ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Підвищення відмовостійкості комп'ютерної системи за допомогою оптимізації апаратної та програмної частин»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Максим ВІТАШ*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. КІД-42
Максим ВІТАШ
Ім'я, ПРІЗВИЩЕ
Керівник: Андрій ЛЕМЕШКО
доктор філософії, доцент Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра 123 Комп'ютерної інженерії

Ступінь вищої освіти бакалавр

Спеціальність 123 Комп'ютерної інженерії

Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Віташу Максиму Олеговичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Підвищення відмовостійкості комп'ютерної системи за допомогою оптимізації апаратної та програмної частин
_____
керівник кваліфікаційної роботи доктор філософії, доцент Андрій ЛЕМЕШКО,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «__ _____» _____2024р. № _____
2. Строк подання кваліфікаційної роботи «_____» _____ 2024р.
3. Вихідні дані до кваліфікаційної роботи:
   1. Політика та програма забезпечення надійності
   2. Інтернет ресурси стосовно забезпечення відмовостійкості мереж
   3. Науково-технічна література

Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Основи надійності та відмовостійкості інформаційної системи
2. Відмовостійкість комп'ютерних систем
3. Забезпечення стійкості та живучості спеціалізованих інформаційних технологій у комп'ютерних мережах
4. Перелік ілюстративного матеріалу: *презентація*
5. Дата видачі завдання «_____» _____ 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	26.02-22.03.2024	Виконано
2	Обробка матеріалу	23.03-05.04.2024	Виконано
3	Основи надійності та відмовостійкості інформаційної системи	06.04-10.04.2024	Виконано
4	Відмовостійкість комп’ютерних систем	11.04-30.04.2024	Виконано
5	Забезпечення стійкості та живучості спеціалізованих інформаційних технологій у комп'ютерних мережах	01.05-03.05.2024	Виконано
6	Висновки за результатами аналізу	04.05-08.05.2024	Виконано
7	Розробка презентації	09.05-11.05.2024	Виконано
8	Передзахист	14.05-17.05.2024	Виконано
9	Здача в деканат	27.05-3.06.2024	Виконано

Здобувач(ка) вищої освіти _____ Максим ВІТАШ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Андрій ЛЕМЕШКО
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 71 стор., 10 рис., 20 джерел.

Мета роботи – Дослідження та оптимізація апаратної та програмної частин комп'ютерної системи з метою підвищення її відмовостійкості.

Об'єкт дослідження – процес оптимізації та підвищення відмовостійкості.

Предмет дослідження – комп'ютерна система.

Короткий зміст роботи: В дипломній роботі розроблено підхід до визначення ефективності ІТ на основі врахування кількісних величин, що характеризують відмовостійкість і живучість, який може бути розширено для врахування інших характеристичних величин. Для забезпечення відмовостійкості та живучості ІТ розроблено систему заходів, у результаті виконання яких отримано ІТ вузькоспеціалізованого використання для різноманітних сфер застосування, де супроводження процесів відносяться до систем ірреального або нереального часу, з достатньо високими параметрами відмовостійкості, живучості та загалом резилентності і, водночас, прийнятними фінансовими витратами на її експлуатацію.

КЛЮЧОВІ СЛОВА: НАДІЙНІСТЬ, ВІДМОВОСТІЙКІСТЬ, КОМП'ЮТЕРНА СИСТЕМА, ОПТИМІЗАЦІЇ, КЕРОВАНІСТЬ ІНФОРМАЦІЙНИХ СИСТЕМ, СТІЙКОСТЬ, ЖИВУЧІСТЬ

ABSTRACT

The text part of the qualification work for the bachelor's degree: 71 pages, 10 figures, 20 sources.

The purpose of the work is Research and optimization of hardware and software parts of the computer system in order to increase its fault tolerance.

The object of research is optimization process and increased fault tolerance.

The subject of research is computer system.

Summary of the work: The thesis developed an approach to determining IT efficiency based on taking into account quantitative values characterizing fault tolerance and survivability, which can be expanded to take into account other characteristic values. To ensure fault tolerance and survivability of IT, a system of measures was developed, as a result of which IT was obtained for highly specialized use for various fields of application, where process support refers to unreal or non-real time systems, with sufficiently high parameters of fault tolerance, survivability and overall resilience and, at the same time, acceptable financial expenses for its operation.

KEY WORDS: RELIABILITY, FAILURE TOLERANCE, COMPUTER SYSTEM, OPTIMIZATION, MANAGEMENT OF INFORMATION SYSTEMS, RESISTANCE, SURVIVAL

ЗМІСТ

ВСТУП

Інформаційні системи являють собою складні програмно-апаратні комплекси і є невід'ємною частиною сучасних технічних систем. Багато з них у складі автоматизованих систем керування здійснюють управління технологічними процесами в реальному масштабі часу. Структурна та функціональна надійність і безпека інформаційних систем мають визначальний вплив на ефективність функціонування автоматизованих систем управління. Звідси очевидно, що без забезпечення надійності інформаційних систем не можна домогтися безперебійної та безпомилкової роботи систем управління.

Завдання побудови надійних інформаційних систем мають комплексний характер - вони не можуть обмежитися тільки структурною або тільки функціональною надійністю. Тут під структурною надійністю мається на увазі надійність продукції - об'єктів (елементів, систем). Функціональна надійність - це надійність надання послуг (виконання процесів збирання, опрацювання, передавання інформації, управління підлеглими об'єктами). Тільки поєднання способів підвищення і однієї, і іншої складових надійності дає відчутний ефект. Цей ефект може виражатися в значному підвищенні надійності системи і при цьому в економії обсягу додаткових ресурсів. Наприклад, контроль стану структурної надійності об'єкта інформаційної системи може здійснюватися за допомогою методів функціональної надійності шляхом оцінки безпомилковості видаваних об'єктом даних.

Надійні відмовостійкі інформаційні системи тому результатів оброблення інформації. З іншого боку, для виявлення збоїв апаратури найчастіше застосовують методи, засновані на аналізі проміжних і вихідних результатів. Відомо, що реалізація мажоритарного резервування здійснюється комплексно методами структурної надійності (структурна надмірність) і методами функціональної надійності (програмна реалізація логічних дій відновлювального органу). Можна навести безліч подібних прикладів. Головне, щоб застосовувані способи комплексного підвищення надійності органічно відповідали структурі та процесам

функціонування досліджуваного об'єкта. Ця умова не завжди здійсненна, особливо щодо традиційних способів. Комплексне розв'язання проблеми забезпечення надійності слугує також методичним фундаментом, на якому створені й розвиваються способи забезпечення функціональної безпеки.

Ці завдання надзвичайно актуальні в даний час для інформаційних систем реального часу, які керують критично важливими об'єктами, що становлять підвищену небезпеку для навколишнього середовища, об'єктами. До подібних систем висувають підвищені вимоги щодо надійності та функціональної безпеки. Підтвердження відповідності системи цим вимогам може бути практично виконано на основі прискорених натурних випробувань при доведенні адекватності цих випробувань реальним умовам. Підтвердження відповідності заданим вимогам до системи тісно пов'язане з підтвердженням відповідності до якості, функціональної та інформаційної безпеки програмного забезпечення.

Для інформаційних технологій (ІТ), що забезпечують життєдіяльність установ чи підприємств у різних спеціалізованих галузях, питання живучості та відмовостійкості є дуже важливими, особливо внаслідок зростання їхніх кількісних параметрів функціонування (збільшення кількості користувачів, серверів, об'ємів інформації в базах даних) та рівня складності. Від забезпечення цих параметрів у прямій залежності перебуває ефективність функціонування всієї ІТ. Для спеціалізованих ІТ, що функціонують у корпоративних комп'ютерних мережах і виконують функцію інформаційного забезпечення в такій вузькоспеціалізованій предметній області, як фінансово-господарська діяльність у різних сферах застосування, цей параметр значно вищий за нуль, але вимоги до таких ІТ також доволі високі, які неможливо виконати за низьких параметрів відмовостійкості та живучості, особливо за умови постійного зростання кількості користувачів, збільшення складності інформаційних потоків та обсягів оброблюваних даних. Забезпечення високої ефективності спеціалізованих ІТ здійснюється на основі реалізації в них принципів відмовостійкості та живучості, що є актуальним науковим завданням, яке починає розв'язуватися ще в процесі розроблення спеціалізованих ІТ.

РОЗДІЛ 1. ОСНОВИ НАДІЙНОСТІ ТА ВІДМОВОСТІЙКОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Життєвий цикл надійності інформаційної системи

Різноманітні рішення та дії, спрямовані на досягнення та збереження надійності інформаційної системи, ухвалюють і здійснюють на різних стадіях її розроблення та експлуатації.

Під життєвим циклом системи розуміють послідовність стадій протягом усього життя системи, від планування до виведення її з експлуатації та утилізації. Життєвий цикл полягає в плануванні, управлінні, перевірці та спостереженні за всіма аспектами системи, включно з інформаційною захищеністю, надійністю та функціональною безпекою, так як система, проходять через окремі стадії для того, щоб бажаний продукт із бажаним рівнем безпеки та за бажаною ціною було надано впродовж обумовленого строку.

На рис. 1.1 показано життєвий цикл надійності інформаційної системи. Він нерозривно пов'язаний із життєвим циклом самої системи і відображає сучасні погляди до організації життєвого циклу для широкого класу складних технічних систем, яскравим представником якого є ІС.

На першій стадії формується концепція створення ІС. Мета цієї стадії полягає у створенні достатнього уявлення про систему для того, щоб усі наступні стадії життєвого циклу надійності ІС могли задовільно виконуватися. При цьому вирішуються три групи завдань:

1) Визначення сфери застосування, концепції та цілей цього проекту; аналіз необхідності й достатності фінансування та регламентація управління проектом;

2) Перевірка зв'язку проекту з надійністю; визначення цілей структурної та функціональної надійності, виявлення джерел загроз надійності, насамперед під час взаємодії з іншими системами та під час взаємодії з людьми; одержання й аналіз інформації з відомими раніше джерелами загроз, досягнутих рівнів надійності для

аналогічних або споріднених систем;

3) Документування всіх результатів цієї стадії (ці документі мають бути ключовими вхідними даними для наступних стадій життєвого циклу;

4) Верифікація пропорційності вхідних даних, коректності та узгодженості з вимогами документації з підтримки надійності життєвого циклу, кваліфікації всіх працівників, що беруть участь у цій роботі, а також оцінювання гарантії того, що систему розробляють згідно з вимогами до неї, а застосовані при цьому технології, методи, способи, інструментальні засоби є адекватними.

На другій стадії проводиться опис системи і встановлюються технічні умови. Вирішуються три групи завдань:

1) Визначення профілю експлуатаційного завдання, розробка опису системи, визначення стратегій і умов експлуатації та технічного обслуговування ІС, визначення обмежень, пов'язаних з інфраструктурою ІС;

2) Оцінювання наявної статистики надійності, попередній аналіз загроз, розробка політики забезпечення надійності на рівні всієї системи, вибір критеріїв ризику;

3) Документування всіх отриманих результатів, включно з політикою безпеки та програмою робіт із надійності й безпеки ІС;

4) Верифікація отриманих документів та розробленого завдання на наступні стадії життєвого циклу, оцінювання кваліфікації співробітників, які виконують завдання цієї стадії.

На третій стадії виконується аналіз ризику. Ця стадія призначена для розв'язання таких завдань: ідентифікація загроз, які пов'язані із системою; ідентифікація подій, що спричиняють ці загрози; визначення ризику, пов'язаного із загрозами; розробка процесу безперервного управління ризиком.

На цій стадії вирішуються такі групи завдань:

1) Систематичний пошук і класифікація всіх ризиків, можливих за нормальних умов, ідентифікація прихованих загроз, виявлення частоти виникнення подій, пов'язаних з наявними загрозами, виявлення/оцінка розміру впливів наявних загроз, виявлення ризику для системи, пов'язаного з кожною загрозою;

2) Аналіз і класифікація допустимості ризиків, що відповідають кожній відомій загрозі, розробляють протокол загроз як базис для управління ризиком. Після того, як виявлено неприпустимо високі ризики, що підлягають зменшенню або ліквідації засобами забезпечення надійності ІС, розглядаються очікувані залишкові ризики та підтверджується їхня прийнятність.

3) Документування всіх отриманих результатів, включно з протоколом загроз безпеці;

4) Верифікація отриманих документів, розробленого завдання на подальші етапи життєвого циклу, оцінювання повноти оцінювання ризику, оцінювання класифікації допустимості ризиків, оцінювання придатності протоколу загроз для системи, що розглядається, оцінювання придатності методів (способів, процесів), інструментів та технічних прийомів (технологій), що використовуються на цьому етапі, оцінювання кваліфікації працівників, що виконують завдання цього етапу.

На четвертій стадії розробляють вимоги до ІС. Вирішуються три групи завдань:

1) розроблення вимог до всієї системи з урахуванням специфіки довкілля, вибір інформаційних технологій, розроблення вимог до якості та організації управління, конфігурації ІС, розроблення критеріїв доказу виконання вимог і приймання, розроблення й узгодження плану атестації системи;

2) розроблення вимог до структурної й функціональної надійності системи, а також критеріїв приймання надійності, регламентація керування надійністю, розроблення так званої програми RAMS (програми комплексного забезпечення надійності, експлуатаційної готовності, відновлюваності та безпеки).

Розроблення системного завдання з безпеки, що містить опис системних вимог безпеки, зокрема перелік ризиків, яким необхідно протистояти, і цілей безпеки, які необхідно досягти за допомогою функцій безпеки, а також опис процедурних і адміністративних регуляторів.

3) Документування отриманих результатів;

4) Верифікація всіх видів розроблених вимог, критеріїв доказу, системного завдання з надійності, а також оцінювання використаних методів, способів, засобів

і оцінювання кваліфікації співробітників.

На п'ятій стадії розподіляють вимоги до підсистем ІС та їхньої надійності, зокрема визначають політики забезпечення надійності для функціонально незалежних підсистем із власними доменами надійності, розробляють профілі захисту, функції (технічні регулятори) безпеки. Уже на цій ранній стадії життєвого циклу слід розпочинати проведення оцінювання та доведення структурної та функціональної надійності ІС. Це дасть змогу розробникам і замовникам краще розуміти систему і передбачуване експлуатаційне середовище, усунути невідповідності та упущення у вимогах безпеки і пропонованих функціях безпеки та регуляторах безпеки. Це надалі полегшить оцінювачам аналіз проєктної та експлуатаційної документації, отримання свідоцтв довіри до досягнутих рівнів структурної та функціональної надійності.

На шостій стадії проводиться безпосереднє розроблення (за необхідності конструювання окремих складових системи) та її реалізація. На цій стадії розв'язуються також завдання архітектурного та детального проєктування програмних засобів ІС, кодування та тестування програмних компонентів, розроблення інтерфейсних засобів, вибору та адаптації інформаційних технологій, протоколів обміну інформацією, а також інтеграції програмних та апаратних засобів, верифікації, виконання планів забезпечення надійності системи та програми RAMS, вироблення спільного доказу надійності ІС, документування програмних та апаратних засобів, розроблення експлуатаційної документації.

На сьомій стадії передбачається виготовлення системи. Вирішуються також завдання реалізації плану забезпечення надійності, тестування апаратних компонентів і налагодження програмних засобів, розроблення документації, навчання персоналу.

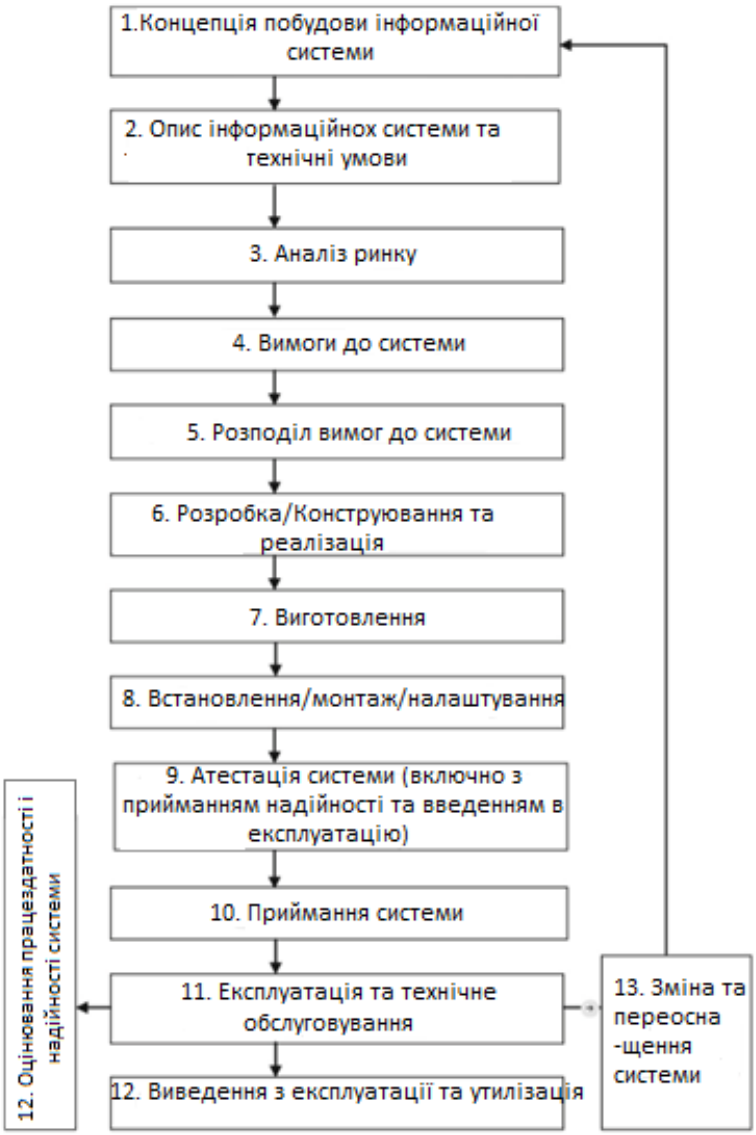


Рисунок 1.1 - Життєвий цикл надійності інформаційної системи

На цій стадії також розв'язуються задачі закупівлі та адаптації необхідного базового та прикладного програмного забезпечення, включно з програмними засобами забезпечення надійності, а також системна інтеграція, конфігурація та тестування розробником/системним інтегратором.

На восьмій стадії проводять монтаж, установку і налаштування системи, навчання обслуговуючого персоналу, доопрацювання експлуатаційної документації. На цій стадії також створюють структуру забезпечення надійності для адміністративного і процедурного рівнів, документують політики, правила і процедури забезпечення надійності з урахуванням завдань системної інтеграції

На дев'ятій стадії проводять підтвердження відповідності системи, включно з прийманням надійності та введенням в експлуатацію. Вирішуються також завдання дослідної експлуатації, навчання персоналу, детального опрацювання та реалізації методів аналітичного та експериментального доказу надійності системи.

На десятій стадії здійснюється приймання системи в постійну експлуатацію. При цьому вирішуються завдання обґрунтування можливості та доцільності введення системи в постійну експлуатацію, інтеграції результатів доказу надійності та функціональної безпеки. Вирішуються завдання оцінювання досягнутих рівнів структурної та функціональної надійності елементів ІС. Потім здійснюється оцінка надійності всієї ІС. Це дасть замовнику ІС незалежне підтвердження, що всі визначені раніше ризики завдяки застосуванню функцій і регуляторів надійності зменшено до прийнятного рівня. У протоколі випробувань перераховуються всі виявлені вразливості, а в матеріалах аудиту описуються рекомендовані дії щодо їх усунення. Формується план усунення недоліків, виявлених під час оцінювання надійності системи. Визначається допустимість залишкових ризиків після реалізації плану усунення виявлених недоліків. Потім реалізується цей план, оцінюється ефективність його виконання. Після чого здійснюється введення ІС у постійну експлуатацію.

Одинадцята стадія - стадія експлуатації та технічного обслуговування ІС охоплює переважну частину життєвого циклу і за вартістю перебуває на рівні (60-70)% від загальної вартості життєвого циклу системи. Вона охоплює завдання поточного технічного обслуговування і матеріально-технічного забезпечення, моніторингу надійності системи, поточного навчання персоналу, ведення протоколу загроз. На цій стадії вирішуються завдання оптимізації технічного обслуговування з метою зниження вартості життєвого циклу системи за умови збереження її надійності на прийнятному рівні.

На дванадцятій стадії проводять оцінювання стану надійності системи на основі збирання, опрацювання та аналізу поточних даних з експлуатації ІС, оцінювання досягнутих рівнів технічної ефективності та надійності ІС з урахуванням попередніх матеріалів доказу надійності системи. Розглядаються й

аналізуються всі пропоновані або зроблені зміни в ІС, включно зі змінами політик, правил і процедур. На цій стадії, так само як і на інших стадіях життєвого циклу, проводиться коригування документа "Доказ надійності системи". На цій стадії також визначаються необхідні роботи з модифікації системи.

На тринадцятій стадії здійснюється модифікація і переоснащення системи. Це природний процес для експлуатації інформаційних систем, спричинений безперервним удосконаленням і розвитком інформаційних технологій, створенням більш ефективних технологій, безперервним удосконаленням обчислювальних і телекомунікаційних засобів, активним розвитком математичного забезпечення. У зв'язку з модифікаціями ІС виникає потреба в оцінках їхнього впливу на функціональну надійність, безвідмовність та інші характеристики RAMS. Створення нових версій програмного забезпечення, зміна інформаційних технологій, зміни в обладнанні зумовлюють необхідність повтору життєвого циклу, починаючи з коригування концепції і завершуючи етапами експлуатації, контролю структурної та функціональної надійності, а потім знову модифікації системи тощо.

Життєвий цикл інформаційної системи завершується стадією (чотирнадцятою згідно з розглянутою нумерацією) планування і виведення з експлуатації, архівуванням, ліквідацією або переміщенням даних на інші системи, а також можливою утилізацією системи. Ці процедури можуть спричинити збитки як економічного, так і соціального характеру. Звідси необхідність в оцінках загроз, ризиків і безпеки виведення системи з експлуатації.

Слід зазначити, що не тільки на перших чотирьох, а й на всіх наступних стадіях життєвого циклу здійснюється документування отриманих результатів і верифікація розроблених матеріалів, проєктів, технологій, а також оцінювання використаних методів, способів, засобів і оцінювання кваліфікації співробітників.

1.2 Програма забезпечення та доказ надійності інформаційної системи

1.2.1 Політика та програма забезпечення надійності

Для створення надійної інформаційної системи слід розробити та узгодити Політику та Програму забезпечення надійності системи.

Програма забезпечення надійності (ПЗН) - це задокументований перелік запланованих за часом заходів, ресурсів і подій, спрямованих на впровадження організаційної структури, розподілу відповідальності за їх розподіл і виконання для того, щоб забезпечити необхідний рівень надійності ІС. Програма може містити такі розділи:

- Політика забезпечення надійності;
- Програма робіт.

У Політиці мають бути відображені цілі та завдання забезпечення структурної та функціональної надійності ІС, основні принципи та підходи до вирішення цього завдання; принципи та підходи до управління ризиками тощо.

Політика забезпечення надійності - це офіційно затверджений керівництвом організації-замовника документ, у якому відображено загальні наміри та напрями діяльності організації в частині забезпечення структурної та функціональної надійності ІС. У цьому документі мають зазначатися принципи і методи забезпечення надійності, які використовуються організацією (наприклад, принцип мінімізації витрат на забезпечення безпомилковості вихідних результатів, механізм перерозподілу ризику, метод стимулювання персоналу, механізми економічної відповідальності підприємства та ін.). У документі "Політика забезпечення надійності" слід навести вимоги до комплексу організаційно-технічних заходів щодо забезпечення надійності функціонування ІС, розподілу відповідальності та повноважень, порядок взаємодії між внутрішніми структурними підрозділами організації; порядок взаємодії зі сторонніми організаціями. Повинен бути також викладений підхід до управління ризиками, прийнятий організацією, прийнятий

принцип допустимості ризику, перелік потенційних небезпек (загроз) для ІС, шляхи та способи визначення пріоритетів під час реалізації заходів з опрацювання ризику (зокрема, захисних заходів), оцінка залишкового ризику.

У Програмі мають міститися методи, способи і технічні рішення щодо забезпечення безвідмовності, готовності, ремонтопридатності, функціональної безпеки складових елементів і системи загалом та безпомилковості виконання інформаційних процесів у розроблюваній ІС (стадії 1-5 життєвого циклу, рис. 1.1), організація робіт із забезпечення надійності ІС на етапах виробництва (стадії 6-9 життєвого циклу, рис. 1.1), а також заходи із забезпечення надійності ІС у процесі її експлуатації (стадії 10-13 життєвого циклу, рис.1.1).

При викладенні заходів щодо забезпечення надійності ІС у Програмі мають бути відображені такі головні положення:

- опис життєвого циклу надійності системи та завдань із забезпечення структурної та функціональної надійності, що мають бути виконані в процесі життєвого циклу ІС;

- результати аналізу та оцінювання надійності ІС протягом життєвого циклу, включно з такими основними позиціями:

- ідентифікація та аналіз небезпек;

- оцінювання ризику та поточне керування ризиком;

- визначення критеріїв допустимості ризику;

- принципи розроблення вимог до структурної та функціональної надійності ІС і докази їх адекватності;

- верифікація складових об'єктів і валідація ІС загалом;

- результати аналізу показників експлуатації та технічного обслуговування ІС;

- специфікація документації з надійності ІС;

- опис взаємозв'язку з іншими об'єктами і системами.

1.3 Доведення надійності інформаційної системи

Доказ надійності (Паспорт надійності) ІС - документоване підтвердження того, що система виконує всі задані вимоги до структурної та функціональної надійності на всіх етапах життєвого циклу. Цей документ являє собою документовану сукупність доказів, які забезпечують переконливі та достовірні аргументи, що система є адекватно надійною для даного застосування в даному навколишньому середовищі. Цей документ має супроводжувати систему на всіх етапах життєвого циклу від початку розроблення до завершення її експлуатації.

Цілі доказу надійності:

- перевірка виконання концепції забезпечення надійності системи;

- перевірка відповідності ІС вимогам надійності;

- перевірка відповідності показників надійності ІС заданим нормам.

Документ "Доказ надійності" акумулює всю сукупність матеріалів доказового характеру і відображає результати робіт з управління якістю, управління надійністю, що проводяться на всіх етапах життєвого циклу ІС. У Доказі надійності в письмовій формі слід обґрунтувати, що ІС відповідає заданим якісним і кількісним вимогам.

Для забезпечення доказів надійності необхідно:

- оформити в явному вигляді набір вимог до системи;

- зробити змістовний доказ (очевидність);

- забезпечити низку аргументів надійності, які пов'язують вимоги до системи з доказами;

- зробити зрозумілими припущення і судження, які лежать в основі аргументів;

- забезпечити різні точки зору й рівні деталізації.

Щоб виконати зазначені вимоги, докази надійності мають бути структуровані таким чином:

Вимоги до властивостей системи або деяких підсистем.

Доказ, який використовується як основа аргументу надійності. Він може містити або факти (наприклад, факти, засновані на встановлених наукових принципах і попередніх дослідженнях), або залежні аргументи, отримані з

аргументів нижчого рівня.

Аргумент, що з'єднує доказ із вимогою, який може бути детермінованим, імовірнісним або якісним.

Умовивід - механізм, який забезпечує правила перетворення, тобто висновок для аргументу.

Докази надійності можуть бути розділені на твердження для різних властивостей різних великих вузлів системи, наприклад: готовність, захист (від зовнішньої атаки), функціональна коректність, ремонтопридатність, забезпечення надійності в разі відмови окремих елементів тощо.

Перелік поданих вище атрибутів (характеристик) є лише прикладом.

Ставлення до надійності може мати й інші атрибути.

Розглянемо процес здійснення доказів надійності. Багато проблем під час реалізації прийнятного доказу надійності є результатом неконструктивної позиції, яка розцінює докази надійності як додатковий аксесуар до системи (часто вироблений після створення системи). На цьому етапі часто виявляється, що "підгонка" доказів надійності, які підтримують систему, є і дорогим, і трудомістким процесом. Рекомендовано інший підхід, де докази надійності беруть до уваги протягом усього проєкту. У цьому підході рекомендується:

- інтегрувати докази надійності інформаційної системи та її складових частин у проект і процес його розроблення.

- застосовувати "порівневі (шаруваті)" докази надійності, тобто високорівневі докази надійності з допоміжними доказами надійності для великих підсистем.

- забезпечувати відстежуваність між рівнями системи та підсистем.

- застосовувати "проект для оцінки", який бере до уваги витрати і складність доказів надійності так само, як і проекту.

Доказ надійності є "живим документом", який розвивається в процесі всього життєвого циклу системи. Оскільки в ньому реєструються аргументи надійності, базова структура має залишатися значною мірою подібною протягом довгого часу, але статус доказу може змінитися. Наприклад, заплановані рівні тестового

покриття замінюються очевидністю результатів випробувань на досягнутому рівні охоплення. На практиці, звісно, докази надійності можуть бути розділені на кілька документів (наприклад, ті, що охоплюють конкретні підсистеми), а також вони можуть посилатися на підтримувальні документи (наприклад, документація проекту, звіти аналізу, звіти про випробування тощо).

Доведення надійності в загальному випадку має містити такі основні розділи:

- призначення, функції та характеристики ІС, зовнішнє обладнання, інтерфейси, режими роботи;

- архітектура системи, підсистем, взаємозв'язок, обмеження проекту.

- опис навколишнього середовища.

- звіт про заходи з управління якістю.

- запланований підхід забезпечення надійності (зокрема, докази, що підтримують, і методи аналізу, наприклад, RAMS).

- звіт про заходи з управління структурною надійністю.

- звіт про заходи з управління функціональною надійністю.

- докази структурної та функціональної надійності складових частин.

- довгострокові вимоги підтримки. Технічне обслуговування системи.

- висновок щодо надійності.

Зміст документа "Доказ надійності ІС" має оновлюватися на всіх етапах життєвого циклу системи за результатами поточних оцінок стану надійності системи.

Висновки, отримані з доказу надійності, повинні дозволяти судити про те, що:

- вимоги, що висуваються до ІС, задано коректно та в повному обсязі;

- вимоги, що висуваються до ІС, у повному обсязі та коректно реалізовано в технічних рішеннях;

- технічні рішення не привносять додаткових негативних властивостей відносно первісних вимог надійності;

- наведені докази обґрунтовано та достовірно.

Викладений вище підхід до доказів надійності інформаційної системи та її

складових частин робить головний акцент на твердженнях щодо поведінки системи (тобто, функціональної поведінки і характеристик системи) і відповідних аргументах для підтримки зазначених тверджень. Ідеї структурування (які використовують твердження, аргументи та докази) досить прості, але вони мають забезпечити переконливі докази надійності, які будуть побудовані та які будуть зрозумілими і простежуваними. Для того, щоб виконати докази надійності, рекомендовано інтегрувати докази надійності в процес здійснення проекту.

Рівнева побудова доказів надійності дає змогу цим доказам розвиватися протягом тривалого часу і допомагає встановлювати вимоги на кожному рівні. Для великих проектів із субпідрядниками, цей "низхідний" підхід до доказів надійності допомагає визначати вимоги до підсистем, і доказ надійності підсистеми може бути зроблений явною договірною вимогою, яку буде поставлено субпідрядником.

1.4 Надмірність і надійність

Якщо надмірність у системі виникла всупереч цілям її розробника, унаслідок незнання або невміння зробити ефективнішу архітектуру інформаційної системи, то таку надмірність можна іменувати як "дика надмірність". Надмірність деяких ресурсів може бути і небезпечною для системи. Надлишкове обладнання, омертвлені фінансові ресурси, роздуті штати підприємств зовсім не йдуть нікому на користь. Однак, є й цікаве відхилення від цього правила - більшість мікрочипів, інтелектуальних пристроїв, починаючи від смартфонів і закінчуючи великими комп'ютерами, мозок людини та багатьох тварин заздалегідь володіють надмірною пам'яттю і "обчислювальними потужностями".

Якщо надмірність стосовно певних функцій з'явилася остільки, оскільки введено надмірність стосовно інших функцій, то таку надмірність слід іменувати як "природну надмірність". Наприклад, у багатоканальній інформаційній системі кількість каналів опрацювання інформації розрахована або на максимальне інформаційне навантаження, або на найімовірніше навантаження. За невисокого інформаційного навантаження вільні канали можуть використовуватися як

природно надлишкові ресурси для контролю стану працездатності зайнятих опрацюванням інформації каналів або для оперативного резервування каналів, що відмовили. Інший приклад. В інформаційних системах реального часу кожен цикл обробки інформації зазвичай має постійну і рівну тривалість, розраховану на максимальне інформаційне навантаження. У нормальних умовах функціонування, як правило, частина циклу обробки інформації вільна. Це означає наявність певного часового ресурсу, який може використовуватися для забезпечення надійності системи.

Часто надмірність вводиться розробниками навмисне для забезпечення надійності та функціональної безпеки, для забезпечення подальших поліпшень системи, для універсалізації її функцій, розширення можливостей її адаптації, застосування з різними цілями тощо. У таких випадках будемо користуватися терміном "штучна надмірність".

У процесі розвитку системи завжди відбувається зменшення надмірності як за рахунок зменшення непотрібних ресурсів, так і за рахунок перетворення надмірності на ресурси, які необхідно використовувати в системі для забезпечення її нормального функціонування. Еволюція надмірності системи може бути описана такими етапами:

1. Система з високою вихідною надмірністю.

2. Зниження надмірності системи - перетворення надмірності на корисні ресурси, що забезпечують:

- підвищення якості, надійності, безпеки та ефективності системи;

- збільшення кількості та якості виконання корисних функцій системи;

- зниження шкідливих чинників;

- зниження матеріаломісткості системи;

- зниження трудомісткості виготовлення, маркетингу та експлуатації системи;

- підвищення ефективності виготовлення системи;

- підвищення ефективності експлуатації системи.

3. Виникнення протиріч між підвищенням якості та зниженням витрат.

4. Розв'язання суперечностей між якістю і витратами шляхом створення систем нового покоління (що мають, як правило, "запас надмірності", який потім перетворюватиметься на ресурси, що використовуються).

З позиції забезпечення надійності інформаційних систем розрізняють такі основні види надмірності:

1. Структурна надмірність.
2. Тимчасова надмірність.
3. Інформаційна надмірність.
4. Функціональна надмірність.
5. Алгоритмічна надмірність.
6. Програмна надмірність.
7. Багаторівнева надмірність.

Структурна надмірність (Structure Redundancy або Hardware Redundancy) - наявність в об'єкта надлишкових елементів, вузлів, пристроїв, за допомогою яких можна вчасно замінити основні вузли або пристрої, що відмовили, для запобігання виходу з ладу всього виробу, системи. Надлишкові вузли або пристрої можуть працювати паралельно з основними, або можуть бути під'єднані й використовуватися в режимі очікування, або застосовуватися за допомогою іншого, наприклад, гібридного способу. Слід зазначити, що можливість застосування структурної надмірності перебуває в прямій залежності від засобів виявлення елемента системи, що відмовив. Причому, процедура заміни елемента, що "відмовив", найбезпосереднішим чином пов'язана зі структурою системи.

Ще один із принципів, необхідний під час забезпечення надійності за допомогою структурної надмірності - модульність (роздільність).

Тимчасова надмірність (Time Redundancy) полягає у використанні деякої частини продуктивності комп'ютера для контролю за виконанням програм і відновлення (рестарту) інформаційного процесу.

Розрізняють природну та штучну тимчасові надмірності. Розглянемо приклад природної часової надмірності. В інформаційній системі реального часу опрацювання інформації та вироблення керівних команд здійснюються в межах

чергового циклу управління. У зв'язку з високою швидкістю обробки інформації в сучасних комп'ютерах у межах багатьох циклів обробки залишаються невикористані відрізки часу, протягом яких комп'ютери вільні від виконання передбачених функціональних завдань. Ці вільні відрізки часу можна використовувати, зокрема, для виконання операцій контролю. Їх можна використовувати для повторного виконання окремих операцій опрацювання інформації або управління та порівняння отриманих результатів. За значної величини вільних від виконання функціональних завдань відрізків часу створюються умови для реалізації легкого рестарту інформаційного процесу.

Штучна часова надмірність передбачає введення резерву часу для повторного виконання передбачених функціональних завдань і порівняння отриманих результатів. Під час проектування комплексу програм має передбачатися запас продуктивності, який використовуватиметься для контролю та підвищення надійності функціонування. Значення часової надмірності залежить від вимог до надійності функціонування системи і становить від 5-10% продуктивності комп'ютера в системі до трьох- і чотириразового дублювання продуктивності в мажоритарних обчислювальних комплексах. Під час функціонування комплексів програм у реальному масштабі часу резерв часу для контролю і відновлення обчислювального процесу та інформації не встановлюється заздалегідь. Для діагностики спотворень і операцій відновлення потрібен у загальному випадку невеликий інтервал часу, який виділяється за рахунок резерву або скорочення часу розв'язання функціональних завдань.

Інформаційна надмірність (Information Redundancy) - деяке повторення інформації в тій чи іншій формі, що дає змогу відновлювати вихідні дані в разі будь-яких порушень у роботі системи. Інформаційна надмірність - це процес дублювання частини даних інформаційної системи для забезпечення надійності та контролю даних. Надмірність використовується для забезпечення достовірності вхідних і проміжних даних, які найбільшою мірою впливають на нормальне функціонування комплексу програм, а також даних, що потребують значного часу відновлення. Для відновлення таких даних потрібне їх тривале накопичення та

обробка. Інформаційна надмірність сприяє не тільки виявленню спотворення даних, а й усуненню помилок. Дані захищають дво- і триразовим дублюванням з відповідною дисципліною контролю збереження і періодичним оновленням. Для менш важливих даних інформаційна надмірність використовується у вигляді завадозахисних кодів, що дають змогу тільки виявляти спотворення. Спотворені дані виключаються з обробки, і відбувається їхнє природне оновлення в процесі подальшого функціонування. Багато даних інформаційної системи не захищаються внаслідок їхнього частого оновлення або слабкого впливу можливих викривлень на розв'язання основних функціональних завдань. Інформаційна надмірність - це збільшення кількості інформації з метою підвищення надійності та достовірності передавання або зберігання даних.

Інформаційна надмірність зазвичай реалізується поетапно шляхом застосування двох типів методів:

1. Методи виявлення помилок.

2. Методи корекції помилок.

Методи виявлення помилок дають змогу тільки зафіксувати факт наявності помилки. Зазвичай, вони застосовуються в сукупності з програмами відновлення даних. Наприклад, якщо помилку виявлено під час пересилання даних, то запитується повторне відправлення. А якщо помилку виявлено в блоці збереженої інформації, то проводиться її відновлення з резервної копії. Наприклад, до методів виявлення помилок належить перевірка на парність. У цьому разі блоки інформації під час надсилання або перед збереженням доповнюються до парної кількості одиниць (у бінарному коді), а під час приймання або зчитування перевіряється, чи є кількість одиниць парною. Також можна доповнювати блоки інформації до непарної кількості одиниць. Однак, цей метод може фіксувати лише непарну кількість помилок. Якщо одночасно відбулася парна кількість збоїв, то така помилка не буде зафіксована, оскільки кількість одиниць залишиться парною (або непарною, якщо використовується доповнення до непарності).

Більш надійним є метод контрольного підсумовування. Найпростіша реалізація методу - це додавання байтів масиву даних з перенесенням розряду

переповнення. Є більш складні та надійні методи контрольного підсумовування, наприклад, контрольне підсумовування CRC, що базується на методах циклічного кодування. Фактично CRC є залишком від ділення многочлена, що відповідає вихідним даним, на породжений многочлен фіксованої довжини. Залежно від довжини породженого многочлена, розрізняють алгоритми контрольного підсумовування CRC8, CRC16, CRC32 тощо.

До методів корекції помилок належить таке кодування інформації, яке дає змогу не тільки виявити помилку, а й виправити її. Число помилок, яке можна виправити, обмежене і залежить від виду застосовуваного кодування. Природно, коди корекції можна використовувати і для виявлення помилок, причому вони можуть виявити одночасних помилок більше, ніж у змозі виправити. Наприклад, до таких кодів належать коди Хеммінга і згорткові коди.

Іншим аспектом поняття інформаційна надмірність є сформоване в теорії інформації уявлення про перевищення кількості інформації, використовуваної для передавання або зберігання повідомлення, над його інформаційною ентропією. Тут під інформаційною ентропією розуміється міра невизначеності інформації, невизначеність появи будь-якого символу. Це уявлення з'явилося в теорії електрозв'язку. Для адміністратора баз даних інформаційну ентропію слід інтерпретувати дещо по-іншому: інформаційна ентропія - це також міра невизначеності інформації, але, яка інформаційна невизначеність може виникнути в базі даних? Будь-яка база даних призначена для зберігання інформації. І під час проектування бази даних слід врахувати те, що якась інформація може повторюватися кілька разів. А кожен повторюваний запис - це зайняте місце на диску. Тобто перевищення кількості інформації необхідної для зберігання даних.

Звичайно, можна сказати, що зараз, з появою терабайтних накопичувачів, відпала потреба заощаджувати місце на диску. Але надмірність інформації веде не тільки до того, що потрібне збільшення об'єму накопичувачів, а й призводить до аномалій у базі даних. Аномалії в базі даних - це проблеми пов'язані з опрацюванням інформації, а точніше, з видаленням даних із бази даних, з модифікацією даних у таблиці бази даних, а також проблеми з додаванням даних

до бази даних, що містить надлишкові дані.

Зупинимося на надмірності джерела повідомлень. З ентропійних оцінок джерел повідомлень зрозуміло, що їхня надмірність залежить від статичних характеристик самих повідомлень. Ентропія максимальна за рівномірної появи літер на будь-якому місці повідомлення. Для характеристики джерела повідомлень з різним алфавітом представляє інтерес порівняння фактичної ентропії джерела $H(X)$ з максимально можливою $H_{max} = logM$, де M - кількість різних букв в алфавіті. У цьому сенсі введено поняття надмірності джерела повідомлень або надмірності алфавіту як $R=(H_{max}-H(X))/H_{max}$, Надмірність джерела R показує, наскільки добре використовуються літери в даному джерелі. Що менша надмірність джерела інформації R, то більша кількість інформації виробляється джерелом на одну букву. Однак, не завжди необхідно прагнути до $R = 0$. З підвищенням надмірності підвищується завадостійкість (надійність) джерела. З'ясування кількості надмірності важливе тому, що ми повинні вводити її розумно, щоб отримати максимальний ефект завадозахищеності, а не покладатися на стихію. Наприклад, надмірність будь-якої мови виявляється порядку (50-70)%, тобто, якби всі літери мали однакову ймовірність використання і можна було б використовувати будь-які комбінації букв, то середню довжину слова можна було б значно зменшити. Однак розбиратися в цьому записі було б значно важче, особливо за наявності помилок (лектора чи студента).

Сучасні системи зв'язку побудовані без урахування обмежень, які існують у мові, а тому є недостатньо ефективними, бо вони пристосовані для передавання рівноймовірних літер алфавіту, що можуть слідувати одна за одною в будь-яких комбінаціях.

Функціональна надмірність (Functional Redundancy). Цей вид надмірності враховує можливість проведення однієї й тієї самої роботи різними засобами. Наприклад, до складу операційної системи (ОС) може входити кілька типів моніторів (модулів супервізора, які керують тим чи іншим видом ресурсу), різні засоби організації комунікацій між обчислювальними процесами. Наявність кількох типів моніторів, кількох систем керування файлами дає змогу

користувачам швидко і найбільш адекватно адаптувати ОС до певної конфігурації обчислювальної системи, забезпечувати максимально ефективне завантаження технічних засобів під час розв'язування конкретного класу завдань, отримувати максимальну продуктивність під час розв'язування заданого класу завдань.

Функціональна надмірність може бути реалізована за допомогою різних засобів, що відрізняються різними рівнями ефективності в проведенні роботи. Так, інформаційне управління безпекою руху локомотива на ділянці колії може здійснюватися за допомогою спеціалізованого локомотивного пристрою безпеки або шляхом передачі команд машиністу за допомогою мобільних засобів зв'язку. У першому випадку досягається висока гарантія безпеки руху. У другому випадку можливість виникнення небезпечної ситуації суттєво вища. Водночас у разі відмови пристрою безпеки зберігається (хоча і зі зниженою ефективністю) можливість здійснювати інформаційне управління безпекою руху локомотива (хоча і зі зниженою ефективністю) у разі відмови пристрою безпеки.

Було розглянуто приклади штучної функціональної надмірності. Однак у самій природі більшості реалізованих у цифровій техніці логічних функцій присутня природна функціональна надмірність. Наприклад, у базових логічних елементах І та АБО, таблиці істинності яких показано на рис. 1.2 а та рис. 1.2 б відповідно, міститься логічна надмірність за вхідними сигналами x_1 і x_2. Так, у разі елемента І за вхідного набору 00 трансформація сигналу x_1 або x_2 внаслідок перешкоди не призведе до помилки у вихідному сигналі у. За вхідного набору 01 або 10 трансформація сигналу x_2 у першому випадку або сигналу x_1 у другому випадку також не призведуть до помилки у вихідному сигналі у. Це природно, оскільки елемент І має функціональну надмірність за вхідними сигналами $x_1=0$ або $x_2=0$. Своєю чергою, елемент АБО має надмірність за вхідними сигналами $x_1 = 1$ або $x_2 = 1$

Елемент "І"

x_1	x_2	y
0	0	**0**
0	**1**	**0**
1	0	**0**
1	1	**1**

Елемент "АБО"

x_1	x_2	y
0	0	**0**
0	1	**1**
1	**0**	**1**
1	1	**1**

Рисунок 1.2 - Таблиці істинності для логічних функцій "І", "АБО"

Функціональна надмірність складових логічних елементів розвивається у функціональну надмірність цифрових пристроїв, мікрооперацій, операцій. Вона може бути реалізована на рівні макрооперацій і алгоритмів. За глибокого розвитку цієї теми функціональна надмірність може слугувати ефективним інструментом для забезпечення функціональної надійності інформаційних систем.

Алгоритмічна надмірність (Algorithmic Redundancy). Цей вид надмірності означає наявність в алгоритмі додаткових правил і приписів понад мінімально необхідних. Мірою чутливості алгоритму може бути похибка обчислень. Результати обчислень спотворюються внаслідок похибок:

а) вихідних даних, трансформованих під час обчислень;

б) заокруглення;

в) похибок методу;

г) похибок, зумовлених відмовами, збоями та помилками в програмі.

Для усунення спотворень у результатах обчислень у кожному конкретному випадку можна застосовувати спеціальні алгоритми, нечутливі до різного роду порушень інформаційного процесу. Наприклад, під час розв'язання задачі оптимального керування за допомогою фільтра Калмана - Б'юсі можливе внаслідок збійної помилки спотворення екстрапольованих параметрів. Результатом цього буде в кращому разі спотворення результатів оцінювання параметрів траєкторії керування, а можливо і втрата траєкторії керування. Для виключення цього та інших негативних і небезпечних явищ, які можуть виникнути в процесі керування об'єктом унаслідок збійних помилок, застосовують таку алгоритмічну надмірність.

Поряд із модулем алгоритму, що реалізує оптимальну фільтрацію, вводять модуль, у якому реалізують згладжування параметрів траєкторії керування методом найменших квадратів на основі ковзної фіксованої вибірки. Ця вибірка містить результати згладжування (оцінювання) параметрів траєкторії в кількох останніх циклах обробки, отриманих за допомогою алгоритму оптимальної фільтрації. У разі виявленого спотворення в поточному циклі опрацювання згладжених або екстрапольованих параметрів, або нових даних здійснюється автоматичний перехід на алгоритм керування на основі методу найменших квадратів. За збереженими у фіксованій вибірці параметрами проводиться повторна екстраполяція параметрів траєкторії, що дає змогу в інший спосіб усунути виявлені спотворення параметрів траєкторії керування. Для виявлення викривлень параметрів можливі різні способи, наприклад, алгоритмічний контроль шляхом стробування параметрів.

Цей приклад цікавий тим, що описаний спосіб забезпечення стійкості до помилок може застосовуватися і в інших додатках. Наприклад, під час опрацювання інформації про рухомий об'єкт, у метрології під час опрацювання результатів вимірювань тощо. Він характеризує штучну алгоритмічну надмірність. В інформаційних системах досить часто зустрічається природна алгоритмічна надмірність. Таку властивість мають алгоритми опрацювання потоків даних, алгоритми розв'язання систем диференціальних (алгебраїчних) рівнянь, алгоритми швидкого перетворення Фур'є та багато інших. Під час реалізації цих алгоритмів поодинокі помилки нейтралізуються безпомилковими результатами паралельного оброблення інших вхідних даних, які в сукупності з прийнятною похибкою забезпечують безпомилковий вихідний результат.

Програмна надмірність (Program Redundancy) - наявність у програмі процедур (операцій) понад мінімально необхідних. Можливий варіант програмної надмірності полягає в тому, щоб періодично запускати еталонні задачі та порівнювати отриманий результат з передбачуваним. Існують інші способи забезпечення надійності програм на основі їхньої надмірності. Істотне підвищення функціональної надійності програм можливе на основі методів багатоверсійного програмування. Розглянемо ці методи. Але насамперед згадаємо специфічні

особливості програм стосовно апаратури інформаційних систем. Ці особливості зводяться до наступного:

- програма фізично не старіє і не зношується. Можливе тільки моральне старіння програми;

- програма не потребує ремонту і профілактичного обслуговування;

- програмні помилки є результатом створення і коригування ПЗ. Вони можуть не проявлятися, якщо вхідні дані не ініціюють виконання тих ділянок програм, у яких містяться помилки. На відміну від технічних засобів деякі вхідні потоки даних можуть призвести до частих помилок, у той час як інші потоки можуть взагалі не призвести до помилок;

- кожна програма є унікальним продуктом. Цей факт можна пояснити таким чином. По-перше, будь-які дві копії програми абсолютно ідентичні. По-друге, програма ніколи не дає погіршення характеристик без зовнішнього втручання. По-третє, процес налагодження програми ніколи не повторюється;

- будь-яке коригування програми (навіть якщо це стосується лише усунення виявленої помилки) призводить до нової версії програми, оскільки обов'язково призводить до зміни хоча б одного елемента об'єктного коду програми. Ця важлива обставина виключає можливість використання раніше наявних статистичних даних і принципово не дає змоги здійснювати статистичну оцінку надійності програм;

- під час виправлення виявленої помилки програміст може внести нові помилки, що можуть згодом проявитися за певного набору вхідних даних, що відрізняється від наслідків ремонту технічних засобів.

З перерахованих особливостей програм стосовно апаратури випливає, що традиційне резервування стосовно програм не має сенсу. Дійсно, у разі резервування програми на одному обчислювальному модулі інформаційної системи та порівняння отриманих результатів помилка, що проявилася в програмі, за певного набору вхідних даних призведе до однакових помилок у вихідних результатах обох абсолютно аналогічних програм і не буде помічена під час порівняння результатів. Якщо застосувати іншу тактику: реалізувати тільки одну програму і контролювати її за допомогою еталонної задачі, а іншу абсолютно

аналогічну їй запускати тільки в разі розбіжності отриманого результату з передбачуваним, то за однакових початкових даних реалізація другої програми призведе до такої самої розбіжності отриманого результату з передбачуваним, як це мало місце під час реалізації першої програми. Якщо ще змінити тактику і запускати другу програму за наступного набору вхідних даних, то можлива відсутність помилки і збіг отриманого результату з передбачуваним. Однак, такий самий результат міг мати місце і під час запуску першої програми за нового набору вхідних даних. Наведені міркування переконують нас у тому, що традиційне структурне резервування не може бути поширене на програмні засоби. Тут слід шукати інший шлях.

Для забезпечення надійності програм запропоновано принцип багатоверсійного програмування. Мета: виявити та замаскувати залишкові помилки проекту програмного забезпечення протягом виконання програм, щоб запобігти відмовам системи, та продовжувати роботу з високою надійністю. У багатоверсійному програмуванні дана специфікація програми реалізується по-різному N разів. Ті ж самі значення вхідних даних задаються N версіям, і результати, вироблені N версіями, порівнюються. Якщо результат вважається достовірним, то він передається обчислювальному модулю як вихідні дані. N версій можуть виконуватися паралельно на окремих модулях. Альтернативно всі версії можуть виконуватися на одному обчислювальному модулі. Вихідні результати піддаються внутрішньому голосуванню. Різні стратегії голосування можуть бути використані на N версіях залежно від вимог застосування. Для критично важливих інформаційних систем необхідна повна згода всіх N версій. Для інших інформаційних систем може бути використана стратегія голосування шляхом простої більшості. Для випадків, де немає колективної згоди, можуть бути використані ймовірнісні підходи, щоб максимізувати шанс вибору правильного значення, наприклад, взявши середнє значення, тимчасово заморозивши вихідні дані до повернення згоди, тощо.

У разі дуального програмування (якщо розробляються дві версії програми) у разі виявлення розбіжності в результатах, необхідно визначити за додатковими

критеріями, який результат правильний і відкинути інший результат. У разі N-версійного програмування правильний результат визначається за мажоритарною ознакою, тобто обирають той результат, який спостерігається в більшості варіантів програми.

Багаторівнева надмірність - це комплексне застосування декількох видів надмірності. Так, можливе поєднання алгоритмічної, програмної, функціональної, і навіть часової надмірності. Широко застосовується поєднання структурної та інформаційної надмірності, поєднання структурної та програмної надмірності. Наочним прикладом введення багаторівневої надмірності в систему, для досягнення відмовостійкості, може послужити система контролю і управління авіалайнера Airbus 320 (fly-by-wire flight control system). У процесі функціонування системи управління, і забезпечення взаємозв'язків між різними компонентами та контролю за останніми, в Airbus 320 задіяно 5 різних незалежних комп'ютерів. Система управління авіалайнером будувалася з розрахунку, що виявлення помилок має здійснюватися як в апаратній, так і в програмній частині системи. З цієї причини, в процесі управління польотом, додатково задіяно два типи програмного забезпечення, від двох незалежних розробників.

РОЗДІЛ 2. ВІДМОВОСТІЙКІСТЬ КОМП'ЮТЕРНИХ СИСТЕМ

Під відмовостійкістю технічних систем зазвичай мають на увазі їхню здатність виконувати передбачені функції в реальних умовах експлуатації за наявності внутрішніх впливів (відмов та/або збоїв складових технічних засобів, програмних помилок). Відмовостійкість - це властивість технічної системи зберігати свою працездатність після відмови одного або декількох складових компонентів. Тут ідеться про апаратну відмовостійкість. Програмна відмовостійкість має місце тоді, коли система має можливість зберігати працездатність після виникнення однієї або декількох програмних помилок.

Відмовостійкість пристрою або системи ґрунтується на наявності надмірності. Тут доречно розрізняти три типи надмірності: штучну, природну, гібридну. Штучна надмірність має місце тоді, коли в об'єкт (пристрій або в систему) для забезпечення відмовостійкості вводять додаткові ресурси понад мінімально необхідних. При цьому додаткові ресурси витрачаються на парирування відмов компонентів. Це забезпечує функціонування об'єкта з постійним максимальним рівнем ефективності. Після того, як усі додаткові ресурси вичерпано, об'єкт уже не є відмовостійким. Цей тип відмовостійкості найбільш витратний, оскільки пов'язаний з підтримкою ефективності об'єкта на максимальному рівні.

Найпростіший приклад штучної надмірності - структурне резервування.

Природна надмірність характерна для багатофункціональних або багатоканальних систем. Об'єкт із цим видом надмірності має властивість відмовостійкості на основі поступової деградації. Слово деградація запозичене з польської мови (своєю чергою, degradacja - від латинського degradatio «розжалування, поступове пониження»). Деградація - поступове погіршення, втрата цінних властивостей і якостей об'єкта. Це поступове скорочення можливостей або поступовий вихід із роботи. Інакше кажучи, це плавне зниження ефективності об'єкта. Наприклад, деградація у телекомунікаціях - втрата якості сигналу. Деградація в системах масового обслуговування - це зниження

ефективності обслуговування заявок (пропускної здатності обслуговування заявок) унаслідок відмов каналів обслуговування Деградація в багатофункціональних системах - це зниження ефективності внаслідок невиконання окремих функцій через відмови компонент об'єкта. Для будь-яких типів об'єктів забезпечення відмовостійкості на основі поступової деградації виправдане тоді й тільки тоді, коли ефективність об'єкта знижується до рівня, що не є меншим за допустимий. В іншому разі має місце руйнування об'єкта, тобто продовження його роботи з неприйнятним рівнем ефективності, що постійно знижується. Цей тип забезпечення відмовостійкості найменш витратний, однак, часто не застосовується через наявні для об'єкта вимоги щодо збереження ефективності.

Найбільш раціональний шлях забезпечення відмовостійкості заснований на застосуванні гібридної надмірності. У цьому випадку можна мінімізувати кількість додаткових ресурсів і разом із цим підвищити поріг допустимого рівня зниження ефективності об'єкта.

Реалізація відмовостійкості в інформаційних системах може здійснюватися на основі забезпечення їхньої спостережливості та керованості.

2.1 Спостережливість інформаційних систем

Під спостережливістю ІС будемо мати на увазі її здатність своєчасно виявляти порушення інформаційних процесів, спричинених відмовами і збоями технічних засобів, помилками в програмах, помилками операторів і у вхідній інформації.

Спостережливість інформаційних систем досягається в результаті виконання етапів виявлення факту несправності, локалізації несправності, класифікації несправності і, потім, виявлення її місцезнаходження.

1. Виявлення факту несправності може здійснюватися як у робочому, так і в неробочому стані. Виявлення в робочому стані відбувається в інформаційній системі (ІС) одночасно з виконанням функціональних алгоритмів і забезпечує здатність виявлення несправностей у режимі реального часу. Під час виявлення в

неробочому стані пристрій проходить тестування і, природно, пристрій нездатний виконувати корисну роботу. Отже, виявлення в неробочому стані забезпечує сигналізацію про несправність пристрою або програми ІС перед виконанням завдання і в початковий момент його виконання, але не протягом усього періоду завдання.

Під час виявлення несправності мається на увазі, що зареєстровано момент її виникнення. В інформаційних системах застосовуються різні апаратні та програмні засоби виявлення факту несправності. Забезпечення якості цих засобів - багатокритеріальне завдання. Воно формулюється таким чином. Досягнення мінімального часу виявлення факту несправності за максимальних рівнях достовірності та повноти контролю в умовах заданих обмежень за обсягом обладнання, габаритами, вагою і, особливо, за додатковою кількістю апаратних відмов і програмних помилок, привнесених в ІС засобами контролю. Це завдання поки що не вирішено, не розроблено також науково обґрунтованої технічної політики в галузі спостережливості інформаційних систем. Разом з тим, помітні певні тенденції і в застосуванні відомих апаратних і програмних способів виявлення фактів несправностей в інформаційних системах. Серед апаратних способів найбільш широкого поширення набули:

Коди, що виявляють помилки (коди з перевіркою на парність у пристроях керування, оброблення, зберігання та передавання інформації, коди Хеммінга, циклічні та ітеративні коди для контролю зберігання і передавання інформації, рівноважні коди для контролю керуючих автоматів);

- дублювання технічних засобів з метою забезпечення їхньої паралельної роботи та порівняння отриманих результатів;

- самотестування мікропроцесорів на основі апаратно-мікропрограмних засобів.

Апаратні способи контролю в багатьох випадках не забезпечують виявлення великого різноманіття помилок, що виникають під час роботи інформаційних систем. Основні з них:

- зациклення і зупинки виконання програм;

- самоблокування (клінчі) виконання програм в ІС;

- перевантаження ІС за пропускною здатністю;

- порушення послідовностей виклику підпрограм;

- помилки взаємного переривання програм;

- спотворення і втрати накопиченої інформації про стан зовнішніх абонентів; спотворення і втрати зовнішніх абонентів.

З метою оперативного виявлення факту подібних помилок у робочому режимі ІС застосовуються програмні методи логічного контролю та охоронного таймера. Ці методи не руйнують інформацію, накопичену в процесі функціонування зовнішніх абонентів. Вони допускають перевірку в межах невеликих квантів часу. На відміну від цих методів контролю традиційні методи подвійного або потрійного рахунку вимагають великих витрат продуктивності та пам'яті ІС. Тому методи повторного рахунку практично не застосовуються для оперативного виявлення фактів помилок в ІС.

Метод охоронних таймерів ґрунтується на програмній реалізації таймерів спільно з лічильниками відносного або поточного часу. Розглянемо два характерні способи реалізації цього методу: без переривання програм і з перериванням виконуваних програм.

Перший спосіб застосовується для контролю програм побудованих за модульним принципом. У цьому випадку перед увімкненням і-го програмного модуля на лічильнику відносного часу встановлюється гранично допустимий час його реалізації, розрахований на виконання найдовшого маршруту модуля. У процесі рахунку показання лічильника рівномірно убуває, що забезпечується подачею на цей лічильник сигналів точного часу. При досягненні нульового часу на лічильнику виробляється сигнал помилки. Якщо ж під час повної реалізації і-го програмного модуля охоронний таймер не виробив сигнал помилки, то на лічильнику виробляється нова гранична тривалість реалізації, розрахована на виконання найдовшого маршруту (i+1)-го програмного модуля.

Другий спосіб полягає в порівнянні поточного часу таймера з показанням допустимого часу закінчення роботи підпрограми, або допустимого часу

очікування заявки в черзі, або тривалості запізнювання ввімкнення періодичних програм тощо.

Загалом метод охоронних таймерів ефективний для вирішення завдань виявлення фактів зациклення, зупинок і самоблокувань виконання програм, а також перевантаження ІС за пропускною здатністю.

Для виявлення порушень послідовностей виклику програм крім методу охоронних таймерів застосовується також контроль ключових кодів, під час якого формується таблиця ключових кодів і кількості вмикань кожної програми.

Логічний контроль ґрунтується на надмірності вихідної, проміжної та результуючої інформації. Він містить у собі низку способів:

- перевірки на логічні невідповідності;

- перевірки за граничними значеннями обчислюваних параметрів. Вони полягають у визначенні низки умов, які характеризуються граничними значеннями контрольованого параметра. Наприклад, значення вихідних даних, проміжних і вихідних результатів не можуть виходити за межі розрядної сітки обчислювального модуля (ОМ), в іншому випадку фіксується помилка; або, обчислюване значення ймовірності p будь-якої події має перебувати в межах $0 \le p \le 1$. Одним із окремих випадків цього способу слід вважати спосіб статичного контролю стробуванням, у якому контрольовані параметри порівнюються з допустимими значеннями. Допустимі значення визначаються заздалегідь або розраховуються під час виконання обчислювального процесу. Вони, як правило, менші за граничні. За допомогою цього способу можуть контролюватися не тільки значення змінних, а й швидкості їх застосування протягом певного кванта часу;

- динамічний контроль стробуванням (контроль гладкості зміни змінних з плином часу);

- перевірки контрольних співвідношень. Існує безліч варіантів цього способу, які можна розділити на дві групи. Перша група варіантів ґрунтується на порівнянні з еталоном, тобто порівнянні розрахованих значень змінних або масивів з їх заздалегідь заданими (еталонними значеннями). Порівняння з еталоном може здійснюватися не тільки стосовно значень змінних, а й до результатів операцій, що

виконуються над їхніми адресами. Друга група варіантів способу контрольних співвідношень ґрунтується на обчисленні контрольних функцій. При цьому поряд з обчислюваною функцією за іншою програмою визначається інша функція, що перебуває з основною обчислюваною функцією у контрольних співвідношеннях. Найпростішим прикладом застосування методу контрольних співвідношень є обчислення функцій sin x і cos x за окремими програмами. Контрольним співвідношенням у цьому випадку буде $sin^2x + cos^2x = 1$.

Контроль із використанням надлишкових змінних. Суть способу полягає у введенні надлишкового перетворення в вихідний алгоритм, а також у введенні надлишкових змінних, які задовольняють певним контрольним умовам.

Способи логічного контролю застосовуються в керуючій ІС для виявлення фактів спотворень і втрат накопиченої інформації про стан зовнішніх абонентів, для оперативного контролю стану і зміни даних у пам'яті інформаційної системи, для виявлення фактів помилок, що виникають під час виконання функціональних програм.

2. Локалізація несправності полягає в обмеженні її поширення в конкретній підсистемі та запобіганні псуванню інших системних складових. Обмеження сфери впливу несправності має здійснюватися як у просторі, так і в часі. Типові приклади поширення несправностей у просторі: відмови або помилки апаратури або програм ОМ спричиняють втрати або спотворення результатів роботи одних ОМ і спотворюють дані інших ОМ, а зрештою призводять до помилок у результатах розв'язання задач керування; деякі несправності спричиняють спотворення передбачених і розроблених циклів, а також утворення непередбачених, помилкових циклів, що в підсумку може призвести до блокування і припинення розв'язання задач керування. Поширення несправностей у часі зазвичай має місце під час розв'язання завдань, що реалізуються за допомогою ітераційних обчислювальних процесів. У цих задачах можливі невеликі спотворення результатів окремих ітерацій через кілька кроків (ітерацій) посилюються і можуть призвести до зривів виконання завдань.

Локалізація несправностей здійснюється як у поточному стані, так і перед

початком розв'язання задач керування (передстартові перевірки та обмеження).

У робочому стані ІС можливі такі способи локалізації несправностей:

- затримка до видачі результатів виконання функцій на час аналізу величини їх спотворень і можливості поширення. Залежно від результатів аналізу в керуючих ІС можуть бути вжиті оперативні заходи для ліквідації їхніх наслідків, такі як: ігнорування виявленої помилки внаслідок слабкого впливу на процес управління; виключення отриманих результатів з подальшої обробки внаслідок їхньої спотвореності та або труднощів відновлення обчислювального процесу;

- короткочасне припинення розв'язання функціональної завдання до моменту оновлення вихідних даних.

Для попередньої локалізації несправностей (передстартової перевірки обмежень) розв'язуються такі завдання:

- контроль збереження програм. Він забезпечує перевірку відповідності запису програм у пам'яті ІС вихідному еталону. При цьому зазвичай застосовується логічний контроль;

- контроль вихідних даних режиму початкового пуску.

Призначений для перевірки повноти складу і правильності значень вихідних змінних, а також для перевірки правильності часу початку функціонування, синхронності і синфазності показань лічильників реального часу всіх компонент ІС. Перше завдання вирішується за допомогою логічного контролю або шляхом порівняння дубльованих файлів даних.

Друге завдання вирішується за допомогою охоронних таймерів;

- контрольне завдання призначене для перевірки функціонування комплексу програм за фіксованими вихідними даними шляхом порівняння результатів із заздалегідь розрахованими еталонними значеннями;

- контроль обміну даними із зовнішніми абонентами призначений для оцінки готовності інтерфейсних каналів ІС і каналів передавання даних до виконання завдань керування. Реалізується, головним чином, шляхом введення в системи передавання даних вирішального, зворотного зв'язку в поєднанні з блокуванням, переспросом і повторенням повідомлень, а також із кодами, що виявляють

помилки.

За наявності достатнього часу до початку функціонування системи в ІС можуть виконуватися діагностичні тести для локалізації несправностей в апаратурі та програмах ПС.

3. Класифікація несправності призначена, головним чином, для виявлення відмови або збійної помилки складового пристрою системи. Класифікація несправності відіграє важливу роль у виборі стратегії відновлення обчислювального процесу.

Для розв'язання цієї задачі доцільно, по-перше, встановити, що тривалість несправності більша або менша за допустимого часу перерви в роботі системи, і, по-друге, проаналізувати відсутність вихідної інформації або наявність спотвореної інформації на виході пристрою. Збійні помилки мають короткочасний характер, їхня тривалість зазвичай набагато менша за допустимий інтервал перерви у виконанні заданої функції ІС. У тому разі, коли тривалість несправності більша за допустимий час, існують серйозні підстави вважати, що пристрій відмовив. У цьому можна остаточно переконатися, якщо буде встановлено, що на виході пристрою відсутні вихідні результати (таку відмову назвемо важкою). На відміну від важкої можлива так звана візантійська відмова, що означає наявність помилкових результатів на виході пристрою. При цьому мають місце різні невірні результати за одних і тих самих вихідних даних і програм.

Таким чином, якщо факт несправності пристрою виявлено і пристрій, очевидно, видає помилкові результати, то для остаточного ухвалення рішення про відмову або збої достатньо реалізувати механізм контрольної точки, що, зі свого боку, дасть змогу усунути збійну помилку пристрою (якщо несправність такого типу).

Під контрольною точкою мається на увазі деяка точка в обчислювальному процесі, для якої сформовано в локальній або загальній пам'яті ІС початкові умови виконання чергової частини процесу і вхідна інформація для нього, отримана за результатами виконання попередніх частин. Таким чином, обчислювальний процес розділений контрольними точками на складові частини, Початковими умовами є,

наприклад, початкова адреса чергової частини програми та вихідний стан схеми тактування, що задає тривалість виконання цієї частини програми. Повернення до контрольної точки здійснюється за початковими умовами чергової частини програми.

Якщо вміст контрольної точки не зіпсовано, то цей спосіб придатний для класифікації несправностей. Він полягає в повторному рахунку з контрольної точки і контролі стану працездатності пристрою під час повторного рахунку. Якщо в цьому випадку несправність не виявлено, то можна вважати, що збійна помилка, яка мала місце під час першого рахунку, ліквідована. В іншому разі слід фіксувати факт відмови пристрою. Загалом етапи локалізації та класифікації несправностей дають змогу не тільки обмежувати сферу впливу несправностей і визначати тип помилок обчислювальних засобів, а й в окремих випадках усувати збійні помилки.

4. Виявлення місцезнаходження несправностей призначене для виявлення типового елемента заміни, модуля, блоку, програмного модуля або блоку програмних модулів, що відмовили. Необхідна глибина виявлення несправностей залежить від прийнятих принципів забезпечення відмовостійкості кожної конкретної ІС.

Цей етап спостережуваності ІС реалізується, головним чином, за допомогою добре розроблених засобів діагностування їхньої апаратури та програм.

Місцезнаходження і подальше виправлення власних програмних помилок в ІС здебільшого здійснюють в інтервалах часу між розв'язаннями задач керування. З цією метою застосовуються методи структурного та функціонального тестування програм.

Структурні методи тестування програм полягають у виборі набору шляхів у структурі програми, що задовольняє деяким критеріям тестування, наприклад, щодо покриття графа сформованим набором шляхів. Унаслідок обмеження часу очікування початку керування природно застосовувати вибіркове тестування програм в окремих точках простору вихідних даних.

Розроблено способи числового та символічного тестування. За допомогою числового тестування перевіряють правильність числових результатів роботи

програм за спеціально підібраних тестових наборах вхідних змінних. Символічне тестування полягає у виконанні процедур, заснованих на символічних входах і виходах, причому, для різних шляхів програми мають місце різні входи і виходи. Символічне тестування знімає багато обмежень щодо простору вихідних даних, властивих числовому тестуванню.

Функціональні методи тестування програм полягають у безпосередній перевірці відповідності виконуваних програмою функцій поставленим вимогам. При цьому необхідно вирішити питання вибору тестів і оцінки правильності результату проходження кожного тесту. Застосовуються два підходи виборів тестів: за змістовною ознакою і шляхом статистичного моделювання вхідних даних (стохастичний вибір тестів). Перший підхід важко формалізується, зате під час моделювання випадкових вихідних даних можна вибрати такі закони розподілу цих даних, які найповніше відображають зміст виконуваної програми і її зв'язки з іншими програмами та зовнішніми абонентами.

Перевірка правильності результатів функціонального тестування здійснюється за допомогою методів охоронного таймера і логічного контролю.

Загалом проблема оперативного виявлення власних програмних помилок в ІС потребує подальшого опрацювання і, мабуть, на основі інших нетрадиційних підходів.

2.2 Керованість інформаційних систем

Під керованістю ІС будемо мати на увазі її здатність своєчасно адаптуватися до функціональних і структурних порушень, що виникли.

Керованість ІС можна розглядати як реакцію на виявлену відмову. Вона досягається шляхом виконання етапів реконфігурації ІС, повного відновлення інформаційного процесу, відновлення апаратури ІС і, нарешті, реінтеграції.

1. Реконфігурація призначена для автоматичної зміни архітектури ІС з метою забезпечення її працездатного функціонування. Завдання реконфігурації: збереження колишніх можливостей системи в умовах відмов апаратних або

програмних компонентів, або виділення мінімальної пріоритетної множини функцій, необхідної для продовження роботи в допустимих межах скорочення можливостей системи. Перше завдання вирішується за допомогою спеціальних передбачених в архітектурі ІС резервних апаратних і програмних засобів. При цьому ізолюються компоненти системи, що відмовили, і замінюються справними резервними.

Друге завдання розв'язують на основі наявної і модульної ІС природної апаратурної та програмної надмірності. Залежно від типів виконуваних обчислювальних процесів можливі два варіанти використання природної надмірності. Перший варіант - виключення з обробки завдання з нижчим пріоритетом на час відновлення відмови компонента і використання звільнених ресурсів апаратури для заміни компонента, що відмовив. Щодо відмови програмного компонента можлива його заміна іншою версією розв'язання цієї ж задачі, якщо така є. Другий варіант - виключення з архітектури ІС компонента, що відмовив, без заміни. Такий підхід має назву поступової (елегантної) деградації.

Під час реконфігурації ІС виконуються такі операції:

- відшукання справного резервного ОМ із захисного середовища;

- виключення з функціональної структури ОМ, що відмовив;

- включення до складу обчислювальної структури функціональної задачі резервного ОМ;

- завантаження резервного ОМ програмами і даними основного ОМ.

Під час розв'язання низки завдань оброблення інформації та керування допускається поступова деградація модульної керівної ІС у разі відмов складових ОМ. Так, наприклад, під час розв'язання завдання згладжування й екстраполяції параметрів рухомих об'єктів за допомогою алгоритмів фіксованої вибірки кожен одиничний вимір може оброблятися окремим ОМ. У разі відмови цього модуля значення відповідного виміру часто виключається з оброблення оскільки зважені суми результатів інших вимірів будуть у допустимих межах відрізнятися від тих значень, які мали б місце за безвідмовної роботи всіх ОМ використаних для розв'язання задачі. Чим більший обсяг фіксованої вибірки, тобто чим більше ОМ

використовуються для розв'язання зазначеної задачі, тим більша природна надмірність сформованої обчислювальної структури і тим більше можливостей для збереження відмовостійкості системи на основі її поступової деградації. Природно, що при цьому погіршуються характеристики ОС внаслідок виключення без заміни ОМ, що відмовили.

У відмовостійких системах із поступовою деградацією обчислювальних структур здійснюється також реконфігурація системних програмних засобів. Так може здійснюватися переміщення ОС у ділянку пам'яті, де немає відмов, перевідображення адрес пам'яті з метою виключення модуля, що відмовив. Для збереження працездатності застосовуються альтернативні алгоритми виконання операцій на іншій апаратурі у разі виходу з ладу основної апаратури в складі ОМ. Так, наприклад, у разі несправності в блоці прискореного множення операція множення може виконуватися за допомогою альтернативної мікропрограми й основного суматора.

Широко застосовується метод поступової деградації для збереження відмовостійкості оперативної, постійної пам'яті ОМ і загальної пам'яті ІС.

Реконфігурація будь-яких елементів системи виконується спеціальними перемикачами, під якими маються на увазі апаратні та програмні засоби керування. Як апаратні перемикачі реконфігурації використовуються складові процесори або обчислювальні модулі системи, а також спеціально синтезовані для цих цілей пристрої керування. Програмні засоби реконфігурації мають бути складовими частинами загальної або розподіленої операційної системи. Під час побудови перемикача реконфігурації ІС слід передбачити можливість появи у його роботі помилок типу «обрив» або «коротке замикання». Помилки першого типу виникають у системі в разі відмов самих перемикачів реконфігурації. Вони можуть призвести до часткового або повного порушення всієї системи забезпечення відмовостійкості ІС. Тому слід звернути особливу увагу на досягнення високих рівнів відмовостійкості перемикачів реконфігурації. Помилки другого типу («коротке замикання») призводять до несанкціонованої активізації механізму перемикання внаслідок збійних або власних програмних помилок, як перемикача,

так і пов'язаних із ним пристроїв.

Існують різні способи усунення зазначених помилок у роботі перемикачів реконфігурації. Так, наприклад, у системі Pluribus доступ до перемикачів реконфігурації контролюється механізмом паролів, а у системі FTMP передбачено триразове виконання програм реконфігурації.

2. Повне відновлення обчислювального процесу безпосередньо слідує за етапом реконфігурації. Відновлення обчислювального або інформаційного процесу можна розглядати за деякою аналогією з відновленням працездатного стану людини, який, своєю чергою, досяжний тільки в тому разі, якщо усунуто порушення функцій організму, що стосуються виконуваної ним роботи. Тобто і проблему відновлення обчислювального процесу слід розглядати з позицій єдиної системи «апаратура - алгоритм». Цей підхід визначає необхідність виправлення порушених функцій у системі, а не тільки усунення виявлених відмов, збійних або програмних помилок. З цих позицій будь-яка ІС являє собою кібернетичну систему з властивими їй властивостями спостережуваності та керованості.

Для відновлення правильного функціонування системи можливе або продовження виконання обчислювального процесу (ОП) після усунення можливих наслідків помилки - метод FER (foward error recovery), або беззастережне повернення ОП до деякого попереднього стану, для якого є підстави припускати, що він не зазнав впливу помилки - метод BER (backward error recovery).

Метод FER застосовано, зокрема, в системі ESS, призначеній для обслуговування телефонної мережі. На рівні програмного забезпечення цей метод дає змогу відновити кожен модуль пам'яті зі зруйнованою інформацією шляхом заміщення інформації на неушкоджену копію з диска.

Найбільш поширеним є метод BER, який реалізує резервування ОП з деякого моменту часу, що передує виявленню несправності. Існує два рівня реалізації процесу «повернення ОП». Перший рівень можливий, якщо помилкою зіпсовано невеликий обсяг інформації, який допустимий для реалізації механізму контрольної точки. Другий рівень відновлення необхідний у разі псування помилкою великого обсягу інформації або якщо в системі не передбачено

механізму контрольних точок. У цих випадках застосовується рестарт (повторний запуск). «Легкий» рестарт ґрунтується на тому, що всі виконувані раніше процеси збережені, і тому з його допомогою поновлюються всі операції з моменту виявлення несправності. «Глибокий» рестарт застосовується в тому разі, якщо низку процесів втрачено, при цьому збережені процеси за необхідності поновлюються від моментів виявлення несправностей, а втрачені процеси поновлюються спочатку. Нарешті, якщо внаслідок несправності (наприклад, типу «зупинка» або «зациклення») жоден із процесів не зберігається, то «глибокий» рестарт трансформується у «перезапуск» системи, під час якого здійснюється її повне перезавантаження.

У відмовостійких керуючих ІС реального часу відновлення ОП на рівнях «глибокого» рестарту або «перезапуску» обов'язково пов'язане з виникненням в ІС часткової або повної функціональної відмови. Тому система захисту від відмов повинна мати можливості підтримувати процес відновлення на першому рівні, або на рівні «легкого» рестарту.

3. Відновлення обчислювальних структур ІС полягає в ремонті складових компонентів, що відмовили, виведених з конфігурації системи. Обчислювальні структури вузького класу ІС, що застосовуються в бортових автоматичних системах автономного функціонування зазвичай не відновлюються.

Широкий клас керуючих ІС у складі різних типів АСУ обслуговують ремонтні бригади. АСУ обслуговується ремонтними бригадами. У цих системах можливі відновлення компонентів, що відмовили, або в робочому стані, або в інтервалах часу між розв'язаннями завдань управління. Ремонт у робочому стані може здійснюватися без затримки після виведення з конфігурації ІС компонентів, що відмовили, у разі спільного дотримання таких чотирьох умов:

1) обчислювальні структури скомпоновані у вигляді типових елементів заміни (ТЕЗ);

2) у конструкції пристроїв ІС передбачено можливість заміни ТЕЗ без вимкнення живлення інших пристроїв;

3) у ЗІПі ІС є хоча б один справний ТЕЗ того ж типу, що й виведений з

конфігурації системи;

4) є вільна від обслуговування ремонтна бригада.

Остання умова менш жорстка, ніж перші три, оскільки існує можливість звільнення ремонтної бригади (або хоча б одного фахівця з експлуатації протягом роботи ІС після усунення відмов інших компонентів ІС). Якщо перераховані умови не виконуються, особливо це стосується перших двох умов, то відновлення обчислювальних структур можливе тільки в неробочому стані.

4. Реінтеграція. Після фізичної заміни компонента відремонтований модуль має бути возз'єднаний із системою. Для ремонту в робочому стані реінтеграція повинна бути виконана без переривання роботи системи.

2.3 Системи забезпечення відмовостійкості

Системи забезпечення відмовостійкості (СЗВ) є складовою частиною інформаційних систем (ІС) і формуються під час їх проектування з передбачених надлишкових апаратних і програмних засобів. Усі системи забезпечення відмовостійкості (СЗВ) ІС за організацією архітектури цих систем можна розділити на такі три групи:

- СЗВ закритого типу;
- СЗВ відкритого типу;
- СЗВ змішаного типу.

Системи забезпечення відмовостійкості закритого типу будуються на базі статичної надмірності. У цих системах застосовують жорсткий алгоритм функціонування. Характерні прийоми побудови алгоритму функціонування СЗВ закритого типу полягають у маскуванні несправностей на локальному (модульному) та/або глобальному (системному) рівнях. Маскування несправностей у чистому вигляді не виконує функцію виявлення несправностей - його призначення полягає в автоматичній нейтралізації несправностей, що виникли, і факти виникнення не реєструються. На локальному рівні для маскування несправностей застосовуються коригувальні коди, що дають змогу як виявляти, так

і виправляти помилки. Для реалізації таких кодів необхідні кодувальні і декодувальні пристрої, що містять mінформаційних і k<m контрольних розрядів. Слід зазначити, що ефективні коди для виправлення помилок розроблено для умов, коли інформація не перетворюється. Тому їхнє застосування обмежене цифровими пристроями зберігання і передавання інформації і, крім того, пов'язане з необхідністю введення великого обсягу апаратурної надмірності. На глобальному рівні для маскування несправностей застосовується мажоритарне резервування, зокрема штучне або природне гібридне мажоритарне резервування.

Перевага СЗВ закритого типу полягає в оперативності усунення несправностей. Недоліки: велика(часом надмірна) надмірність, специфічна придатність для об'єктів. У СЗВ відкритого типу реалізуються властивості спостережуваності та керованості, можливий гнучкий алгоритм функціонування. Отже, є всі вихідні передумови для реалізації механізму адаптації до несправностей ІС (апостеріорної адаптації), а також механізму адаптації до інформаційного навантаження ІС з метою здійснення динамічної стабілізації процесів обробки інформації в ІС (апріорної адаптації) обмеженої групи завдань, що реалізуються пристроями ІС тощо.

У таких СЗВ застосовується динамічна надмірність, що дає змогу значно знизити надмірність порівняно зі статичною її організацією. Переваги СЗВ відкритого типу очевидні: значно більша універсальність порівняно з з маскуванням несправностей, існування здатності СЗВ пристосування до характеру несправності, наявність можливості цілеспрямованої перебудови структури ІС, щоб у разі відмови модуля включити в роботу інші наявні ресурси тощо. Разом з тим, проявляються і суттєві недоліки: збільшується час адаптації до відмов модулів ІС, значно підвищується роль засобів виявлення несправностей, ефективність яких можна забезпечити знову – таки за рахунок введення значної структурної, інформаційної або часової надмірності. Це замкнене коло вдається розірвати за допомогою оригінальних методів адаптивної відмовостійкості (активного захисту).

Певною мірою недоліки, притаманні закритим і відкритим СЗВ, усуваються в СЗВ змішаного типу. Граф станів цих систем показано на рис. 2.1. У стані 0

інформаційна система перебуває в режимі підготовки до функціонування. Запобігають несправностям шляхом створення комфортної роботи апаратури та програм, а також здійснюється попередня локалізація несправностей шляхом розв'язання контрольних задач, контролю функціонування ІС, діагностичного тестування програм і апаратури.

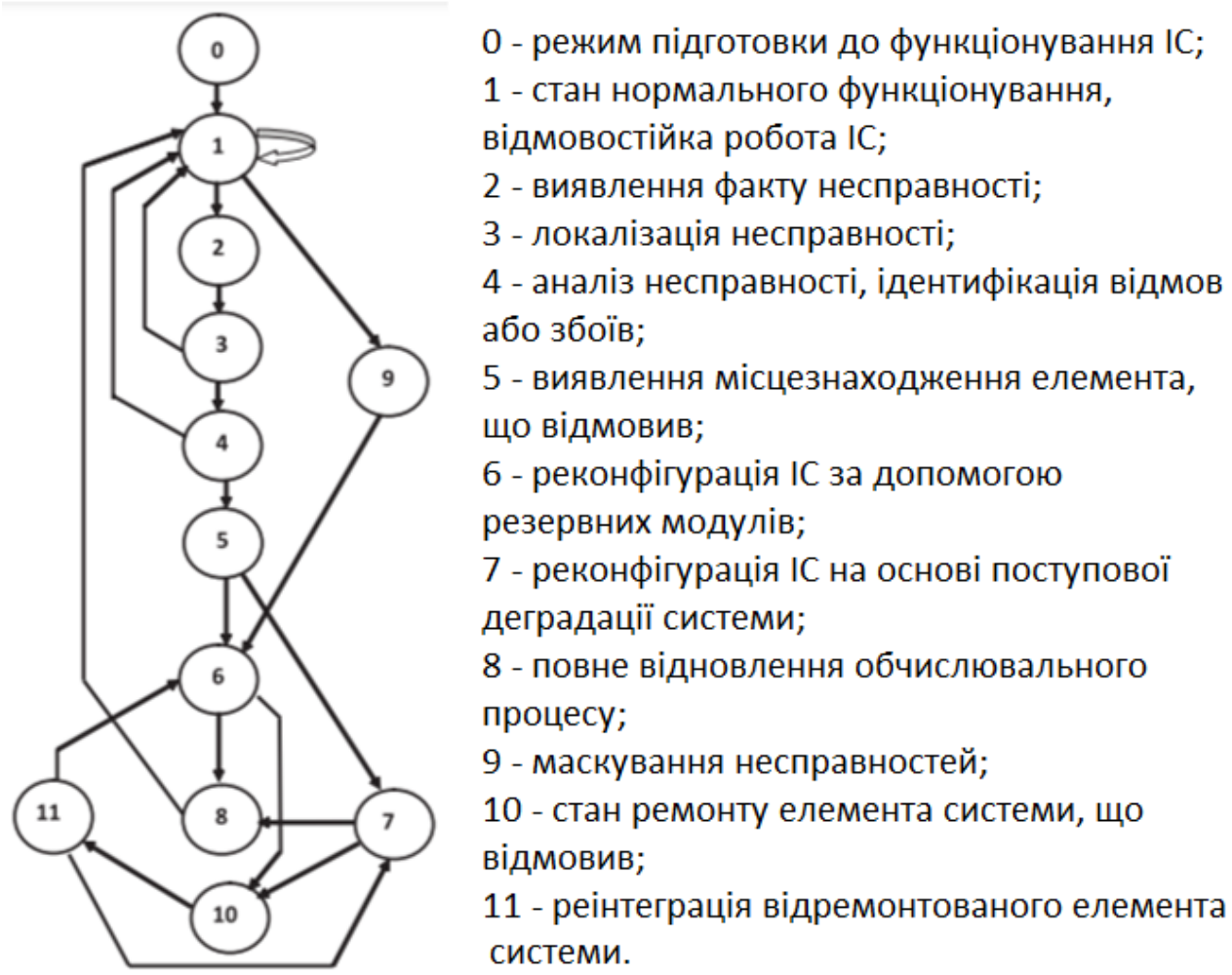


Рисунок 2.1 - Граф станів системи забезпечення відмовостійкості інформаційної системи

У стан 1 система переходить під час надходження заявок на обслуговування. У цьому стані здійснюється динамічна стабілізація відмовостійкості ІС. У разі виникнення несправності відбувається або виявлення і локалізація несправності (стани 2 і 3), або маскування несправності (стан 9), якщо це передбачено. Якщо допустимо виключення спотвореної інформації або ступінь спотворення незначний

і ним можна знехтувати, то відбувається перехід у стан 1. У протилежних і найбільш характерних випадках потрібно продовжити аналіз несправності і класифікувати її на відмови або збої (стан 4). У разі збою відбувається перехід 4-1.

Якщо в результаті класифікації несправності встановлено відмову, то в стані 5 вмикаються механізми виявлення модуля, що відмовив. Потім відбувається перехід 5-6 і проводиться реконфігурація системи за допомогою резервних модулів. Якщо резервні модулі відсутні (не передбачені або використані), то відбувається перехід у стан 7 для реконфігурації системи за допомогою природної надмірності (вільних основних модулів, або шляхом поступової деградації).

За будь-якого варіанта реконфігурації відбудеться перехід у стан 8 повного відновлення обчислювального процесу з подальшим переходом 8-1, а також ремонт модуля, що відмовив (стан 10). Реінтеграція відремонтованого модуля (або заміненого на справний із ЗІПу) проводиться з пріоритетом стану 7 системи щодо стану 6.

РОЗДІЛ 3. ЗАБЕЗПЕЧЕННЯ СТІЙКОСТІ ТА ЖИВУЧОСТІ СПЕЦІАЛІЗОВАНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У КОМП'ЮТЕРНИХ МЕРЕЖАХ

Відомі методи забезпечення відмовостійкості та живучості спеціалізованих ІТ орієнтовані на їхні різні типи, додатки, функції комп'ютерів і особливості реалізації в різних компонентах ІТ. Крім того, нагальною галуззю, що потребує дослідження, є вплив зовнішніх чинників (розподілені атаки, шкідливе програмне забезпечення (ПЗ)) на функціонування ІТ, стійкість і живучість.

Методи забезпечення відмовостійкості та живучості спеціалізованих ІТ недостатньо систематизовані та не завжди можуть бути реалізовані через специфіку використання та структуру спеціалізованих ІТ. Отже, необхідні подальші дослідження і розробка нових методів і технологій, які можуть підвищити відмовостійкість і живучість спеціалізованих ІТ, включно з кібератаками і шкідливим ПЗ.

3.1 Критерії ефективності та стійкості спеціалізованих інформаційних технологій у комп'ютерних мережах

Уявімо спеціалізовану ІТ у корпоративних комп'ютерних мережах безліччю її компонентів:

$$S_{IT} = \{S_1, S_2, ..., S_n\} , \quad (3.1)$$

де S_i - компонента спеціалізованої ІТ у корпоративних комп'ютерних мережах, i = 1,2,....,n , n - кількість компонентів.

Для кожної компоненти S_i застосуємо функцію, яка включатиме всі критерії ефективності в корпоративних комп'ютерних мережах, застосування яких під час розроблення ІТ необхідне для подальшого користування нею. Зокрема, серед таких критеріїв будуть також критерії забезпечення відмовостійкості та живучості. Задамо критерії ефективності спеціалізованих ІТ вектором, компонентами якого

будуть функції ефективності, що відповідають конкретним критеріям:

$$K_e = (f_1, f_2,...,f_m), \quad (3.2)$$

де f_j - функція, що задає один із критеріїв ефективності, j = 1,2,...,m, m - кількість функцій.

З огляду на те, що загалом завдання досягнення максимальної ефективності залежить від конкретних критеріїв, які можуть бути пов'язані між собою і, відповідно, впливати один на одного, при цьому поліпшення ефективності одного може призвести до погіршення іншого. Крім того, оскільки спеціалізовані ІТ складаються з компонентів, до яких застосовуються ті самі критерії із заданого вектора, то завдання ускладнюється тим, що частина компонентів ІТ є різною і, відповідно, досягнення ефективності за тими самими наборами критеріїв буде різним. Тому вибір оптимальних рішень є складним багатокритеріальним завданням. Загальну постановку задачі пошуку найкращої ефективності для спеціалізованих ІТ у корпоративних комп'ютерних мережах сформулюємо так:

$$\begin{cases} K_e(S_{IT}) \to max; \\ f_j(S_i) \to max,\ i = 1,2,...,n;\ j = 1,2,...,m. \end{cases} \quad (3.3)$$

Крім того, деякі ІТ-компоненти можуть бути функціонально повторюваними залежно від завдань і місця знаходження в корпоративних комп'ютерних мережах. Це вплине на загальну ефективність спеціалізованих ІТ. Однак досягнення продуктивності за певними критеріями в одних і тих самих компонентах спеціалізованої ІТ не обов'язково повинно бути однаковим, бо ці компоненти вирішуватимуть різні завдання або одні й ті самі завдання, але в різний час вони проходитимуть різні стадії. Пам'ятати про ці функції важливо, тому ми деталізуємо задачу з пошуку найкращої продуктивності для спеціалізованих ІТ у корпоративних комп'ютерних мережах таким чином:

$$\begin{cases} K_e(S_{IT}) \to max; \\ f_{j,q}(S_{i,p}) \to max,\ i = 1,2,...,n;\ j = 1,2,...,m \\ q = 0,1,...,n_q;\ j = 0,1,...,n_p \end{cases} \quad (3.4)$$

де q - номер компонента спеціалізованої ІТ у певному вузлі корпоративної

комп'ютерної мережі; j - індекс для критерію ефективності компонентів спеціалізованих ІТ у певному вузлі корпоративної комп'ютерної мережі; q = 0,1,..., n_q, j = 0,1,....,n_p; n_q - кількість однакових компонентів спеціалізованої ІТ у корпоративній комп'ютерній мережі; n_p - номер критерію для однакових компонентів спеціалізованої ІТ у корпоративній комп'ютерній мережі.

Уведемо функцію, яка визначатиме максимальне значення критерію ефективності:

$$F : K_e(S_{IT}) \to \max . \tag{3.5}$$

Значення критерію ефективності задамо виразом з урахуванням вагових коефіцієнтів:

$$K_e(S_{IT}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{p=0}^{n_p} \sum_{q=0}^{n_q} \left(\alpha_{i,j,p,q} \cdot f_{j,q}(S_{i,p}) \right), \tag{3.6}$$

де $a_{i, j, p, q}$ - вагові коефіцієнти.

Розглянемо досягнення максимізації критеріїв за показниками відмовостійкості та живучості в конфігураціях інформаційних технологій, побудованих на основі архітектури «клієнт-сервер» з їхнім забезпеченням за всіма ланками системи від користувачів (клієнтська частина) до критично важливої серверної частини. Вибір для розгляду саме архітектури «клієнт-сервер» залежить від її особливостей, які проявляються в такому: базові функції клієнтського додатка розподіляються між клієнтом і сервером; програмне забезпечення автоматизованого робочого місця клієнтського комп'ютера працює з даними через запити до серверного програмного забезпечення; здійснюється повна підтримка багатокористувацької роботи; гарантується цілісність даних. Це відрізняє її від інших архітектур і дає змогу здійснити забезпечення відмовостійкості та живучості в кожній із ланок системи окремо.

Основними напрямками для підвищення живучості та відмовостійкості ІТ є внесення надмірності в конфігурацію апаратних і програмних засобів, підтримуючої інфраструктури, резервування інформаційних ресурсів (програм і даних).

Відомі два підходи в побудові відмовостійкої ІТ. Перший підхід базується на

використанні відмовостійких компонентів. Така ІТ забезпечує свої функції навіть у разі виходу з ладу підкомпонентів деяких компонентів. Це найпростіший метод, але водночас і найдорожчий, через застосування найдорожчих складових - відмовостійких компонентів ІТ. Другий спосіб полягає в побудові відмовостійкої ІТ з використанням компонентів, які не є відмовостійкими. Відмовостійкість у таких системах досягається за рахунок введення в них надмірності через резервування критичних ланок апаратного забезпечення, програмного забезпечення, міжкомпонентних зв'язків та спеціальних алгоритмів функціонування ІТ, які передбачають її реконфігурацію в разі відмови деяких компонентів.

Головною властивістю відмовостійкості є прозорість відмов її окремих компонентів для кінцевого користувача. Це означає, що відмовостійка система автоматично змінює свою конфігурацію в разі відмови. Її програмне забезпечення в процесі виконання шукає обхідні шляхи, намагаючись в умовах відмови привести виконувану функцію до успішного завершення.

Задамо функцію $f_1(S_i)$, i = 1,2,....,n визначення відмовостійкості в комп'ютерних системах у кількісному вигляді так:

$$f_1(S_i) = \frac{T_{f_1(S_i),1}}{T_{f_1(S_i),1} - \left(T_{f_1(S_i),2} - T_{f_1(S_i),3}\right)}, \tag{3.7}$$

де i - кількість компонентів спеціалізованої ІТ, i = 1,2,...,n , $T_{f1(Si),1}$ - час між сусідніми збоями; $T_{f1(Si),2}$ - час, необхідний для виявлення збою та пошуку шляху його обходу; $T_{f1(Si),3}$ - час, необхідний для відновлення ІТ після збою.

Як видно з формули (3.7), для ІТ з автоматичною системою забезпечення відмовостійкості вона наближатиметься до максимуму через швидкість реакції. Для побудови таких систем немає теоретичних перешкод, але на практиці під час їхньої реалізації потрібно враховувати низку важливих чинників: фінансові витрати на реалізацію автоматичної системи забезпечення живучості та відмовостійкості; складність системи. Для ІТ, призначених для інформаційного забезпечення у вузькій спеціалізованій предметній області, наприклад фінансово-

господарська діяльність закладу вищої освіти, буде доцільним відмовитися від автоматичної системи управління відмовостійкістю на користь автоматизованої.

За такого підходу частину дорогих функцій управління резервуванням, присутніми в ІТ, буде покладено на людину. Тоді, згідно з формулою (3.7), відмовостійкість $f_1(S_i)$ буде нижчою, ніж у першому випадку. Але розв'язанням задачі побудови ІТ (подібно до того, як і в інших задачах проектування) є не забезпечення максимально можливої відмовостійкості системи, а знаходження прийнятного балансу параметрів системи в межах певного технологічного базису.

Дослідимо розв'язання питань забезпечення відмовостійкості ІТ при використанні такої стратегії. Проаналізуємо фактори, що негативно впливають на відмовостійкість ІТ з боку клієнта. Схему впливу негативних факторів на відмостійкість клієнтської частини спеціалізованої ІТ зображено на рис. 3.1.

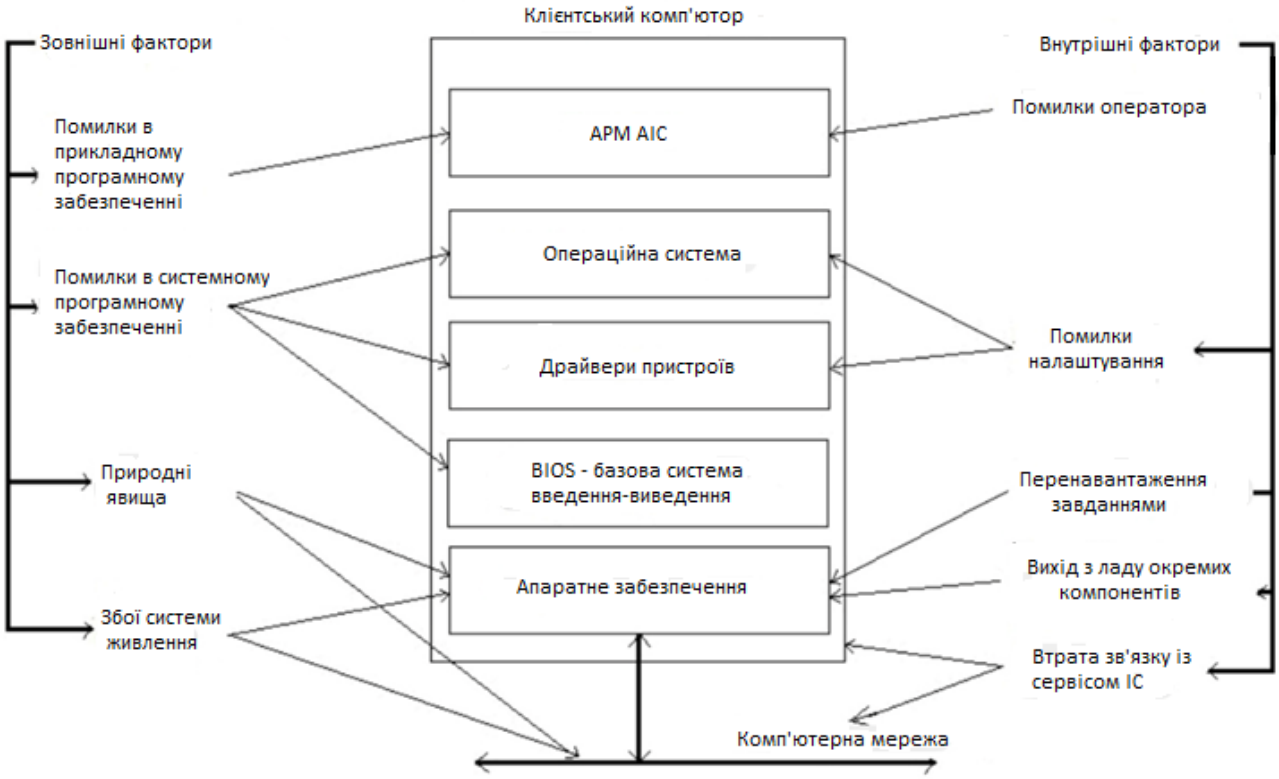


Рисунок 3.1 - Схема дії негативних факторів, що впливають на відмовостійкість клієнтської частини спеціалізованої ІТ

Як видно із запропонованої моделі, негативні чинники, що впливають на відмовостійкість клієнтської частини ІТ, поділяються на зовнішні та внутрішні.

Серед зовнішніх чинників найбільшу загрозу становлять збої в роботі енергосистем живлення та природні явища, які можуть призвести до відмов компонентів комп'ютерів і комп'ютерних мереж.

Іншим важливим фактором є помилки в коді системного програмного забезпечення. Причина в тому, що системне програмне забезпечення являє собою складну систему і кожне виправлення раніше знайденої помилки не гарантує відсутності привнесення нової. Цей аспект, пов'язаний із системним програмним забезпеченням, є породженням ще одного фактора, який може зменшити відмовостійкість. Через його складність під час налаштування програмного забезпечення можуть вноситися помилки налаштування. Зменшити його прояв можна шляхом використання програмного забезпечення з автоматичним налаштуванням, що не завжди прийнятно, та залученням більш кваліфікованого персоналу.

Для прикладного програмного забезпечення, до якого належать клієнтські частини спеціалізованої ІТ, критичними є помилки, що проявилися під час експлуатації робочих місць, фіксуються разом зі своїми параметрами в реєстрі системи в автоматичний спосіб і надалі використовуються для аналізу з метою усунення причин, що їх спричинили. Це досягається завдяки підходу, що ґрунтується на привнесенні деякої надмірності в програмне забезпечення клієнтської частини ІТ. З цією метою всі розрахункові процедури, які можуть містити критичні для функціонування помилки, розроблено з дотриманням заданого однотипного шаблону побудови алгоритму її виконання. Суть алгоритму відображено на рис. 3.2.

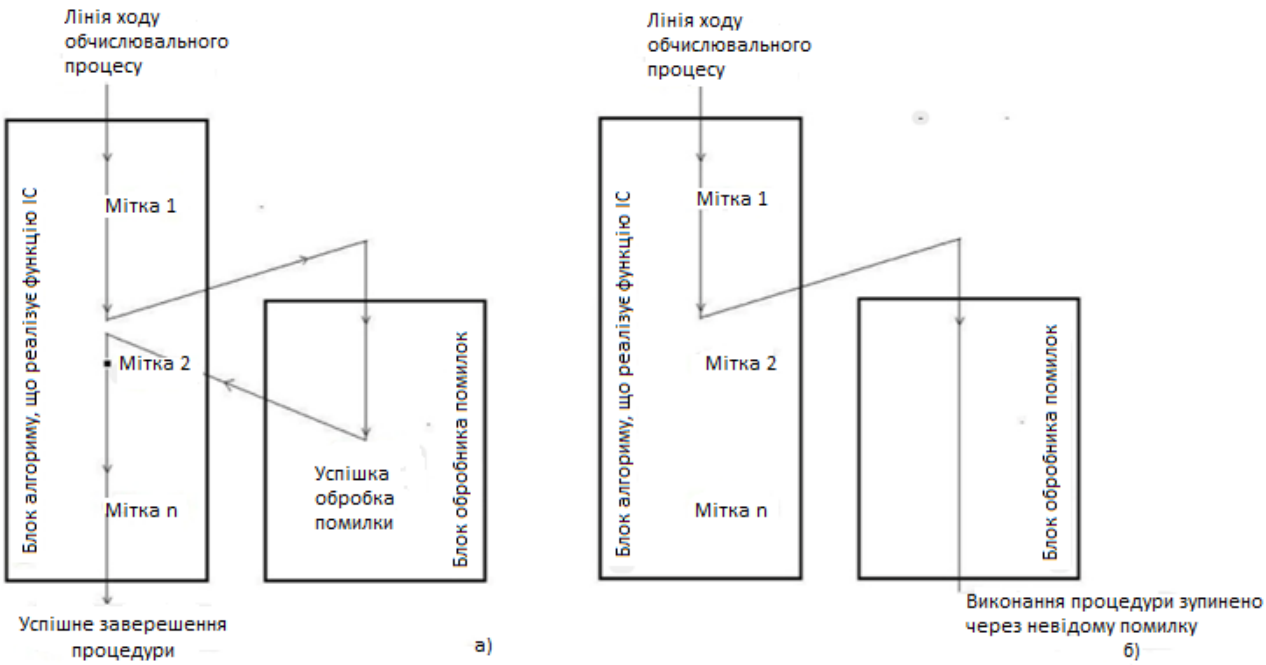


Рисунок 3.2 - Модель роботи алгоритму відмовостійкої процедури а) для випадку успішного завершення процедури; б) для випадку, коли помилка невідома для обробника помилок

У цій структурі алгоритм виконання будь-якої нетривіальної процедури поділяється на два взаємодіючі блоки. У першому блоці реалізується функція процедури ІТ, а в другому - обробник помилок. У процесі виконання деякої процедури, яка реалізує одну з функцій компонентів ІТ, обидва блоки взаємодіють між собою, передаючи управління обчислювальним процесом один одному, поки виконувана функція не завершиться. Його суть полягає в тому, що алгоритм, який реалізує функцію ІТ, розділяється маркерами (мітка 1, ..., мітка n на рис. 3.2) на фрагменти за принципом функціональної завершеності.

Таке рішення, окрім усього, дає змогу збирати інформацію про фатальні помилки, що сталися в процесі функціонування ІТ, і в процесі подальшого аналізу дає змогу виявити слабкі ланки в ІТ з метою їхнього усунення шляхом удосконалення програмного забезпечення клієнтської частини ІТ.

Програмне забезпечення клієнтських частин ІТ упродовж життєвого циклу ІТ з різних причин, зокрема й через виявлення в ньому помилки, може змінюватися,

проходячи свої власні цикли оновлення.

Наступним зі значущих внутрішніх чинників, що негативно впливають на відмовостійкість, є перевантаження апаратної платформи клієнтського комп'ютера задачами, що може різко погіршити часові параметри виконуваних завдань клієнтською частиною ІТ, або навіть унеможливити його роботу через вичерпання технічних ресурсів.

Щоб нейтралізувати дію цього чинника в ІТ, під час розроблення програмного забезпечення застосовано функціональне резервування, а саме тієї його частини, яка відповідальна за реалізацію «бізнес-логіки».

Наявність функціонального резерву «важких» розрахункових функцій дає змогу здійснювати маневр обчислювальними потужностями апаратної платформи ІТ у разі перевантаження окремих її ланок, підвищуючи таким чином відмовостійкість. Оскільки процедура, яка функціонально резервується, розробляється у двох варіантах за одним алгоритмом, але в різних програмних середовищах, для виконання в різних технічних засобах. У цьому проявляється позитивна мультиплікативність ефекту функціонального резервування, що підвищує загальну відмовостійкість ІТ.

Загалом, відмовостійкість клієнтської частини забезпечено шляхом виконання комплексу заходів, що включають як перелічені вище, так і деякі додаткові. Наприклад, використання нетривіальних редакторів даних, що включають у свій алгоритм роботи інтерактивну процедуру, що унеможливлює неконтрольоване маніпулювання даними бази даних (БД) з боку оператора.

У результаті аналізу факторів, бази даних яких негативно впливають на відмовостійкість серверної частини ІТ, було побудовано модель дії негативних факторів, зображену на рис. 3.3.

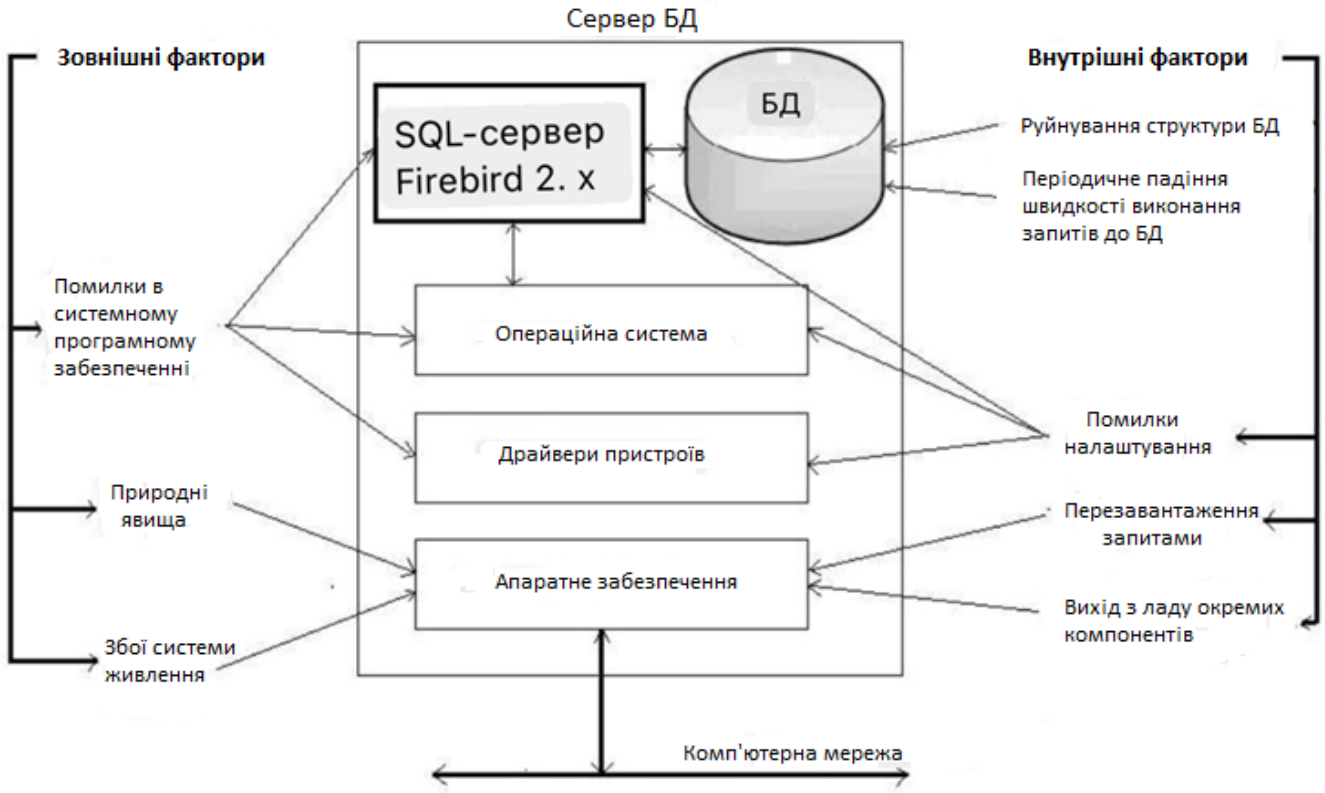


Рисунок 3.3 - Модель дії негативних чинників, що впливають на відмовостійкість сервера ІС

У представленій моделі всі негативні фактори поділяються на зовнішні, спричинені причинами, що перебувають поза межами системи, та внутрішні. Зовнішні чинники, які зменшують відмовостійкість серверної частини, нейтралізуються в той самий спосіб, що й у клієнтській частині ІТ. Але через свою значущість цього недостатньо. Оскільки сервер ІТ є місцем знаходження бази, де зосереджена вся інформація, що обробляється в системі, то для нього такий фактор, як збої в системі живлення, особливо небезпечний. Це пов'язано з тим, що БД, будучи сама по собі складно організованою системою, є чутливою до порушення технології поводження з нею. Раптове зникнення живлення або вихід із ладу апаратного забезпечення сервера через флуктуації напруги може призвести з високою ймовірністю до пошкодження бази даних з усіма подальшими негативними наслідками для інформації.

З метою недопущення такого розвитку подій, у контур системи живлення було введено пристрій безперебійного живлення з подвійним перетворенням

напруги і достатнім часом забезпечення автономної роботи сервера. Крім того, пристрій безперебійного живлення повинен мати контролер стану, який містить у собі вихід із послідовним інтерфейсом. Він необхідний для передавання серверу сигналу розвантаження в разі, коли через тривале зникнення зовнішнього живлення, буде вичерпано до неприпустимої межі внутрішній запас електроенергії пристрою безперебійного живлення. За цим сигналом сервер коректно завершить усі додатки, які були запущені, не допустивши руйнування БД.

Не менш загрозливими для серверної частини, які зменшують її відмовостійкість, є внутрішні чинники. Серед них найважчий за наслідками вихід з ладу апаратних засобів, а саме накопичувачів. Для сервера ІТ вони є найвідповідальнішою ланкою, відмова якої може призвести не лише до тривалої недоступності до інформаційних ресурсів, а й до безповоротної втрати даних. Оскільки втрата БД є неприйнятною, то необхідні заходи для нейтралізації загрози раптової втрати накопичувача. Це завдання може бути вирішене шляхом його резервування. Замість окремого накопичувача для зберігання БД та інших критичних даних може бути застосовано RAID масив накопичувачів типу 1. З погляду оцінки додаткових витрат на резервування вартість додаткового накопичувача невелика порівняно з можливими втратами, викликаними втратою БД. Крім того, організація періодичного діагностування накопичувачів дасть змогу в більшості випадків завчасно виявляти проблеми накопичувача.

Ще одним способом недопущення втрати БД є організація її резервного копіювання. Незважаючи на описані заходи забезпечення відмовостійкості серверної частини ІТ, вони все ж не можуть претендувати на абсолютність. Оскільки БД і вся послідовність програмного забезпечення, що забезпечує її роботу, є складною системою, то наявність помилок її функціонування залишається досить високою.

Щоб зменшити вплив таких деструкцій, як невідомі помилки, що можуть призвести до руйнування БД, у контур програмного забезпечення сервера можна включити підсистему резервного копіювання БД. Вона працює в автоматичному режимі згідно з графіком. Репозитарієм копій БД слугує інший комп'ютер,

територіально віддалений від основного сервера. Оскільки копія БД є досить великим масивом інформації, тому, щоб не залежати від мережевого трафіку, основний сервер і комп'ютер із репозитарієм копій БД мають власний канал зв'язку. Для недопущення неконтрольованих змін даних у БД, які можуть порушити цілісність даних ІТ, усі зміни виконуються під управлінням транзакцій. Такий підхід забезпечує перехід БД з одного узгодженого стану в інший під час маніпулювання даними.

Важливо підкреслити, що процес забезпечення відмовостійкості є безперервним протягом усього життєвого циклу ІТ. Він починається з планування заходів забезпечення відмовостійкості ІТ, проєктується і триває до моменту завершення її функціонування взагалі.

Показники живучості в складній системі: багатофункціональність окремих компонентів; наявність єдиної (головної) мети функціонування всієї системи; можливість не тільки інформаційного обміну між окремими компонентами, а й інформаційної взаємодії з користувачами; наявність засобів захисту, контролю, діагностики та самоорганізації. Завдання аналізу структурної живучості вимагає визначення: системної архітектури, необхідної для виконання мети функціонування ІТ у деякий момент або проміжок часу, коли виникають небажані впливи на систему; вимог до окремих видів ресурсів системи та їхнього взаємозв'язку; вимог до функціональних можливостей ресурсів системи; особливостей характеру небажаних впливів або їхніх наслідків.

Задамо функцію $f_2(S_i)$, у якій i = 1,2,....,n визначення живучості в кількісних одиницях комп'ютерних мереж виразимо так:

$$f_2(S_i) = \frac{T_{f_2(S_i),1} - T_{f_2(S_i),2}}{T_{f_2(S_i),1}}, \tag{3.8}$$

де $T_{f2(Si),1}$ - час функціонування процесі ІТ у стандартному режимі роботи; $T_{f2(Si),2}$ - час, витрачений на процеси забезпечення живучості, i = 1,2,...,n .

Таке визначення функції живучості дає змогу відобразити стандартний режим роботи значенням одиниці, а в разі виникнення потреби в забезпеченні живучості та в разі набагато тривалішого часу, ніж час стандартного режиму

роботи, значення функції відображатиме кількісну порядкову величину.

Резервування сервера гарантує достатню живучість ІТ загалом, але не дає гарантії втрати нею деяких своїх функцій, пов'язаних із виходом з ладу апаратних компонентів клієнтського комп'ютера, що критично впливають на функціонування клієнтської частини загалом.

Розв'язання задачі, в таких випадках, полягає у створенні певного резерву. Його особливістю є те, що як резерв тут слугує будь-який інший клієнтський комп'ютер, який згідно з планом подолання критичної ситуації, може взяти на себе забезпечення роботи програмного забезпечення клієнтської частини, чий комп'ютер вийшов з ладу.

При цьому модулі програмного забезпечення в налаштованому вигляді зберігаються в репозитарії програмного забезпечення ІТ і на тих клієнтських комп'ютерах, де їх планують використати в критичні моменти згідно з планом резервування. У разі виходу з ладу критичного обладнання комп'ютера, воно переноситься на інший комп'ютер, згідно з планом резервування.

Така реконфігурація клієнтської частини стала можливою завдяки тому, що на клієнтських комп'ютерах, на яких виконується програмне забезпечення, не зберігаються абсолютно ніякі дані. А сам програмний модуль, для зручності скомпонований в один файл, і не потребує процедури інсталяції. Її достатньо скопіювати на інший комп'ютер, після чого вона буде готова до роботи. Є тільки одне обмеження - кожен екземпляр програмного забезпечення клієнтської частини попередньо має бути зареєстрований в ІТ. Інакше спроба запуску такої програми розглядатиметься як спроба несанкціонованого доступу до системи, навіть за правильних реєстраційних даних. Контроль ІТ за всіма екземплярами своїх клієнтських частин дає змогу блокувати спроби зловмисників, яким вдалося заволодіти даними акаунтами користувача, отримати доступ до системи. При цьому програма, яку опанував зловмисник, не отримує доступу до даних ІТ, а сам факт спроби такої програми під'єднатися до системи фіксується в реєстрі фатальних помилок із даними, що дає змогу їх використовувати для вжиття організаційних заходів проти зловмисників.

Таким чином, живучість ІТ забезпечується резервуванням серверної частини ІТ з територіальним рознесенням основного і резервного сервера, резервуванням програмного забезпечення клієнтської частини. Особливістю резервування є те, що як резерв слугують не апаратні модулі, а резерв продуктивності окремих клієнтських комп'ютерів, які, згідно з планом резервування, у критичний момент будуть використовуватися як штатні, не допускаючи втрати функціональності ІТ.

На основі формул (3.6)-(3.8) отримаємо значення ефективності для ІТ з урахуванням показників відмовостійкості та живучості:

$$K_e(S_{IT}) = \sum_{j=1}^{m} \sum_{p=0}^{n_p} \sum_{q=0}^{n_q} \left(\alpha_{1,j,p,q} \times \right.$$
$$\left. \times \frac{T_{f_1(S_i),1}}{T_{f_1(S_i),1} - \left(T_{f_1(S_i),2} - T_{f_1(S_i),3} \right)} + \alpha_{2,j,p,q} \cdot \frac{T_{f_2(S_i),1} - T_{f_2(S_i),2}}{T_{f_2(S_i),1}} \right), \quad (3.9)$$

де $a_{1,j,\ p,\ q}$ - коефіцієнт для значення, що визначає відмовостійкість у кількісних одиницях; $a_{2,j,\ p,\ q}$ - коефіцієнт для значення, що визначає живучість у кількісних одиницях; $a_{1,j,\ p,\ q} + a_{2,j,\ p,\ q} = 1$.

Аналогічно, доданками у формулі (3.6) та її конкретизації формулі (3.9) для двох величин можуть бути інші показники, що характеризують ефективність ІТ.

У результаті використання перелічених заходів було отримано ІТ вузькоспеціалізованого використання для різноманітних сфер застосування, де процеси супроводження відносяться до ірреального або нереального часу з достатньо високими параметрами відмовостійкості, живучості та загалом резилентності й водночас прийнятною відносно фінансових витрат на її експлуатацію.

3.2 Експерименти та оцінка

Для визначення, наскільки ефективними є пропоновані рішення щодо забезпечення відмовостійкості та живучості, проведемо порівняння критерію ефективності для ІТ без забезпечення відмовостійкості та живучості і з включенням

цих характеристик на основі формули (3.9).

Значення величини критерію ефективності ІТ, у якій не забезпечуються вимоги відмовостійкості та живучості, отримуємо з формули (3.9) так:

1) вирішення проблем, пов'язаних із відсутністю забезпечення в ІТ реалізованих відмовостійкості та живучості, покладено на оператора чи адміністратора, який постійно моніторить функціонування ІТ; розв'язання проблемних ситуацій здійснюється лише в разі їх виявлення. У першому випадку розрахунок за формулою (3.9) може бути аналогічним і отримаємо значення величини, які на порядок вищі за значення критерію для ІТ, де забезпечується відмовостійкість і живучість.

Якщо ж розглядати другий варіант, тоді $K_e(S_{IT}) = 1$. У цьому разі відношення між значеннями визначається за формулою (3.10) і дає змогу встановити ефективність запропонованих рішень щодо забезпечення відмовостійкості та живучості, а також поліпшити досягнення ефективності за рахунок коригування коефіцієнтів:

$$\mu = \frac{1}{\sum_{j=1}^{m}\sum_{p=0}^{n_p}\sum_{q=0}^{n_q}\left(\alpha_{1,j,p,q}\cdot\frac{T_{f_1(S_i),1}}{T_{f_1(S_i),1}-\left(T_{f_1(S_i),2}-T_{f_1(S_i),3}\right)}+\alpha_{2,j,p,q}\cdot\frac{T_{f_2(S_i),1}-T_{f_2(S_i),2}}{T_{f_2(S_i),1}}\right)}, \quad (3.10)$$

де відсутність збоїв у роботі спеціалізованої ІТ або зовнішніх впливів означає, що час, витрачений на їхнє опрацювання, дорівнює нулю і відповідно відношення дорівнюватиме одиниці.

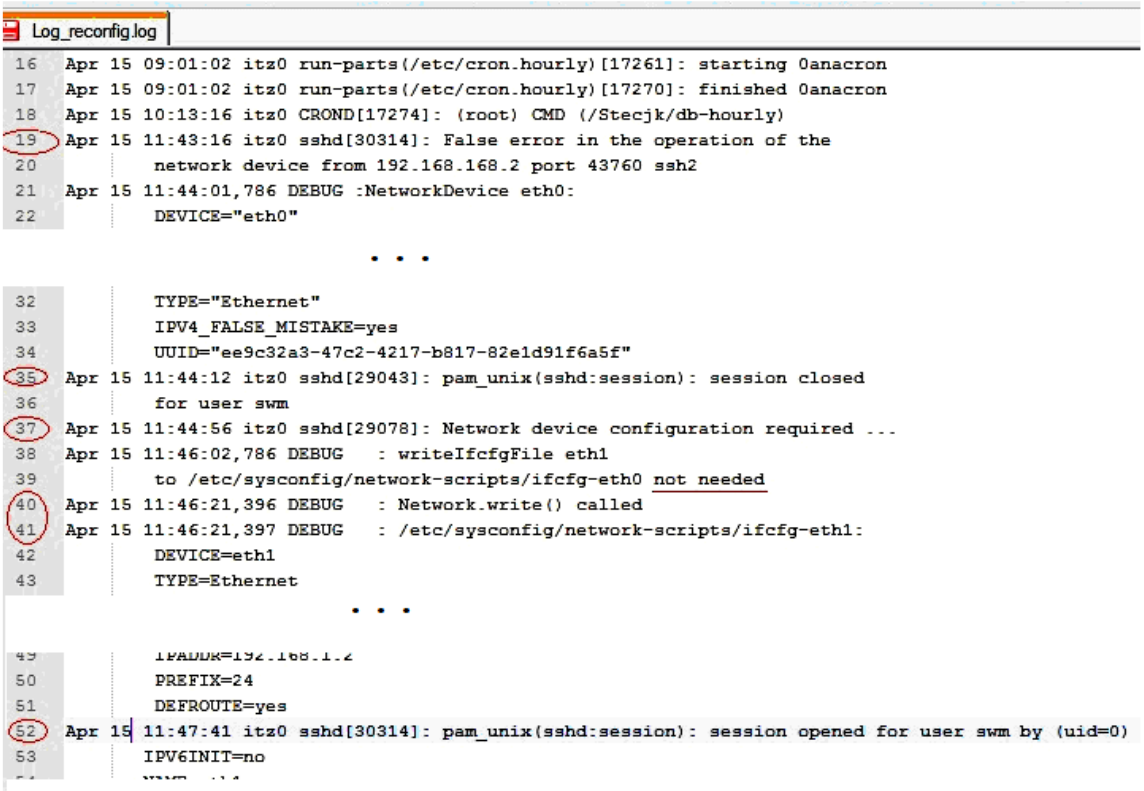


Рисунок 3.4 - Фрагмент Log-файлу підсистеми контролю роботи мережевих пристроїв

Якщо ж станеться збій або зовнішнє втручання, тоді значення буде більшим за одиницю. Ефективним значенням є значення, мінімально відхилене від одиниці.

Результати забезпечення відмовостійкості та живучості спеціалізованої ІТ зображено в реалізованій ІТ на рис. 3.4.

Для зручності всі рядки фрагмента log-файлу було пронумеровано, а критичні позиції виділено.

У позиції 19 виявлено фатальну помилку в роботі мережевого адаптера «eth0» у момент звернення комп'ютера користувача з IP 192.168.168.2. У позиції 35,36 закривається поточна сесія користувача SWM. У позиції 37 система повідомляє, що потрібна реконфігурація мережевих пристроїв. У позиції 38 повідомляється, що пристрій «eth0» відключається. У позиціях 40,41 повідомляється, що активується резервний мережевий адаптер «eth1». У позиції 52 повідомляється, що відкривається сесія користувача SWM, яка була припинена через вихід з ладу мережевого адаптера «eth0».

Графіки (рис. 3.5-3.7) отримано за розрахунками за формулою (3.10) для результатів відмовостійкості (рис. 3.5), живучості (рис. 3.6) і для випадку поєднання проявів як відмовостійкості, так і живучості (рис. 3.7).

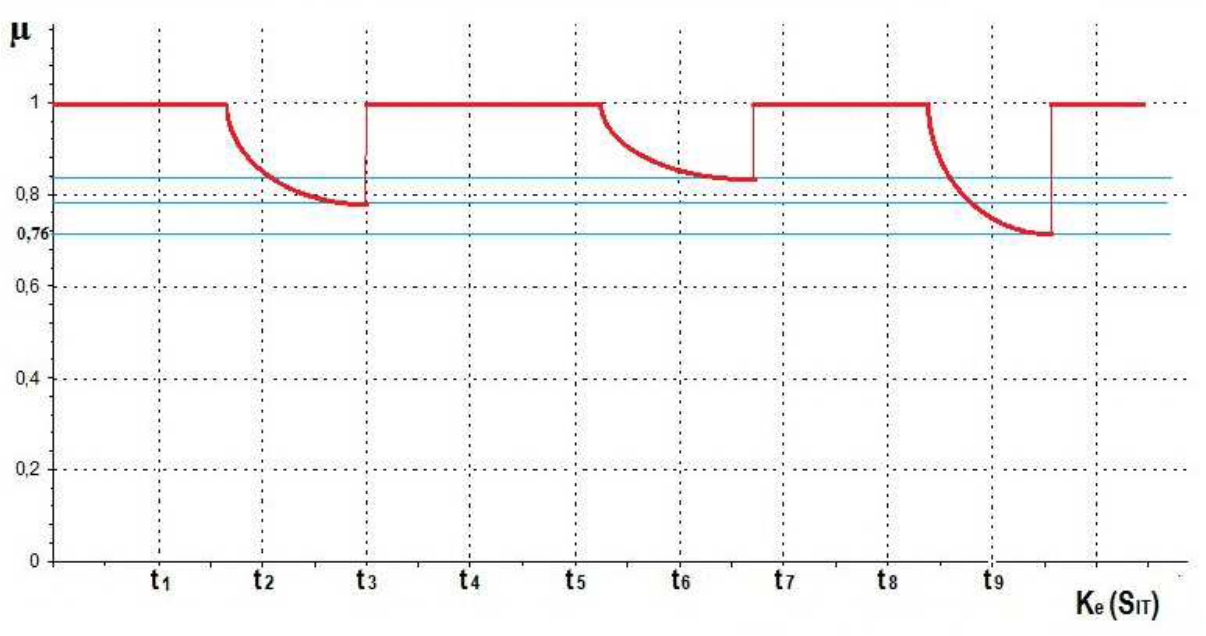


Рисунок 3.5 - Графік відмовостійкості

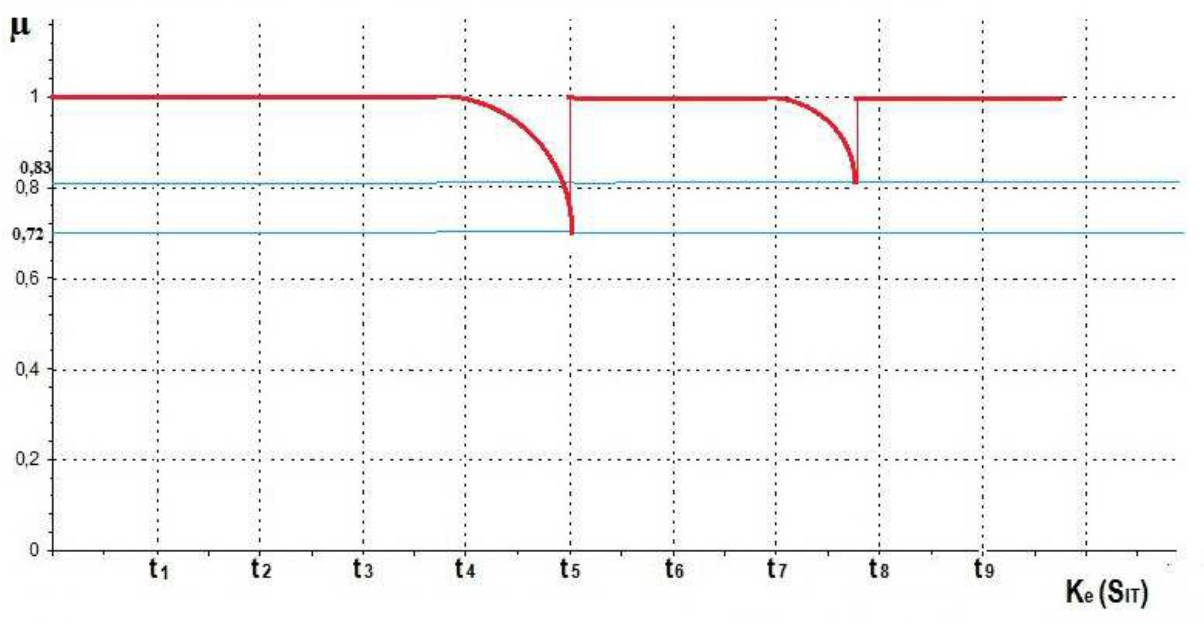


Рисунок 3.6 - Графік проявів живучості

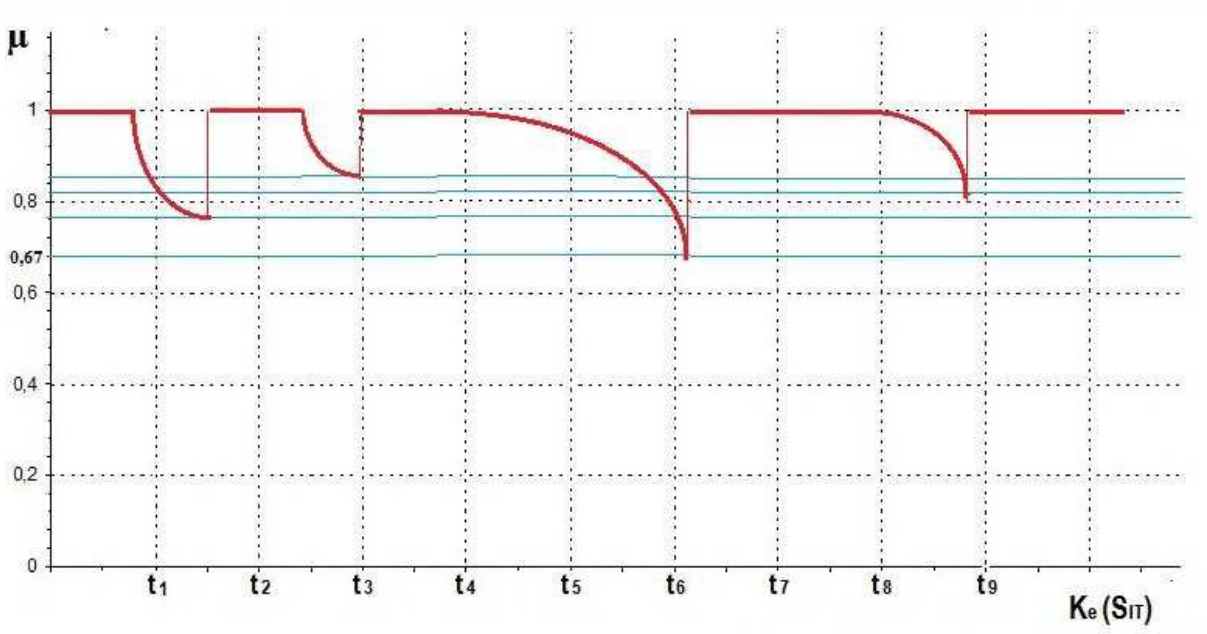


Рисунок 3.7 - Графік відображення одночасних проявів відмовостійкості та живучості

Результати дослідження підтверджують високий рівень відмовостійкості та живучості корпоративних комп'ютерних мереж, що становить понад 75 %.

Важливим напрямом подальших досліджень для підвищення ефективності ІТ є розробка методу забезпечення ефективного захисту інформації безпосередньо в структурі ІТ та обчислювальних процесах, що протікають у процесах обчислень. Їхнє врахування в загальному критерії визначення ефективності ІТ дасть змогу збалансувати такі величини, як живучість, відмовостійкість і захист інформації, виражені в кількісному вигляді, і стане основою розроблення спеціалізованої ІТ із поліпшеними характеристиками.

ВИСНОВКИ

Весь спектр методів і способів побудови надійних відмовостійких інформаційних систем поділяється на дві групи, що складають:

1. Методи аналізу і синтезу структурної надійності об'єктів інформаційних систем; методи аналізу і синтезу функціональної надійності виконання інформаційних процесів; методи аналізу і синтезу побудови відмовостійких інформаційних систем реального часу, методи аналізу та синтезу функціональної безпеки інформаційних систем управління.

2. Технологія створення надійної елементної бази з урахуванням кліматичних умов, режимів роботи, рівня інтеграції та технології виготовлення, інформаційних навантажень технології виготовлення, інформаційних навантажень, перегонів і змагань сигналів тощо; технологія розроблення та супроводу функціонально надійних програмних засобів; способи побудови якісних людино-машинних (ергономічних) інформаційних систем на основі теорії та практики ергономіки.

Таким чином, розроблено підхід до визначення ефективності ІТ на основі врахування кількісних величин, що характеризують відмовостійкість і живучість, який може бути розширено для врахування інших характеристичних величин. Для забезпечення відмовостійкості та живучості ІТ розроблено систему заходів, у результаті виконання яких отримано ІТ вузькоспеціалізованого використання для різноманітних сфер застосування, де супроводження процесів відносяться до систем ірреального або нереального часу, з достатньо високими параметрами відмовостійкості, живучості та загалом резилентності і, водночас, прийнятними фінансовими витратами на її експлуатацію.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Li. K., Hudak. P. Memory Coherence in Shared Virtual Memory Systems// ASM Trans. On Computer Systems. – 2019. – Vol.7, N11. – P. 321 – 359.
2. Baker T.P. Stack Based Scheduling of Real-Time Processes// Journal on Real-Time Systems. – 2015. –Vol 3, N1. – P. 67 – 99.
3. Silva J.G.D., Wood J.V. Design of processing subsystems for Manchester data flow computer // IEEE Proc. N.Y. – 2021. – Vol. 128, N 5. – P. 218 – 224.
4. Watson R., Guard J. A practical data flow computer// Computer. – 2018. – Vol. 15, N 2.– P. 51 – 57.
5. Johnson D. Data flow machines thereaten the program counter// Electronic Design. – 2005. – N 22. – P. 255 – 258.
6. «Resiliency in Distributed Systems». [Електронний ресурс]. — Режим доступу: https://blog.gojekengineering.com/resiliency-in-distributed-systems-efd30f74baf4
7. «How to build Resilience in large scale Distributed Systems». [Електронний ресурс]. — Режим доступу: https://blog.gojekengineering.com/how-to-build-resilience-in-large-scale-distributed-systems-ddd1496b6c0
8. «2016 Stock Market Investor Profile» [Електронний ресурс]. — Режим доступу: http://www.pseacademy.com.ph/LM/investors~details/id-1498096241729/2016_Stock_Market_Investor_Profile.html
9. Mike Ebbers Introduction to the New Mainframe: z/OS Basics [Text] / Mike Ebbers, John Kettner, Wayne O'Brien, Bill Ogden.— 3rd edition. — IBM Redbooks, 2012. — 96 p.
10. Діаграма послідовності [Електронний ресурс]. — Режим доступу : https://uk.wikipedia.org/wiki/Діаграма_послідовності.

11. Соколов А. Методи та алгоритми паралельних обчислень [Текст] / Соколов А.В., Барковський Є.А., Кучумов Р.І., Сазонов А.М - Петрозаводськ: ПетрГУ, 2016. - 48 с.

12. Шпаковський Г.І. Програмування для багатопроцесорних систем у стандарті MPI [Текст]/Г.І.Шпаковський, Н.В.Серикова. - Мн.: БДУ, 2012. - 323 с.
13. «Принципи побудови стійких до відмови IT-систем» [Електронний ресурс]. Режим доступу: https://www.pcweek.ua/themes/detail.php?ID=152353&THEME_ID=13879 -
14. В. П. Котляров Проектування відмово розподілених інформаційних систем для студентів [Текст] / Котляров В. П., - 2019. - 246 с.
15. P. M. Mell and T. Grance, “Sp 800-145. the nist definition of cloud computing,” National Institute of Standards & Technology, Gaithersburg, MD, United States, Tech. Rep., 2011.
16. Oracle. 1.1.1. Brief History of Virtualization. [Online]. Available: https: //docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1010.html.
17. VMware, Inc. Understanding Full Virtualization, Paravirtualization and Hardware Assist. [Online]. Available: https://www.vmware.com/files/pdf/ VMware_paravirtualization.pdf.
18. R. Vanover. Type 1 and Type 2 Hypervisors Explained. (Accessed on 05/24/2016). [Online]. Available: https://virtualizationreview.com/blogs/ everyday-virtualization/2009/06/type-1- and-type-2-hypervisors-explained.aspx.
19. VMware, Inc., “vSphere ESXi Hypervisor,” 2016. [Online]. Available: http: //www.vmware.com/se/products/vsphere/features/esxi-hypervisor.html.
20. Vmware vsphere with operations management: High availability | vmware sverige.

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

«ПІДВИЩЕННЯ ВІДМОВОСТІЙКОСТІ КОМП'ЮТЕРНОЇ СИСТЕМИ ЗА ДОПОМОГОЮ ОПТИМІЗАЦІЇ АПАРАТНОЇ ТА ПРОГРАМНОЇ ЧАСТИН»

виконав студент: **Максим ВІТАШ**
керівник: **Андрій ЛЕМЕШКО**, доктор філософії, доцент

Мета бакалаврської роботи:

Дослідження та оптимізація апаратної та програмної частин комп’ютерної системи з метою підвищення її відмовостійкості

Об’єкт дослідження:

процес оптимізації та підвищення відмовостійкості

Предмет дослідження:

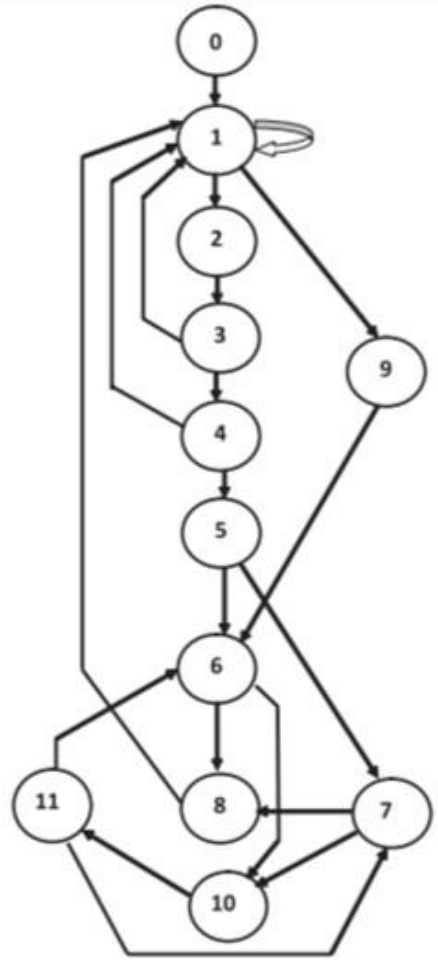
комп’ютерна система

Актуальність:

Для інформаційних технологій, що забезпечують життєдіяльність установ чи підприємств у різних галузях, питання відмовостійкості є дуже важливим, особливо внаслідок зростання кількісних параметрів функціонування (збільшення кількості користувачів, серверів, об'ємів інформації в базах даних) та рівня складності. Від забезпечення цих параметрів у прямій залежності перебуває ефективність функціонування всієї комп'ютерної системи. Тому дуже важливо забезпечити її надійність та стійкість до відмов для запобігання можливих проблем і втрат.

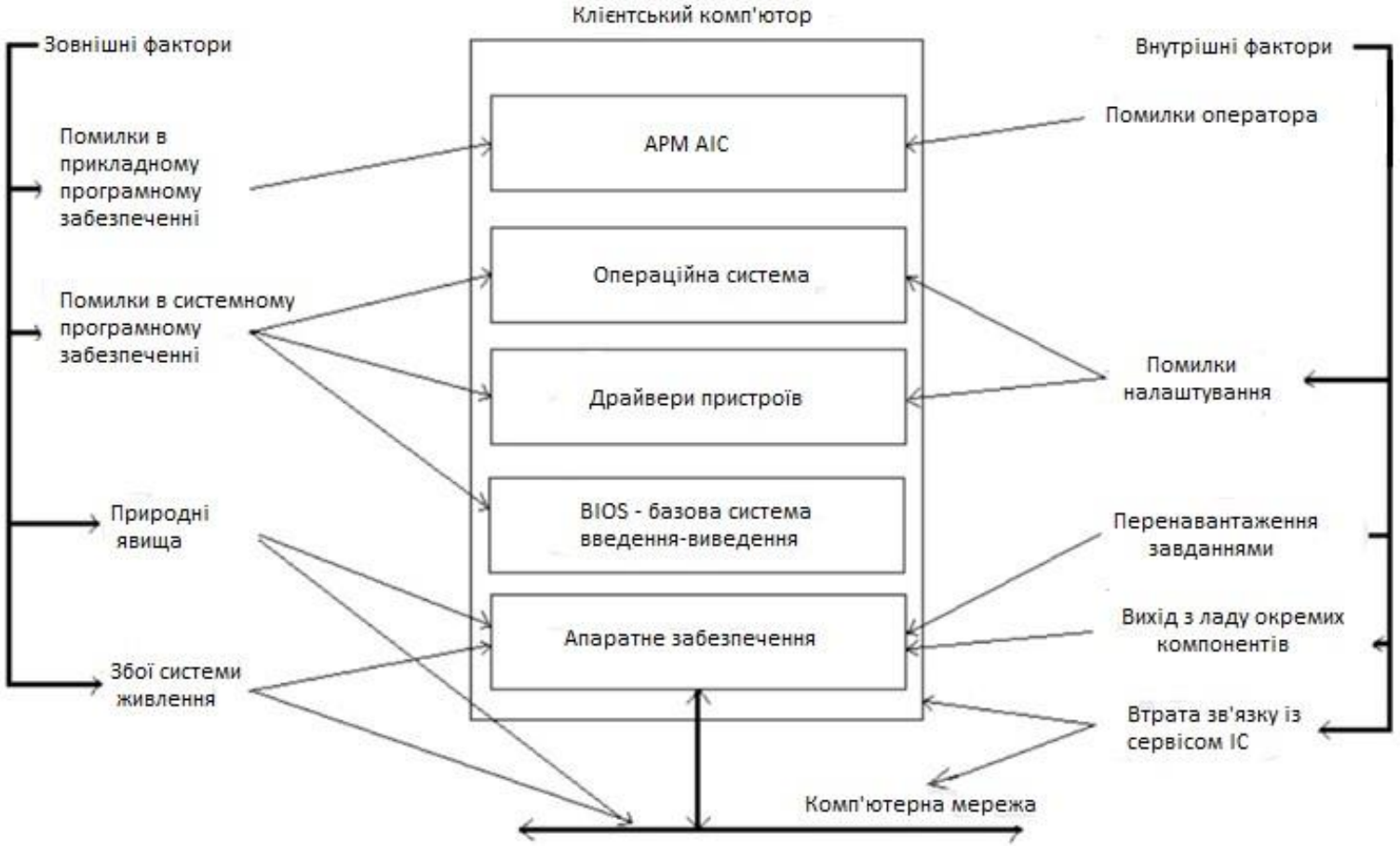
Життєвий цикл надійності інформаційної системи

Граф станів системи забезпечення відмовостійкості інформаційної системи



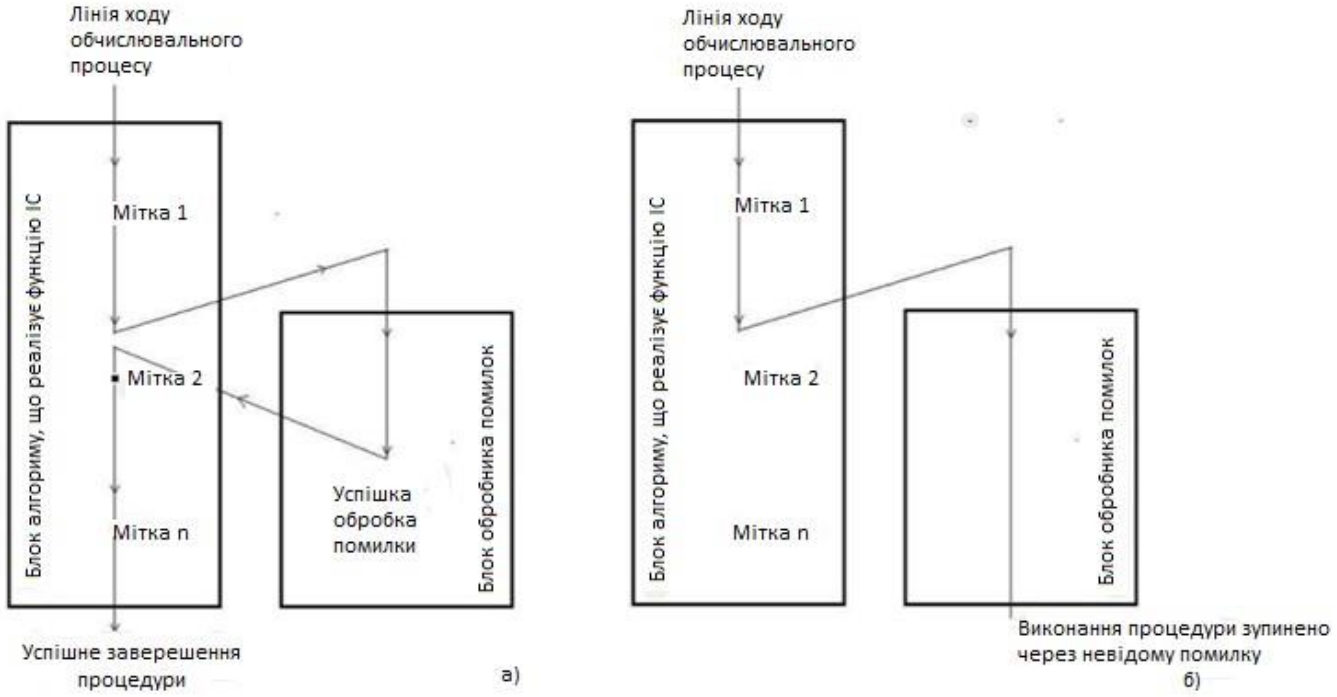
0 - режим підготовки до функціонування ІС;
1 - стан нормального функціонування, відмовостійка робота ІС;
2 - виявлення факту несправності;
3 - локалізація несправності;
4 - аналіз несправності, ідентифікація відмов або збоїв;
5 - виявлення місцезнаходження елемента, що відмовив;
6 - реконфігурація ІС за допомогою резервних модулів;
7 - реконфігурація ІС на основі поступової деградації системи;
8 - повне відновлення обчислювального процесу;
9 - маскування несправностей;
10 - стан ремонту елемента системи, що відмовив;
11 - реінтеграція відремонтованого елемента системи.

Схема дії негативних факторів, що впливають на відмовостійкість клієнтської частини спеціалізованої ІТ

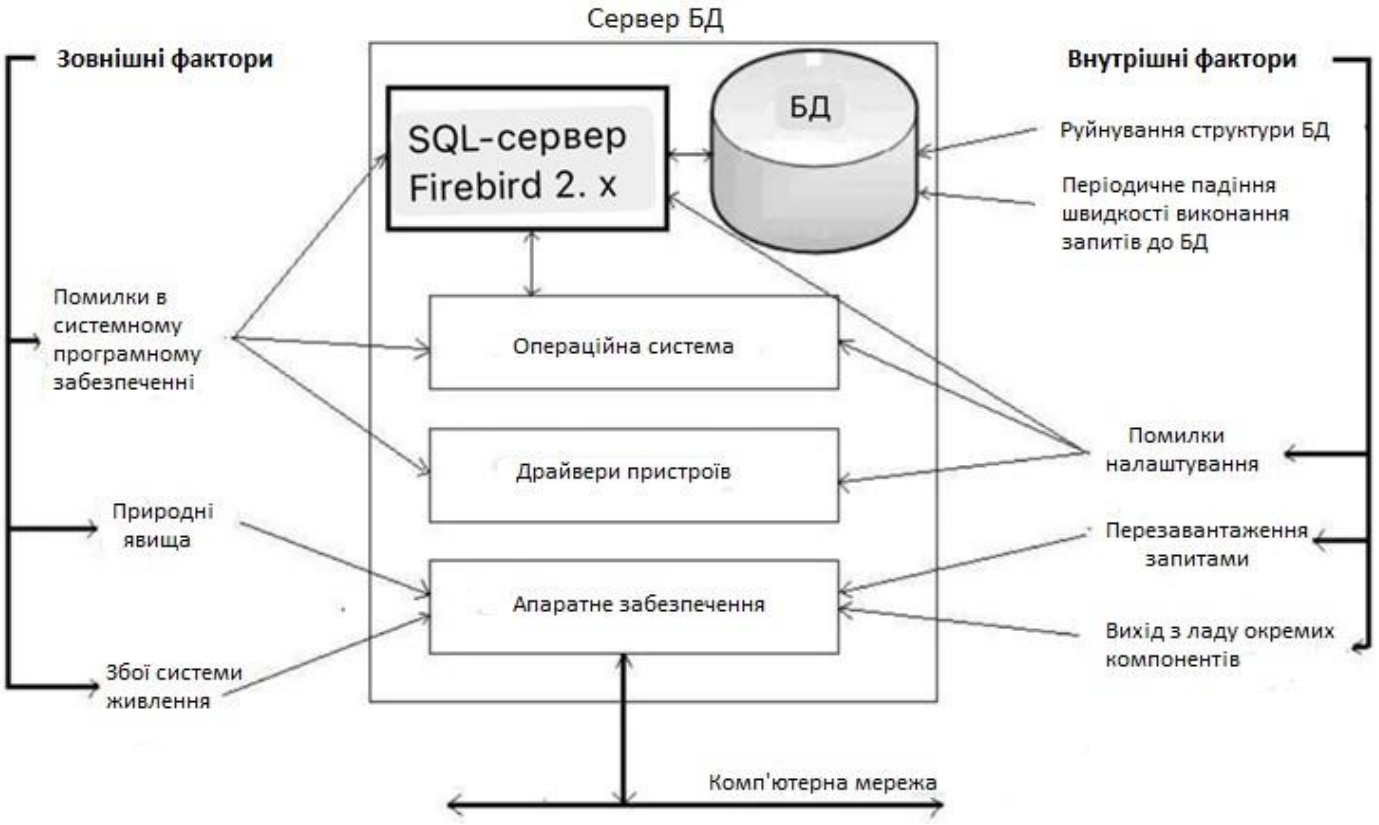


Модель роботи алгоритму відмовостійкої процедури

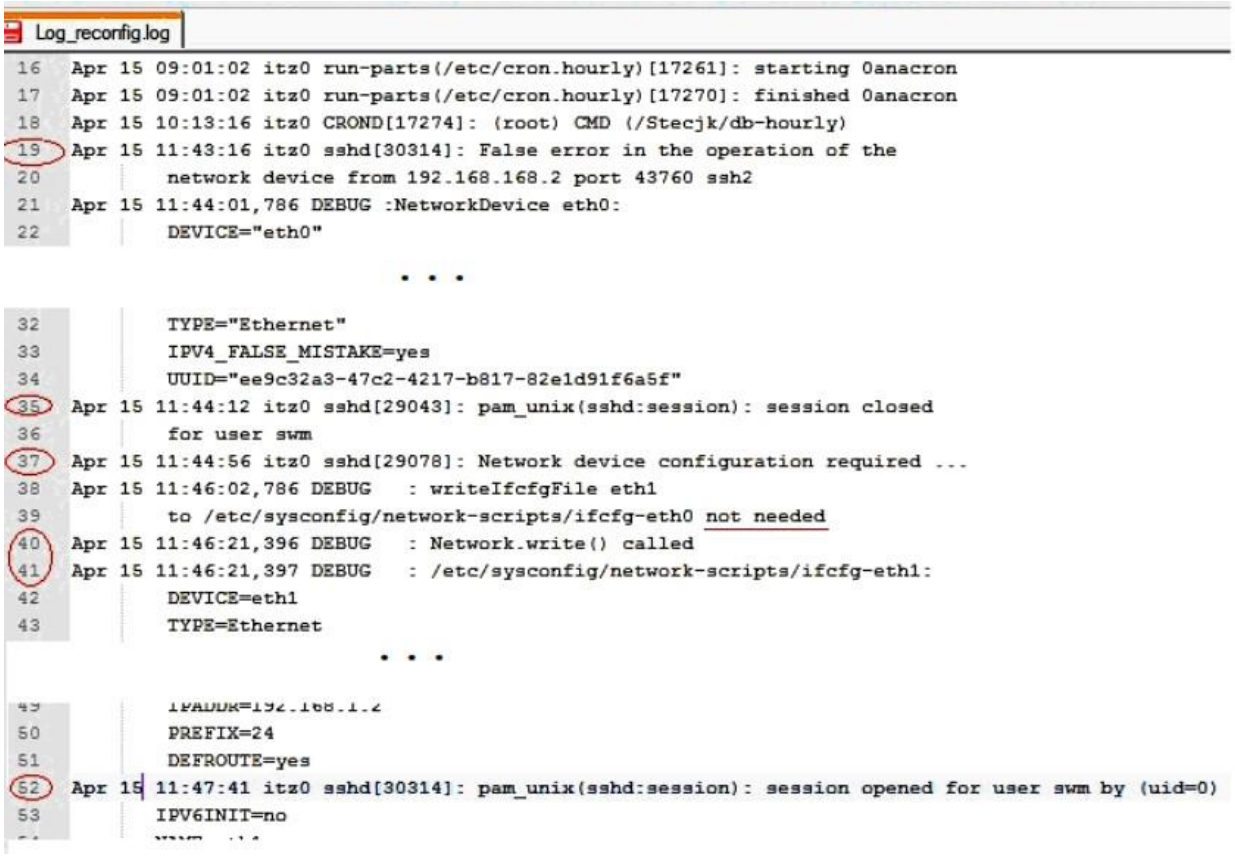
а) для випадку успішного завершення процедури;
б) для випадку, коли помилка невідома для обробника помилок



Модель дії негативних чинників, що впливають на відмовостійкість сервера ІС



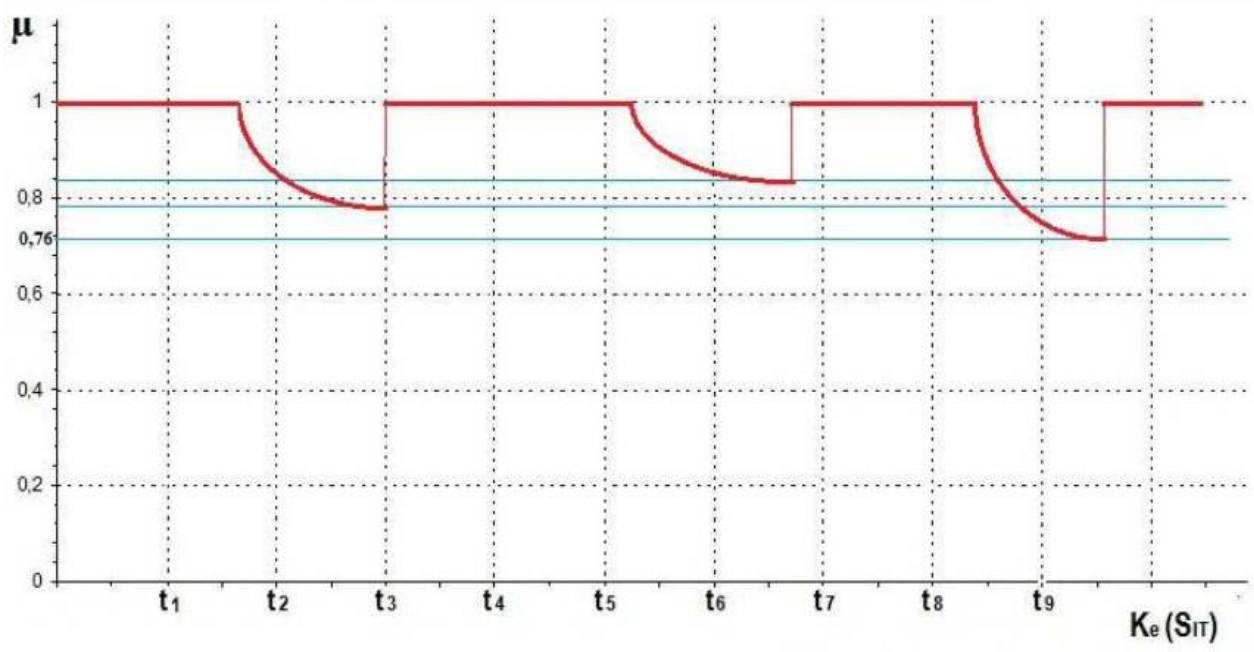
Фрагмент Log-файлу підсистеми контролю роботи мережевих пристроїв



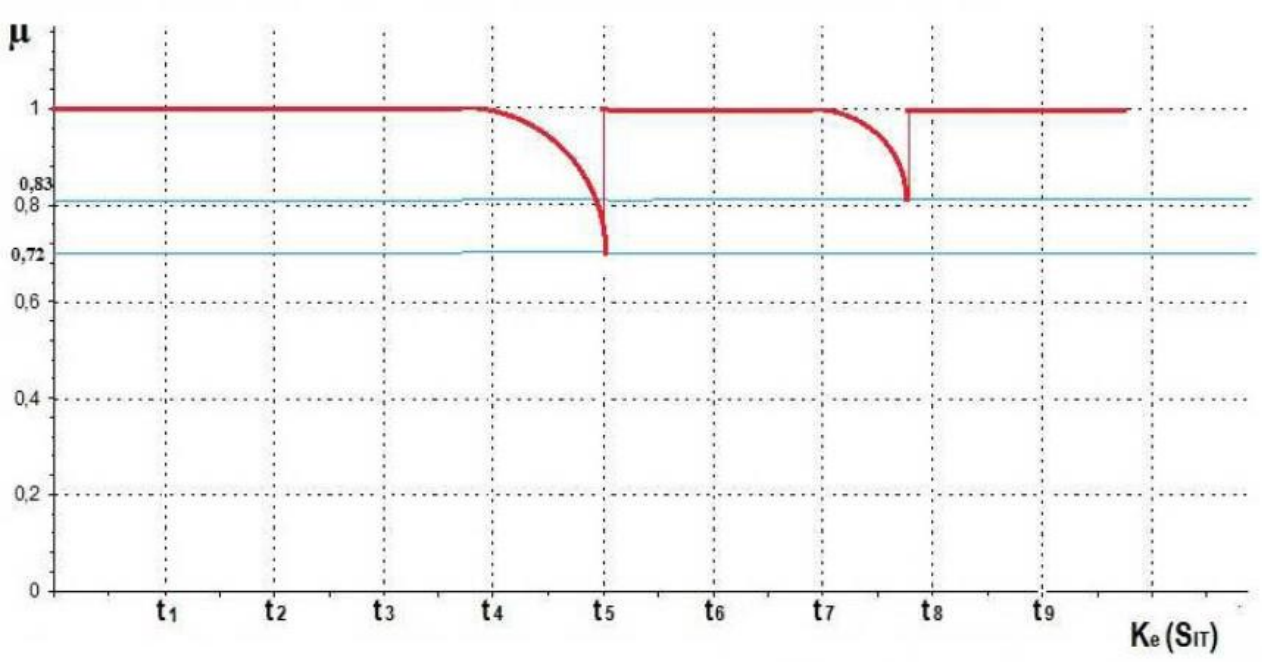
Log_reconfig.log

```
16  Apr 15 09:01:02 itz0 run-parts(/etc/cron.hourly)[17261]: starting 0anacron
17  Apr 15 09:01:02 itz0 run-parts(/etc/cron.hourly)[17270]: finished 0anacron
18  Apr 15 10:13:16 itz0 CROND[17274]: (root) CMD (/Stecjk/db-hourly)
19  Apr 15 11:43:16 itz0 sshd[30314]: False error in the operation of the
20          network device from 192.168.168.2 port 43760 ssh2
21  Apr 15 11:44:01,786 DEBUG :NetworkDevice eth0:
22          DEVICE="eth0"
                              . . .
32          TYPE="Ethernet"
33          IPV4_FALSE_MISTAKE=yes
34          UUID="ee9c32a3-47c2-4217-b817-82e1d91f6a5f"
35  Apr 15 11:44:12 itz0 sshd[29043]: pam_unix(sshd:session): session closed
36          for user swm
37  Apr 15 11:44:56 itz0 sshd[29078]: Network device configuration required ...
38  Apr 15 11:46:02,786 DEBUG   : writeIfcfgFile eth1
39          to /etc/sysconfig/network-scripts/ifcfg-eth0 not needed
40  Apr 15 11:46:21,396 DEBUG   : Network.write() called
41  Apr 15 11:46:21,397 DEBUG   : /etc/sysconfig/network-scripts/ifcfg-eth1:
42          DEVICE=eth1
43          TYPE=Ethernet
                              . . .
49          IPADDR=192.168.1.2
50          PREFIX=24
51          DEFROUTE=yes
52  Apr 15 11:47:41 itz0 sshd[30314]: pam_unix(sshd:session): session opened for user swm by (uid=0)
53          IPV6INIT=no
```

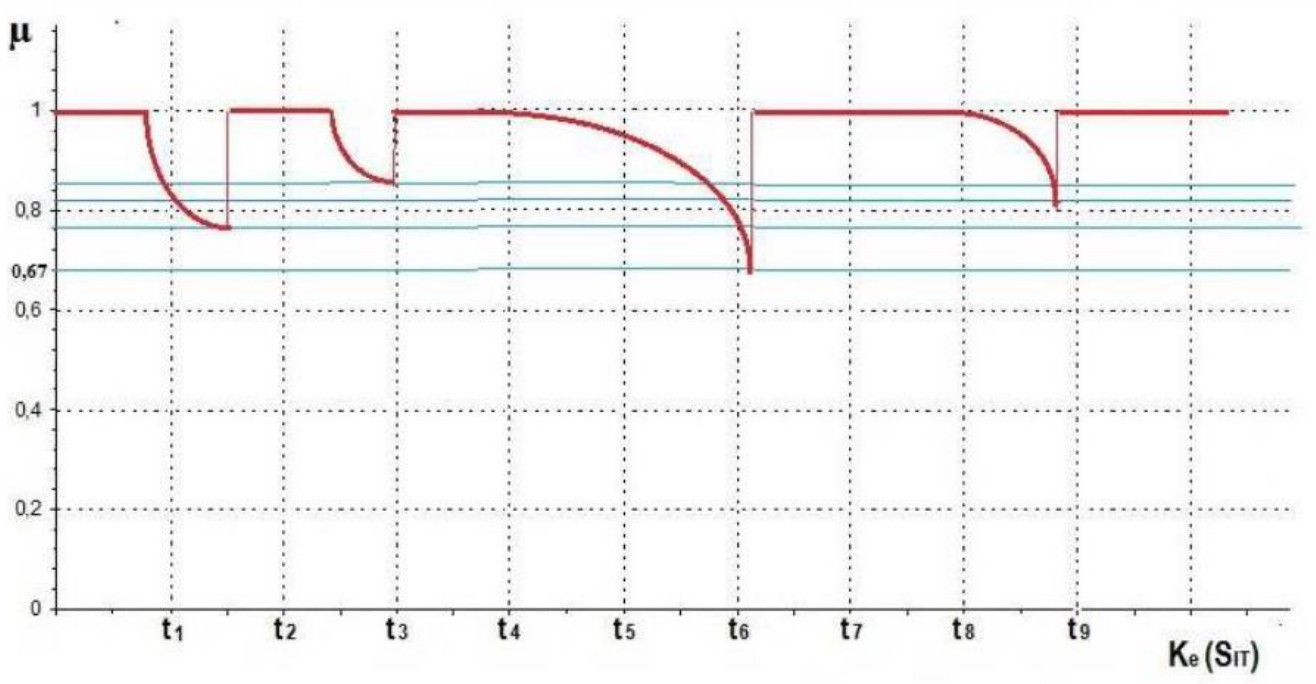
Графік відмовостійкості



Графік проявів живучості



Графік відображення одночасних проявів відмовостійкості та живучості



Висновки

В кваліфікаційній роботі представлено підхід до визначення ефективності ІТ на основі врахування кількісних величин, що характеризують відмовостійкість і живучість, який може бути розширено для врахування інших характеристичних величин. Для забезпечення відмовостійкості та живучості ІТ розроблено систему заходів, у результаті виконання яких отримано ІТ вузькоспеціалізованого використання для різноманітних сфер застосування, де супроводження процесів відносяться до систем ірреального або нереального часу, з достатньо високими параметрами відмовостійкості, живучості та загалом резилентності і, водночас, прийнятними фінансовими витратами на її експлуатацію.