ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти магістр

на тему: **«Імітаційна модель функціонування інформаційної системи в умовах атак на її готовність»**

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи та технології
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти магістр
групи 126ІСТмд_21
Бражник Ю.І.
Керівник: Поночовний Ю.Л.
Рецензент: Брикун О.М.

Полтава – 2023 року

ВСТУП

Актуальність теми. Протягом останніх десятиліть, сервіс-орієнтована архітектура зростає надзвичайно швидко у сфері ІТ та інформаційних систем. Вона стала не лише стандартом для взаємодії різних додатків, а й важливою складовою для численних бізнес-орієнтованих систем, таких як Інтернет-банкінг, онлайн-магазини, системи резервування та продажу туристичних послуг, електронний бізнес та наукові платформи.

Проте сучасні виклики відображаються в плані розробки та впровадження відмовостійких архітектур. Наприклад:

– кількість станів відмовостійких сервісів інформаційних систем зростає, спричиняючи різні відмови через фізичні дефекти у апаратному забезпеченні, проблеми у програмному проєктуванні та атаки на взаємодію вебсервісів;

– проведення процедур забезпечення відмовостійкості враховує вплив їх реалізації на надійність вебсервісів інформаційної системи;

– характеристики SOA та цільових сервісів, які формують композитні SOA, є невизначеними та змінними.

Відомо, що існують недоліки у відомих методах, таких як складність реалізації архітектур для бізнес-критичних інформаційних систем, відсутність уніфікованих процедур забезпечення відмовостійкості та моделей оцінки й забезпечення відмовостійкості інформаційних систем. Останні дослідження, такі як ті, які проводили В. Харченко, А. Горбенко, К. Тріведі, А. Боярчук, зосереджуються на розвитку методів та моделей оцінювання надійності та готовності інформаційних систем. Вони аналізують аналітичні моделі вебсервісів та інших компонентів інформаційних систем з використанням спеціалізованих інструментів моделювання.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в межах науково-дослідної роботи «Розвиток підприємництва: управлінські, економічні, інноваційна та правові аспекти» відповідно до договору №9 від 15.05.2023 р. між ТОВ «ПАФ Гарант» та Полтавським державним аграрним

університетом (розділ «Обґрунтування показників оцінювання гарантоздатності розподілених інформаційних систем»).

Метою кваліфікаційної роботи є оцінювання адекватності результатів аналітичного моделювання та розробка рекомендацій щодо застосування імітаційних моделей інформаційних систем.

Завданнями кваліфікаційної роботи є:

– аналіз існуючих методів і засобів розробки, моделювання, оцінки характеристик і забезпечення надійності та кібербезпеки компонентів інформаційних систем;

– розробка і дослідження марковських моделей для оцінки готовності вебсервісів інформаційних систем;

– розробка і дослідження імітаційних моделей функціонування інформаційних систем з врахуванням атак на їх компоненти.

Об'єктом дослідження є процеси моделювання функціонування інформаційних систем в умовах прояву відмов та атак на їх компоненти.

Предметом дослідження є аналітичні та імітаційні моделі оцінки готовності вебсервісів інформаційних систем.

Методи дослідження – проведені в роботі дослідження базуються на методах теорії ймовірності, системного і марковського аналізу, систем масового обслуговування, які використовувалися при розробці комплексу марковських та імітаційних моделей вебсервісів інформаційних систем.

Інформаційна база кваліфікаційної роботи складається з наукових статей, міжнародних аналітичних видань і звітів, матеріалів наукових конференцій інтернет-ресурсів, що місять інформацію про архітектуру сучасних інформаційних систем, а також даних, отриманих від провідних IT-компаній у сфері надійності та безпеки вебскладових інформаційних систем.

Елементи наукової новизни полягають у розроблені та досліджені аналітичних та імітаційних моделей функціонування вебсервісів інформаційних систем в умовах прояву дефектів апаратних і програмних засобів, атак на компоненти сервісу, проведення оновлень ПЗ, загальних та роздільних обслуговувань.

Практична значущість роботи полягає в можливості повторного застосування та модифікації розробленого програмного коду моделі для оцінювання показників якості обслуговування інформаційних систем з відмовами. Отримані результати також можуть бути корисними для ІТ фахівців при моделюванні спеціалізованих інформаційних управляючих систем.

Апробація результатів дослідження відбувалася шляхом оприлюднення доповідей на наукових конференціях, семінарах.

Публікації. За результатами проведеного дослідження опубліковано тези: «Імітаційна модель для оцінювання безпеки системи з віртуалізованою інфраструктурою з врахуванням атак на компоненти», Матеріали XII Міжнар. наук. конференції «Інформаційні технології в енергетиці та агропромисловому комплексі», м. Львів, 04-06 жовтня 2023 р.; «Застосування методів та технологій для забезпечення безпеки баз даних в сучасному цифровому світі», Матер. науково-практичної конференції за підсумками виробничої практики здобувачів вищої освіти спеціальності «Інформаційні системи та технології», 17 вересня 2023 р., м. Полтава.

Структура та обсяг кваліфікаційної роботи логічно пов'язані з задачами досліджень. Робота містить перелік умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг текстової частини дипломної роботи складає 70 сторінок формату А4. Вона містить 28 рисунків і 6 таблиць. В роботі використано 48 науково-технічних джерел.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗРОБКИ ТА МОДЕЛЮВАННЯ ГОТОВНОСТІ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Архітектури вебсервісів та технології їх застосування в інформаційних системах

У парадигмі сервіс-орієнтованого комп'ютерного моделювання (service-oriented computing, SOC) [1], вебсервіси використовуються для підтримки розробки швидкодіючих, маловитратних, інтероперабельних, еволюційних і розподілених додатків. Вони виконують широкий спектр функцій від простих відповідей на запитання до виконання складних бізнес-процесів, для яких потрібні взаємозв'язки між різними рівнями споживачів і постачальників сервісів.

Ключем для розуміння цієї концепції є сервіс-орієнтована архітектура (service-oriented architecture, SOA). Вона виступає як логічний засіб розробки програмних систем для надання сервісів додатків, орієнтованих на кінцевих користувачів або інших сервісів, що розподіляються в мережі, через публікацію і виявлення інтерфейсів [2]. SOA забезпечує клієнтським організаціям гнучку інфраструктуру та середовище обробки за рахунок підтримки незалежних, повторно використовуваних прикладних функцій у вигляді сервісів та створення надійної основи для використання цих сервісів.

У сучасному світі найбільш перспективною технологією, що базується на парадигмі SOA, є вебсервіси. Вони представляють собою одну з клієнт-серверних технологій, яка дозволяє додаткам взаємодіяти незалежно від платформи та мови програмування, на яких вони розроблені [3]. Фактично, вебсервіс є набором інтерфейсів, що описують певний набір функцій, які можна викликати віддалено по мережі, використовуючи стандартизовані XML повідомлення.

У цій технології Інтернет виступає як комунікаційне середовище, для передачі даних застосовується протокол SOAP (Simple Object Access Protocol), для визначення сервісів – мова WSDL (Web Services Description Language), а для їх

адміністрування – мова BPEL4WS (Business Process Execution Language for Web Services).

Проєкти додатків вебсервісів розробляються з використанням трьох технологій: мови опису вебсервісів (WSDL), протоколу простого доступу до об'єктів (SOAP) і специфікації універсального опису, пошуку і інтеграції (UDDI). SOAP (Simple Object Access Protocol) є стандартом для відправлення та отримання повідомлень через Інтернет [4]. Початково цей протокол був запропонований компанією Microsoft як засіб для віддаленого виклику процедур (RPC) через протокол HTTP, а специфікація SOAP 1.0 (Userland, Microsoft, Developmentor) була тісно пов'язана з Component Object Model. Специфікація SOAP визначає XML-обгортку для передачі повідомлень, метод кодування програмних структур даних у форматі XML, а також засоби зв'язку через протокол HTTP. SOAP-повідомлення можуть бути двох типів: запити (Request) і відповіді (Response). Якщо запит викликає метод віддаленого об'єкта, відповідь повертає результат виконання цього методу. Специфікація SOAP визначає формат кодування, який визначає спосіб представлення даних у форматі XML. Для того, щоб програми могли використовувати вебсервіси, програмні інтерфейси останніх мають бути детально описані – з цієї точки зору мова WSDL відіграє ту ж роль, що і мова Interface Definition Language (IDL) у розподілених обчисленнях. Опис може включати таку інформацію, як протокол, адресу сервера, номер використовуваного порту, список доступних операцій, формат запиту і відповіді тощо. Для опису цієї інформації було запропоновано кілька мов. Однією з них є мова Service Description Language (SDL), розроблена компанією Microsoft і включена до першої версії Microsoft SOAP Toolkit. Коли компанія IBM переробила специфікацію і, використовуючи специфікацію Network Accessible Service Specification Language (NASSL), випустила NASSL Toolkit як частину SOAP4J, ідеї, реалізовані в NASSL, вплинули на специфікацію мови SOAP Contract Language (SCL), запропоновану Microsoft. Наразі обидві специфікації (NASSL і SDL/SCL), а також пропозиції інших компаній враховані у специфікації мови WSDL.

Завдання UDDI (Universal Description, Discovery and Integration) – надати механізм для виявлення вебсервісів. UDDI задає бізнес-реєстр, в якому провайдери вебсервісів можуть реєструвати сервіси, а розробники – шукати необхідні послуги. Компанії IBM, Microsoft і Ariba створили власні UDDI-реєстри, в кожному з яких розробники можуть зареєструвати свої вебсервіси, після чого дані автоматично реплікуються в інші реєстри.

Мова BPEL (BPEL4WS, Business Process Execution Language for Web-Services) є моделлю і граматикою опису поведінки бізнес-процесів на основі взаємодії процесу (координатора) і його партнерів. Взаємодія з кожним партнером здійснюється за допомогою вебсервісів.

Процес, що визначається в BPEL4WS, складається з ряду ключових елементів:

– операції, які визначають окремі етапи бізнесу всередині цього процесу;

– посилання на партнерів, що визначають зовнішні сутності, що взаємодіють з вибраним процесом чи використовують WSDL-інтерфейси;

– змінні, які зберігають повідомлення, передані між операціями, і тим самим відображають стан процесу;

– кореляційні набори, використовувані для співставлення кількох запитів та відповідей із конкретним випадком бізнес-процесу;

– обробник виключень, що вирішує несподівані ситуації, які можуть виникнути під час роботи бізнес-процесу;

– обробник подій, що приймає та опрацьовує повідомлення паралельно зі звичайним виконанням процесу;

– коригувальні обробники, які реалізують логіку виправлення для відміни дій або кількох операцій у разі виникнення виняткових ситуацій [5].

Процес, визначений в BPEL4WS, складається з дій, посилань на партнерів, змінних, кореляційних наборів, обробників несправностей, обробників подій і коригувальних оброблювачів [6]. Отже, мова BPEL4WS є моделлю для опису взаємодії процесів та їх партнерів через вебсервіси, а описаний процес включає різноманітні компоненти для впорядкування бізнес-процесів [7].

1.2 Аналіз засобів для розробки вебкомпонентів інформаційної системи

Серед багатьох комерційних програмних продуктів, які використовуються для розробки вебсервісів, варто розглянути інструменти від Borland, HP, IBM, Microsoft, Oracle та Sun.

Borland є піонером у розробці засобів для вебсервісів на різних платформах. Їх Borland e-business platform створює основу для повноцінної інформаційної інфраструктури сучасного підприємства. Це включає Borland Enterprise Server, який базується на промислових стандартах CORBA 2.4 і J2EE 1.3, Borland Enterprise Server AppServer Edition для розробки EJB-компонентів і Borland Enterprise Server Web Edition з використанням Apache і Tomcat.

Hewlett Packard займає власну нішу на ринку засобів для вебсервісів порівняно з IBM і Sun. Вони мають апаратні та програмні платформи для виконання і управління вебсервісами. HP Service Composer дозволяє розробникам створювати WSDL-інтерфейси для Java-об'єктів і автоматично впроваджувати їх на HP Application Server.

IBM пропонує дві лінійки продуктів, що цікаві для розробників вебсервісів: WebSphere Studio і WebSphere Application Server.

Microsoft пропонує .NET Framework та набір корпоративних серверних додатків (.NET Enterprise Servers) як платформу для вебсервісів. Visual Studio .NET є основним інструментом для створення .NET-сервісів, підтримуючи мови програмування, такі як Visual Basic, C # і J #. Однак остання, хоч і формально підтримує синтаксис мови Java, не дозволяє створювати стандартні Java-додатки – код буде працювати тільки під керуванням Microsoft .NET.

Oracle має два підходи до створення та використання вебсервісів: спочатку, вони надають програмну інфраструктуру для розробників, яку можна використовувати при створенні вебсервісів; подруге, вони самі розробляють та продають програмні продукти у вигляді вебсервісів [6]. Група засобів розробки Oracle 9i представляє собою комплект інтегрованих програмних продуктів для створення та запуску вебсервісів. Цей набір складається з трьох основних

компонентів: Oracle 9i JDeveloper (для розробки вебсервісів), Oracle 9i Application Server (для виконання вебсервісів) і Oracle 9i Database (для створення додатків, які працюють з даними і можуть бути використані разом з вебсервісами).

Основний продукт від компанії Sun – Sun ONE – включає в себе архітектуру, платформу та набір інструментів для створення та впровадження вебсервісів, які в термінах Sun називаються «Services on Demand» [7]. Платформа Sun ONE базується на таких основних компонентах, як операційна система Solaris, платформа Java 2 Platform, набір серверів сімейства iPlanet і засоби розробки Forte Development Tools. Java є ключовою технологією в компанії Sun і становить основу більшості їх продуктів і сервісів. Згідно з поглядом компанії Sun, додатки, які виконують функції вебсервісів, повинні бути написані мовою Java.

1.3 Застосування мікросервісів у інформаційних системах

Існує багато визначень мікрослужб, які можна знайти в Інтернеті разом з корисними ресурсами, що містять різноманітні точки зору на цю тему. Проте є загальні особливості, які характеризують мікрослужби:

- ізоляція сценаріїв клієнтів або бізнес-сценаріїв;
- розробка малими командами програмістів;
- можливість написання коду на будь-якій мові програмування та використання будь-якої платформи;
- компоненти включають код та при необхідності стани, що можуть бути розгорнуті та масштабовані незалежно, а також керувати їх версіями незалежно один від одного;
- взаємодія з іншими мікрослужбами за допомогою чітко визначених інтерфейсів та протоколів;
- наявність унікальних ідентифікаторів (URL-адрес), що дозволяють визначати їх місцезнаходження;
- забезпечення стабільності роботи та доступності у разі збоїв.

Використання принципу монолітності при розробці додатків має свої переваги. Часто такі програми вимагають менше зусиль на етапі розробки, а взаємодія між їх компонентами відбувається швидше завдяки механізму взаємодії між процесами. Крім того, всі користувачі можуть тестувати один продукт, що сприяє ефективному використанню людських ресурсів. Однак цей підхід має свої обмеження: тісний зв'язок компонентів додатка в рамках рівнів та обмежена можливість масштабування окремих компонентів. Також, для виконання виправлень або оновлень потрібно чекати завершення тестування інших компонентів, що негативно впливає на гнучкість.

Мікрослужби дозволяють подолати ці недоліки та досягти більш точної відповідності бізнес-вимогам. Проте вони мають як переваги, так і недоліки. Серед переваг мікрослужб – це включення простих бізнес-функцій у кожну з них та можливість незалежного вертикального масштабування, тестування, розгортання і управління. Основна перевага полягає в тому, що команди розробників акцентують увагу на бізнес-сценаріях, а не на конкретних технологіях, що відзначається при розробці складних додатків. На практиці, невеликі команди розробляють мікрослужби, спираючись на сценарії клієнта та використовуючи технології за власним вибором. Іншими словами, організаціям не потрібно стандартизувати технології для підтримки своїх мікрослужб. Окремі команди, що відповідають за служби, можуть робити це за своїм бажанням, виходячи зі своїх знань або відповідно до потреб задачі. У практиці, краще користуватися набором рекомендованих технологій, наприклад, при обранні певного сховища NoSQL чи вебплатформи.

Недоліки мікрослужб проявляються в управлінні зростаючою кількістю окремих сутностей, складному розгортанні та керуванні версіями. Обсяг мережевого трафіку між мікросервісами збільшується, що призводить до мережевих затримок. Велика кількість «вузькопрофільних» служб може спричинити серйозне зниження продуктивності. Відсутність засобів виявлення залежностей ускладнює можливість огляду системи в цілому.

Мікрослужби стають корисними завдяки застосуванню стандартів, а не жорстких контрактів, які потрібно строго дотримуватися. Тут важливі правила обміну даними, що враховують лише завдання служби. Тому при розробці таких контрактів вони повинні бути чітко визначені наперед, оскільки служби оновлюються незалежно одна від одної. Ще одна важлива характеристика розробки додатків на мікрослужбах – це дрібномодульна сервісно-орієнтована архітектура (SOA). Фактично, при створенні застосунків на мікрослужбах використовуються незалежні фабрики служб з можливістю окремих змін у кожній з них та узгоджені стандарти для обміну даними. З розвитком хмарних додатків люди усвідомлюють, що декомпозиція єдиної програми на окремі, сценарієорієнтовані служби – це оптимальний підхід у довгостроковій перспективі (рис. 1.1).

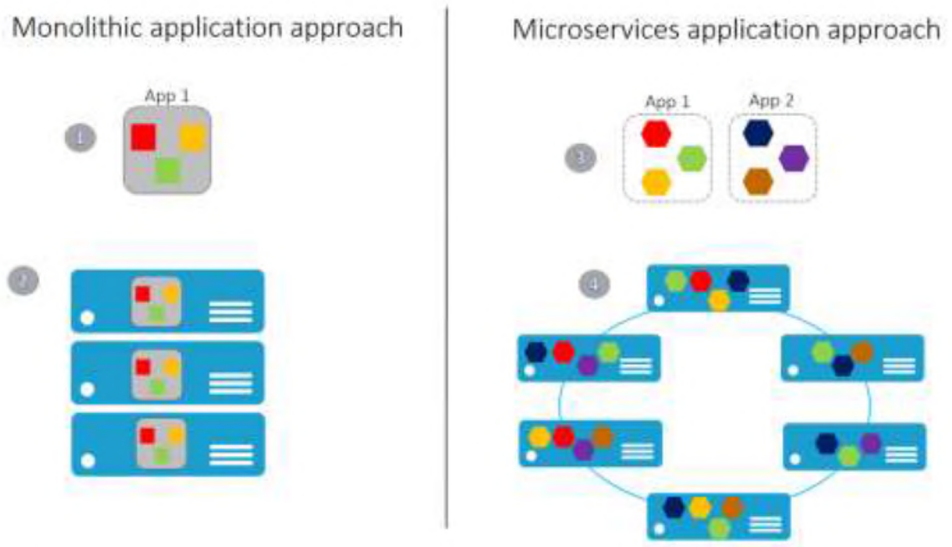


Рисунок 1.1 – Порівняння підходів до розробки додатків

При розробці вебдодатків можна використати наступні підходи (рис. 1.1):

1. Монолітні програми обладнані функціональними можливостями конкретної сфери та зазвичай структуровані за рівнями функціональності, такими як мережа, бізнес-логіка та дані.

2. Такі програми можна розширити шляхом їх розміщення на декількох серверах, віртуальних машинах або у різних контейнерах шляхом їх клонування.

3. У додатках, що складаються з мікрослужб, функціональні можливості розділені між окремими службами меншого розміру.

4. Використання мікрослужб дозволяє розширювати додатки, розгортаючи кожну службу незалежно від інших, створюючи екземпляри цих служб на різних серверах, віртуальних машинах та різних контейнерах.

Використання мікрослужб при розробці – не універсальний підхід. Але він дозволяє досягти більш точної відповідності бізнес-цілям, описаним вище. Застосувати монолітний підхід можна, якщо відомо, що пізніше у вас не буде можливості переробити код для впровадження мікрослужб. Але найчастіше розробники спочатку створюють монолітний додаток, а потім повільно і поетапно поділяють його, починаючи з функціональних областей, в яких слід збільшити масштабованість і гнучкість.

Таким чином, підхід з використанням мікрослужб є збіркою додатку з декількох дрібних служб. Ці служби виконуються в контейнерах, розгорнутих в кластерах комп'ютерів. Невеликі команди розробляють службу, орієнтовану на сценарії. При цьому тестування, управління версіями, розгортання і масштабування виконується для кожної служби окремо, дозволяючи додатку розвиватися як єдине ціле.

1.4 Аналіз надійності та кібербезпеки вебкомпонент інформаційної системи

Згідно з основною ідеєю гарантування надійності [8], для комп'ютерних систем загрозами є помилки, несправності і відмови. Помилка – це стан системи, який може спричинити відмову. Відмова відбувається, коли помилка впливає на інтерфейс системи та змінює системну послугу. Несправність – це реальна або потенційна причина помилки, яка може бути активною або прихованою. Несправності системи поділяються на несправності розробки, фізичні несправності і несправності взаємодії.

Несправності внаслідок злого наміру можуть бути розділені на ті, які спричинені руйнівною логікою (троянські коні, віруси, черв'яки) і вторгнення.

Зловмисник може використовувати внутрішні несправності системи або зовнішні впливи (відключення живлення, перехоплення трафіку) для здійснення вторгнень. В цьому випадку особливо цікаво моделювати наявність несправностей у системі через вторгнення. Табл. 1.1 містить потенційні загрози для системи безпеки і способи забезпечення безпечної роботи проєктованої системи.

Таблиця 1.1 – Потенційні загрози для системи безпеки

Методи забезпечення кібербезпеки	Розрізнення прав доступу на рівні					Шифрування даних			Інші методи криптографії	
Порушення системи безпеки	ОС	Web-сервера	СУБД	Мови сценаріїв	Простори документів серверів	Асиметричні	Симетричні	Комбіноване	Сертифікати VeriSign	Електронний цифровий підпис
Несанкціонований доступ до ресурсів	*									
Відсутність доступу	*									
Порушення конфіденційності							*			
Порушення цілісності									*	

Три основні загрози для вебслужб і XML-ресурсів полягають у фальсифікації ідентифікаційної інформації, спотворенні контенту та операційних атаках [9]. Розглянемо кожну з них у більш детально.

1. Фальсифікація ідентифікаційної інформації може статися під час обміну даними через протокол HTTP. Хоча вебслужби не обмежені цим протоколом, багато трафіку SOAP передається саме ним. WSS 1.0 має набір стандартів для авторизації та аутентифікації для захисту від таких атак. Технології безпеки XML, такі як XML-Encryption і XML-Signature, також допомагають захистити від атак на рівні додатків. Протоколи транспортного рівня, такі як SSL для HTTP і TLS для SMTP, знижують ймовірність «підслуховування», але вони не завжди підходять для вебслужб через проміжні вузли SOAP. Для уникнення цього, процеси шифрування та перевірки ідентичності мають відбуватися на рівні додатків.

2. Контент-атака спрямована на сервер з метою несанкціонованих дій, таких як отримання, знищення або маніпуляція даними або SOAP-повідомленнями. Це призводить до використання надмірних ресурсів сервера та його нереагування на запити.

3. Операційні атаки мають на меті відмову в обслуговуванні, викликану маніпуляцією XML-повідомлень. Наприклад, спроба аналізу XML-документа, який містить рекурсивні описи XML-об'єктів або великий обсяг даних, може призвести до вичерпання пам'яті або значного уповільнення роботи сервера через велику кількість обробки даних [10].

Захист від таких атак вимагає попередньої перевірки вхідних SOAP-повідомлень на відповідність встановленій схемі. Це включає перевірку типів даних та відповідність максимальній/мінімальній довжині строкових даних.

У системах сервіс-орієнтованої архітектури можуть виникати відмови з різних джерел:

- апаратне забезпечення;
- програмне забезпечення (операційна система, середовище додатків: вебсервер, сервер додатків, СУБД);
- конкретні додатки вебсервісів (сервлети, CGI-додатки, збережені процедури та тригери СУБД).

Враховуючи критерії таксономії (залежність відмов, джерела відмов, сталість прояву, вплив на систему та її компоненти, процедури відновлення), побудовано граф залежностей відмов системи (Додаток А).

Оцінюючи вплив відмов F на роботу системи, визначено унікальні групи відмов {F1, ..., F8}. Згідно графу, існують унікальні ланцюжки процедур відновлення функціонування системи. Множина процедур R включає в себе вісім процедур, згрупованих за логікою функціонування: R = {R1, ..., R8}.

Аналізуючи ці множини, можна вважати, що існує чітка взаємодія між вісьмома унікальними групами відмов і вісьмома ланцюжками процедур відновлення після відповідних відмов: F1 – R1; F2 – R2; F3 – R3; F4 – R8; F5 – R4; F6 – R5; F7 – R6; F8 – R7. На основі цього аналізу можна припустити, що після

виникнення відмови F4 (помилка програми), систему можна відновити за допомогою процедури R8 (без спеціального методу), яка передбачає відновлення системи залежно від типу помилки програми за певним алгоритмом.

1.5 Аналіз математичного апарату для моделювання вебкомпонент інформаційних систем

Деякі методи забезпечення гарантоздатності систем вебсервісів інформаційних систем були представлені у дослідженнях вітчизняних та зарубіжних авторів [11, 12]. Однак сучасний план створення та впровадження стійких архітектур відображає нові виклики у сфері моделювання та оцінки надійності та кібербезпеки:

– у стійких сервісах збільшується кількість станів системи внаслідок різних видів відмов: фізичних дефектів обладнання, дефектів у програмному забезпеченні та проблем у взаємодії вебсервісів через різні атаки;

– кожна процедура забезпечення стійкості враховує вплив засобів їх реалізації на надійність вебсервісів;

– характеристики SOA і цільових сервісів, які формують композитні SOA, залишаються невизначеними і змінними.

Існують кілька підходів до моделювання вебсервісів, які базуються на експериментах з реальними сервісами [13], методі Монте-Карло [14], аналітичних методах дослідження Байєсових моделей [15] і Марковських систем [16]. Розглянемо різні методи та математичні підходи для оцінки готовності вебсервісів.

У системах масового обслуговування (СМО) обслуговуючі пристрої або канали обслуговування є основою, яка задовольняє вимоги, названі вимогами обслуговування [17]. Теорія СМО визначає об'єкт обслуговування як вимогу, а основне призначення СМО полягає у задоволенні цих вимог. Загалом, вимога – це запит на задоволення певної потреби, а СМО класифікуються за різними критеріями, такими як час очікування у черзі перед обслуговуванням і дисципліна

обслуговування вимог. За кількістю каналів СМО бувають одноканальні (з одним обслуговуючим пристроєм) і багатоканальними (з декількома обслуговуючими пристроями). Багатоканальні системи можуть складатися з обслуговуючих пристроїв як однакової, так і різної продуктивності.

За часом перебування вимог у черзі до початку обслуговування системи діляться на три групи:

1) з необмеженим часом очікування (з чергою); при зайнятості системи заявка надходить в чергу і далі буде виконана;

2) з відмовами (нульове очікування або явні втрати); «відмовлена» заявка знову надходить в систему, щоб її обслужили;

3) змішаного типу (обмежене очікування); є обмеження на довжину черги чи на час перебування заявки в СМО.

У системах з певною дисципліною обслуговування вимога, заставши всі пристрої зайнятими, в залежності від свого пріоритету, або обслуговується позачергово, або стає в чергу. Основними елементами СМО є: вхідний потік вимог, черга вимог, пристрої (канали) що обслуговують, і вихідний потік вимог.

Використання марковських випадкових процесів дозволяє вирішити проблему розмірності в системах подібного типу [18]. Ця проблема виникає через складність СОА, різноманітність факторів, що впливають на їх працездатність та якість обслуговування, а також можливість створення складних систем сервісів із простіших компонентів. Застосування вкладених марковських процесів дозволяє формалізувати структуру вебсистеми та уявити схему впливів на неї.

Суть підходу до розробки математичної моделі полягає в наступному:

– декомпозиція структури до рівня, що забезпечує можливість врахування і оцінки параметрів. В даному випадку можна говорити про два рівні декомпозиції: вертикальна декомпозиція (поділ системи на окремі функціональні модулі) і функціональна декомпозиція (поділ функціонального модуля на його складові структурні елементи);

– проведення структурування впливів;

– здійснення обліку зміни параметрів шляхом використання багатофрагментних марковських моделей.

Таким чином, за допомогою розробки уніфікованих і розширюваних марковських моделей, можна враховати композитність структури, зовнішні впливи і динамічну зміну параметрів.

Слід зазначити, що для аналізу і побудови графіків функції готовності неможливо використовувати алгебраїчні рівняння за причини неергодічності марковського графа системи. Коректне рішення можливо тільки за допомогою апарату диференціальних рівнянь [19].

Для дослідження функціонування вебсервісів в умовах прояву різних впливів запропонований математичний апарат багатофрагментним моделювання [12, 20]. Особливістю багатофрагментним моделювання є збільшення кількості розглянутих станів системи, що в деяких випадках призводить до суттєвої громіздкості моделі. У випадках, коли побудова аналітичної моделі з певних причин ускладнюється, застосовують метод статистичних випробувань або метод Монте-Карло [21, 22]. Ідея методу Монте-Карло полягає в наступному: замість опису випадкового явища за допомогою аналітичних виразів, проводиться розіграш – моделювання випадкового явища процедурою, яка дозволяє отримати випадковий результат. Застосування методу статистичних випробувань виправдано для складних систем, які складаються з великої кількості елементів і в яких випадкові фактори певним чином пов'язані між собою. Крім того, моделювання випадкових явищ методом Монте-Карло проводиться з метою перевірки достовірності результатів, отриманих при застосуванні певного математичного апарату [23].

Основними аспектами, лежать в основі побудови імітаційної моделі досліджуваного процесу є:

– обґрунтування необхідності імітаційного моделювання;

– обґрунтування вибору показника ефективності процесу функціонування системи;

– вибір способу формалізації моделі;

– визначення параметрів і змінних моделі.

Як показує проведений аналіз методів моделювання сервіс-орієнтованих систем, найбільш придатними є методи, засновані на апараті марковських випадкових процесів. Використання апарату марковських або напівмарковських процесів є, кращим підходом, оскільки дозволяє систематизувати сам процес моделювання (визначення множини станів, переходів між ними, інтенсивностей переходів). Однак використання цього підходу в вебсервісах обмежується низкою причин, перш за все відсутністю детальної та уніфікованої технології, яка враховує особливості різних варіантів СОА, їх функціонування при відмовах, викликаних різними збоями.

При цьому виникає проблема розмірності системи, обумовлена наступними факторами:

– складністю розподілених сервіс-орієнтованих систем;

– різноманіттям факторів, що впливають на їх працездатність;

– можливістю композиції складних сервісів з простих.

Для побудови моделі, що адекватно описує дану систему, необхідно виконати операції з регуляризації (побудови регулярного опису) структури моделі і впливів на неї; а також використання апаратів вкладених марковських процесів для опису процесів, що відбуваються «всередині» комплексних станів системи (станів, що включають в себе підмножини інших станів) та імітаційного моделювання.

1.6 Постановка завдання на дослідження

Загальним завданням дослідження є розробка аналітичних та імітаційних моделей вебкомпонент інформаційної системи для оцінювання їх готовності з врахуванням атак на них.

У роботі вирішуються такі часткові задачі:

– аналіз існуючих методів і засобів розробки, моделювання, оцінки характеристик і забезпечення надійності та кібербезпеки інформаційних систем;

– розробка і дослідження марковських моделей для оцінки готовності вебкомпонент інформаційних систем;

– розробка і дослідження імітаційних моделей функціонування вебсервісів з врахуванням атак на них.

1.7 Обґрунтування вибору методів дослідження

Методика досліджень, результати яких викладені в наступних розділах, базується на використанні методології системного підходу при постановці і рішенні загального і часткових завдань дипломної роботи. Це проявляється в наступному:

– у визначенні етапів вирішення поставлених завдань і логічної послідовності їх виконання;

– у виборі адекватного математичного апарату, методів досліджень і їх співвідношення з завданнями окремих етапів.

Загальне завдання вирішуються в два етапи. На першому етапі будується модель функціонування вебсервісів (з втратою і без втрати готовності), розробляються і досліджуються марковські моделі оцінки готовності вебсервісу. На другому етапі розробляється імітаційна модель вебсервісів в умовах атак на них, після чого виконується порівняння отриманих результатів та висновки про доцільність використання марковських та імітаційних моделей вебсервісів.

Висновки до розділу 1

У першому розділі виконано аналіз сучасного стану галузі розробки та використання інформаційних систем з вебкомпонентами. У процесі проведеного аналізу предметної області встановлено, що у зв'язку зі збільшенням складності таких систем та їх вебсервісів і специфікою вимог до них зросла необхідність в

моделюванні, розрахунку і розробці більш надійних систем і компонентів інфраструктур.

Підвищення безвідмовності і готовності вебсервісів шляхом оцінки та вибору варіантів відмовостійких сервіс-орієнтованих архітектур є головним завданням при побудові гарантоздатних SOA-інфраструктур бізнес-критичного застосування.

На основі проведеного аналізу сформульовані загальне завдання, яке полягає у розробці аналітичних та імітаційних моделей вебсервісів інформаційної системи для оцінювання їх готовності з врахуванням атак на них. Загальне завдання декомпозовано на три часткові задачі, дві з яких розглянуті у наступних розділах.

На сьогоднішній день завдання побудови моделей систем вебсервісів не вирішене повною мірою, зокрема, в повному обсязі не враховано фактор взаємодії різних множин функціональних станів системи в різних режимах роботи системи. Відомі кілька підходів до моделювання вебсервісів, засновані на експериментуванні з реальними сервісами, методі Монте-Карло, аналітичних методах на основі баєсовських моделей. Використання цих підходів в вебсервісах обмежується, перш за все відсутністю детальної та уніфікованої техніки, яка враховує особливості варіантів архітектур SOA, їх функціонування при відмовах, викликаних різними помилками.

РОЗДІЛ 2
РОЗРОБКА І ДОСЛІДЖЕННЯ АНАЛІТИЧНИХ МОДЕЛЕЙ ОЦІНКИ ГОТОВНОСТІ ВЕБСЕРВІСІВ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1 Принципи і послідовність розробки марковських моделей вебсервісів

Першим етапом розробки і дослідження моделей вебсервісу є побудова базових моделей функціонування відмовостійких вебсервісів. Новизною даних базових моделей є можливість врахування різних сценаріїв поведінки вебсервісу при відмовах, що дозволяє розробити типові моделі та оцінити показники безвідмовності, готовності економічної ефективності проєктованих сервісів.

Для цього слід визначити стани функціонування вебсервісу, виділити серед них множини станів готовності, активної готовності, відмови, відновлення. Другим кроком буде визначення відповідностей між станами зазначених множин шляхом розмітки переходів між станами (множинами станів) як відображень Ω_{X*Y}: $M_{SX}* \Rightarrow M_{SY}$.

Наступним етапом структурного синтезу моделі є ідентифікація станів, що входять в різні множини, і переходів між цими станами (відображень). Для цього проаналізовано логіку функціонування системи, описавши її поведінку на верхньому рівні декомпозиції. Розглянуто по черзі «вміст» кожного з цих станів, виконавши його вертикальну декомпозицію.

Об'єднавши всі вищеописані частини, отримують модель системи з станами, декомпозованими до зазначеного рівня. У разі надмірної деталізації станів мінімізують модель таким чином, щоб вона залишалася чутливою до потоку вхідної черги, потоку впливів, а також параметри процедур відновлення були збережені. Позначивши переходи між станами, отримують орієнтований граф функціонування системи.

Для розробки і дослідження марковських моделей оцінки надійності вебсервісів здійснено перехід від функціонального графа до марковської моделі,

вершинами якого будуть стани системи, а орієнтованими дугами – переходи між станами. У разі порушення умов марковості проводять необхідну декомпозицію станів. Згідно з результатами раніше проведеного аналізу позначено параметри інтенсивностей переходів між усіма станами. Таким чином, отримують розмічений марковський граф системи з дискретними станами і неперервним часом.

Нарешті, для моделювання та дослідження системи проводять обґрунтування і вибір значень постійних і змінних параметрів інтенсивностей переходів моделі, використовуючи джерела вітчизняної та зарубіжної літератури, результати проведеного експерименту [24] і закономірності – підтверджені [25].

2.2 Модель функціональних (справних) станів вебсервісу інформаційної системи бронювання квитків

Першим етапом розробки (структурного синтезу) моделі є визначення (ідентифікація) станів, що входять в різні множини і переходів між цими станами (відображень). Для цього необхідно проаналізувати логіку функціонування системи. Опишемо її поведінку на верхньому рівні декомпозиції. Дана модель має всього чотири стани, що відображають реакцію системи на потік заявок, що надходить. Розглянемо докладно кожен зі станів.

1. Ініціалізація. Початковий стан системи, в який вона переходить після включення. Під час ініціалізації відбувається старт всіх модулів системи; встановлення з'єднання з БД; формування первинного набору цільових компонент вебсервісу; конфігурація системи; ініціалізація черги клієнтських запитів.

2. Отримання запиту з черги. Другий стан системи після ініціалізації. Всі модулі системи запущені, цільові компоненти вебсервісу вже зареєстровані в системі. Черга заявок порожня і очікує надходження запитів користувачів. У чергу починають надходити перші запити, система відбирає їх і відправляє на декомпозицію.

3. Декомпозиція запитів. Вибрані з черги запити надходять на вхід блоку, що займається декомпозицією запитів на верхньому рівні. Визначається категорія запиту, тобто кількість композитів в ньому (для туристичного агентства – підзапити бронювання авіаквитків, бронювання готелю, оренди автомобіля); з множини зареєстрованих цільових вебсервісів вибираються ті, які можуть задовольнити отримані підзапити.

4. Обробка запитів. На цьому етапі система безпосередньо виконує обробку запитів за допомогою цільових вебсервісів. Відповідно до розглянутої раніше структури композитного сервісу запити, що приходять декомпозуються в WSCA-заявки на кожному рівні «вкладеності» композитних і компонентних серверів, після чого передаються на обробку до цільових сервісів. У разі прийняття користувачем запропонованого результату, система формує відповідь і відсилає користувачеві. В іншому випадку отримані результати відбраковуються і користувачеві повертається повідомлення про неможливість задоволення його запиту. Дана структура ілюструється на рисунку 2.1.

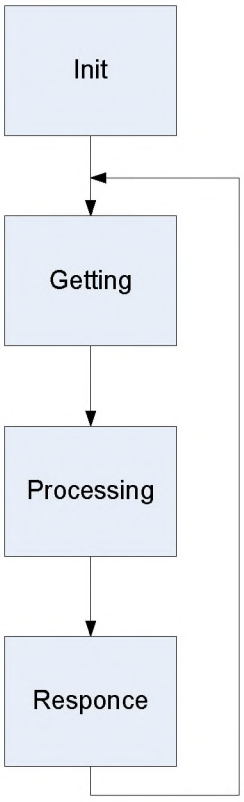


Рисунок 2.1 – Послідовність виконання операцій на верхньому рівні декомпозиції

Розглянемо по черзі «вміст» кожного з цих станів, виконавши їх вертикальну декомпозицію. Стани ініціалізації Init представлені на рисунку 2.2.

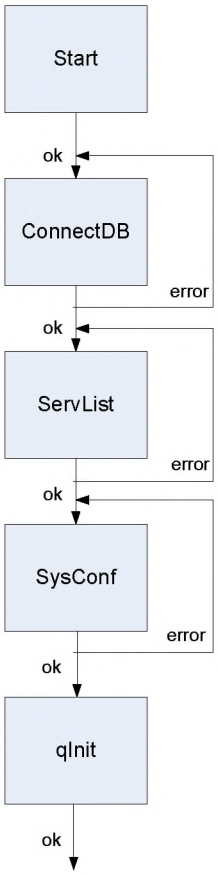


Рисунок 2.2 – Декомпозиція стану «Ініціалізація»

На рисунку 2.2 Start – старт модулів системи; ConnectDB – підключення модулів до СУБД; ServList – формування первинного набору цільових сервісів; SysConf – конфігурація системи з урахуванням заданого режиму її функціонування; QInit – виконання ініціалізації черги вхідних заявок.

У разі успішного виконання кожної дії система послідовно переходить в наступний стан, в разі ж помилкового завершення будь-якого стану система спробує виконати його ще раз. Рисунок 2.3 ілюструє декомпозицію стану Getting отримання клієнтського запиту з черги вхідних заявок. На рисунку 2.3 Wait – стан очікування надходження заявки (циклічна перевірка черги); Get – стан отримання клієнтського запиту шляхом вибірки його з черги; Decompose – декомпозиція запиту (поділ його на компоненти).

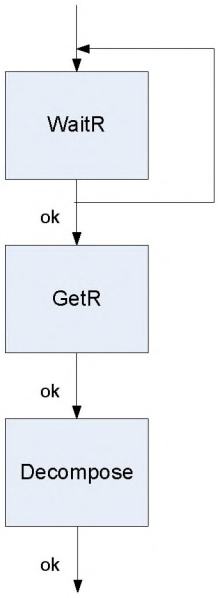


Рисунок 2.3 – Декомпозиція стану «Отримання клієнтського запиту»

Основний функціональний елемент системи – блок обробки клієнтських запитів – представлений на рисунку 2.4. Оскільки безпосередньо виконання операцій по суті запиту не є функцією, що моделюється, обмежимося описом трьох станів в даному блоці – відсилання запиту на обробку, збору результатів і перевірки їх на валідність.

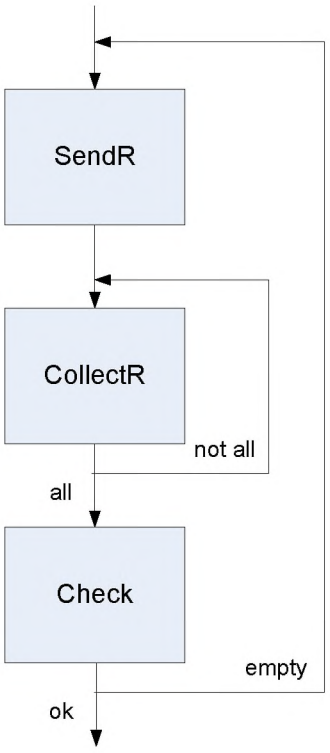


Рисунок 2.4 – Декомпозиція стану «Обробка клієнтського запиту»

На рисунку 2.4 SendR – стан відсилання запитів на обробку цільовим вебсервісам; CollectR – збір результатів виконання запитів; CheckR – перевірка результатів запиту на валідність.

В даному блоці станів можливі переходи трьох типів:

– послідовний перехід в наступний стан в разі успішного виконання операції в поточному стані;

– циклічний повернення в стан збору результатів доти, поки не будуть зібрані результати всіх оброблених запитів;

– повернення в стан повторного відсилання запитів зі стану перевірки запитів на валідність. Даний перехід матиме місце у разі негативного результату перевірки на валідність (результат обробки надійшов, але не може бути відправлений клієнту через наявність некоректних даних).

Заключним модулем моделі сервісу є блок відправки валідного результату на сторону клієнта. Як випливає з рисунку 2.5, зазначений блок складається всього з двох станів – запиту клієнта на прийнятність результатів і відправки схваленого результату клієнтові

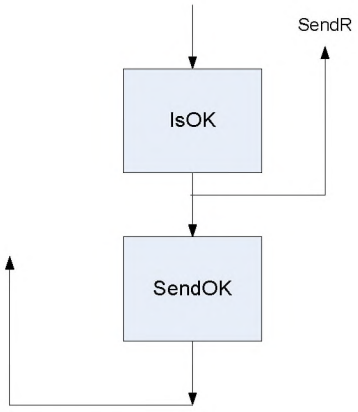


Рисунок 2.5 – Декомпозиція стану «Відправлення результатів»

На рисунку 2.5 IsOk – запит клієнта на прийнятність результату обробки запиту, SendOk – відправка результату клієнтові в разі схвалення. Як видно з рисунку 2.5, в цьому блоці можливі три переходи:

– послідовний перехід зі стану перевірки в стан відправки (в разі схвалення клієнтом отриманого результату);

– повернення до повторного виконання стану SendR – відсилання запиту на обробку – в разі неприйняття клієнтом початкового результату;

– повернення в стан очікування нового запиту WaitR з черги клієнтських запитів.

Об'єднавши всі вищеописані частини, отримуємо модель системи з станами, декомпозованими до зазначеного рівня. Однак, деякі стани є надлишковими, через що модель може стати перевантаженою. Мінімізуємо модель таким чином, щоб виконувалися наступні умови:

– модель залишалася чутливою до потоку вхідної черги;

– модель залишалася чутливою до потоку впливів;

– параметри процедур відновлення були збережені.

Модель з укрупненими станами показана на рисунку 2.6.

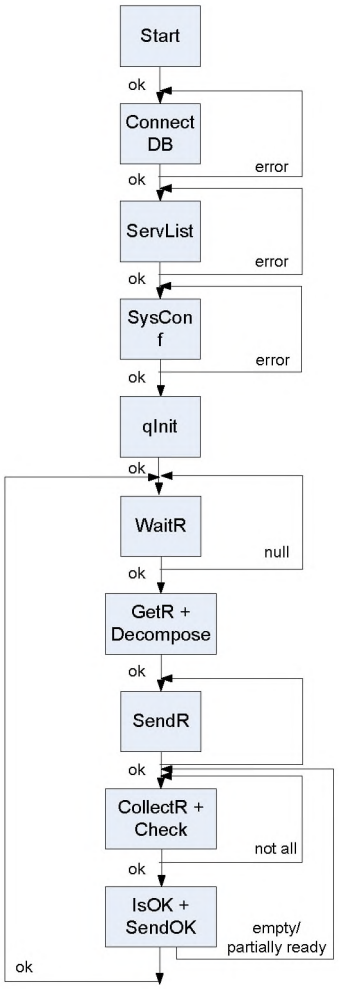


Рисунок 2.6 – Модель функціонування вебсервісу з деталізованими станами

Система (вебсервіс) може бути представлена марковським процесом з дискретними станами і неперервним часом. На рисунку 2.7 показано модель системи у вигляді графа станів, де вершини – стани системи, а орієнтовані дуги – переходи зі стану в стан.

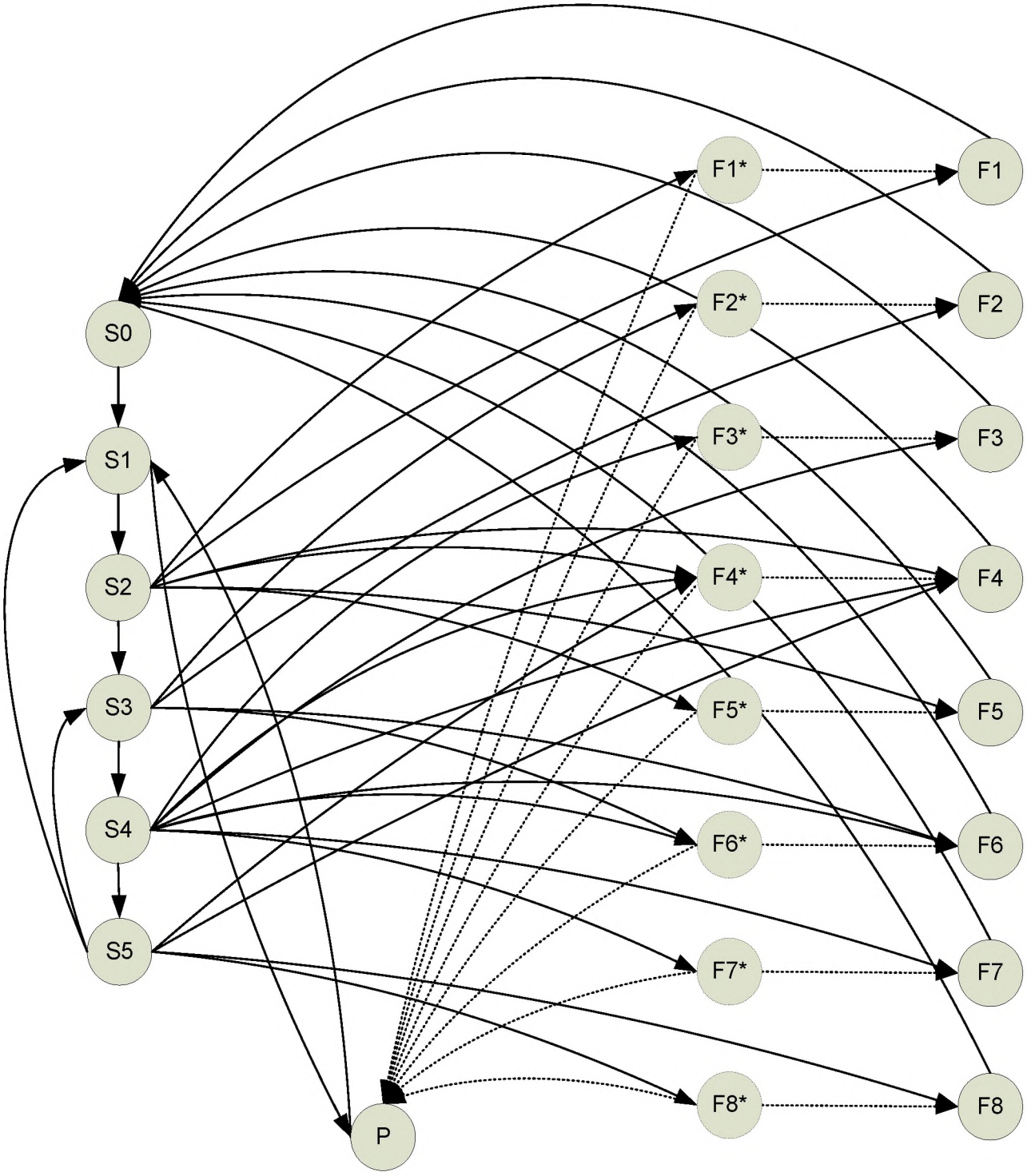


Рисунок 2.7 – Модель вебсервісу як нерозмічений граф станів системи

На рисунку 2.7 показано модель з 24-ма станами:

S0 – виконання ініціалізації черги вхідних заявок;

S1 – очікування надходження заявки (циклічна перевірка черги);

S2 – отримання клієнтського запиту і його декомпозиція;

S3 – відсилання запитів на обробку цільовим вебсервісам;

S4 – збір результатів виконання запитів і перевірка їх на валідність;

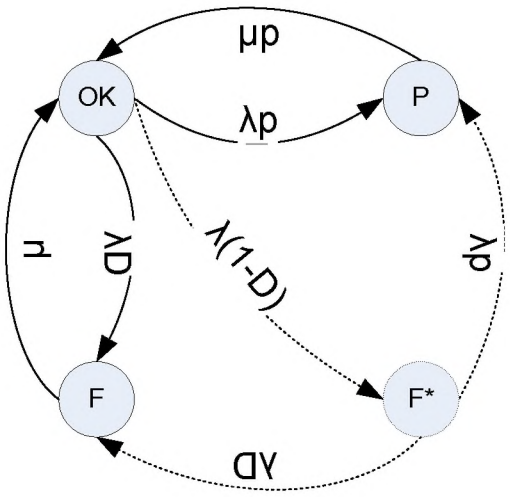
S5 – запит клієнта на прийнятність результату обробки запиту і відправка результату клієнтові в разі схвалення;

P – профілактика системи;

F * – стан прихованої відмови;

F1-F8 – стани виявлених відмов системи;

Для розмітки даного марковського графа слід керуватися правилами розстановки інтенсивностей переходів відповідно до логіки функціонування моделі. Проілюструємо цей підхід на прикладі узагальненого фрагмента графа (рис. 2.8).



OK – працездатний стан;

P – стан профілактики;

F – стан відмови;

λ – інтенсивність переходу.

Рисунок 2.8 – Розмічений фрагмент марковського графа

Як відомо, найпростіший потік характеризується наступними властивостями:

– стаціонарністю;

– ординарністю;

– відсутністю післядії.

Додамо два припущення, що характеризують цільову модель:

– заявки, які прийшли від різних джерел, не залежать одна від одної;

– заявки, які прийшли від одного джерела, залежать тільки в тому випадку, якщо цільовий сервіс не надав послугу, однак імовірність настання даної події надзвичайно мала.

Позначимо параметри інтенсивностей переходів між станами. Так, переходи з працездатного стану ОК в стан профілактики P і назад позначимо як λp і μp відповідно, перехід з ОК в стан прихованої відмови F* позначимо як λ*(1-D), в стан профілактики P як λp, а в стан діагностованої відмови F як λ*D. Нарешті, переходи з ОК в F і назад будуть розмічені λ*D і μ відповідно.

Застосовуючи даний підхід отримаємо наступний розмічений граф (рис. 2.9).

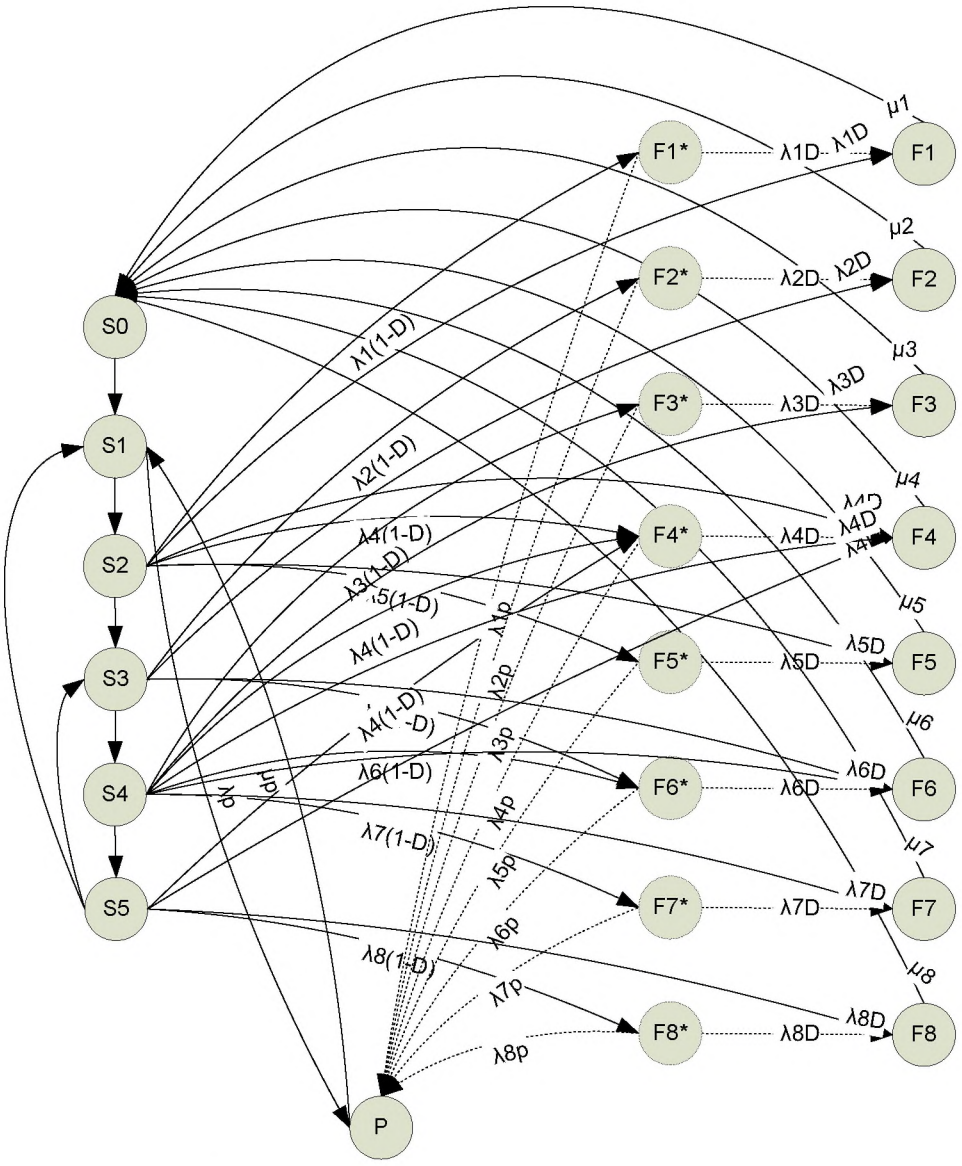


Рисунок 2.9 – Розмічений марковський граф моделі вебсервісу

Для розрахунку показників завантаження вебсервісів приймається інтенсивність запитів в інтервалі від 1 до 100 запитів в секунду [25]. Вони, в свою чергу, виходять з даних лабораторії Hewlett Packard Labs, отриманих при аналізі файлів журналу доступу до сайтів [26]. Таким чином, середнім значенням інтенсивності надходження запитів можна вважати 10 запитів/с або 36000 запитів/годину.

З іншого боку, була проаналізована статистика ресурсу BASIS WS, метою якої став вимір показників гарантоздатності сервіс-орієнтованих систем біомедичного профілю. Згідно зі статистичними даними цієї роботи, максимальна кількість запитів становила 15,74 запитів/с (56664 запитів/годину), мінімальна – 2,16 запитів/с (7776 запитів/годину). Отже, середнє значення інтенсивності запитів користувачів для реальних SOA-систем можна прийняти 8,95 запитів/с (32220 запитів/годину).

У загальному випадку, якщо враховувати відхилення від середнього значення, обумовлені активністю користувачів порталів туристичних агентств в залежності від часу доби (найбільша інтенсивність надходження запитів спостерігається в 15-16 годин дня, найменша – 4-5 годин ранку), загальноприйнятих сезонів відпусток (серпень-вересень та січень), а також виходу реклами туристичних акцій («гарячі путівки») і пропозицій промо-турів, можна визначити діапазон змін інтенсивності в межах 100 – 100000 запитів/годину.

Що стосується інтенсивностей переходу системи в стан збою і вторгнень, то тут слід керуватися наступними положеннями. Для кожного сервера, що входить в кластерну систему, інтенсивність збою можна прийняти за 0.001 1/год (тобто 10^{-3}), а інтенсивність відмови – 0.00001 1/год (тобто 10^{-5}). Інтенсивність збою, викликаного програмним забезпеченням, можна вважати на порядок більше, ніж інтенсивність збою апаратного забезпечення системи, тобто 10^{-2} 1/год, а інтенсивність відмов, відповідно, 10^{-4} 1/год. Крім цього, прийнято, що інтенсивність збоїв ОС може бути в два рази вище, ніж інтенсивність збоїв системного ПЗ (СУБД, вебсервери і т.д.), а інтенсивність збоїв прикладного

програмного забезпечення може бути в три рази вище інтенсивності збоїв системного ПЗ [22].

Для розрахунку інтенсивностей вторгнень можна застосувати таке положення: інтенсивність вторгнень в початковий момент часу приблизно становить 0,01–0,1% від потоку вхідних заявок; в разі знаходження зловмисником потенційної уразливості, потік вторгнень, що атакують знайдену уразливість, може скласти до 1% від потоку вхідних запитів [25].

Для дослідження поведінки системи при різних значеннях вхідних параметрів, в моделі були прийняті наступні постійні вихідні дані, представлені в таблиці 2.1.

Таблиця 2.1 – Постійні значення параметрів марковської моделі вебсервісу

λ	Значення (1/годину)	μ	Значення (1/годину)
λ_1	10^{-5}	μ_1	0,5
λ_2	10^{-4}	μ_2	1,429
λ_3	$5*10^{-3}$	μ_3	8,571
λ_4	$5*10^{-4}$	μ_4	2,4
λ_5	$9*10^{-4}$	μ_5	15
λ_6	$3*10^{-3}$	μ_6	6
λ_7	$2*10^{-3}$	μ_7	20
λ_8	$9*10^{-4}$	μ_8	0,6

Далі, для ряду параметрів були прийняті інтервальні оцінки, що дозволило дослідити динаміку поведінки системи. Межі зміни цих параметрів і їх значення за замовчуванням вказані в таблиці 2.2.

Таблиця 2.2 – Змінні значення параметрів марковської моделі вебсервісу

Параметр	Інтервал зміни	Значення за замовчуванням
$\rho_1 - \rho_7$	0,1…100000	32000 (1/годин)
D	0…1	0,8
μ_p	0,3…0,6	0,5 (1/годин)
λ_p	0,00595…1	0,00595 (1/годин)

Інтенсивності обробки заявок вебсервісом ρ_1-ρ_7 при відсутності затримок в обслуговуванні рівні. Інтенсивності профілактики λ_p і μ_p розраховуються наступним чином: μp = MIN ($\mu 1$ ... $\mu 8$) = $\mu 1$ = 0,5 (1/год); λp буде залежати від

частоти проведення профілактики, наприклад: профілактика проводиться раз в тиждень – Тр = 7*24 = 168 годин, λр = 0,00595 (1/год); профілактика проводиться раз на два дні – Тр=2*24=48 годин, λр=0,02083 (1/год); профілактика проводиться раз на добу – Тр = 24 години, λр = 0,0417 (1/год).

2.3 Аналіз марковської моделі функціонування вебсервісів інформаційної системи

Модифікований граф (рис. 2.10) отримано з вихідного (рис. 2.9) шляхом внесення наступних змін:

– введена наскрізна нумерація станів: таким чином, непрацездатні стани – це стану прихованих відмов (S14 – S21), стани явних відмов (S6 – S13), і стан профілактики – S22;

– розмічені переходи між справними станами як ρ_1 – ρ_7;

– додані прямі переходи з справних станів в стани явних відмов;

– інтенсивності переходів із станів прихованих відмов в стан профілактики розмічені як λр.

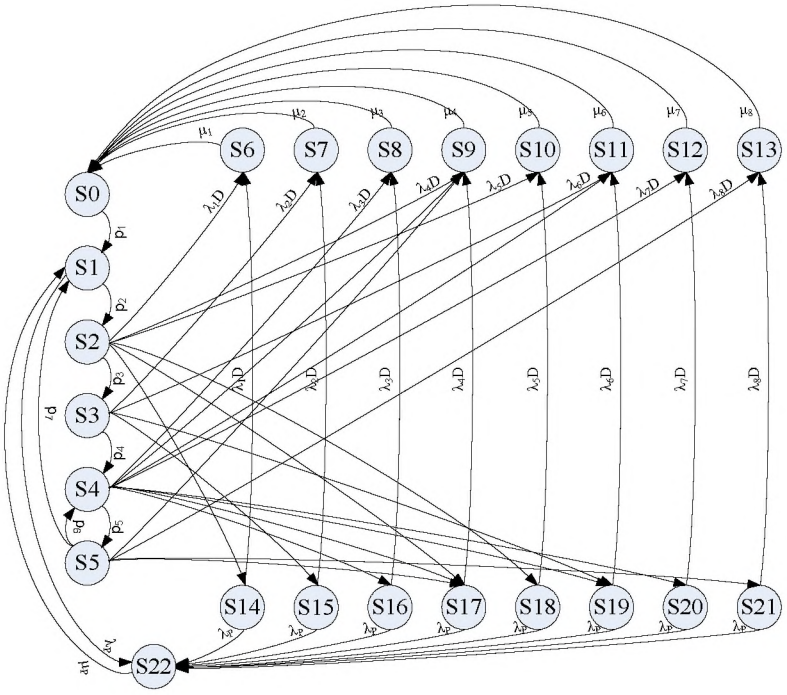


Рисунок 2.10 – Розмічений марковський граф моделі вебсервісу

Після рішення системи лінійних диференціальних рівнянь Колмогорова, побудованої за графом на рисунку 2.10, для параметрів з табл. 2.1 отримані результати, наведені на рисунку 2.11.

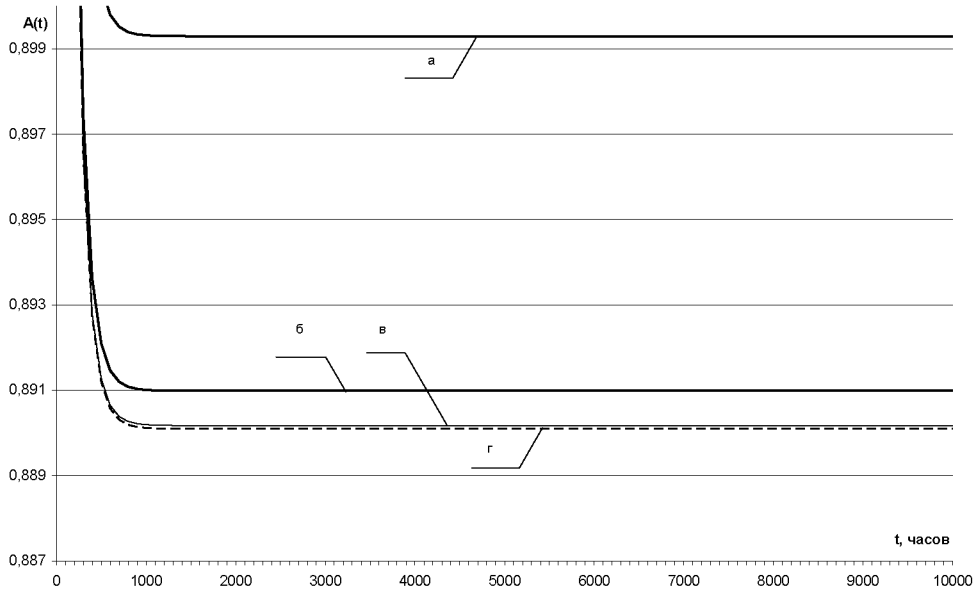


Рисунок 2.11 – Залежність коефіцієнта готовності системи А для різних значень параметрів ρ: а) ρ = 0,1; б) ρ = 1; в) ρ = 5; г) ρ = 8.

Для побудови матриці системи диференціальних рівнянь Колмогорова-Чепмена використовується функція matrixA [27]. Рішення СДР Колмогорова було виконано у системі Matlab за допомогою методу ode15s для часового інтервалу [0…10000] годин. Результати рішення представлені в графічному виді на рисунку 2.11, де показані результати розрахунків коефіцієнта готовності для занижених параметрів ρ. Як видно з графіків, динаміка зміни коефіцієнта готовності стабільна (постійна), а змінюється тільки його стаціонарні значення. Так, з похибкою 10^{-5} система переходить на стаціонарні значення через 900 годин, а з похибкою 10^{-6} – через 1200 годин для будь-яких значень ρ. Тому зроблено припущення, що і для великих значень ρ динаміка зміни коефіцієнта готовності залишиться тією ж самою. Для розрахунку значення коефіцієнта готовності в стаціонарному режимі досить вирішити систему лінійних рівнянь. Всі подальші дослідження моделі вебсервісу проведені для стаціонарного режиму, де

досліджуються залежності коефіцієнта готовності не від часу функціонування, а від інших параметрів.

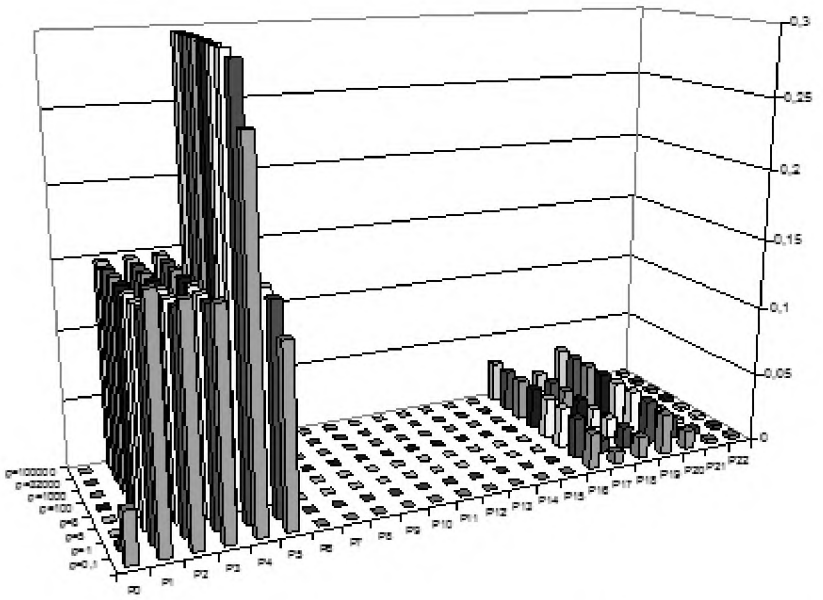


Рисунок 2.12 – Діаграма розподілу рівнів ймовірностей станів системи моделі вебсервісу при різних значеннях параметрів ρ1-ρ7

Аналіз даних на рисунку 2.13 показав, що для прийнятих значень параметрів системи основну загрозу готовності вебсервісу несуть з одного боку – приховані відмови програмної компоненти (ймовірності P16-P21), з іншого боку – прискорене проведення профілактики прихованих відмов (P22).

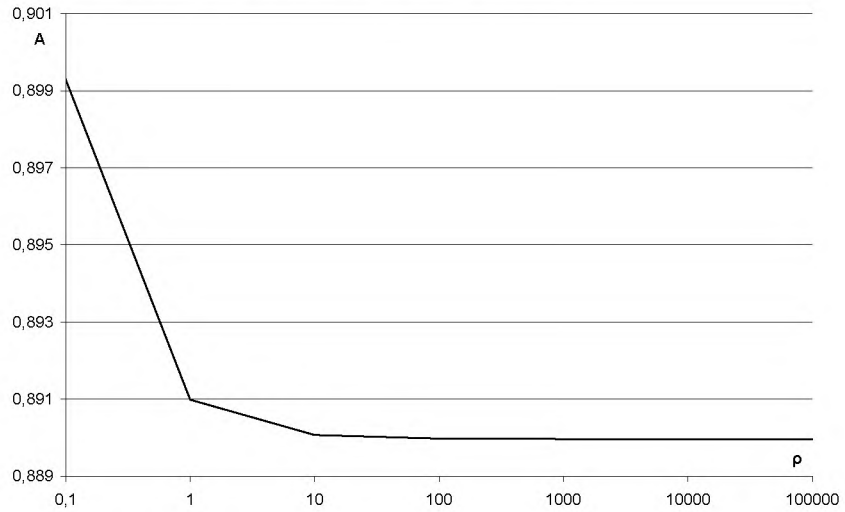


Рисунок 2.13 – Залежність коефіцієнта готовності системи A від значень параметрів ρ1-ρ7

Збільшення інтенсивності запитів (ρ_1-ρ_7) на кілька порядків практично не впливає на готовність системи, так як викликає рівномірний перерозподіл ймовірностей початкових станів між P1, P2, P3 і P5. ймовірність знаходження системи в початковому стані P0 при збільшенні ρ_1-ρ_7 буде прагнути до нуля.

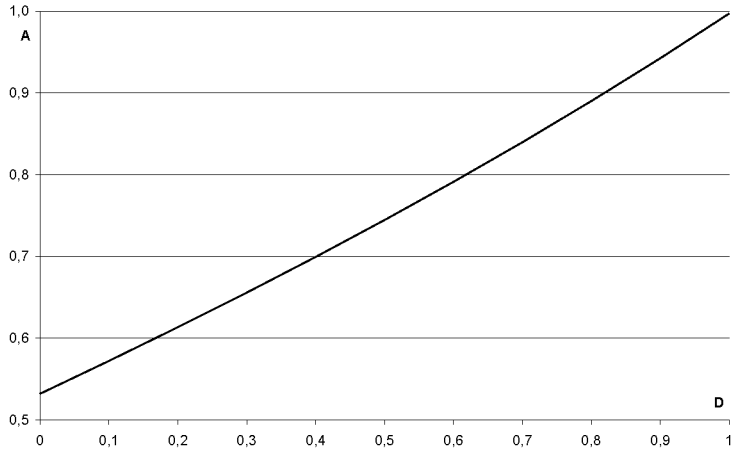


Рисунок 2.14 – Залежність коефіцієнта готовності системи A від значень параметра D

Аналіз даних на рисунку 2.14 привів до очікуваного результату – зі збільшенням повноти контролю D ймовірність знаходження системи в стані прихованих відмов буде прагнути до нуля, внаслідок чого збільшиться готовність системи в цілому. Спостерігається практично лінійна залежність коефіцієнта готовності від величини повноти контролю.

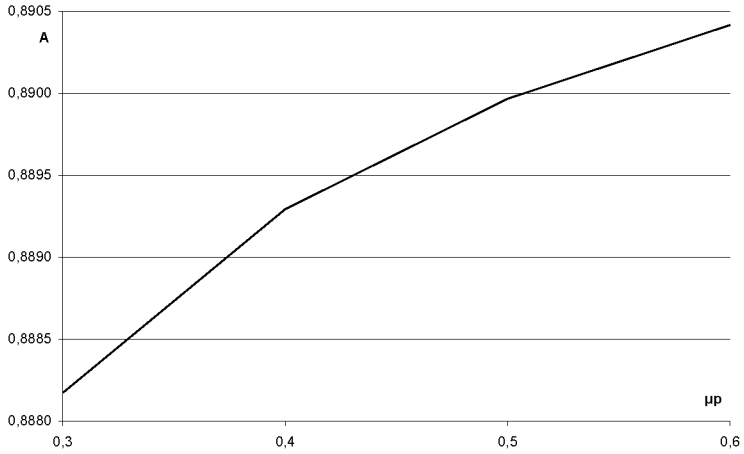


Рисунок 2.15 – Залежність коефіцієнта готовності системи A від значень параметра μ_p

Аналіз залежності готовності вебсервісу від середньої тривалості профілактики (величини, зворотної параметру μ_p) на рисунку 2.15 показав, що з одного боку спостерігається нелінійна залежність $A(\mu_p)$, з іншого – зменшення тривалості профілактики в середньому на 2 години дає виграш в надійності на 0,0025 щодо рівня 0,888. Це показує низьку ефективність прискорення профілактики при прийнятих за замовчуванням змінних параметрах системи.

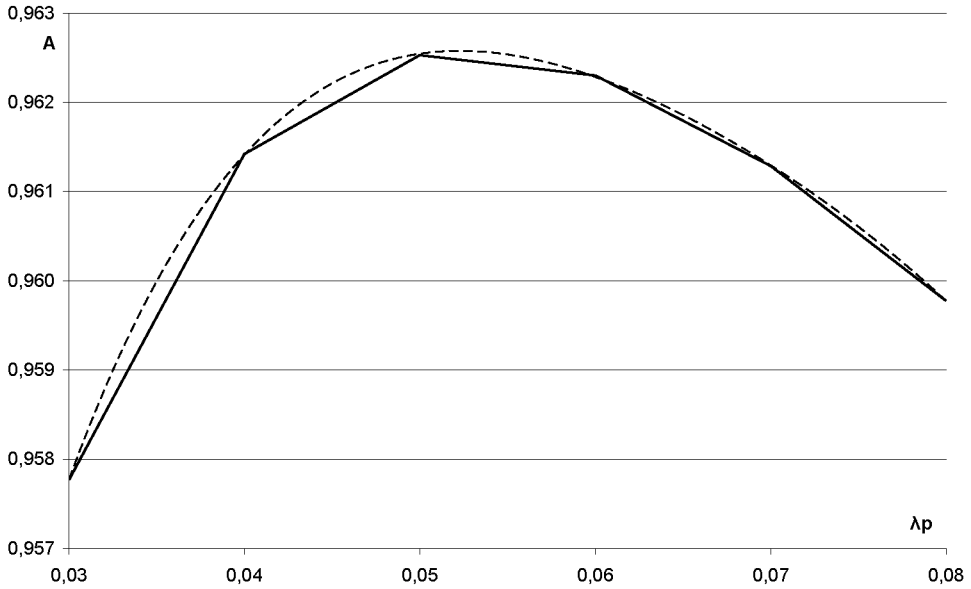


Рисунок 2.16 – Залежність коефіцієнта готовності системи A від значень параметра λ_p

Аналіз даних на рисунку 2.16 ілюструє, що збільшення інтенсивності проведення профілактики λ_p дозволяє ефективно боротися з прихованими відмовами до певного моменту, після якого основним фактором неготовності стає знаходження системи в стані профілактики S22. Відповідно, для певного набору значень параметрів системи існує оптимальне значення λ_p, при якому система має максимальну надійність. Визначення цього оптимального значення – окреме завдання, вирішення якої виходить за рамки даної роботи. Одним із способів вирішення такого завдання можна вважати апроксимування функції $A(\lambda_p)$ з подальшим визначенням аргументу її максимального значення. На рис. 2.16 штриховою лінією показана апроксимуюча функція поліномом 4-го порядку. Як

видно з рисунку 2.16, максимальне значення функції готовності відповідає значенню λ_p, близькому до 0,05, що відповідає періоду виконання профілактики $T_p = 20$ год.

Висновки до розділу 2

У другому розділі розглянуто моделі архітектур відмовостійких вебсервісів. На їх основі можуть бути отримані моделі для детального дослідження відмовостійких сервіс-орієнтованих систем. Для цього необхідно вибрати відповідну базову модель, деталізувати марковський граф і дані для моделювання.

У розділі виконана побудова марковського графа для моделі композитних сервісів. Розрахований коефіцієнт готовності, а також виконаний аналіз впливу вхідних параметрів на його стале значення. Результати моделювання показали, що найбільш ефективними методами підвищення надійності сервіс-орієнтованої системи є поліпшення якості роботи контролюючого модуля, що виявляє приховані відмови, а також оптимальний вибір інтенсивності профілактики прихованих відмов системи.

РОЗДІЛ 3
РОЗРОБКА І ДОСЛІДЖЕННЯ ІМІТАЦІЙНОЇ МОДЕЛІ ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ З ВРАХУВАННЯМ АТАК НА ЇЇ ВЕБКОМПОНЕНТИ

3.1 Визначення обмежень та алгоритму імітаційного моделювання

Необхідність імітаційного моделювання для дослідження вебсервісу як відновлюваної системи, що функціонує в умовах прояву дефектів АЗ і ПЗ, проведення оновлень ПЗ обумовлена наступним [22]:

– необхідністю перевірки достовірності результатів, отриманих в ході аналітичного моделювання [23];

– необхідністю розробки підходу до побудови імітаційних моделей функціонування розглянутих систем з метою зняття накладених в ході аналітичного моделювання обмежень.

Аналіз завдання імітаційного моделювання показав, що з точки зору формалізації розглянутий процес підпадає під клас математичних схем систем масового обслуговування (так званих Q-схем) [26]. Однак, використання принципів формалізації Q-схем при побудові алгоритму і подальша його реалізація із застосуванням спеціалізованих мов імітаційного моделювання (GPSS, SLAM, GASP, SIMSCRIPT і ін.), або модулів імітаційного моделювання математичних програмних пакетів (наприклад, пакету візуально-орієнтованого програмування SIMULINK , що входить складовою частиною в систему MATLAB), є досить трудомістким в силу специфіки зміни інтенсивності прояву дефектів і функціонального представлення очікуваного результату. Тому, при побудові імітаційного алгоритму доцільно використовувати метод безпосереднього моделювання (без прив'язки до типових математичних схем) з використанням однієї з мов програмування високого рівня [25]. При цьому функціональну залежність коефіцієнта готовності від часу функціонування системи A(t) можна

отримати при поєднанні декількох значень коефіцієнта готовності для різних часових моментів t_i.

Через специфіку проведення статистичних випробувань, в якості вхідних параметрів імітаційної моделі двоканального вебсервісу (крім використовуваних в аналітичних моделях параметрів λ_{HW}, λ_{SW}, μ_{HW}, μ_{SW}, λ_{up}, μ_{up} виступають:

α – достовірність (довірча ймовірність) оцінки коефіцієнта готовності за допомогою статистичного моделювання;

ε – похибка (абсолютна точність) визначення коефіцієнта готовності за допомогою статистичного моделювання;

Коефіцієнт готовності для даної задачі є суть випадкова величина (A_i), яка в результаті експерименту може приймати одне з можливих значень, тому до його оцінювання пред'являються вимоги незсуненості та ефективності [23]. Його статистичною оцінкою є вибіркове середнє (середнє арифметичне), яке можна визначити з заданою точністю ε>0 і достовірністю 1-α. Необхідна кількість реалізацій $n_{тр}$ досліджуваного процесу при його моделюванні [26] невідома, для визначення її найбільш прийнятним є послідовний метод, в якому для досягнення заданої точності забезпечується мінімальна кількість реалізацій на початковому етапі моделювання. Метод складається з наступних етапів:

1. Побудова першої реалізації A_1 випадкової величини A.

2. Побудова реалізації A_n ($n \geq 2$) випадкової величини A, що не залежить від попередніх A_1 ... A_{n-1} реалізацій.

3. Визначення оцінки дисперсії реалізацій величини A_n.

$$S_n^2 = \frac{1}{n-1}\left[\sum_{i=1}^{n} A_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} A_i\right)^2\right]. \qquad (3.1)$$

4. Визначення точності δ оцінки A за формулою:

$$\delta = \frac{t_{\alpha/2,n-1} \cdot S_n}{\sqrt{n}}, \qquad (3.2)$$

де $t_{\alpha/2,n-1}$ – квантиль розподілу Стьюдента з n-1 ступенями свободи.

5. Перевірка умови $\delta \leq \varepsilon$, якщо умова не виконана, то здійснюється повернення до другого етапу з заміною n на n+1, в іншому випадку приймається $n_{тр}=n$.

Моделюючий алгоритм процесу функціонування вебсервісу розроблений на основі «принципу dz» (принципу особливих станів) [29]. При моделюванні події, що зумовлюють зміну станів системи, обробляються послідовно і системний час зміщується кожен раз до початку наступної події. Таким чином, моменти часу зміни станів системи є випадковими і визначаються моментами відмов, відновлень АЗ, ПЗ, моментами оновлень ПЗ вебсервісу. Ці часові проміжки розігруються по наступному закону:

$$t = \frac{1}{\pi}\ln(Random), \tag{3.3}$$

де t – час безвідмовної роботи каналу по одній з компонент;

Random – випадкова величина, рівномірно розподілена на інтервалі (0...1), отримана за допомогою програмного датчика ПСЧ RANDOM системи програмування;

π – відповідний параметр закону розподілу (λ або μ).

Блок-схема моделюючого алгоритму зображена на рис. 3.1, а варіант програмної реалізації цього алгоритму, що дозволяє провести дослідження вебсервісу, представлений в додатку Б.

У роботах [21,22] запропоновані схожі з побудови і характером імітаційні моделі, однак детальний аналіз дозволив виділити і усунути такі недоліки попередніх моделей:

1. На кожен канал (АЗ або ПСЗ) раніше резервувалася чотири динамічних масиви, в яких зберігалися дані про часові проміжки – тривалості відмов і відновлень кожного каналу і їх стани (справний/несправний). У запропонованій моделі використовується по одному динамічному масиву на кожен канал, так як відпадає необхідність зберігання кожного розіграшу часового інтервалу, а стан кожного каналу в будь-якому випадку по черзі змінюються починаючи з справного.

2. У запропонованій моделі збільшена розмірність масиву значень квантиля розподілу Стьюдента з [3,100] до [4,500], що дозволило підвищити достовірність моделювання до 0,999.

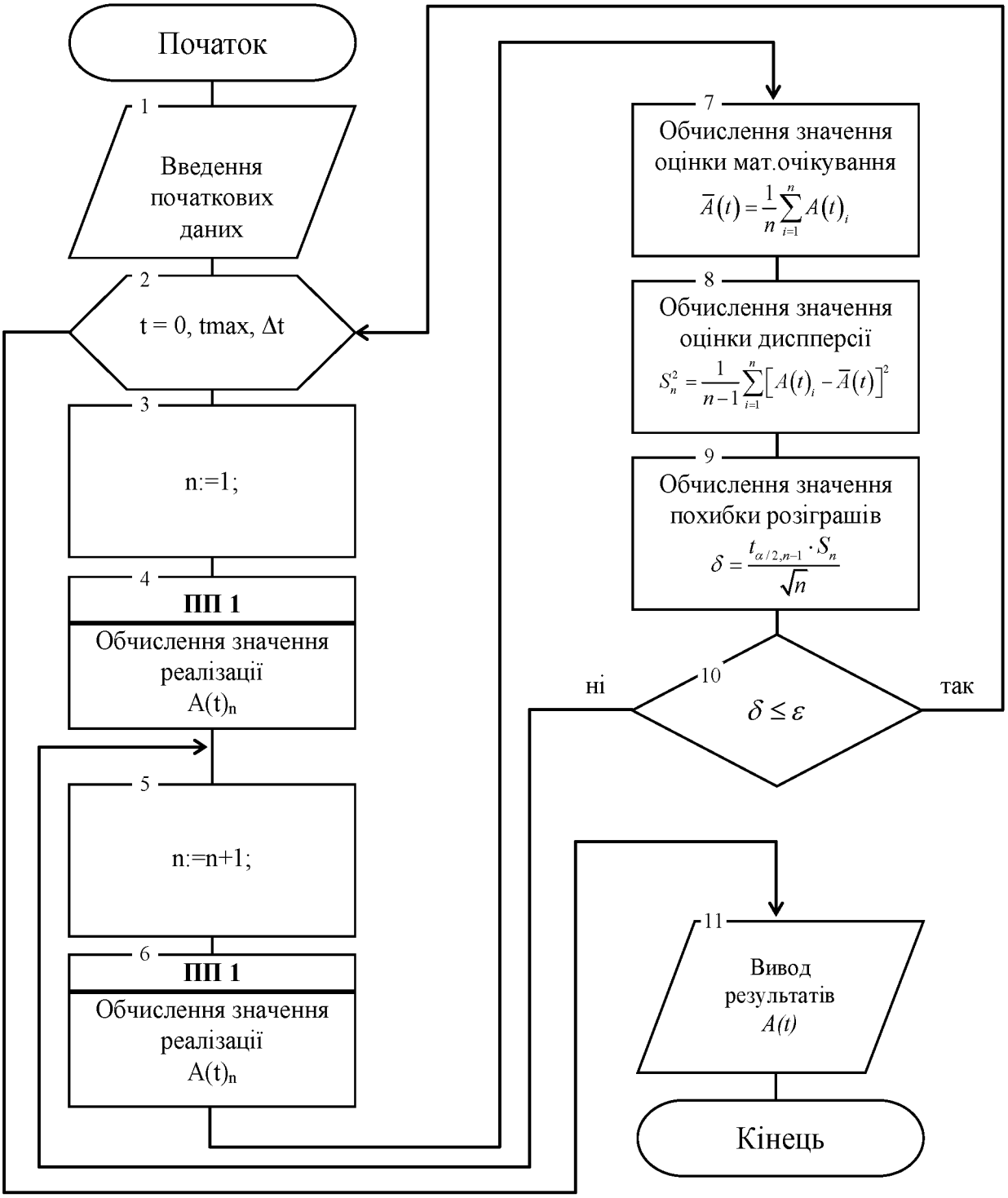


Рисунок 3.1 – Алгоритм імітаційного моделювання вебсервісу за певною кількістю розіграшів $n_{тр}$

3. Для визначення справності системи в цілому використовується метод множення логічних станів апаратних каналів з подальшим складанням з логічним станом програмного каналу, а також станів оновлення АЗ (в попередніх моделях використовувався метод вкладених логічних конструкцій if ... then).

4. Кількість початкових розіграшів збільшено з 10 до 50, що дозволяє зменшити похибку розрахунку вибіркового середнього значення, відносного якого визначається результат розіграшу $A(t_i)$ із заданою точністю ε.

3.2 Розробка імітаційної моделі функціонування вебсистеми в умовах атак на її компоненти без визначення кількості розіграшів

На даний час кількість запропонованих аналітичних моделей невелика (у порівнянні з моделями [28,29]), що дозволяє відмовитися від розробки складних альтернативних імітаційних моделей вебсерверів. У даній роботі метою імітаційного моделювання є визначення зміни значень функції готовності вебсервера в процесі експлуатації системи для випадків проведення атак на службу DNS [29]. За результатами статистичних випробувань здійснюється перевірка достовірності аналітичних моделей.

Аналіз завдання імітаційного моделювання показав, що з точки зору формалізації розглянутий процес підпадає під клас математичних схем систем масового обслуговування (так званих Q-схем) [31]. Для досягнення поставленої мети побудова моделює алгоритму і подальша його реалізація (як і реалізація принципів формалізації Q-схем) були покладені на модуль імітаційного моделювання математичного програмного пакету MATLAB. В силу обмеженості вирішуваних завдань, пакет візуально-орієнтованого програмування SIMULINK не використовується, а використовуються елементи командного вікна.

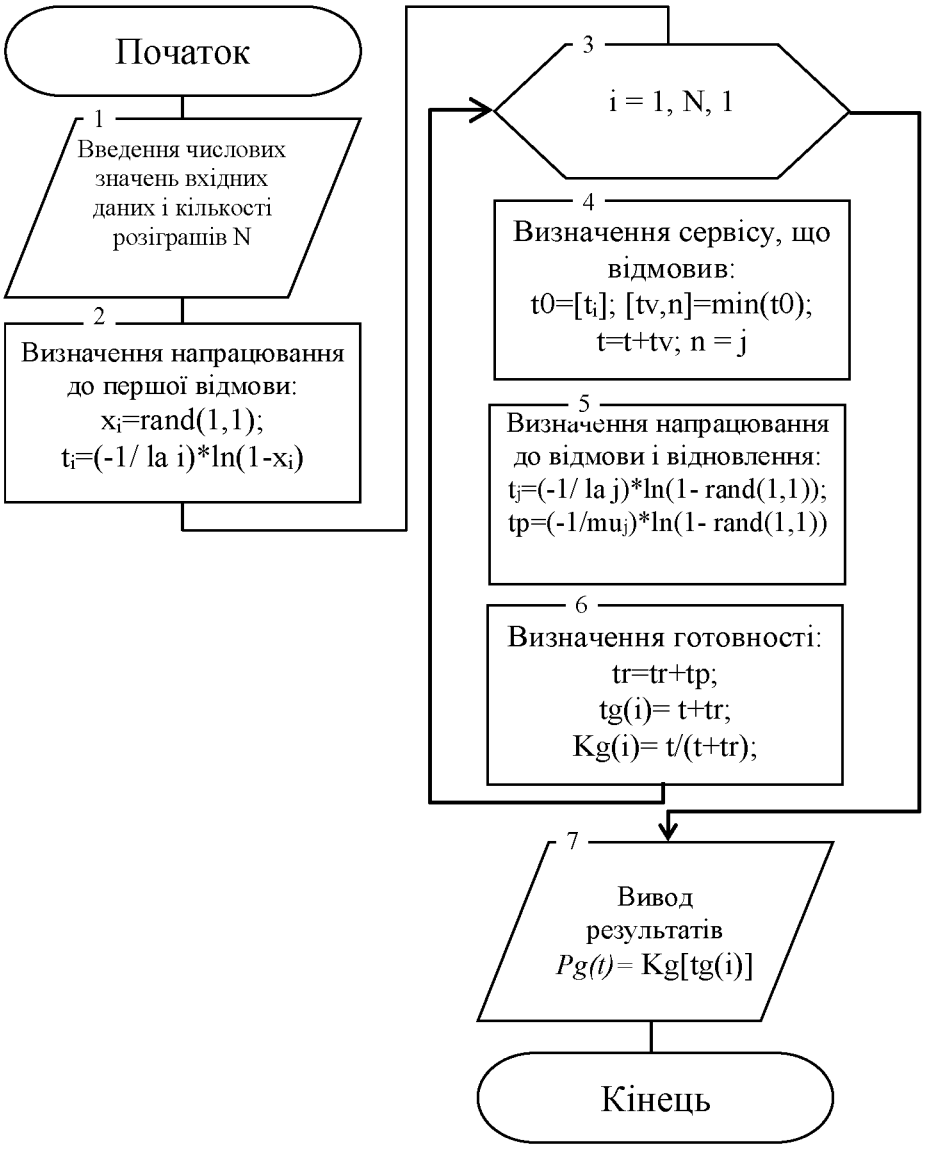


Рисунок 3.2 – Алгоритм імітаційного моделювання вебсистем

Детальний опис команд пакета Matlab для імітаційного моделювання викладено в [32]. В рамках даної роботи на базі алгоритму імітаційного моделювання (рис. 3.2) були побудовані наступні статистичні моделі вебсервісу:

– з урахуванням відмов і відновлень трьох служб;

– з урахуванням атак на службу DNS з подальшим її перезапуском;

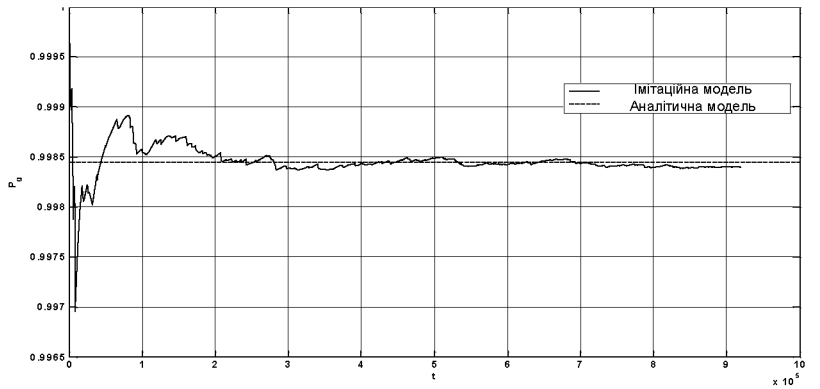
– з урахуванням атак на три служби з подальшим їх перезапуском;

– з урахуванням атак на службу DNS з подальшим усуненням несправностей конфігурації.

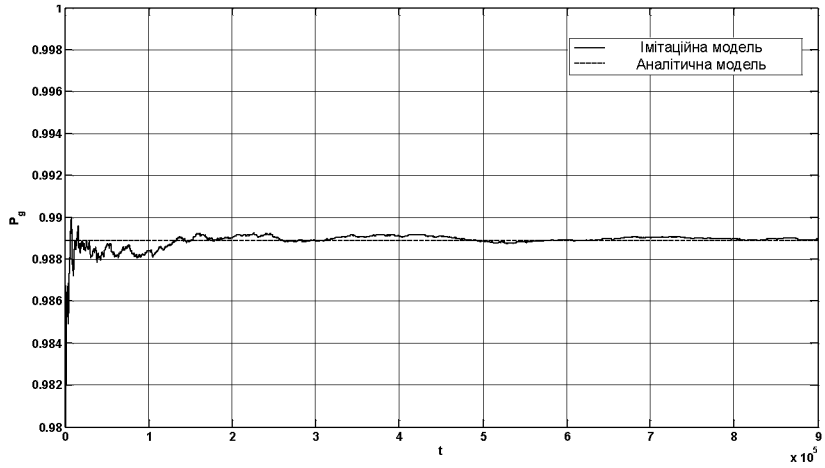
Програмні конструкції Matlab для зазначених імітаційних моделей представлені в додатку Б.

3.3 Порівняльний аналіз результатів аналітичного і імітаційного моделювання

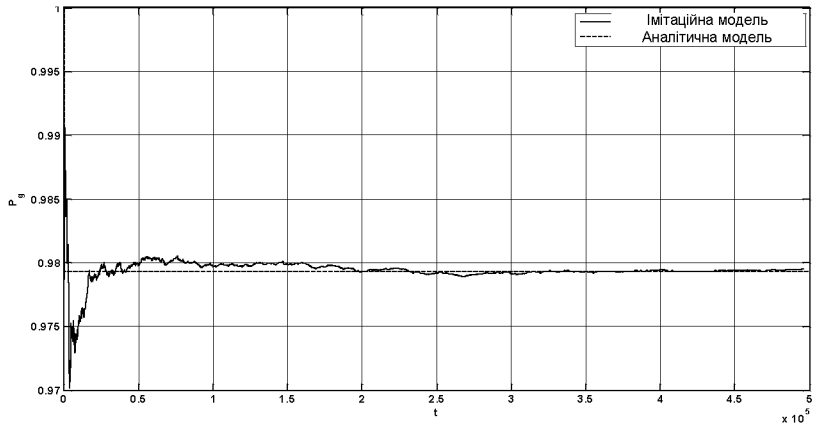
Результати порівняння імітаційних та аналітичних моделей представлені на рисунку 3.3 і на рисунку 3.4. Для прийнятного відображення часовий інтервал дослідження аналітичних моделей був спеціально збільшений на порядок.



а) модель з урахуванням відмов і відновлень трьох служб



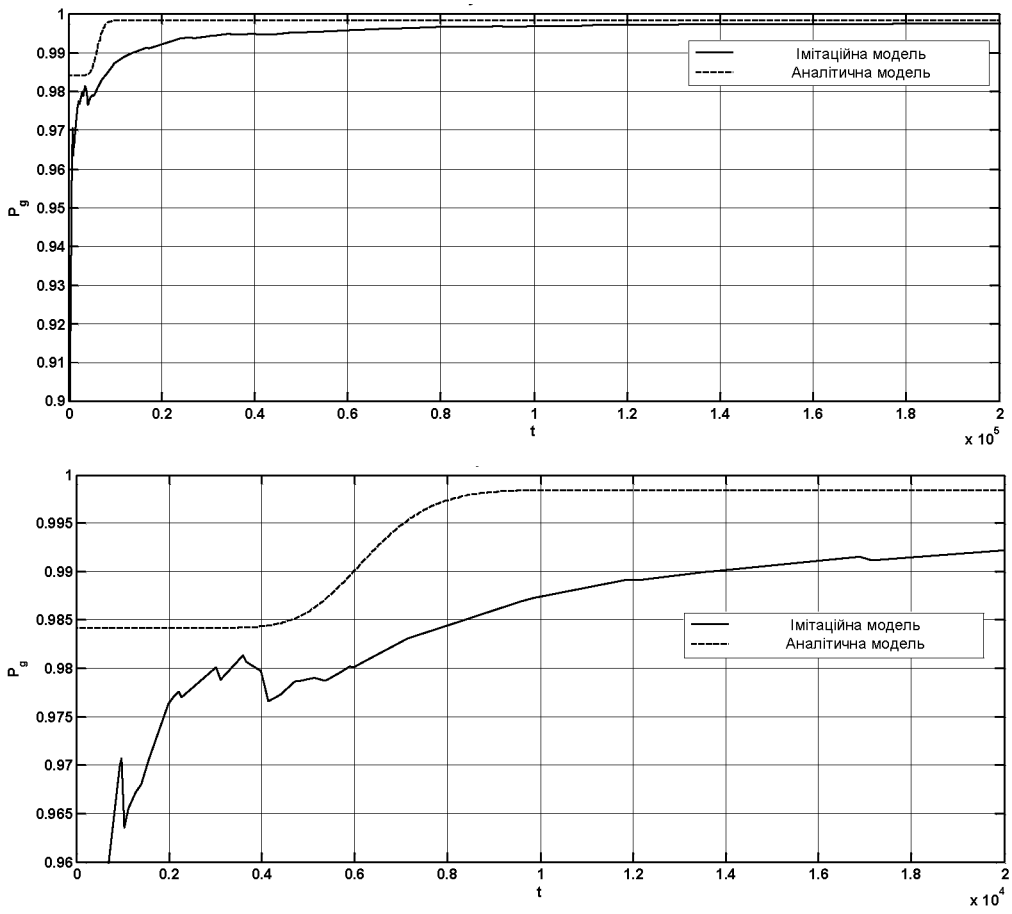
б) модель з урахуванням атак на службу DNS з подальшим її перезапуском



в) модель з урахуванням атак на три служби з подальшим їх перезапуском

Рисунок 3.3 – Порівняння результатів імітаційного моделювання вебсервісів і аналітичних моделей

Як показали результати випробувань, отримані графіки функції готовності мають характерні фази розвитку: перехідну на початковому етапі функціонування системи і фазу переходу в сталий (стаціонарний) стан. Для імітаційних моделей характерно неоднозначна поведінка функції готовності на перехідному етапі – на рисунку 3.3, а крива готовності кілька разів перетинає лінію горизонтальної асимптоти сталого значення, а на рисунку 3.3, б – підходить до асимптоти знизу. Період переходу в усталений режим у функції готовності, отриманої за допомогою імітаційних моделей, становить $3*10^5$ годин, що на 4 порядки більше, ніж у відповідних аналітичних моделей. При цьому значення коефіцієнта готовності в усталеному режимі практично збігаються, що підтверджує адекватність аналітичних моделей.



а) модель на часовому інтервалі [0...200000] годин

б) модель на часовому інтервалі [0...20000] годин

Рисунок 3.4 – Порівняння результатів імітаційного моделювання вебсервісів і аналітичної моделі з урахуванням атак на службу DNS з подальшим усуненням несправностей конфігурації

Порівняння аналітичної та імітаційної моделей вебсервісу з усуненням вразливостей конфігурації також показало високу збіжність значень коефіцієнтів готовності в усталеному режимі. Що характерно, як видно з рисунку 3.4, б, імітаційна модель в точності повторює динаміку аналітичної моделі (відповідність виходу кривої готовності з точки мінімуму після 5000 годин експлуатації). Однак при цьому видно, що крива готовності імітаційної моделі «не встигає» досягти усталеного значення в ті ж терміни, що і аналітична модель. При цьому слід врахувати, що для ілюстрації такої поведінки в вихідні дані було внесено підвищену кількість вразливостей (nv = 30).

3.4 Імітаційне моделювання функціонування вебсервісу в умовах прояву дефектів та атак на його вразливості

Процес моделювання здійснюється наступним чином. Після запуску скрипта здійснюється введення значень вхідних параметрів інтенсивностей переходів і розмірності моделі. Окремо відзначимо введення параметрів Taim_interval – часового інтервала дослідження поведінки системи і Nr – кількості повних розіграшів імітаційної моделі.

Надалі часовий інтервал Taim_interval уточнюється як подвоєне значення часу переходу в сталий стан функції готовності, отриманий за допомогою аналітичної моделі вебсервісу.

Під одним розіграшем тут розглядається повна «реалізація» функції готовності на інтервалі [0..Taim_interval]. У процесі одного розіграшу також фіксується мінімальне значення функції готовності Amin і значення моменту часу tconst переходу системи в крайній правий (поглинаючий) стан (якщо така подія спостерігається при розіграші).

Процедура розіграшу з використанням даних орграфа здійснюється наступним чином. У момент часу t=0 система знаходиться в початковому стані S0, який відповідає першому елементу масиву V:V(1,:)=[0 0 0 0], тому допоміжній

змінній n присвоюється значення 1. Зі стану S0 можливі два переходи в стани S9 і S15, описані елементами масиву E: E(1,:)=[1 10 0.003]; E(19,:)=[1 16 0.0005]. Формування вибірки цих елементів у масив Q здійснюється за допомогою виразу: Q=E(find(E(:,1)==n),:).

Далі формується вектор t1, що містить розіграші часових інтервалів, що визначають вихід зі стану S0. Розіграші отримані за допомогою функції t1=exprnd(1/mu), де mu – відповідна вага дуги графа із масиву E (і відповідно його вибірки Q). Мінімальне значення часового інтервалу визначає перехід в наступний стан, тому допоміжній змінній k присвоюється значення k=Q(q+find(t1==min(t1))).

Далі здійснюється перерахунок напрацювання Tbf між відмовами (оскільки перехід виконувався із працездатного стану S0). Якби перехід здійснювався із непрацездатного стану (n>(Nd+1)*(Nv+1)), то виконувався б перерахунок напрацювання Tbr відновлень і обслуговувань.

Далі здійснюється присвоювання n=k і, якщо система не досягла поглинаючого стану S9 (для Nd=2, Nv=2), або сумарний час всіх розіграшів не перевищив Taim_interval; виконується наступний розіграш переходу в новий стан. Якщо в процесі розіграшів система перейшла у поглинаючий стан, то для всіх наступних часових відліків готовність системи приймається рівною 1.

Повним розіграшем функції готовності є сукупність часткових розіграшів, що забезпечують побудову функції готовності на інтервалі [0..Taim_interval].

Після завершення заданої кількості Nr повних розіграшів формуються множини [tconst] і [Amin], які обробляються за допомогою типових статистичних функцій Matlab. Складнішою є ситуація з множиною реалізацій функції готовності, оскільки кожна реалізація має унікальні часові відліки і значення функції готовності в них. На рис. 3.5 показаний результат 10000 повних розіграшів функції готовності у порівнянні з аналітичною моделлю вебсервісу. Очевидно, що проводити порівняння результатів аналітичного і імітаційного моделювання для такого представлення складніше, і потребує додаткової обробки результатів розіграшів.

На основі отриманих значень розіграшів (а саме – значення часових відліків і функції готовності в них) формується масив Ai. На першому етапі необхідно відсікти детерміновані значення елементів матриці Ai в точках t=0 і t=Taim_interval(A(t)=1). Для цього послідовно виконуються операції унікалізації елементів матриці Ai=unique(Ai,'rows') і відсікання першого і останнього рядків масиву: Ai(1,:)=[]; Ai(end,:)=[].

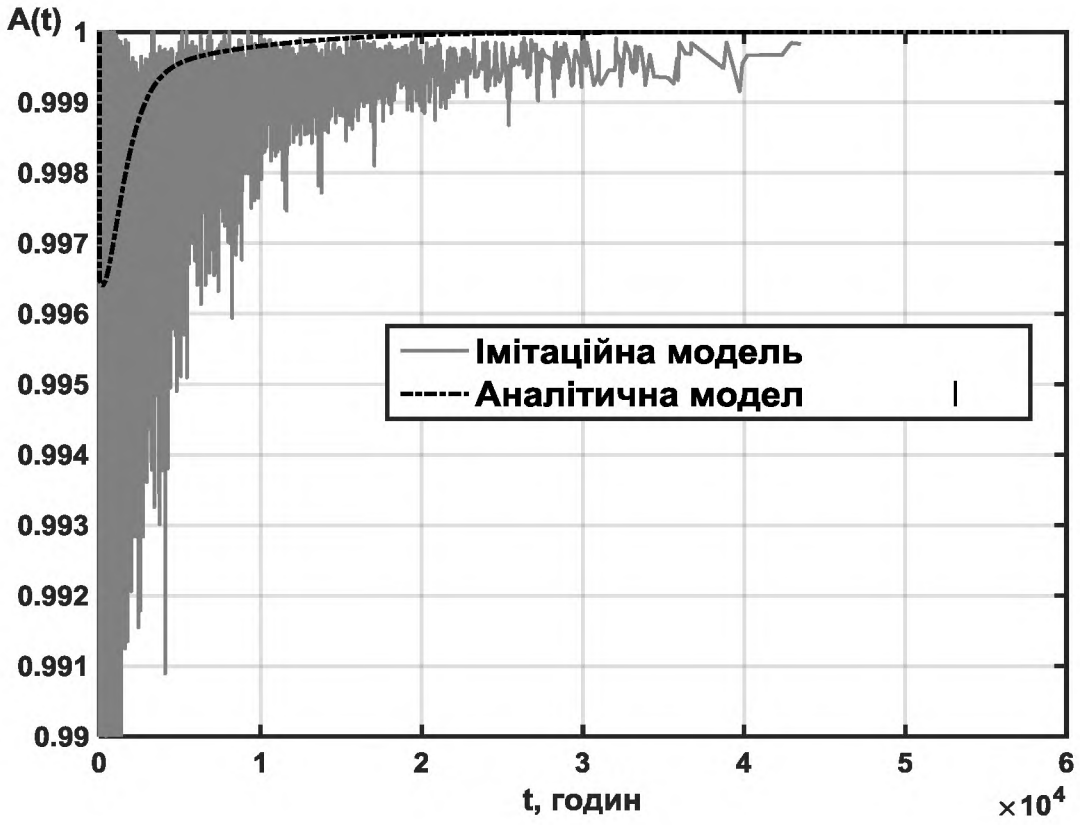
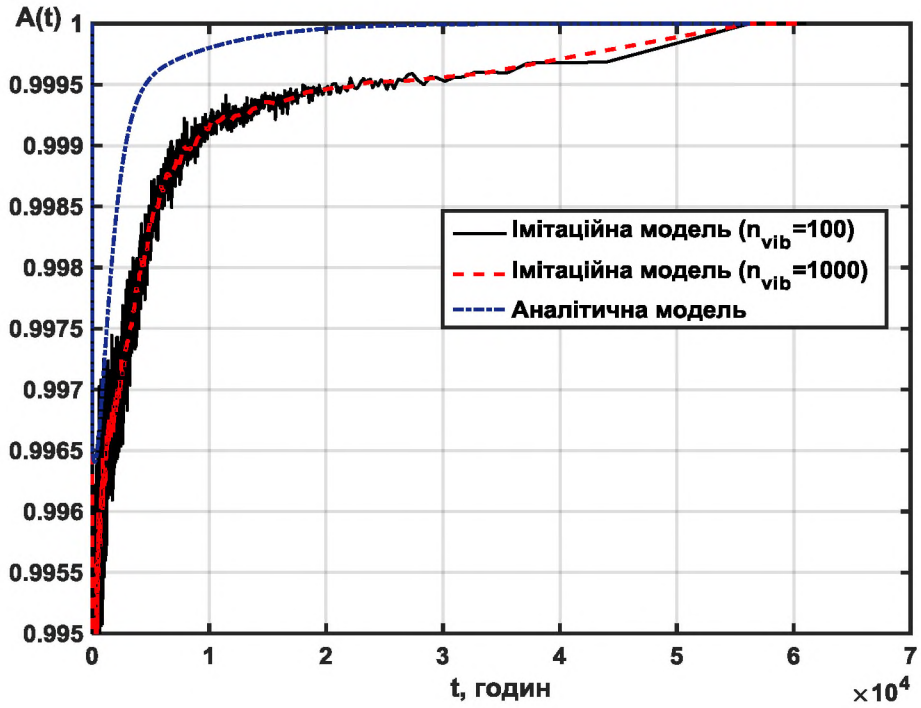


Рисунок 3.5 – Порівняння результатів аналітичного моделювання і всіх повних розіграшів Монте-Карло моделі вебсервісу

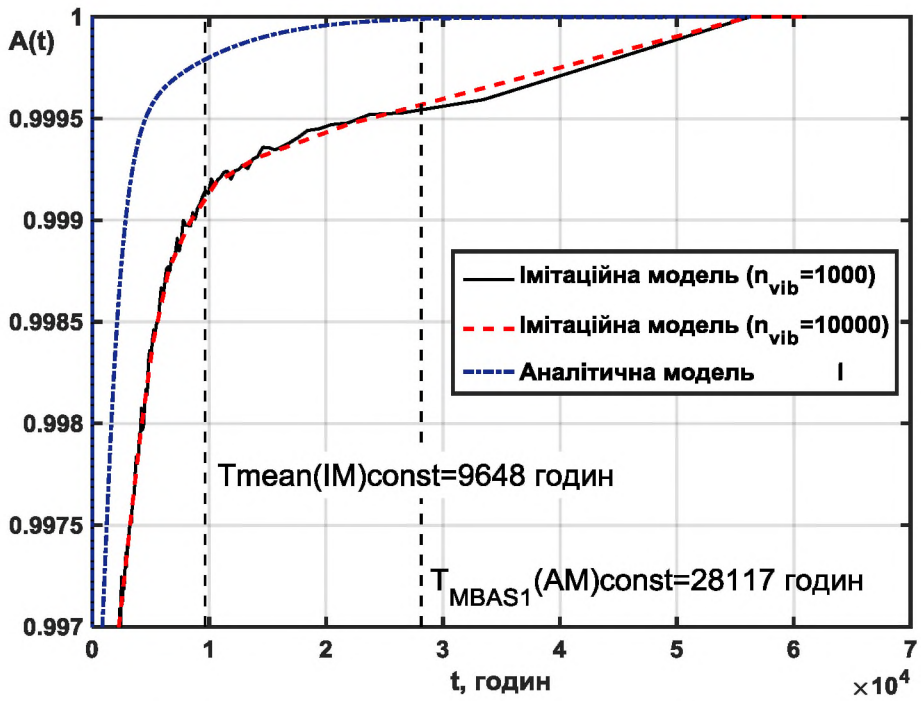
Далі у процесі статистичної обробки елементи матриці Ai групуються у вибірки (розмір групуючої вибірки визначається змінною n_vibor) і для кожної вибірки визначається усереднене значення часового відліку і значення функції готовності в ньому за допомогою функції mean(X). Результати обробки представлені на рисунку 3.6 для різних розмірів вибірки n_vibor.

Збільшення розмірів вибірки дозволяє отримати більш гладкий графік, що зручно для візуального порівняння результатів аналітичного і імітаційного моделювання. Також на рисунку 3.6, б у масштабі часової вісі показано

розташування значень результуючого показника T_{MBAS1}const аналітичної моделі (визначений з похибкою 10^{-5}) і усередненого значення tconst всіх повних розіграшів моделі Монте-Карло.



а)



б)

Рисунок 3.6 – Порівняння результатів обробки генеральної сукупності повних розіграшів Монте-Карло з різним обсягом вибірок (а – 100 і 1000; б – 1000 і 10000) і аналітичної моделі вебсервісу

З графіка (рис. 3.6,б) видно, що значення показника T_1const аналітичної моделі розташовано приблизно посередині між значеннями mean(tconst) і max(tconst) імітаційної моделі.

Для полегшення порівняння з аналітичною моделлю прийнято рішення інтерполювати результат аналітичного моделювання (функцію готовності) на множині часових відліків імітаційної моделі за допомогою функції Pg_a=[interp1(ta, Pga, tg_I_avg, 'spline')].

Значення результуючого показника tconst формують масив розіграшів деякої випадкової величини в діапазоні [367.7…5.1576e+04]. Розподіл випадкової величини tconst відрізняється від нормального і представлено на рисунку 3.7.

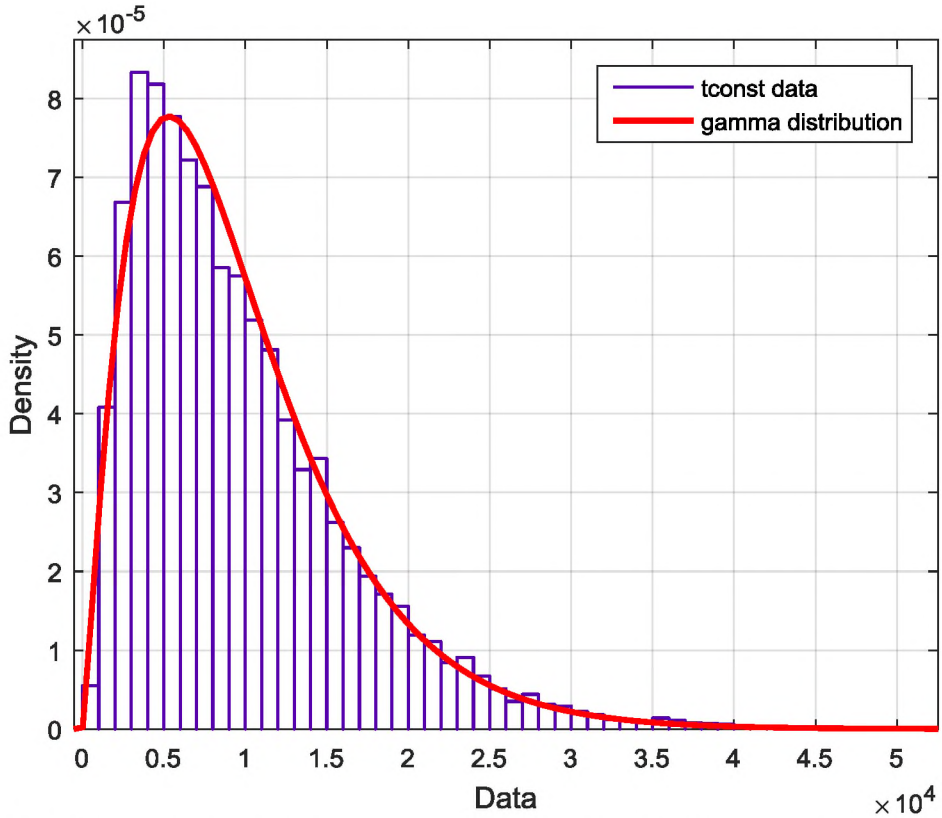


Рисунок 3.7 – Функція розподілу щільності ймовірностей результуючого показника tconst та її представлення гамма-розподілом

Оскільки закон розподілу випадкової величини tconst апріорі невідомий, було прийнято рішення визначити його, використовуючи спеціальну функцію Matlabfitmethis.m [30] для визначення оптимального за критерієм “Loglikelihood →

min” закону розподілу із множини:

Cdist= {'normal'; 'lognormal'; 'exponential'; 'gamma'; 'logistic';...
'loglogistic'; 'uniform'; 'weibull'; 'ev'; ...
'gev'; 'gp'; 'inversegaussian'; 'birnbaumsaunders';...
'beta'; 'nakagami'; 'rayleigh';'rician'}

У результаті виконання функції fitmethis(tconst) був підібраний гамма-розподіл з функцією:

$$f(t)=\frac{t^{a-1}}{b^{a}\cdot\Gamma(a)}e^{-\frac{t}{b}};t\geq 0\,. \qquad (3.4)$$

Параметри розподілу були визначені методом максимальної правдоподібності (mle), та їх значення склали: a=2.22245 (з похибкою 0.0293806) і b=4341.19 ((з похибкою 64.3568). Функція гамма-розподілу для вказаних значень параметрів показана на рис. 3.7.

З рисунку 3.7 очевидно, що відмінність між значеннями показника T_{MBAS1}const аналітичної моделі і mean(tconst) і max(tconst) імітаційної моделі суттєві. При цьому необхідно врахувати той факт, що в аналітичній моделі при визначенні T_{MBAS1}const використовувалась похибка 10^{-5}. Приймаючи до уваги значення похибки, для mean(tconst)=9648,1 годин значення функції готовності аналітичної моделі складе Pga=0.999791. Це відповідає визначенню T_{MBAS1}const з похибкою 2.09e-04.

Відповідно, для max(tconst)=51576 годин, значення функції готовності аналітичної моделі складе Pga=0.999999855652. Це відповідає визначенню T_{MBAS1}const з похибкою 1.4435e-07.

Другий результуючий показник Amin розіграний на множині [0.0078…0.9996]. Враховуючи той факт, що мінімум функції готовності розташований на початковому етапі експлуатації системи (близько t=0), зібрати достовірну статистику зміни випадкової величини Amin складно. Це підтверджується видом функції щільності розподілу розіграшів величини Amin, що представлена на рис. 3.8.

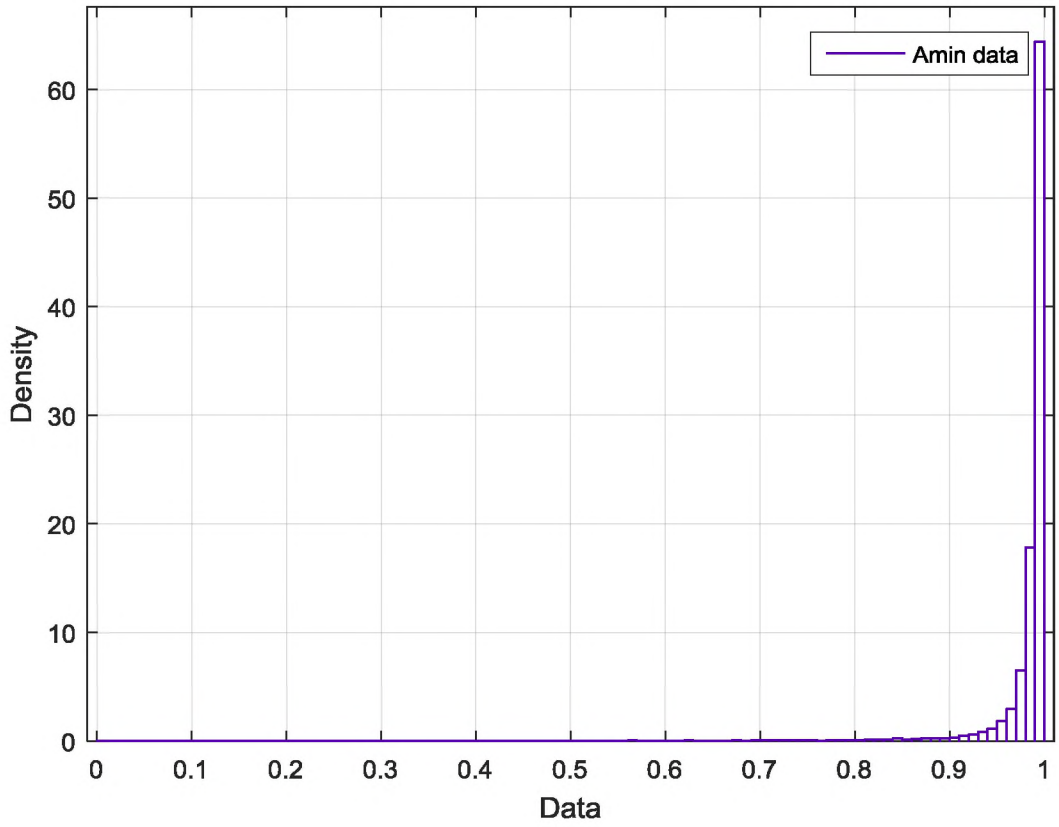


Рисунок 3.8 – Функція розподілу щільності ймовірностей розіграшів результуючого показника Amin моделі вебсервісу

Із зібраної статистики (рис. 3.8) видно, що розіграші показника Amin дають довгий лівий хвіст значень з низькою щільністю розподілу, які навряд чи несуть корисне статистичне навантаження. Усереднене значення mean(Amin)=0.9808 відносно мінімуму функції готовності аналітичної моделі A_1min=0.9964 має абсолютну похибку 0.0156, або відносну похибку в 1,57%.

У рамках даної роботи на базі алгоритму імітаційного моделювання (рис. 3.1) були побудовані наступні статистичні моделі вебсервісів для графів з поглинаючими станами:

– з урахуванням програмних дефектів і вразливостей;

– з урахуванням проведення обмеженої кількості загальних обслуговувань;

– з урахуванням проведення обмеженої кількості роздільних обслуговувань.

Програмні конструкції Matlab для вказаних імітаційних моделей представлені у додатку Б.

Також, на основі запропонованого алгоритму (рис. 3.1) були розроблені моделі вебсервісів для графів без поглинаючих станів:

– з урахуванням проведення необмеженої кількості загальних обслуговувань;

– з урахуванням проведення необмеженої кількості роздільних обслуговувань.

Розглянемо ключові особливості таких моделей на прикладі моделі з необмеженою кількістю загальних обслуговувань. На відміну від попередньої моделі, при відсутності поглинаючих станів для визначення результуючого параметра tconst необхідно фіксувати час першого «входження» системи в стан S9 (для Nd=2, Nv=2). Цей прийом реалізується через використання допоміжної змінної tc, яка обнуляється з початком випробувань, і приймає значення tc=1 з подією переходу в стан S9. Також важливою є зміна умови закінчення повного розіграшу – якщо сумарний час всіх розіграшів перевищив Taim_interval (оскільки після переходу в стан S9 в системі продовжують проводити заходи загального обслуговування). Результати статистичного моделювання системи з необмеженим загальним обслуговуванням показані на рисунку 3.9.

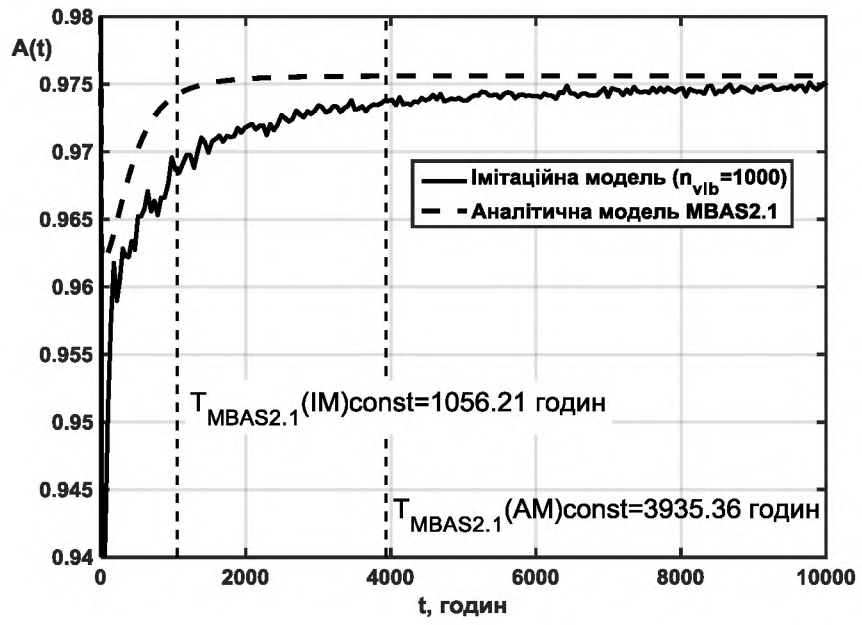


Рисунок 3.9 – Порівняння результатів розіграшів Монте-Карло й аналітичної моделі вебсервісу з урахуванням проведення обмеженої кількості загальних обслуговувань

Графіки на рисунку 3.9 ілюструють зближення кривих готовності аналітичної й імітаційної моделей при t→∞. На кінці інтервалу дослідження (t=20000 годин, на рис. 3.9 не показано) абсолютна похибка визначення A_2const для імітаційної моделі склала 2.0011e-04 (відносна похибка 0,02%).

Отримані розіграші результуючого показника tconst знаходяться в інтервалі [63.13…5773.3 годин]. Як і в попередньому випадку, значення показника $T_{MBAS2.1}$const аналітичної моделі розташоване приблизно посередині між значеннями mean(tconst) і max(tconst) імітаційної моделі. Розподіл випадкової величини tconst представлено на рисунку 3.10.

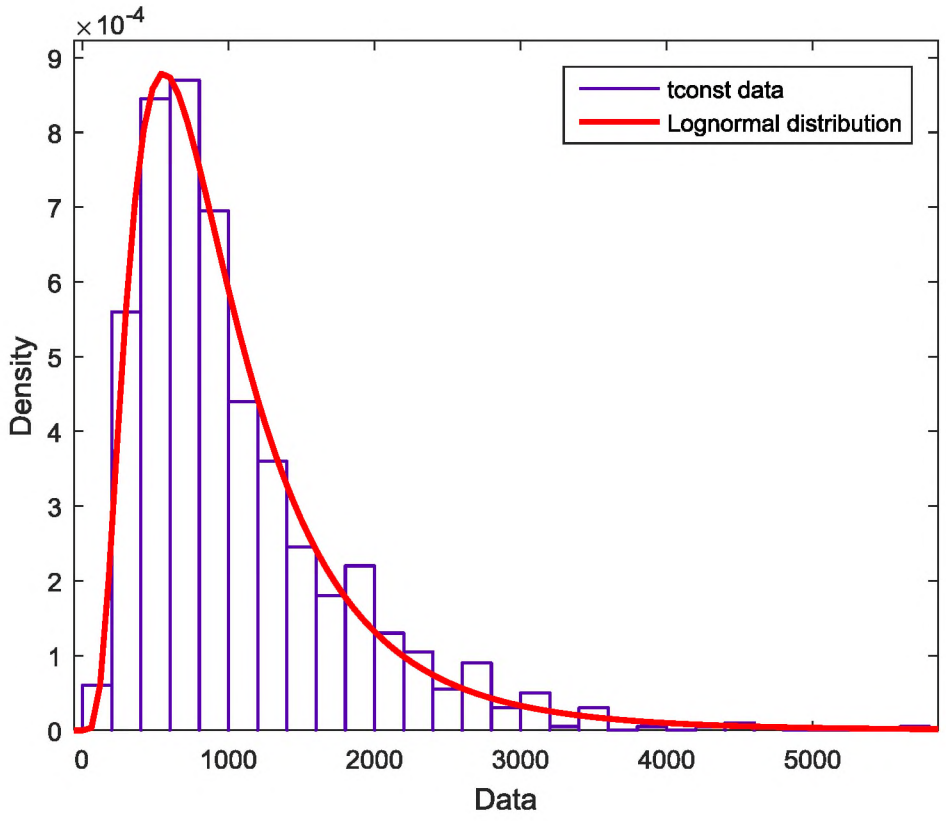


Рисунок 3.10 – Функція розподілу щільності ймовірностей результуючого показника tconst та її представлення логнормальним розподілом

У результаті виконання функції fitmethis(tconst) для випадкових реалізацій tconst був підібраний логнормальний розподіл з функцією:

$$f(t)=\frac{1}{t\cdot\sigma\cdot\sqrt{2\pi}}e^{-\frac{(\ln t-\mu)^2}{2\sigma^2}} \quad (3.5)$$

Параметри розподілу були визначені методом максимальної правдоподібності (mle), і їх значення склали: mu=6.75208 (з похибкою 0.0208161) і sigma=0.658262 (з похибкою 0.0147302). Функція логнормального розподілу для вказаних значень параметрів показана на рисунку 3.7.

Приймаючи до уваги значення похибки, для mean(tconst)=1056,2 годин значення функції готовності аналітичної моделі складе Pga=0,9751. Це відповідає визначенню T_2const з похибкою 4.7166e-04.

Відповідно, для max(tconst)=5773,3 годин, значення функції готовності аналітичної моделі складе Pga=0,9756. Це відповідає визначенню $T_{MBAS2.1}$const з похибкою 5.1651e-07.

Другий результуючий показник Amin розіграний на множині [0.0362…0.9794]. Функцію щільності розподілу розіграшів величини Amin, представлено на рисунку 3.11.

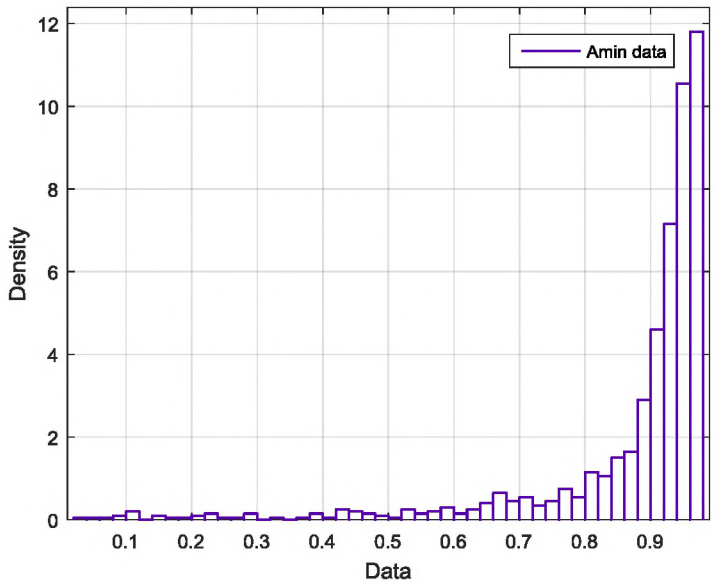


Рисунок 3.11 – Функція розподілу щільності ймовірностей розіграшів результуючого показника Amin моделі з урахуванням проведення обмеженої кількості загальних обслуговувань

Усереднене значення mean(Amin)=0.8761 відносно мінімуму функції готовності аналітичної моделі A_2min=0.9619 має абсолютну похибку 0.0859, або відносну похибку в 8,6%.

3.5 Порівняльний аналіз результатів аналітичного й імітаційного моделювання вебсервісу в умовах прояву дефектів та атак на його вразливості

Результати порівняння імітаційних і аналітичних моделей, що представлені на рис. 3.4 і рис. 3.7, ілюструють однакову динаміку зміни функції готовності. Як показали результати випробувань, отримані графіки функції готовності мають характерні фази розвитку: перехідну на початковому етапі функціонування системи і фазу переходу в усталений (стаціонарний) стан. Для імітаційних моделей характерно заниження значень функції готовності на перехідному етапі, але це явище пояснюється тим, що значення функції готовності на часових інтервалах, близьких до початку відліку t=0, мають яскраво виражений ненормальний розподіл з розташуванням максимуму, близьким до 1 (рис. 3.6, рис. 3.9). Як показали розрахунки для моделей вебсервісу, розіграші результуючого показника tconst мають ненормальний розподіл зі зміщеним вліво максимумом. Усереднене значення розіграшів mean(tconst) відповідає аналогічному результуючому показнику аналітичної моделі, визначеному з точністю порядка 10^{-4}; а максимальне значення вибірки max(tconst) відповідає показнику аналітичної моделі, визначеному з точністю порядка 10^{-7}.

Результати порівняння всіх імітаційних моделей з аналітичними за результуючим показником A_{MBAS}min зведені у таблицю 3.1.

Результати імітаційного моделювання для всіх розглянутих моделей вебсервісів показали задовільне сходження з відповідними аналітичними моделями. Найменші похибки відхилення показника A_{MBAS}min (аналітична модель) від усередненого на множині Amin або максимального на множині Amin (імітаційна модель) спостерігаються в системі без обслуговувань; а максимальні

похибки – в системах із загальним обслуговуванням. Загалом, похибка визначення результуючого показника A_{MBAS}min не перевищує 9,8% відносно mean(Amin) і 4,7% відносно max(Amin).

Таблиця 3.1 – Порівняння результатів аналітичного і імітаційного моделювання за показником A_1min

№ моделі	A_1min	mean (Amin)	Δ	ξ, %	max (Amin)	Δ	ξ, %
1	0,9964	0,9800	0,0164	1,7	0,9996	0,0032	0,3
2	0,9619	0,8761	0,0858	9,8	0,9828	0,0209	2,2
3	0,9786	0,9281	0,0505	5,4	0,9867	0,0081	0,8
4	0,9547	0,8730	0,0817	9,4	0,9994	0,0447	4,7
5	0,9693	0,9009	0,0684	7,6	0,9997	0,0304	3,1

Результати порівняння всіх імітаційних моделей з аналітичними за результуючим показником T_2const зведені у таблицю 3.2.

Таблиця 3.2 – Порівняння результатів аналітичного й імітаційного моделювання за показником T_2const

№ моделі	T_{MBAS}const, годин	mean (tconst)	ε	max (tconst)	ε	Опт. закон розподілу
1	28117	9648,1	2.09e-04	51576	1.4435e-07	gamma
2	3935,36	1056,2	4.7166e-04	5773,3	5.1651e-07	lognormal
3	16810	5.454,2	2.6902e-04	25445	1.9118e-06	gamma
4	22810	5477,7	2.0197e-04	44081	2.1758e-07	lognormal
5	21683,4	6162,5	1.9552e-04	41542	2.6956e-07	gamma

За показником T_2const результати імітаційного моделювання для всіх розглянутих моделей показали задовільне сходження. Найменшу похибку визначення часу переходу в усталений стан при використанні усереднення на множині tconst показала модель з обмеженим роздільним обслуговуванням. При використанні максимальних значень множини tconst, найменшу похибку показала модель без обслуговувань. Імітаційні моделі систем із загальним обслуговуванням сформували множини розіграшів показника tconst, розподіленого за

логнормальним законом; в решті моделей оптимальний закон визначений як гамма-розподіл. У загальному випадку, при використанні імітаційних моделей час переходу функції готовності в усталений режим може бути визначений з похибкою не більше $4,7*10^{-4}$ при використанні усередненої оцінки mean(tconst), і не більше $5,1*10^{-7}$ при використанні максимальної оцінки max(tconst).

Результати порівняння всіх імітаційних моделей з аналітичними за результуючим показником A_2const зведені у таблицю 3.3.

Результати імітаційного моделювання для систем з необмеженим обслуговуванням показали задовільну збіжність. В загальному випадку, при використанні імітаційних моделей значення функції готовності в сталому режимі може бути визначено з похибкою не краще $2,4*10^{-4}$.

Таблиця 3.3 – Порівняння результатів аналітичного й імітаційного моделювання за показником A_2const

№ моделі	A_{MBAS}const	Aconst	Δ	ξ, %
1	1	1	-	-
2	0,97561	0,97537	2,40E-04	2,46E-02
3	0,98522	0,98519	2,68E-05	2,72E-03
4	1	1	-	-
5	1	1	-	-

Таким чином, результати імітаційного моделювання показали високу збіжність значень функції готовності в усталеному режимі з відповідними аналітичними моделями. Було прийнято рішення надалі відмовиться від статистичного оцінювання функції готовності, оскільки воно не дозволяє однозначно отримати значення результуючих показників T_2const і A_2min. Однак застосування методів Монте-Карло незамінне при дослідженні моделей функціонування вебсервісів з немарковськими потоками відмов і відновлень. Розгляд цього класу задач виходить за рамки досліджень даної роботи.

3.6 Оцінювання економічної ефективності функціонування інформаційної системи з вебкомпонентами

Зазвичай, економічний ефект E системи визначається різницею між прибутками від її використання та витратами $G_П$, пов'язаними із її функціонуванням [41]. Отже, формула для оцінки економічної ефективності інформаційної системи з вебкомпонентами виглядає наступним чином:

$$E = p_{об} \cdot \lambda \cdot c \cdot T - G_П, \tag{3.6}$$

де c – середній економічний ефект, отриманий під час обслуговування однієї заявки;

T – інтервал часу, на якому досліджується функціонування системи;

$G_П$ – величина втрат у системі.

У контексті системи обслуговування з відмовами втрати включають в себе:

- витрати на експлуатацію системи,
- збитки через небажане залишення заявок необслуженими у системі,
- збитки внаслідок примусових простоїв у вебкомпонентах інформаційної системи.

Отже, для системи з відмовами обсяг втрат обчислюється за формулою [42]:

$$G_П = \left(q_K \cdot N + q_B \cdot p_B \cdot \lambda + q_{Пр} \cdot N_B\right) \cdot T, \tag{3.7}$$

де q_K – вартість експлуатації однієї вебкомпоненти інформаційної системи за одиницю часу;

q_B – вартість збитків внаслідок необслуговування заявок за одиницю часу;

$q_{Пр}$ – вартість одиниці часу простою компоненти інформаційної системи;

N_B – середня кількість вільних комопнент.

У системі обслуговування з обмеженою чергою заявок втрати пов'язані з:

- витратами на експлуатацію,
- збитками від неприйняття заявок,

– збитками від простої пристроїв,

– збитками від втрати заявок.

Отже, обсяг втрат $G_П$ у такій системі обчислюється за формулою [43]:

$$G_П = \left(q_K \cdot N + q_O \cdot K_O + q_B \cdot p_B \cdot \lambda + q_{Пр} \cdot N_B\right) \cdot T, \tag{3.8}$$

де q_o – вартість збитків, пов'язаних з простоєм черги за одиницю часу.

Для системи обслуговування з очікуванням у необмеженій черзі втрати пов'язані з:

– збитками від простою черги;

– збитками від простою приладів.

Таким чином, величина $G_П$ втрат у системи обслуговування з очікуванням у необмеженій черзі буде обчислюватися за формулою:

$$G_П = \left(q_K \cdot N + q_O \cdot K_O + q_{Пр} \cdot N_B\right) \cdot T. \tag{3.9}$$

Можна зауважити, що крім зазначених показників, таких як товарообіг, дохід і прибуток, як критеріїв економічної ефективності можуть вибиратися інші показники. Оптимальні значення параметрів системи обслуговування можуть бути досягнуті шляхом максимізації цих показників.

Наприклад, при вхідних даних [44], де інтенсивність обслуговування заявок вебсервісом складає 25,3 заявки/годину, інтенсивність заявок до інформаційної системи в годину пік – 500 заявок/годину, в інший час – 85 заявок/годину, та кількість місць у черзі – 2, були отримані такі результати. Для сценарію з інтенсивністю заявок до інформаційної системи 500 заявок/годину, оптимальна кількість вебкомпонентів за моделлю становить 29. Імовірність відмови в обслуговуванні складає 0,5%, середня довжина черги – 0,02 заявки, а середнє завантаження вебсервера – 70%. У випадку з інтенсивністю заявок до інформаційної системи 85 заявок/годину, оптимальна кількість вебкомпонентів, обчислена моделлю, становить 10. Імовірність відмови в обслуговуванні – 1,5%, середня довжина черги – 0,05 заявки, а середнє завантаження вебсервера – 66%.

Результати моделювання свідчать, що система повністю обслуговує всі вхідні заявки. Однак, незважаючи на мінімальні витрати, у обох випадках вона функціонує з неповним завантаженням.

Спроба встановити статистичні параметри при повному завантаженні інформаційної системи принесла такі результати. Повне навантаження системи за інтенсивності вхідних заявок 500 заявок/годину можливе при використанні 19 обслуговуючих вебкомпонентів. Проте при цьому ймовірність відмови в обслуговуванні зросте до 14%, що викличе збільшення витрат на чергування заявок та втрату неприйнятих клієнтів. Подібна ситуація спостерігається і при інтенсивності вхідних заявок 85 заявок/годину, коли для досягнення повного навантаження системи потрібно n=6 обслуговуючих вебкомпонентів. Тут ймовірність відмови вже становить 22%.

Висновки до розділу 3

У третьому розділі було розроблено метод визначення кількості розіграшів імітаційної моделі. Побудовано та досліджено імітаційні моделі вебсервісів, шо функціонують в умовах прояву дефектів АЗ та ПЗ, оновлення ПЗ, атак на вразливості компонент, проведення загального та роздільного обслуговування. Результати імітаційного моделювання показали високу збіжність значень функції готовності в усталеному режимі з відповідними аналітичними моделями. Застосування метода Монте-Карло дозволяє відстежити динаміку зміни функції готовності, аналогічну аналітичним моделям. Закон розподілу результуючого показника часу переходу в усталений режим має максимум, зміщений вліво; а пошук його оптимального (за критерієм “Loglikelihood → min”) закону розподілу не дав однозначного рішення, оскільки визначені гамма- і логнормальний закони. Тому було прийнято рішення відмовитися від статистичного оцінювання функції готовності вебсервісів, оскільки воно не дозволяє однозначно та із заданою точністю визначити значення результуючих показників T_2const і A_2min.

ВИСНОВКИ

У роботі була поставлена і вирішена актуальна наукова задача розробки аналітичних та імітаційних моделей вебсервісів інформаційних систем для оцінювання їх готовності з врахуванням атак на них.

У першому розділі виконано аналіз сучасного стану галузі розробки та використання інформаційних систем з вебкомпонентами. У процесі проведеного аналізу предметної області встановлено, що у зв'язку зі збільшенням складності систем вебсервісів і специфікою вимог до них зросла необхідність в моделюванні, розрахунку і розробці більш надійних систем і компонентів інфраструктур.

У другому розділі розглянуто моделі архітектур відмовостійких інформаційних систем та їх вебсервісів. На їх основі можуть бути отримані моделі для детального дослідження відмовостійких сервіс-орієнтованих систем. Для цього необхідно вибрати відповідну базову модель, деталізувати марковський граф і дані для моделювання. Виконана побудова марковського графа для моделі композитних сервісів. Розрахований коефіцієнт готовності, а також виконаний аналіз впливу вхідних параметрів на його стале значення. Результати моделювання показали, що найбільш ефективними методами підвищення надійності сервіс-орієнтованої системи є поліпшення якості роботи контролюючого модуля, що виявляє приховані відмови, а також оптимальний вибір інтенсивності профілактики прихованих відмов системи.

У третьому розділі було розроблено метод визначення кількості розіграшів імітаційної моделі. Побудовано та досліджено імітаційні моделі вебсервісів, шо функціонують в умовах прояву дефектів АЗ та ПЗ, оновлення ПЗ, атак на вразливості компонент, проведення загального та роздільного обслуговування. Результати імітаційного моделювання показали високу збіжність значень функції готовності в усталеному режимі з відповідними аналітичними моделями. Застосування метода Монте-Карло дозволяє відстежити динаміку зміни функції готовності, аналогічну аналітичним моделям. Однак при цьому крива готовності імітаційної моделі на початковому етапі показує занижені результати внаслідок

розподілу розіграшів з правостороннім максимумом. Закон розподілу результуючого показника часу переходу в усталений режим має максимум, зміщений вліво; а пошук його оптимального (за критерієм “Loglikelihood → min”) закону розподілу не дав однозначного рішення, оскільки визначені гамма- і логнормальний закони. Тому було прийнято рішення відмовитися від статистичного оцінювання функції готовності вебсервісів, оскільки воно не дозволяє однозначно та із заданою точністю визначити значення результуючих показників T_2const і A_2min.

Виконано розрахунок економічної ефективності роботи інформаційної системи з вебкомпонентами, враховуючи втрати від неприйнятих заявок та простоїв черги. Повне навантаження системи за умови інтенсивності вхідних заявок 500 заявок/годину можливе при використанні 19 обслуговуючих вебкомпонент. Однак у такому випадку ймовірність відмови в обслуговуванні збільшиться до 14%, що призведе до зростання витрат через збільшення черги заявок та втрату необслужених клієнтів.

Таким чином, поставлені задачі розв‘язано у повному обсязі. Напрямками подальших досліджень є:

– розробка та дослідження імітаційних моделей композитних вебсервісів;

– розробка та дослідження моделей вебсервісів, що функціонують в умовах впливу непуасонівських потоків подій.