );
    // Задание значений параметров
    object.StartUpTimeout = StartUpTimeout;
    object.ConsistencyTimeout = ConsistencyTimeout;
    object.AcceptTimeout = AcceptTimeout;
    object.CandidateTimeout = CandidateTimeout;
    object.ElectionTimeoutHigh = ElectionTimeoutHigh;
    object.ConflictTimeout = ConflictTimeout;
    object.SynchroTimeout = SynchroTimeout;
    object.ResolveTimeoutHigh = ResolveTimeoutHigh;
    object.ResolveTimeoutLow = ResolveTimeoutLow;
    object.ElectionTimeoutLow = ElectionTimeoutLow;
    object.MTTR = MTTR;
    object.NoMasterTimeout = NoMasterTimeout;
    object.MTTF = MTTF;
    object.NoOfMachines = NoOfMachines;
    object.radius = radius;
    // Завершение создания вложенного объекта
    create_machines_xjal( object, index );
    object.start();
    return object;
}

/**

```

```

    * Этот метод удаляет заданный вложенный объект из коллекции экземпляров
    реплицированного объекта machines<br>
    * Данный объект будет уничтожен, но в общем случае это произойдет позже.
    * @param object активный объект - элемент реплицированного вложенного объекта
    machines, который должен быть удален
    * @return <code>true</code>, если объект успешно удален, <code>false</code>,
    если он не принадлежит machines
    */
    public boolean remove_machines( Machine object ) {
        if( ! machines._remove( object ) ){
            return false;
        }
        object.setDestroyed();
        return true;
    }

    /**
    * Создает экземпляр вложенного объекта и добавляет его в конец списка
    экземпляров реплицированных объектов<br>
    * <i>Пользователь не должен вызывать этот метод</i>
    */
    private Machine instantiate_machines_xjal( final int index ) {
        Machine object = new Machine( getEngine(), this, machines );

        machines._add(object);

        return object;
    }

    /**
    * Инициализация параметров экземпляра вложенного объекта<br>
    * Пользователь не должен вызывать этот метод
    */
    private void setupParameters_machines_xjal(Machine object, final int index ) {
        object.StartupTimeout = object._StartupTimeout_DefaultValue_xjal();
        object.ConsistencyTimeout = object._ConsistencyTimeout_DefaultValue_xjal();
        object.AcceptTimeout = object._AcceptTimeout_DefaultValue_xjal();
        object.CandidateTimeout = object._CandidateTimeout_DefaultValue_xjal();
        object.ElectionTimeoutHigh =
object._ElectionTimeoutHigh_DefaultValue_xjal();
        object.ConflictTimeout =
ConflictTimeout
;
        object.SynchroTimeout =
SynchroTimeout
;
        object.ResolveTimeoutHigh = object._ResolveTimeoutHigh_DefaultValue_xjal();
        object.ResolveTimeoutLow = object._ResolveTimeoutLow_DefaultValue_xjal();
        object.ElectionTimeoutLow = object._ElectionTimeoutLow_DefaultValue_xjal();
        object.MTTR =
MTTR
;
        object.NoMasterTimeout = object._NoMasterTimeout_DefaultValue_xjal();
        object.MTTF = object._MTTF_DefaultValue_xjal();
        object.NoOfMachines =
NoOfMachines
;
        object.radius = object._radius_DefaultValue_xjal();
    }

    /**
    * Инициализация экземпляра вложенного объекта<br>
    * Пользователь не должен вызывать этот метод
    */
    private void create_machines_xjal(Machine object, final int index ) {

```

```

        object.create();

        // Port connections
        // connector:
        network.port.connect( object.port );
    }
    /**
     * Создает экземпляр вложенного объекта<br>
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    private Network instantiate_network_xjal() {
        Network object = new Network( getEngine(), this, null );

        return object;
    }

    /**
     * Инициализация параметров экземпляра вложенного объекта<br>
     * Пользователь не должен вызывать этот метод
     */
    private void setupParameters_network_xjal(Network object ) {
        object.MeanLatency =
MeanLatency
;
        object.LossRate =
LossRate
;
    }

    /**
     * Инициализация экземпляра вложенного объекта<br>
     * Пользователь не должен вызывать этот метод
     */
    private void create_network_xjal(Network object ) {
        object.create();
    }

    // Области
    public ViewArea MainView = new ViewArea( this, "MainView", 0, 0,
ViewArea.TOP_LEFT, ViewArea.NONE, 1.0, 100, 100 );

    static final Font _text_Font = new Font("SansSerif", 1, 14 );
    static final Font _text1_Font = new Font("SansSerif", 0, 11 );
    static final Font _text2_Font = _text1_Font;
    static final Font _text3_Font = _text1_Font;
    static final Font _text4_Font = _text1_Font;
    static final Font _text5_Font = _text1_Font;
    static final Font _text6_Font = _text1_Font;
    static final Font _text7_Font = _text1_Font;
    static final Font _text8_Font = _text1_Font;
    static final Font _text9_Font = _text1_Font;
    static final Font _Text_Font = new Font("SansSerif", 1, 11 );
    static final Font _Text1_Font = _Text_Font;
    static final Font _Text2_Font = _text1_Font;
    static final Font _Text3_Font = _text1_Font;
    static final Font _Text4_Font = _Text_Font;
    static final Font _Text5_Font = _text1_Font;
    static final Font _Text6_Font = _text1_Font;
    static final Font _Text7_Font = _Text_Font;
    static final Font _Text8_Font = _Text_Font;
    static final Font _Text9_Font = _text1_Font;
    static final Font _text15_Font = _text_Font;
    static final Font _Text10_Font = _Text_Font;
    static final Font _Text11_Font = _Text_Font;

```

```

static final int _Slider = 1;
static final int _Slider1 = 2;
static final int _Slider2 = 3;
static final int _Slider3 = 4;
static final int _Slider4 = 5;
static final int machines_presentation = 6;
static final int _line = 7;
static final int _group = 8;
static final int _text = 9;
static final int _rectangle1 = 10;
static final int _text1 = 11;
static final int _text2 = 12;
static final int _rectangle2 = 13;
static final int _rectangle3 = 14;
static final int _text3 = 15;
static final int _rectangle4 = 16;
static final int _text4 = 17;
static final int _text5 = 18;
static final int _rectangle5 = 19;
static final int _rectangle6 = 20;
static final int _text6 = 21;
static final int _rectangle7 = 22;
static final int _text7 = 23;
static final int _rectangle8 = 24;
static final int _text8 = 25;
static final int _text9 = 26;
static final int _rectangle9 = 27;
static final int _Text = 28;
static final int _Text1 = 29;
static final int _Text2 = 30;
static final int _Text3 = 31;
static final int _Text4 = 32;
static final int _Text5 = 33;
static final int _Text6 = 34;
static final int _Text7 = 35;
static final int _Text8 = 36;
static final int _Text9 = 37;
static final int _text15 = 38;
static final int _line1 = 39;
static final int _line2 = 40;
static final int _Text10 = 41;
static final int _rectangle = 42;
static final int _rectangle10 = 43;
static final int _Text11 = 44;
static final int _machinesStatesCharts = 45;

/**
 * Идентификатор группы presentation верхнего уровня
 */
static final int _presentation = 0;

/**
 * Идентификатор группы icon верхнего уровня
 */
static final int _icon = -1;

@Override
public void executeShapeControlAction( int _shape, int index, double value ) {
    switch( _shape ) {
        case _Slider:
            set_MTTR( value );
            break;
        case _Slider1:
            set_LossRate( value );
    }
}

```



```

        break;
    case _Slider2:
        set_ConflictTimeout( value );
        break;
    case _Slider3:
        set_SynchroTimeout( value );
        break;
    case _Slider4:
        set_MeanLatency( value );
        break;
    default:
        super.executeShapeControlAction( _shape, index, value );
        break;
    }
}

@Override
public double getShapeControlMinimum( int _shape, int index ) {
    switch( _shape ) {
        case _Slider: return
20
;
        case _Slider1: return
0.0
;
        case _Slider2: return
1.0
;
        case _Slider3: return
1.0
;
        case _Slider4: return
0.2
;
        default: return super.getShapeControlMinimum( _shape, index );
    }
}

@Override
public double getShapeControlMaximum( int _shape, int index ) {
    switch( _shape ) {
        case _Slider: return
500
;
        case _Slider1: return
1.0
;
        case _Slider2: return
20.0
;
        case _Slider3: return
7.0
;
        case _Slider4: return
5.0
;
        default: return super.getShapeControlMaximum( _shape, index );
    }
}

@Override
public double getShapeControlDefaultValueDouble( int _shape, int index ) {
    switch( _shape ) {
        case _Slider: return
MTTR

```

```

;
    case _Slider1: return
LossRate
;
    case _Slider2: return
ConflictTimeout
;
    case _Slider3: return
SynchroTimeout
;
    case _Slider4: return
MeanLatency
;
    default: return super.getShapeControlDefaultValueDouble( _shape, index );
}
}

@Override
public String getNameOfShape( int _shape ) {
    switch( _shape ) {
        case machines_presentation: return "machines_presentation";
        default: return super.getNameOfShape( _shape );
    }
}

@Override
public int getShapeType( int _shape ) {
    switch( _shape ) {
        case machines_presentation: return SHAPE_EMBEDDED_OBJECT;
        default: return super.getShapeType( _shape );
    }
}

@Override
public int getShapeReplication( int _shape ) {
    switch( _shape ) {
        case machines_presentation: return
machines.size()
;
        default: return super.getShapeReplication( _shape );
    }
}

@Override
public double getShapeX( int _shape, int index ) {
    switch( _shape ) {
        case machines_presentation: return 0;
        default: return super.getShapeX( _shape, index );
    }
}

@Override
public double getShapeY( int _shape, int index ) {
    switch( _shape ) {
        case machines_presentation: return 0;
        default: return super.getShapeY( _shape, index );
    }
}

@Override
public Object getShapeEmbeddedObject( int _shape ) {
    switch( _shape ) {
        case machines_presentation: return machines;
        default: return super.getShapeEmbeddedObject( _shape );
    }
}

```

```

    }
}

/**
 * <i>Пользователь не должен вызывать этот метод</i>
 */
private void _Slider_SetDynamicParams_xjal( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _Slider ), getShapeControlMaximum(
_Slider ) );
}

ShapeSlider Slider;

/**
 * <i>Пользователь не должен вызывать этот метод</i>
 */
private void _Slider1_SetDynamicParams_xjal( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _Slider1 ), getShapeControlMaximum(
_Slider1 ) );
}

ShapeSlider Slider1;

/**
 * <i>Пользователь не должен вызывать этот метод</i>
 */
private void _Slider2_SetDynamicParams_xjal( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _Slider2 ), getShapeControlMaximum(
_Slider2 ) );
}

ShapeSlider Slider2;

/**
 * <i>Пользователь не должен вызывать этот метод</i>
 */
private void _Slider3_SetDynamicParams_xjal( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _Slider3 ), getShapeControlMaximum(
_Slider3 ) );
}

ShapeSlider Slider3;

/**
 * <i>Пользователь не должен вызывать этот метод</i>
 */
private void _Slider4_SetDynamicParams_xjal( ShapeSlider shape ) {
    boolean _visible;
    shape.setRange( getShapeControlMinimum( _Slider4 ), getShapeControlMaximum(
_Slider4 ) );
}

ShapeSlider Slider4;
public Color _machinesStatesCharts_ColorFromDouble( double value ) {
    if (
value == Machine.Consistency

```

```

        ) return blue;
    if (
value == Machine.StartUp
        ) return plum;
    if (
value == Machine.NoMaster
        ) return darkGoldenRod;
    if (
value == Machine.Slave
        ) return steelBlue;
    if (
value == Machine.Master
        ) return darkOrange;
    if (
value == Machine.Accept
        ) return lime;
    if (
value == Machine.Candidate
        ) return blueViolet;
    if (
value == Machine.Conflict
        ) return red;
    if (
value == Machine.Crashed
        ) return silver;
    return null;
}

```

```
TimeColorChart machinesStatesCharts;
```

```

/**
 * <i>Пользователь не должен вызывать этот метод</i>
 */
private void _line_SetDynamicParams_xjal( ShapeLine shape, int index ) {
    boolean _visible;
    shape.setX(
machines.get( index ).x
    );
    shape.setY(
machines.get( index ).y
    );
    shape.setDx(
machines.get( index ).linkExists() ? machines.get( index ).MyMaster.x -
machines.get( index ).x : 0
    );
    shape.setDy(
machines.get( index ).linkExists() ? machines.get( index ).MyMaster.y -
machines.get( index ).y : 0
    );
    shape.setColor(
machines.get( index ).linkExists() ? slateBlue : null
    );
}

/**
 * <i>Пользователь не должен вызывать этот метод</i>
 */
public ShapeLine _line_createShapeWithStaticProperties( final int _index ) {
    ShapeLine shape = new ShapeLine(
        true, 0, 0,
        slateBlue,
        60, 0,
        1, LINE_STYLE_SOLID
    );
}

```

```

        return shape;
    }

    /**
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    private int _line_Replication() {
        return
NoOfMachines
    ;
    }

    ReplicatedShape<ShapeLine> line;
    ShapeGroup group;
    ShapeText text;
    ShapeRectangle rectangle1;
    ShapeText text1;
    ShapeText text2;
    ShapeRectangle rectangle2;
    ShapeRectangle rectangle3;
    ShapeText text3;
    ShapeRectangle rectangle4;
    ShapeText text4;
    ShapeText text5;
    ShapeRectangle rectangle5;
    ShapeRectangle rectangle6;
    ShapeText text6;
    ShapeRectangle rectangle7;
    ShapeText text7;
    ShapeRectangle rectangle8;
    ShapeText text8;
    ShapeText text9;
    ShapeRectangle rectangle9;

    /**
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    private void _Text_SetDynamicParams_xjal( ShapeText shape ) {
        boolean _visible;
        shape.setText(
LossRate
    );
    }

    ShapeText Text;

    /**
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    private void _Text1_SetDynamicParams_xjal( ShapeText shape ) {
        boolean _visible;
        shape.setText(
MeanLatency
    );
    }

    ShapeText Text1;
    ShapeText Text2;
    ShapeText Text3;

    /**
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    private void _Text4_SetDynamicParams_xjal( ShapeText shape ) {
        boolean _visible;

```

```

        shape.setText(
ConflictTimeout
);
    }

    ShapeText Text4;
    ShapeText Text5;
    ShapeText Text6;

    /**
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    private void _Text7_SetDynamicParams_xjal( ShapeText shape ) {
        boolean _visible;
        shape.setText(
SynchroTimeout
);
    }

    ShapeText Text7;

    /**
     * <i>Пользователь не должен вызывать этот метод</i>
     */
    private void _Text8_SetDynamicParams_xjal( ShapeText shape ) {
        boolean _visible;
        shape.setText(
MTTR
);
    }

    ShapeText Text8;
    ShapeText Text9;
    ShapeText text15;
    ShapeLine line1;
    ShapeLine line2;
    ShapeText Text10;
    ShapeRectangle rectangle;
    ShapeRectangle rectangle10;
    ShapeText Text11;

    // Статическая инициализация элементов, у которых разрешено программное
управление
    {
        Slider = new ShapeSlider(
            Main.this, true, 230, 170,
            180, 30,
            transparent,
            false, getShapeControlMinimum( _Slider ), getShapeControlMaximum(
_Slider ), ShapeControl.TYPE_DOUBLE
        ) {

            @Override
            public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
                _Slider_SetDynamicParams_xjal( this );
                super.draw( panel, graphics, xform, publicOnly );
            }

            @Override
            public void action(){
                executeShapeControlAction( _Slider, 0, value );
            }

            @Override

```

```

        public void setValueToDefault() {
            setValue( limit( getMin(), getShapeControlDefaultValueDouble(
_slider1, 0 ), getMax() ) );
        }
        /**
         * Это число используется при сохранении состояния модели
         */
        private static final long serialVersionUID = 1193920047681L;
    };
    Slider1 = new ShapeSlider(
        Main.this, true, 230, 50,
        180, 30,
        transparent,
        false, getShapeControlMinimum( _slider1 ), getShapeControlMaximum(
_slider1 ), ShapeControl.TYPE_DOUBLE
    ) {

        @Override
        public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
            _slider1_SetDynamicParams_xjal( this );
            super.draw( panel, graphics, xform, publicOnly );
        }

        @Override
        public void action(){
            executeShapeControlAction( _slider1, 0, value );
        }

        @Override
        public void setValueToDefault() {
            setValue( limit( getMin(), getShapeControlDefaultValueDouble(
_slider1, 0 ), getMax() ) );
        }
        /**
         * Это число используется при сохранении состояния модели
         */
        private static final long serialVersionUID = 1193920047703L;
    };
    Slider2 = new ShapeSlider(
        Main.this, true, 230, 140,
        180, 30,
        transparent,
        false, getShapeControlMinimum( _slider2 ), getShapeControlMaximum(
_slider2 ), ShapeControl.TYPE_DOUBLE
    ) {

        @Override
        public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
            _slider2_SetDynamicParams_xjal( this );
            super.draw( panel, graphics, xform, publicOnly );
        }

        @Override
        public void action(){
            executeShapeControlAction( _slider2, 0, value );
        }

        @Override
        public void setValueToDefault() {
            setValue( limit( getMin(), getShapeControlDefaultValueDouble(
_slider2, 0 ), getMax() ) );
        }
        /**

```

```

        * Это число используется при сохранении состояния модели
        */
        private static final long serialVersionUID = 1193920047715L;
    };
    Slider3 = new ShapeSlider(
        Main.this, true, 230, 80,
        180, 30,
        transparent,
        false, getShapeControlMinimum( _Slider3 ), getShapeControlMaximum(
        _Slider3 ), ShapeControl.TYPE_DOUBLE
    ) {

        @Override
        public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
        boolean publicOnly ) {
            _Slider3_SetDynamicParams_xjal( this );
            super.draw( panel, graphics, xform, publicOnly );
        }

        @Override
        public void action(){
            executeShapeControlAction( _Slider3, 0, value );
        }

        @Override
        public void setValueToDefault() {
            setValue( limit( getMin(), getShapeControlDefaultValueDouble(
            _Slider3, 0 ), getMax() ) );
        }
        /**
         * Это число используется при сохранении состояния модели
         */
        private static final long serialVersionUID = 1193920047723L;
    };
    Slider4 = new ShapeSlider(
        Main.this, true, 230, 110,
        180, 30,
        transparent,
        false, getShapeControlMinimum( _Slider4 ), getShapeControlMaximum(
        _Slider4 ), ShapeControl.TYPE_DOUBLE
    ) {

        @Override
        public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
        boolean publicOnly ) {
            _Slider4_SetDynamicParams_xjal( this );
            super.draw( panel, graphics, xform, publicOnly );
        }

        @Override
        public void action(){
            executeShapeControlAction( _Slider4, 0, value );
        }

        @Override
        public void setValueToDefault() {
            setValue( limit( getMin(), getShapeControlDefaultValueDouble(
            _Slider4, 0 ), getMax() ) );
        }
        /**
         * Это число используется при сохранении состояния модели
         */
        private static final long serialVersionUID = 1193920047733L;
    };
    line = new ReplicatedShape<ShapeLine>() {

```



```

@Override
public int getReplication() {
    return _line_Replication();
}

@Override
public ShapeLine createShapeWithStaticProperties( int index ) {
    return _line_createShapeWithStaticProperties( index );
}

@Override
public void setShapeDynamicProperties( ShapeLine shape, int index ) {
    _line_SetDynamicParams_xjal( shape, index );
}
/**
 * Это число используется при сохранении состояния модели
 */
private static final long serialVersionUID = 1193817306127L;
};
text = new ShapeText(
    true,30, 260, 0.0,
    black,"Machine states",
    _text_Font, ALIGNMENT_LEFT
);
rectangle1 = new ShapeRectangle(
    true,210, 560, 0.0,
    null, blue,
    20, 10,
    1, LINE_STYLE_SOLID
);
text1 = new ShapeText(
    true,240, 560, 0.0,
    black,"Consistency",
    _text1_Font, ALIGNMENT_LEFT
);
text2 = new ShapeText(
    true,340, 560, 0.0,
    black,"StartUp",
    _text2_Font, ALIGNMENT_LEFT
);
rectangle2 = new ShapeRectangle(
    true,310, 560, 0.0,
    null, plum,
    20, 10,
    1, LINE_STYLE_SOLID
);
rectangle3 = new ShapeRectangle(
    true,390, 560, 0.0,
    null, darkGoldenRod,
    20, 10,
    1, LINE_STYLE_SOLID
);
text3 = new ShapeText(
    true,420, 560, 0.0,
    black,"NoMaster",
    _text3_Font, ALIGNMENT_LEFT
);
rectangle4 = new ShapeRectangle(
    true,130, 560, 0.0,
    null, steelBlue,
    20, 10,
    1, LINE_STYLE_SOLID
);
text4 = new ShapeText(
    true,160, 560, 0.0,

```

```

        black, "Slave",
        _text4_Font, ALIGNMENT_LEFT
    );
    text5 = new ShapeText(
        true, 80, 560, 0.0,
        black, "Master",
        _text5_Font, ALIGNMENT_LEFT
    );
    rectangle5 = new ShapeRectangle(
        true, 50, 560, 0.0,
        null, darkOrange,
        20, 10,
        1, LINE_STYLE_SOLID
    );
    rectangle6 = new ShapeRectangle(
        true, 480, 561, 0.0,
        null, lime,
        20, 10,
        1, LINE_STYLE_SOLID
    );
    text6 = new ShapeText(
        true, 510, 560, 0.0,
        black, "Accept",
        _text6_Font, ALIGNMENT_LEFT
    );
    rectangle7 = new ShapeRectangle(
        true, 560, 560, 0.0,
        null, blueViolet,
        20, 10,
        1, LINE_STYLE_SOLID
    );
    text7 = new ShapeText(
        true, 590, 560, 0.0,
        black, "Candidate",
        _text7_Font, ALIGNMENT_LEFT
    );
    rectangle8 = new ShapeRectangle(
        true, 660, 560, 0.0,
        null, red,
        20, 10,
        1, LINE_STYLE_SOLID
    );
    text8 = new ShapeText(
        true, 690, 560, 0.0,
        black, "Conflict",
        _text8_Font, ALIGNMENT_LEFT
    );
    text9 = new ShapeText(
        true, 780, 560, 0.0,
        black, "Crashed",
        _text9_Font, ALIGNMENT_LEFT
    );
    rectangle9 = new ShapeRectangle(
        true, 750, 560, 0.0,
        null, silver,
        20, 10,
        1, LINE_STYLE_SOLID
    );
    Text = new ShapeText(
        true, 190, 60, 0.0,
        blue, "123",
        _Text_Font, ALIGNMENT_LEFT
    ) {

        @Override

```

```

        public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
            _Text_SetDynamicParams_xjal( this );
            super.draw( panel, graphics, xform, publicOnly );
        }
/**
 * Это число используется при сохранении состояния модели
 */
        private static final long serialVersionUID = 1193920047691L;
    };
    Text1 = new ShapeText(
        true,190, 120, 0.0,
        blue,"123",
        _Text1_Font, ALIGNMENT_LEFT
    ) {

        @Override
        public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
            _Text1_SetDynamicParams_xjal( this );
            super.draw( panel, graphics, xform, publicOnly );
        }
/**
 * Это число используется при сохранении состояния модели
 */
        private static final long serialVersionUID = 1193920047693L;
    };
    Text2 = new ShapeText(
        true,50, 120, 0.0,
        black,"Network mean latency:",
        _Text2_Font, ALIGNMENT_LEFT
    );
    Text3 = new ShapeText(
        true,50, 90, 0.0,
        black,"Synchro timeout:",
        _Text3_Font, ALIGNMENT_LEFT
    );
    Text4 = new ShapeText(
        true,190, 150, 0.0,
        blue,"123",
        _Text4_Font, ALIGNMENT_LEFT
    ) {

        @Override
        public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
            _Text4_SetDynamicParams_xjal( this );
            super.draw( panel, graphics, xform, publicOnly );
        }
/**
 * Это число используется при сохранении состояния модели
 */
        private static final long serialVersionUID = 1193920047701L;
    };
    Text5 = new ShapeText(
        true,50, 150, 0.0,
        black,"Conflict timeout:",
        _Text5_Font, ALIGNMENT_LEFT
    );
    Text6 = new ShapeText(
        true,50, 180, 0.0,
        black,"Mean recovery time:",
        _Text6_Font, ALIGNMENT_LEFT
    );
    Text7 = new ShapeText(

```

```

        true,190, 90, 0.0,
        blue,"123",
        _Text7_Font, ALIGNMENT_LEFT
    ) {

        @Override
        public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
            _Text7_SetDynamicParams_xjal( this );
            super.draw( panel, graphics, xform, publicOnly );
        }
        /**
         * Это число используется при сохранении состояния модели
         */
        private static final long serialVersionUID = 1193920047725L;
    };
    Text8 = new ShapeText(
        true,190, 180, 0.0,
        blue,"123",
        _Text8_Font, ALIGNMENT_LEFT
    ) {

        @Override
        public void draw( Panel panel, Graphics2D graphics, AffineTransform xform,
boolean publicOnly ) {
            _Text8_SetDynamicParams_xjal( this );
            super.draw( panel, graphics, xform, publicOnly );
        }
        /**
         * Это число используется при сохранении состояния модели
         */
        private static final long serialVersionUID = 1193920047731L;
    };
    Text9 = new ShapeText(
        true,50, 60, 0.0,
        black,"Network loss rate:",
        _Text9_Font, ALIGNMENT_LEFT
    );
    text15 = new ShapeText(
        true,30, 20, 0.0,
        black,"Parameters",
        _text15_Font, ALIGNMENT_LEFT
    );
    line1 = new ShapeLine(
        true,30, 280,
        silver,
        790, 0,
        1, LINE_STYLE_SOLID
    );
    line2 = new ShapeLine(
        true,30, 40,
        silver,
        390, 0,
        1, LINE_STYLE_SOLID
    );
    Text10 = new ShapeText(
        true,50, 210, 0.0,
        black,"Tip: click on a machine to view its statechart",
        _Text10_Font, ALIGNMENT_LEFT
    );
    rectangle = new ShapeRectangle(
        true,40, 210, 0.0,
        null, red,
        5, 5,
        1, LINE_STYLE_SOLID
    );

```

```

);
rectangle10 = new ShapeRectangle(
    true, 40, 225, 0.0,
    null, red,
    5, 5,
    1, LINE_STYLE_SOLID
);
Text11 = new ShapeText(
    true, 50, 225, 0.0,
    black, "Tip: click a circle near the machine to force it to crash or
recover",
    _Text11_Font, ALIGNMENT_LEFT
);
}
ShapeGroup presentation;
ShapeGroup icon;

@Override
public Object getPersistentShape( int _shape ) {
    switch( _shape ) {
        case _presentation: return presentation;
        case _icon: return icon;

        case _Slider: return Slider;
        case _Slider1: return Slider1;
        case _Slider2: return Slider2;
        case _Slider3: return Slider3;
        case _Slider4: return Slider4;
        case _machinesStatesCharts: return machinesStatesCharts;
        case _line: return line;
        case _group: return group;
        case _text: return text;
        case _rectangle1: return rectangle1;
        case _text1: return text1;
        case _text2: return text2;
        case _rectangle2: return rectangle2;
        case _rectangle3: return rectangle3;
        case _text3: return text3;
        case _rectangle4: return rectangle4;
        case _text4: return text4;
        case _text5: return text5;
        case _rectangle5: return rectangle5;
        case _rectangle6: return rectangle6;
        case _text6: return text6;
        case _rectangle7: return rectangle7;
        case _text7: return text7;
        case _rectangle8: return rectangle8;
        case _text8: return text8;
        case _text9: return text9;
        case _rectangle9: return rectangle9;
        case _Text: return Text;
        case _Text1: return Text1;
        case _Text2: return Text2;
        case _Text3: return Text3;
        case _Text4: return Text4;
        case _Text5: return Text5;
        case _Text6: return Text6;
        case _Text7: return Text7;
        case _Text8: return Text8;
        case _Text9: return Text9;
        case _text15: return text15;
        case _line1: return line1;
        case _line2: return line2;
        case _Text10: return Text10;
        case _rectangle: return rectangle;
    }
}

```

```

        case _rectangle10: return rectangle10;
        case _Text11: return Text11;
        default: return null;
    }
}

/**
 * Конструктор
 */
public Main( Engine engine, ActiveObject owner, ActiveObjectCollection<?
extends Main> collection ) {
    super( engine, owner, collection );
}

@Override
public void create() {
    // Создание экземпляров вложенных объектов
    for ( int i = 0; i <
NoOfMachines
; i++ ) {
        instantiate_machines_xjal( i );
    }
    network = instantiate_network_xjal();
    // Динамическая инициализация элементов, у которых разрешено программное
управление
    {
        DataSet _item;
        List<DataSet> _items = new ArrayList<DataSet>( 0 );
        List<String> _titles = new ArrayList<String>( 0 );
        machinesStatesCharts = new TimeColorChart(
            Main.this, true, 10, 280,
            820, 270,
            null, null,
            40, 10,
            770, 240, whiteSmoke, null, black,
            30, Chart.NONE,

1000
            , null, 0.8,
            Chart.GRID_DEFAULT,
            darkGray, darkGray, _items, _titles,
            new TimeColorChart.ColorMap() {
                @Override
                public Color colorFromDouble( double value ) {
                    return _machinesStatesCharts_ColorFromDouble( value );
                }
            }
            /**
             * Это число используется при сохранении состояния модели
             */
            private static final long serialVersionUID = 1193821314892L;
        }
    };
}
{
    group = new ShapeGroup( Main.this, true, 630, 140, 0.0

    , machines_presentation
    , line

    );
}
presentation = new ShapeGroup( Main.this, true, 0, 0, 0, Slider, Slider1,
Slider2, Slider3, Slider4, group, text, rectangle1, text1, text2, rectangle2,
rectangle3, text3, rectangle4, text4, text5, rectangle5, rectangle6, text6,
rectangle7, text7, rectangle8, text8, text9, rectangle9, Text, Text1, Text2,

```

```

Text3, Text4, Text5, Text6, Text7, Text8, Text9, text15, line1, line2, Text10,
rectangle, rectangle10, Text11, machinesStatesCharts );
    icon = new ShapeGroup( Main.this, true, 0, 0, 0
);
    // Создание реплицированных фигур
    line.createShapes();
    // Создание нереплицированных вложенных объектов
    setupParameters_network_xjal( network );
    create_network_xjal( network );
    // Соединители с нереплицированными объектами
    // Создание реплицированных вложенных объектов
    for ( int i = 0; i < machines.size(); i++ ) {
        setupParameters_machines_xjal( machines.get(i), i );
        create_machines_xjal( machines.get(i), i );
    }
    assignInitialConditions();
    Slider.setValueToDefault();
    Slider1.setValueToDefault();
    Slider2.setValueToDefault();
    Slider3.setValueToDefault();
    Slider4.setValueToDefault();
    onCreate();
}

@Override
public void start() {
    machinesStatesCharts_autoUpdateEvent_xjal.start();
    for (ActiveObject embeddedObject : machines){
        embeddedObject.start();
    }
    network.start();
    onStartup();
}

public void onStartup() {
    super.onStartup();

    for ( Machine m : machines ){
        machinesStatesCharts.addDataSet( m.StatechartStateDS );
    }
}

public List<Object> getEmbeddedObjects() {
    LinkedList<Object> list = new LinkedList<Object>();
    list.add( machines );
    list.add( network );
    return list;
}

public void onDestroy() {
    super.onDestroy();
    machinesStatesCharts_autoUpdateEvent_xjal.onDestroy();
    for (ActiveObject embeddedObject : machines) {
        embeddedObject.onDestroy();
    }
    network.onDestroy();
}

/**
 * Это число используется при сохранении состояния модели
 */
private static final long serialVersionUID = 1196475115179L;

private void writeObject(java.io.ObjectOutputStream _stream) throws
java.io.IOException {

```

```
        _stream.defaultWriteObject();
        writeCustomData( _stream );
    }

    private void readObject(java.io.ObjectInputStream _stream) throws
java.io.IOException, ClassNotFoundException {
        _stream.defaultReadObject();
        machines.restoreOwner( this );
        network.restoreOwner( this );
        MainView.restoreOwner( this );
        _machinesStatesCharts_autoUpdateEvent_xjal.restoreOwner( this );
        finishReadObject_xjal( _stream, Main.class );
    }
}
```