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: **«Вибір композиції вебсервісів для системи із сервісноорієнтованою архітектурою»**

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи та технології
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти магістр
групи 126ІСТ_мд_22
Линюк О.І.
Керівник: Поночовний Ю.Л.
Рецензент: Брикун О.М.

Полтава – 2023 року

ВСТУП

Актуальність теми. Основні положення сервісноорієнтованої архітектури (СОА) відкривають шлях до нових архітектурних рішень у створенні сучасних інформаційних систем. Підприємства сьогодення зіткнулися з рядом викликів: зростання обсягу даних, потреба у партнерських та внутрішніх інтеграціях, апдейт програмних рішень, що співіснують із перевіреними часом системами [1]. Сервісорієнтована архітектура стала одним із варіантів вирішення цих проблем.

Така архітектура підходить до більш відкритих стандартів, не залежить від певних програмних або апаратних рішень, акцентує увагу на мережевій взаємодії, використовує машинні описи сервісів [2]. Наразі СОА широко застосовується у великих інформаційних системах підприємств і бізнесу, таких як Ощадбанк, Приватбанк, eBay, Adobe, HP та інші. Відкритість цього підходу обумовила появу декількох платформ для організації СОА з відкритим кодом, що дозволяє меншим компаніям використовувати нові можливості.

Основна ідея СОА полягає в злитті автономних вебсервісів для вирішення конкретних завдань [3, 4]. Інформаційні системи сучасності користуються як власними, що підконтрольні, так і зовнішніми сервісами, що представлені як «чорні ящики» для споживачів послуг. Часто кілька вебсервісів надають одну й ту ж функцію, наприклад, картографію, інформацію про географічні об'єкти, мультимедійний контент тощо.

Це створює два взаємопов'язаних завдання при проєктуванні систем СОА: обдуманий вибір композиції вебсервісів для розв'язання завдань та розробка критеріїв порівняння цих сервісів [5]. Раціональний вибір враховує усю доступну інформацію для досягнення найкращих результатів. Існує велика кількість стандартів (понад 80), що описують різні аспекти роботи систем з СОА, проте питання відбору та формування значень критеріїв порівняння при проєктуванні систем з СОА, що використовують зовнішні сервіси, залишається актуальним.

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 «Обгрунтування показників оцінювання гарантоздатності розподілених інформаційних систем»).

Метою кваліфікаційної роботи є вдосконалення моделі системи з сервісноорієнтованою архітектурою для вирішення завдання багатокритеріального вибору композицій вебсервісів з урахуванням переваг особи, яка приймає рішення.

Завданнями кваліфікаційної роботи є:

– аналіз особливостей проблеми раціонального вибору вебсервісів, а також сучасних методів, моделей та програмних засобів її вирішення;

– розробка алгоритму раціонального вибору композиції вебсервісів;

– розробка моделі вебсервісу, яка включає критерії порівняння, що вимірюються.

Об'єктом дослідження є процеси проєктування вебсистем з сервісноорієнтованою архітектурою.

Предметом дослідження є модель вебсервісу підтримки процесів вибору компонент сервісноорієнтованої архітектури.

Методи дослідження – проведені в роботі дослідження базуються на методах теорії ймовірності, системного і марковського аналізу, математичної статистики, машинного навчання, теорії прийняття рішень, теорії важливості критеріїв, багатокритеріального аналізу.

Інформаційна база кваліфікаційної роботи складається з наукових статей, міжнародних аналітичних видань і звітів, матеріалів наукових конференцій інтернет-ресурсів, що місять інформацію про архітектуру сучасних вебсистем, а також даних, отриманих від провідних IT-компаній у сфері проєктування та впровадження різних вебсервісів.

Елементи наукової новизни полягають у вдосконаленні параметричної моделі вебсервісу на основі критеріїв, що характеризують продуктивність та надійність, та автоматизованих методів оцінки їх значень.

Практична значущість роботи полягає в можливості повторного застосування та модифікації розробленого програмного коду моделі для оцінювання показників якості функціонування сервісноорієнтованих архітектурних рішень. Сформульовані в роботі моделі та методи можуть бути корисними для розвитку галузі системної інженерії, теорії побудови великомасштабних інформаційних систем, а також у сфері штучного інтелекту як приклад використання методів машинного навчання.

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

Публікації. За результатами проведеного дослідження опубліковано тези: «Організація та планування завдань у віртуалізованих системах для розподілених обчислень», Матеріали XII Міжнар. наук. конференції «Інформаційні технології в енергетиці та агропромисловому комплексі», м. Львів, 04-06 жовтня 2023 р.

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 61 сторінку формату А4. Вона містить 15 рисунків і 6 таблиць. У роботі використано 43 науково-технічних джерела.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ТА ТЕНДЕНЦІЙ РОЗВИТКУ ПРОБЛЕМИ РАЦІОНАЛЬНОГО ВИБОРУ КОМПОЗИЦІЇ ВЕБСЕРВІСІВ

1.1 Аналіз основної концепції сервісноорієнтованої архітектури

За останні роки розвитку галузі інформаційних технологій (ІТ) намітився перехід від всеосяжних інформаційних систем (ІС), що включають усі необхідні для роботи дані та функції, до розподілених систем, що використовують зовнішні ресурси [5]. Такий перехід виправданий наступними факторами.

1. Зростання обсягу даних. Одна система часто вже не здатна зберігати весь обсяг необхідних даних та забезпечувати прийнятний рівень продуктивності обчислень.

2. Інтеграція з партнерами та новими підрозділами. Злиття та поглинання організацією веде до потреби поєднання гетерогенних систем та додатків, що функціонують за різними (часто пропрієтарними) протоколами.

3. Оновлення платформи підприємства. Багато організацій не бажають відмовлятися від перевірених та надійних технологій, проте розвиток бізнесу призводить до необхідності використання нових програмних засобів, часто несумісних із наявною ІС [6].

Одним із сучасних рішень зазначених завдань є побудова систем на базі сервісноорієнтованої архітектури (СОА). СОА, згідно [7], – це тип архітектури розподілених систем, що характеризується такими принципами:

1. Логічне уявлення. Сервіс є абстрактним, логічним поданням програм, баз даних, бізнес-процесора тощо. Подання визначено у термінах фактично виконуваних дій, зазвичай операцій рівня бізнес-логіки.

2. Орієнтованість на обмін повідомленнями. Сервіс визначається повідомленнями, якими він обмінюється із постачальниками та споживачами послуг. Внутрішня структура сервісу (наприклад, мова програмування, з допомогою якої реалізовано сервіс, структура процесів, схема БД) навмисно невідома: споживач сервісу повинен знати деталі реалізації.

3. Орієнтованість на машиннозалежний опис. Сервіс описується метаданими, доступними для машинної обробки. Такий опис відповідає публічній якості СОА: лише ті деталі, які мають бути відомі споживачеві послуг, повинні бути включені в опис.

4. Ступінь деталізації. Сервіси повинні мати якомога менше доступних операцій для роботи з відносно великими та складними повідомленнями.

5. Орієнтованість на мережеву взаємодію. Доступ до сервісів, як правило, реалізований за допомогою комп'ютерної мережі, однак це не є обов'язковим.

6. Незалежність від платформ. Повідомлення, якими обмінюються сервіси за інтерфейсами, надсилаються у платформонезалежному, стандартизованому форматі. XML (англ. eXtensible Markup Language (розширювана мова розмітки) є найбільш наочним форматом, який задовольняє це обмеження.

СОА ґрунтується на ряді стандартів, прийнятими основними постачальниками послуг у сфері інформаційних технологій: IBM, Oracle, HP, Dell, Microsoft та ін. За рахунок цього існуючі системи з СОА можна прозоро об'єднувати та поєднувати в рамках стандартизованих процедур.

The Organization for the Advancement of Structured Information Standards (OASIS) у документі «Reference Model for Service Oriented Architecture» [8] визначає СОА, як «парадигму для організації та використання розподілених можливостей, які у свою чергу можуть бути керовані різними постачальниками». Визначення The Open Group Standard в «SOA Governance Framework» [9]: «СОА – архітектурний підхід, що сприяє скороченню відстані між бізнесом та IT шляхом надання доцільної бізнес функціональності у своєчасній та ефективній манері».

На рис. 1.1 представлена еталонна модель СОА [10], що відображає концептуальний рівень її рішень. Ця модель, або «багатошарова архітектура СОА», включає такі поняття, як бізнес-процес, сервіс, сервісний компонент, а також взаємозв'язки між ними [10].

Структура складається з 5 функціональних шарів [11, 12]:

– експлуатаційні системи: представляють існуючі IT-рішення, показують

цінність та важливість вкладень у IT для СОА та можливість їх використання;

– сервісні компоненти: реалізують сервіси, при цьому можуть використовувати одну або більше програм із рівня експлуатаційних систем; користувачі та бізнес-процеси, на відміну від сервісів, не мають прямого доступу до компонентів, а існуючі компоненти можуть бути повторно використані або, якщо вони для цього підходять, впроваджені у СОА;

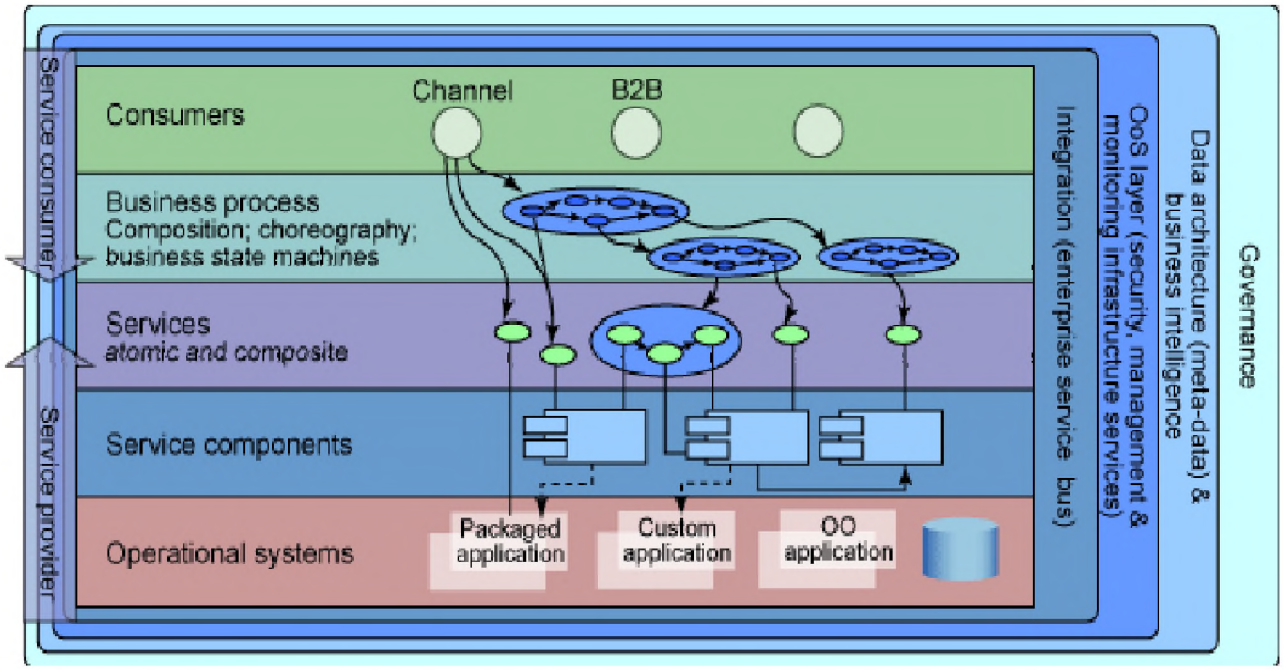

Рисунок 1.1 – Графік зміни загальної кількості обслуговуваних комп‘ютерів

– сервіси: представляють розміщені серед сервіси. Ці послуги є керованими видимими сутностями;

– бізнес-процес: представляє операційні програми, що створюють бізнес-процеси у вигляді хореографій сервісів;

– користувачі: представляють канали, які використовуються для доступу до бізнес-процесів, сервісів та програм.

Дана структура інтегрує усі аспекти функціонування ІС та забезпечує ефективну взаємодію між користувачами, бізнес-процесами та компонентами для досягнення максимальної продуктивності та ефективності. Поняття вебсервісу, як основного елемента СОА представлено у наступному підрозділі.

1.2 Призначення та функції вебсервісів

Відповідно до [7] вебсервіс – це абстрактне поняття, яке має бути реалізовано конкретним агентом. Агент – це конкретний апаратний або програмний засіб, який відправляє та приймає повідомлення, сервіс – це ресурс, що характеризується абстрактним набором функцій, що надаються. Таким чином, певний вебсервіс може бути реалізований декількома агентами, які надають однаковий набір функцій.

З технічного боку, вебсервіс [7, 16] – це програмна система, спроєктована з метою надання можливості взаємодії між машинами через комп'ютерну мережу. Вебсервіс має інтерфейс доступу, описаний у машинно-оброблюваному форматі (зазвичай це WebService Description Language, WSDL). Інші системи взаємодіють із вебсервісом через інтерфейс засобами обміну повідомленнями у стандартизованому форматі за протоколом HTTP.

Основні ідеї, що реалізуються вебсервісами [10, 13]:

– орієнтація на бізнес: послуги орієнтуються не на можливості IT, а на потреби бізнесу, орієнтація сервісів на бізнес підтримується аналізом сервісу та технікою проєктування;

– інструкції: сервіси самодостатні та описуються в термінах інтерфейсів, операцій, семантики, динамічних характеристик, політик та властивостей сервісу;

– повторне використання: повторне використання сервісів забезпечується їх модульним плануванням;

– угоди: сервісні угоди укладаються між сутностями, які називаються постачальниками та користувачами; Ці угоди ґрунтуються на сервісних інструкціях та не впливають на реалізацію самих сервісів;

– розміщення та видимість: протягом свого життєвого циклу сервіси розміщуються та стають видимими завдяки сервісним метаданим, реєстрам та сховищам;

– агрегація: на слабко пов'язаних сервісах будуються об'єднуючі бізнес-процеси та складні програми для одного або кількох підприємств.

Ці принципи відображають стратегічне спрямування вебсервісів на досягнення бізнес-цілей та визначають їхню самостійність, стандартизацію та можливості повторного використання. Побудовані на цих принципах вебсервіси забезпечують надійну основу для створення ефективних бізнес-процесів та програмних рішень у рамках СОА.

1.3 Сервісноорієнтована архітектура як тенденція розвитку інформаційних систем

Кожна нова архітектура розробляється внаслідок підвищення складності завдань: розмір інформації щороку подвоюється, кількість користувачів систем постійно збільшується, все більше вузькоспеціалізованих сервісів, що працюють на різних платформах і т.д. СОА на даний момент є рішенням, що найбільш швидко розвивається, для великих і середніх підприємств. В історії розвитку архітектур ІС можна виділити низку основних етапів [14, 15]:

– централізована архітектура;

– мейнфрейм;

– дворівнева архітектура (клієнт-сервер);

– трирівнева архітектура (клієнт – вебсервер – сервер баз даних (БД));

– багатоланкова архітектура (клієнт – вебсервер – сервер додатків – сервер БД);

– розподілена архітектура;

– однорангова архітектура (Peer to Peer, P2P);

– сервісноорієнтована архітектура;

– сервіснокомпонентна архітектура;

Сервіснокомпонентна архітектура (Service Component Architecture, SCA, СКА) – набір специфікацій, розроблений для опису моделі створення додатків та систем на базі сервісноорієнтованої архітектури [19]. СКА надає модель побудови додатків, що складаються із сукупності сервісів та бізнес-функціоналу. Таким

чином, СКА є надбудовою на СОА, тому дослідження СОА корисні і застосовні для СКА.

Дедалі більше компаній та інтеграторів використовують переваги СОА у своєму робочому процесі [20-22]. Серед великих організацій, які модернізували свої системи за допомогою СОА, можна відзначити: Ощадбанк [23], Райффайзенбанк [24], Приватбанк [25], Ebay [26], та ін.

Серед постачальників ПЗ для організації систем з СОА можна виділити: Oracle, IBM, HP, Microsoft, SAP та ін. Існує близько 15 реалізацій елементів СОА з відкритим вихідним кодом (open -source) таких як: WSO2 Platform, Mule Platform, Apache ServiceMix, Petals ESB та ін.

Згідно з дослідженням [29], проведеним у 2021 році, ринок послуг, пов'язаних із СОА, за рік зріс на 38% і на 2021 рік склав 5.5 млрд. доларів.

Одним з індикаторів підвищення інтересу до СОА може бути кількість вакансій, пов'язаних з СОА. З 2015 по 2022 частка вакансій із ключовим словом «SOA» зросла на 0,5% щодо загальної кількості вакансій у сфері IT.

Основним рушійним фактором є потенціал СОА для [6]:

– підвищення окупності (Return On Investment, ROI) завдяки зменшеній вартості реалізації за рахунок впровадження стандартів, повторного використання, відображення сервісів та інтеграції з партнерами;

– зменшення часу виходу на ринок за рахунок повторного використання активів та впровадження сервісів, що надаються партнерами;

– поліпшення видимості інформаційних активів та їх узгодження з бізнес-цілями;

– підвищення гнучкості як внутрішньої (при взаємодії), так і зовнішньої (при роботі з партнерами);

– реалізації більш ефективних процесів за рахунок повторного використання інформаційних активів та стандартів;

– забезпечення гнучкості бізнес-діяльності та здатності легко та швидко адаптуватися до ринкових змін;

– зменшення витрат з організації.

З наведеного аналізу можна дійти невтішного висновку про всебічне зростання інтересу до технології СОА. З одного боку, великі підприємства всього світу знаходять вигоду в інтеграції СОА, з іншого боку, підтримка СОА з боку спільноти open-source дозволяє використовувати блага технології малим та середнім підприємствам.

Наукове співтовариство також зацікавлене у СОА та проводить дослідження неосвітлених стандартами її аспектів. Зокрема, одними з основних тем дослідження є тема оптимальної композиції сервісів та розробка критеріїв порівняння сервісів.

1.4 Аналіз стандартів, що регламентують галузь сервісноорієнтованої архітектури та постановка завдання

Існує близько 80 стандартів, що стосуються СОА та вебсервісів. Стандарти СОА можна розділити на кілька наступних категорій [11]. Стандарти, що описують еталонну модель (reference model). У документі OASIS «Reference Model for SOA» [28] еталонна модель окреслюється абстрактна основа розуміння істотних відносин між суб'єктами деякого довкілля. Еталонна модель включає мінімальний набір об'єднуючих концепцій, аксіом і зв'язків з будь-якою певною предметною областю, незалежною від конкретних стандартів, технологій, реалізацій або інших деталей. Стандарт [8] також офіційно було видано українською.

Стандарти, що описують еталонну архітектуру. У документі The Open Group «SOA Reference Architecture» [29] Еталонна архітектура визначена як багатошарова структура, заснована на узагальненні набору попередніх успішних рішень. Такі рішення узагальнені та структуровані для зображення логічної та фізичної архітектур, заснованих на збиранні шаблонів, що описують спостереження низки успішних впроваджень. Крім того, документ показує, як поєднувати ці впровадження з метою створення певного рішення. Також еталонна архітектура описується в документі OASIS «Architecture for SOA Foundation» [30], який у цілому містить схожі концепції та поняття, але в більш абстрактному описі.

Онтології СОА представляю стандартизовану мову для опису еталонних моделей та архітектур. Організація The Open Group підготувала документ « Service-Oriented Архітектура Ontology» [14, 31] в якому описані в т.ч. у машинно-читаному форматі OWL (Web Ontology Language) основні терміни, семантика та концепції.

Моделі зрілості (Maturity Models) описує можливі ступені інтеграції СОА, які може реалізувати організація, а також способи їх досягнення. Таку модель описує документ The Open Group «Service Integration Maturity Model» [15].

Мови моделювання включають метамодель і нотацію, які можуть бути використані для представлення артефактів за допомогою певних інструментів і автоматизованих середовищ. До таких мов належать розширення мови Unified Modeling Language (UML) від Object Managment Group (OMG) SysML [16] та SoaML [17]. Детальний огляд та порівняння регламентуючих документів СОА можна знайти в роботах [32] та [33].

Стандарти, що стосуються вебсервісів (також їх називають стандартами сімейства WS-*) поділяють на такі категорії:

– стандарти бізнес-процесів описують потенційний порядок виконання операцій у межах набору вебсервісів;

– стандарти управління описують питання доступності, продуктивності, статистика використання, керування та конфігурації вебсервісів;

– стандарти надійності стосуються гарантії доставки повідомлень;

– стандарти транзакційності описують процеси координації механізмів контролю операцій;

– стандарти захищеності стосуються питань захищеної взаємодії вебсервісів;

– опис та виявлення вебсервісів;

– обмін повідомленнями у розподіленому середовищі;

– стандарти транспортних каналів пропонують структуру для створення протоколів обміну даними між додатками.

Одним із мотивуючих факторів цієї роботи є відсутність у регламентуючих документах процедур вибору сервісів у рамках бізнес-процесів. Часто одна й та ж функція надається різними сервісами. Наприклад, інформаційний вебпортал про

розваги для роботи може користуватися картографічними, відео- та фотохостинговими функціями, сервісами авторизації та іншими. Сервіси картографування надаються Google Maps, Bing Maps (від Microsoft), Nokia Maps і т. д. Сервіси відеохостингу надаються YouTube, Vimeo, та ін.. Таким чином, кожна потрібна функція може бути реалізована як одним постачальником послуг, так і кількома різними.

У вебсервісів можуть бути різні характеристики: доступність, час обробки запитів, час відновлення після збою тощо. Системи, які використовують вебсервіси, можуть мати вимоги, описані вимогами до якості обслуговування (QoS) та/або угодою про рівень обслуговування (SLA) [34]. Таким чином, інженери, які розробляють систему з використанням СОА, стикаються з завданням раціонального вибору складу вебсервісів за певною метою.

Для реалізації такого вибору необхідно мати модель вебсервісу, що відображає його основні функції та характеристики якості обслуговування, а також методи оцінки цих характеристик. Також потрібен метод для раціонального вибору складу вебсервісів.

В іноземній літературі проблема вибору вебсервісів у системі з СОА позначається як «QoS-aware Service Selection», «Optimal Service Selection» або «Optimal Service Composition».

У роботі [31] розглядається застосування теорії нечітких чисел у задачі вибору сервісів для реалізації певних бізнес-процесів у рамках корпоративної інформаційної системи. Використовуючи термінологію, можна сказати, що в [31] виділяють ряд функціональних критеріїв (критерії реалізації бізнес-процесів) та нефункціональних. До нефункціональних відносяться оцінки економічних витрат: одноразових, періодичних та непрямих. Інші нефункціональні критерії не розглядаються. Оцінки якості реалізації функціональних вимог визначаються експертно. Загалом, у роботі в основному розглядається економічний аспект вибору вебсервісів, технічна сторона питання, що стосується надійності та продуктивності вебсервісів, практично не висвітлена.

Робота [35] має схожу тематику: представлений набір критеріїв та методи вибору найкращого проєкту, що реалізує СОА в ІС. Виділено 18 критеріїв, поділених на 4 групи. Описуються такі групи критеріїв: внутрішня організація ІС із СОА, організація змін та політик, готовність процесів та бізнес-процесів, мінімізація операційних ризиків. Як метод вибору використовується метод порогового агрегування та лінійна згортка. Основна відмінність від цієї дисертаційної роботи – вибір критеріїв порівняння: у роботі [32] критерії порівняння мають суб'єктивну природу, їх значення набуті неавтоматизованим шляхом, мають однорідну порядкову шкалу.

У роботах [36, 37] пропонуються онтологія та розширення мови XML, які повинні використовуватися: 1) провайдерами сервісів для опису якості обслуговування, 2) споживачами сервісів для опису необхідної якості обслуговування. Запропонована онтологія поділена на три рівні. За допомогою верхнього рівня описуються відомості про основні концепції, пов'язані з вимірюванням показників якості обслуговування, таких як спосіб оцінки показника (суб'єктивний або об'єктивний), наявність кореляцій між показниками, що розглядаються і т.д. За допомогою середнього рівня онтології описуються відомості про найбільш значущі на думку авторів показники якості обслуговування розподілених систем, такі як доступність сервісу, продуктивність, надійність і т.д. Нижній рівень призначений для опису відомостей про значення показників, специфічних для предметної сфери сервісу. Автори пропонують використовувати багатоагентну систему, у якій мають реєструватися провайдери сервісів. Передбачається, що вони надалі надсилатимуть їй значення показників якості обслуговування, а споживачі вебсервісів звертатимуться до неї для вибору найбільш відповідного вебсервісу. Пропонується спосіб вибору вебсервісу для двох показників якості обслуговування, що враховує кореляцію між показниками. Недоліком підходу є необ'єктивність оцінок показників якості, заснована на довірі до показників, що заявляються провайдерами.

У роботі [38] наводяться варіанти вирішення проблеми пошуку набору, або композиції, вебсервісів, що мінімізує загальний час виконання бізнес-процесу

щодо вартості та часу виконання. При цьому під бізнес-процесом розуміється послідовність викликів вебсервісів у рамках виконання певного завдання, що описується мовою виконання бізнес-процесів BPEL (Business Process Execution Language). У роботі представлено формальну постановку завдання у вигляді проблеми нелінійної оптимізації з обмеженнями за значенням вартості та часу виконання запиту. Також представлено два алгоритми, що вирішують поставлене завдання: точний алгоритм на базі повного перебору та евристичний алгоритм, що має меншу складність та меншу точність. У роботі враховуються лише дві нефункціональні вимоги: час виконання та вартість. Вирішується завдання лише мінімізації вимог, тобто. не враховуються можливі переваги.

У роботах [39, 40] вводиться новий критерій оцінки якості композиції вебсервісів – облік затримки обробки запитів, що виникає через географічну віддаленість вебсервісу від споживача. Пропонується метод на основі генетичного алгоритму вибору найкращої композиції з урахуванням географічної віддаленості. Мінусом підходу є фактична відсутність будь-якої інформації про затримки, що вносяться мережею. Автори використали дані за 2008 рік, отримані у рамках проєкту Planet Lab Trace Data Set. У цьому наборі даних міститься інформація про середню затримку між вузлами у складі мережі Planet Lab. Відповідно, достовірні розрахунки можуть бути здійснені лише якщо споживач та провайдер сервісу знаходяться на одному з вузлів мережі Planet Lab.

Висновки до розділу 1

У першому розділі виконано оцінку сучасного стану проблеми оптимальної композиції вебсервісів у рамках сервісноорієнтованої архітектури, підкреслено її актуальність та значущість. Вирішення цієї проблеми неможливе без використання моделювання вебсервісів, визначення ключових функціональних та нефункціональних характеристик, а також без створення моделі самої системи на основі сервісноорієнтованої архітектури.

Дотепер моделювання вебсервісів здійснювалось для оцінки їх статичних показників. Проте аналіз перспективної поведінки вебсервісів проведено недостатньо ретельно. Наприклад, певний сервіс може показувати гарні результати при окремих запитах, але при зростанні навантаження час обробки та кількість необроблених запитів можуть суттєво зрости. З іншого боку, правильне проєктування сервісів, застосування балансувальників навантаження, хмарних технологій може допомогти уникнути збільшення часу обробки запитів при зростанні навантаження. Таким чином, важливо мати такі характеристики вебсервісів, які відображають зміни їх продуктивності та надійності при збільшенні навантаження.

На основі проведеного аналізу сформульовані загальне завдання, яке полягає у вдосконаленні моделі системи з сервісноорієнтованою архітектурою для вирішення завдання багатокритеріального вибору композицій вебсервісів. Загальне завдання розділено на три часткові задачі, дві з яких розглянуті у наступних розділах.

РОЗДІЛ 2
АЛГОРИТМ РАЦІОНАЛЬНОГО ВИБОРУ КОМПОЗИЦІЇ ВЕБСЕРВІСІВ У СИСТЕМАХ ІЗ СЕРВІСНООРІЄНТОВАНОЮ АРХІТЕКТУРОЮ

2.1 Особливості технологічного процесу обробки даних при композиції вебсервісів

Часто одну і ту саму функцію надають різні вебсервіси. Наприклад, мобільний додаток, що відображає найближчі лікарні в рамках обробки одного запиту, може використовувати картографічний, геопозиційний, географічний та інформаційний сервіси з базою даних медичних установ (рис. 2.1). Постачальниками функцій геопозиціонування та картографії можуть бути сервіси Google Maps, Bing Maps (сервіс Microsoft), Nokia Maps та інші. Як географічні та інформаційні сервіси з БД медичних установ можливо використовувати різні регіональні каталоги, соціальні сервіси FourSquare, AlterGeo, Facebook places та ін. [42]. Використання кількох сервісів при обробці запиту називають композицією сервісів.

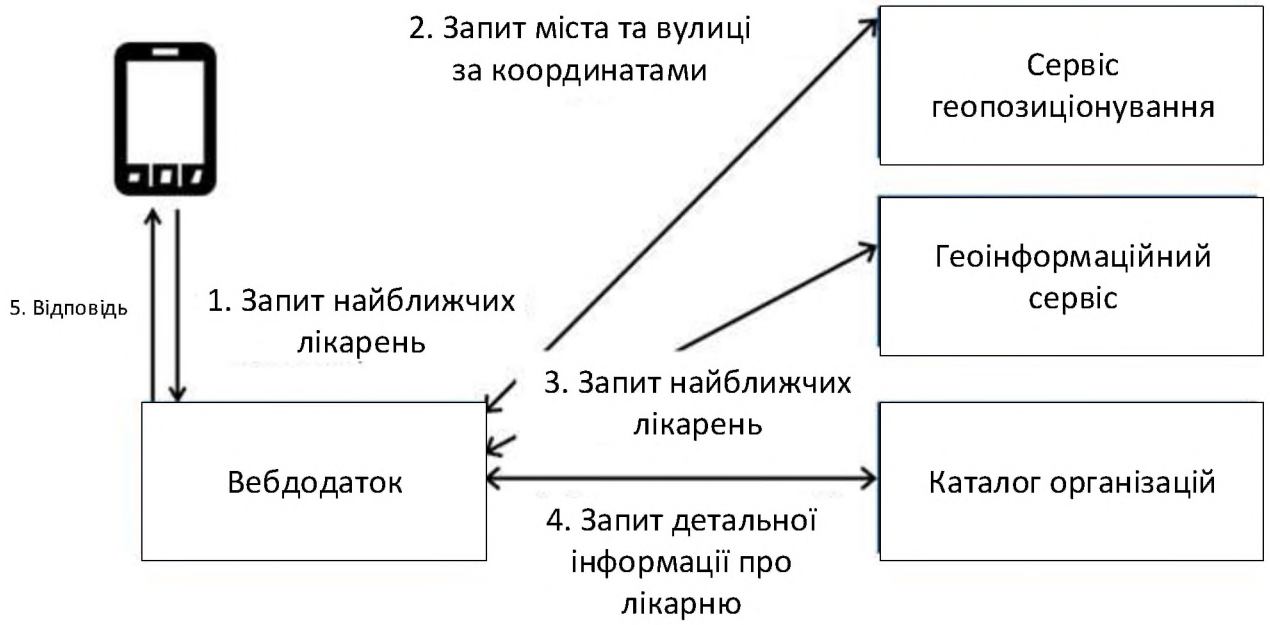

Рисунок 2.1 – Приклад композиції сервісів у рамках отримання списку найближчих лікарень

Вебсервіси мають різні параметри: доступність, час обробки запитів, час відновлення після збою і т.д [43]. Замовники систем, які використовують вебсервіси, можуть висувати вимоги, що описуються у політиках QoS та/або у документі SLA. Інженери, оцінюючи основні показники проєктованої системи, також можуть надавати перевагу швидкості роботи системи, її надійності, малому часу відновлення після збоїв і т.д. Таким чином, перед інженерами, що проєктують систему з СОА, ставиться багатокритеріальна задача вибору оптимальною за перевагами композиції вебсервісів у рамках певного завдання.

2.2 Формальна постановка задачі

У моделюванні завдання використовується нотація теорії важливості критеріїв [26]. Нотація сервісів та функцій має надрядковий індекс; нотація оцінок значень критеріїв – підрядковий індекс.

Введемо такі позначення (рис. 2.2):

– Завдання T складається з ряду підзавдань $t^{i,}$ $i = 1,N_{t,}$ де N_{t} – число підзавдань задачі T.

– $S^{i,}$ $i = 1,N_s$ – i-ий сервіс у реєстрі вебсервісів, де N_s – число вебсервісів у реєстрі.

– $S^{i} \cdot f^{j,}$ $j = 1,N_{fi}$ – j-а функція, що реалізується i-им вебсервісом, N_{fi} – Число функцій i-го сервісу.

– $Implement\ (t^{k)} = \{\ S^{\,i,}f^{\,j} \mid S^{\,i,}f^{\,j,} = t^{\,k}\ \}$ – множина можливих реалізацій підзавдання t^{k} у вигляді функцій сервісів $S^{\,i,}f^{\,j,}$

– $Implement\ (T) = \{Implement\ (t^{k)} \mid k = 1,N_t\}$ – множина можливих реалізацій задачі T, або можливі композиції сервісів у рамках задачі T.

– $K\ (S^{\,i,}f^{\,j)}$ – векторна оцінка i – го вебсервісу $K_{n,}$ за $n = 1,N_{\,k}$ критеріями.

Наприклад визначимо деякі критерії вибору функцій вебсервісів:

– $ServiceTime\ (S^{\,i,}f^{\,j)}$ – час обробки запиту;

– *Availability* ($S^{i}.f^{j}$) – доступність функцій;

– *Cost* ($S^{i}.f^{j}$) – вартість обробки запиту.

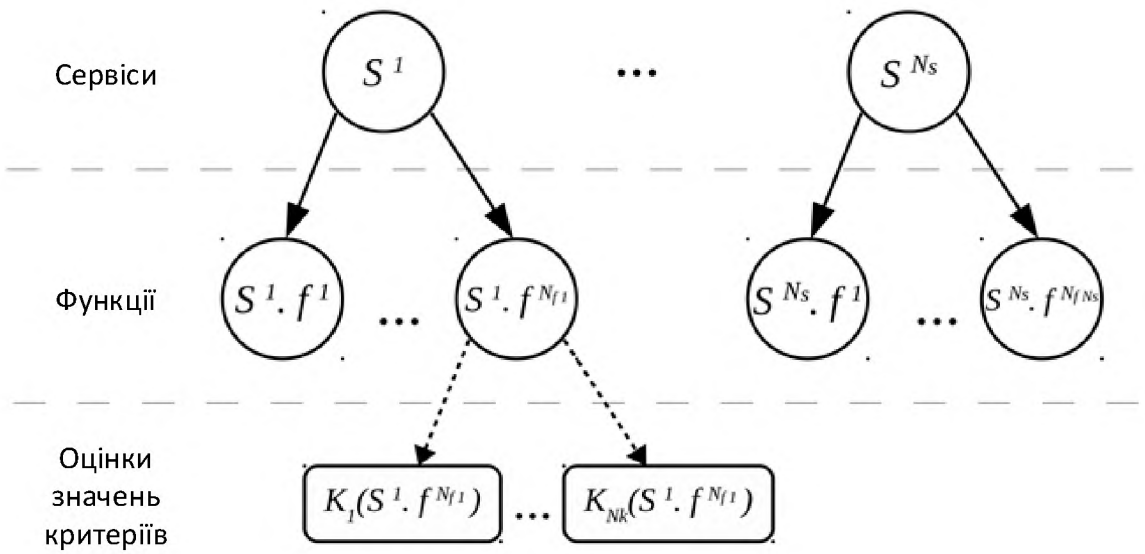

Рисунок 2.2 – Модель сервісів, функцій та їх критеріальних оцінок

Введемо нотацію вибору варіантів за перевагами:

– $\Omega = \{ServiceTime \succ Availability;\ Availability = Cost\}$ – позначення переваги критеріїв вибору від особи, що приймає рішення (ОПР). У прикладі вказується, що час обробки важливіше, ніж доступність, доступність рівноважна вартості. Можна дійти невтішного висновку, що час обробки також важливіше вартості.

– P_Ω – відношення переваги між двома варіантами (у нашому випадку, функціями вебсервісів). Наприклад, запис $S^1.f^1\ P_\Omega\ S^2.f^1$ означає, що функція f^1 сервісу S^1 краще функції f^1 сервісу S^2 з урахуванням інформації про перевагу Ω.

– Функція $S^i.f^{j\,*}$, що реалізує задачу t^k така, що для неї немає функції $S^i.f^j$, кращої по відношенню P_Ω, називається недомінованою, або оптимальною за Еджворт – Парето.

– *Implement* $(t^k)_\Omega$ – множина оптимальних за перевагами реалізації k-ої задачі.

– *Implement* $(T)_\Omega$ – множина оптимальних за перевагами Ω композицій сервісів у рамках завдання T.

$$Implement\ (T)_\Omega = \{Implement\ (t^k)_\Omega, k = 1, N_t\} \qquad (2.1)$$

Таким чином, завдання вибору можна сформулювати наступним чином: необхідно зробити вибір оптимальної за перевагами Ω композиції вебсервісів $Implement(T)_{\Omega}$ в рамках певного завдання T.

2.3 Алгоритм розв'язання задачі вибору

Для вирішення поставленого завдання розробки алгоритму вибору необхідно реалізувати такі етапи (рис. 3.3):

1. Для визначення кола вебсервісів та їх функцій, які братимуть участь у вирішенні завдання вибору, необхідно знайти множину реалізацій підзадач різними вебсервісами.

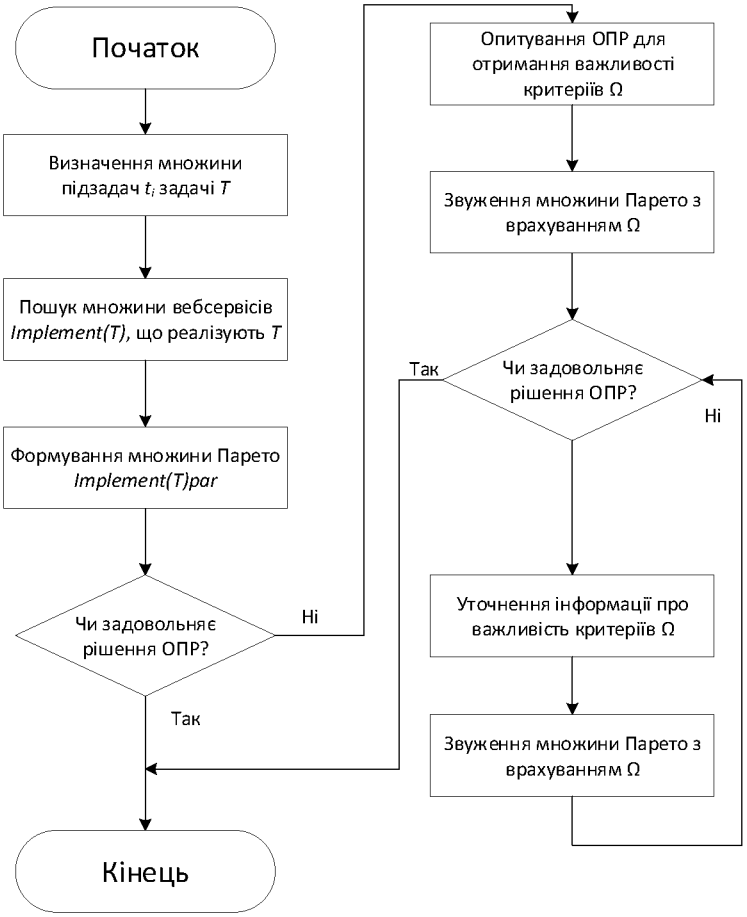

Рисунок 3.3 – Блок-схема алгоритму розв'язання задачі раціонального вибору композиції вебсервісів

2. Провести оцінку отриманої множини функцій вебсервісів за набором наявних критеріїв.

3. Сформувати множину Парето, або множину варіантів, що недомінують.

4. Отримана множина Парето може бути досить великою для прийняття будь-якого рішення, тому з метою його звуження необхідно визначити інформацію про кількісні переваги ОПР.

5. Звуження множини Парето за рахунок використання інформації про відносну важливість критеріїв з використанням ітераційної людино-машинної процедури.

На рис. 3.3 представлена блок-схема алгоритму, що описується. У наступних підрозділах будуть розглянуті методи реалізації кожного етапу алгоритму.

2.4 Формування множин можливих реалізацій підзавдань

На даний момент запропоновано декілька механізмів пошуку вебсервісів, які реалізують необхідні завдання. Серед них можна виділити групу стандартів, що належать до галузі семантичного опису вебсервісів (Semantic Web Services, SWS):

– Universal Description, Discovery and Integration (стандарт універсального опису, виявлення та інтеграції (UDDI)) – перший і до теперішнього часу стандарт, розроблений організацією OASIS, що не використовується, стандарт описує процедури розташування WSDL-описів вебсервісів та їх публікації для подальшого пошуку іншими організаціями [27].

– Ontology Web Language for Services (мова семантичного опису вебсервісів на базі OWL (OWL-S) – стандарт опису вебсервісу у вигляді онтології на базі мови побудови онтологій OWL (Ontology Web Language). Складається з трьох частин: профілю вебсервісу, що містить рекламний опис та дані для знаходження вебсервісу, модель процесів, що містить детальний опис функцій, що надаються, і базовий опис, що містить інформацію про способи взаємодії з вебсервісом за допомогою повідомлень [18].

– Semantic Annotations for WSDL (мова семантичного опису вебсервісів на базі WSDL (SAWSDL) – стандарт, що доповнює WSDL семантичними описами вхідних та вихідних даних, операцій та умов вебсервісу. SAWSDL передбачає використання OWLS як бази для семантичного опису вебсервісу. Багато в чому SAWSDL схожий на OWL-S з тією відмінністю, що зберігається WSDL-подібний (заснований XML) синтаксис опису вебсервісу [19].

Також можна відзначити інші стандартизовані мови семантичного опису вебсервісів, які можуть бути використані у вирішенні задачі пошуку необхідного вебсервісу: WSMO (Web Service Modeling Ontology, мова семантичного опису вебсервісів на базі онтологій) [20], WSML (Web Service Modeling Language, мова моделювання вебсервісів) [31], SWSL (Semantic Web Services Language, мова семантичного опису вебсервісів) [22], SA-REST (Semantic Annotation of Web Resources (семантичний опис вебресурсів) [13].

На даний момент запропоновано безліч методів пошуку необхідних вебсервісів (наприклад, [14]). Деякі методи формалізовані міжнародними організаціями стандартизації.

Зараз існує і розвивається велика кількість стандартизованих підходів до семантичного опису вебсервісів. Таке різноманіття насамперед пов'язане з популярністю ідей семантичного Інтернету та технології СОА. На жаль, на даний момент багато стандартизованих мов семантичного опису не знайшли застосування в загальнодоступному та корпоративному секторах (UDDI, SWSL та ін.), інші мови не мають належної технологічної підтримки та підтримки корпоративних користувачів (OWL-S, SAWSDL, WSMO) [18]. У зв'язку з тим, що повноцінна реалізація методу пошуку вебсервісів, що реалізують необхідне завдання, не є однією з цілей цієї роботи, надалі буде використано спрощений підхід.

Опишемо підхід на прикладі. Нехай завдання T складається з трьох послідовно виконуваних підзадач:

$$T = (t^1, t^2, t^3) \tag{2.2}$$

Репозитарій вебсервісів містить опис трьох сервісів:

$$\begin{gathered} S^1 = \{ S^1.f^1, S^1.f^2 \} \\ S^2 = \{ S^2.f^1, S^2.f^2, S^3.f^3 \} \\ S^3 = \{ S^3.f^1 \} \end{gathered} \tag{2.3}$$

Проаналізувавши документацію вебсервісів, можна скласти множини *Implement* ($t^{k)}$, $k = 1, 2, 3$:

$$\begin{gathered} Implement\ (t^{1)} = S^1.f^1, S^3.f^1 \\ Implement\ (t^{2)} = S^2.f^1 \\ Implement\ (t^{3)} = S^1.f^2, S^2.f^2 \end{gathered} \tag{2.4}$$

Таким чином, у подальшому при виборі оптимальної за перевагами композиції *Implement* $(T)_\Omega$ необхідно зробити оптимальний вибір для кожної підзадачі окремо.

2.5 Критерії порівняння та способи обчислення їх оцінок

У цій роботі досліджувалися критерії порівняння вебсервісів, значення яких можна отримати автоматизованим або напівавтоматизованим способом. У першому розділі були представлені у структурованому вигляді основні функціональні та нефункціональні критерії порівняння вебсервісів. У цьому розділі наводяться способи обчислення значень оцінок вибраних критеріїв.

Критерій «чутливість» визначається через метод, який пропонує оцінку, засновану на поділі класів типів реакцій вебсервісів підвищення навантаження. Перший клас (низька чутливість) є найбільш сприятливим і включає вебсервіси, які демонструють практично незмінний час обробки запитів і відсутність помилок

обробки при підвищенні навантаження. Відповідно, п‘ятий клас (висока чутливість) є гіршим варіантом і включає вебсервіси, що показують різке підвищення часу обробки запитів і числа помилок при підвищенні навантаження.

Оскільки час обслуговування запиту є випадковою величиною, доцільно характеризувати його двома величинами: середнім і стандартним відхиленням.

Вебсервіси працюють за протоколом прикладного рівня HTTP та протоколом мережного рівня TCP. Типовий запит до вебсервісу складається з наступних етапів:

1. Пошук IP-адреси вебсервісу за його URL (англ. DNS Lookup). Цей етап може складатися з двох підетапів. Спочатку здійснюється пошук відповідного запису до DNS-кешу клієнта. Якщо запис не знайдений або застарілий, то робиться запит до DNS-сервера. Затримка, що вноситься цим етапом, залежить від швидкості роботи локального DNS-кешу, від швидкості роботи мережі та DNS-сервера.

2. Підключення до сервера (TCP connection setup, 3-way handshake). Оскільки використовується надійний транспортний протокол TCP, спочатку необхідно встановити з'єднання з віддаленим сервером, на якому знаходиться вебсервіс. Затримка, що вноситься цим етапом, залежить від характеристик мережі між клієнтом і сервером. Етап підключення складається з наступних підетапів:

2.1. Надсилання клієнтом пакету з виставленим прапором SYN;

2.2. Отримання клієнтом пакета з виставленими прапорами SYN та ACK;

2.3. Надсилання клієнтом пакета з виставленим прапором ACK.

3. Відправка запиту (англ. Sending) включає в себе час необхідний клієнту на формування та повне відправлення запиту. Як правило, цей час незначний, але може займати більше часу, якщо надсилаються великі файли (HTTP POST запит).

4. Очікування відповіді (англ. Waiting). Протягом цього етапу відбувається очікування на обробку запиту віддаленим вебсервісом. Цей час залежить від продуктивності вебсервісу та затримки мережі.

5. Отримання відповіді (англ. Receiving). Протягом цього етапу відбувається

пакетне отримання відповіді вебсервісу. Час цього етапу залежить від затримки, що вноситься мережею, і від розміру відповіді.

6. Закриття TCP з'єднання (англ. TCP connection close).

Діаграма процесу проходження HTTP-запиту представлена на рис. 2.4. Діаграма представляє деяке поширене спрощення процесу проходження HTTP-запиту. Головне спрощення полягає в тому, що за фактом вебсервіс на діаграмі є об'єднання вебсервера і самого вебсервісу. Насправді етапи, що стосуються встановлення та закриття TCP-з'єднання, проводяться в рамках взаємодії «клієнт – вебсервер». Етапи відправлення, обробки та отримання відповіді на HTTP-запит відноситься до ланцюжка взаємодії «клієнт – вебсервер – вебсервіс».

У роботах, що розглядаються, оцінка часу обробки запиту вебсервісом проводиться як вимірювання часу проходження всіх етапів відправки та обробки запиту. Проте така оцінка може бути некоректною з великого розкиду значень часів проходження кожного етапу даного процесу.

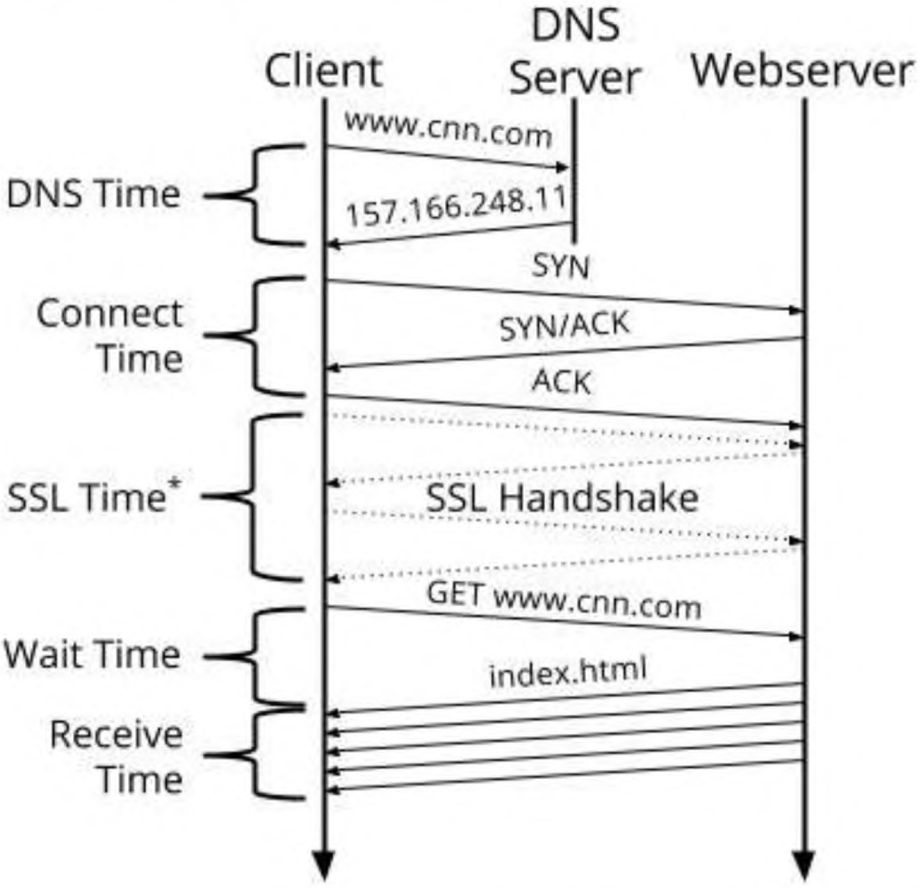

Рисунок 2.4 – Діаграма процесу проходження HTTP-запиту

Такий розкид значень зумовлюється такими факторами [29]:

1. Затримка, яка вноситься різними середовищами передачі. Від клієнта запит фізично може кілька разів пройти бездротові, кабельні та оптичні середовища.

2. Затримка, обумовлена відстанню між клієнтом та сервером, а також між проміжними вузлами. Згодом топологія мережі Інтернет може зазнавати суттєвих змін, що також робить внесок у варіацію мережевої затримки.

3. Затримка, яка вноситься мережевим обладнанням. У ході передачі запиту від клієнта до сервера сигнал може проходити безліч роутерів, комутаторів та інших мережних пристроїв, кожен з яких робить свій внесок у варіацію мережної затримки через різну завантаженість та використання складових пристроїв з непостійним часом обробки запитів, таких як жорсткі диски, оперативна пам'ять, процеси і т.д.

Кожен етап процесу проходження HTTP-запиту може мати досить велику варіацію за часом, оскільки включає використання мережного середовища. Також етап пошуку IP-адреси вебсервісу може змінюватись за часом залежно від того, чи є відповідний DNS-запис у локальному кеші клієнта або його потрібно буде отримати або оновити.

Таким чином, якщо вебсервіс знаходиться на значній відстані від клієнта, але досить близько до системи оцінки часу обслуговування запиту, то для клієнта представлений у системі час обслуговування запиту буде неправильним.

Ця проблема вивчалася в деяких роботах, що оглядаються. Наприклад, у роботі [36] пропонується метод усунення похибки, що вноситься мережею для оцінки часу обслуговування. Метод ґрунтується на припущенні про те, що вже існує інформація про час затримки мережі між вузлами вебсервісу та клієнтом. Як подібна інформація автори використовували дані, зібрані в рамках проєкту Planet Lab у 2018 році.

Мінус цього підходу в тому, що за фактом топологія Інтернету постійно змінюється, і одного разу зібрана інформація не може в майбутньому гарантувати

коректну роботу даного методу.

Далі буде запропоновано підхід, що дозволяє отримати час обробки запиту зі зведеним до мінімуму шумом, що вноситься мережевою затримкою, яка зважаючи на вищевикладене часто має великий розкид значень.

Приймемо за x_i – час обслуговування i – го запиту певної функції вебсервісу. Алгоритм обчислення часу обслуговування:

1. Здійснити кілька послідовних запитів певної функції вебсервісу. Мінімально необхідну кількість запитів можна визначити, наприклад, за допомогою методу довірчих інтервалів [40], виходячи з припущення, що час обробки запитів має експоненційний розподіл.

2. Зберегти інформацію про обробку запитів як x_i, $i = 1,..., m$, де m – кількість вироблених послідовних запитів.

3. Обчислити середній час обробки запиту:

$$\mu(X) = \frac{1}{m}\sum_{i=1}^{m} t(x_i) \tag{2.5}$$

де $t(x_i) = Waiting(x_i) - Connecting(x_i)$,

Waiting (x_i) – час відправки запиту, очікування відповіді та отримання першого пакета з заголовком HTTP-відповіді,

Connecting (x_i) – *час* встановлення TCP -з'єднання із сервером.

4. Обчислити стандартне відхилення часу обробки запиту:

$$\sigma(X) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(t(x_i) - \mu(X))^2} \tag{2.6}$$

Залежно від характеру функцій, що надаються, середнє навантаження на вебсервіс може змінюватися протягом дня, тижня, місяця, року. За теоремою Літла середній час обслуговування запитів прямо пропорційний навантаженню. Таким чином, для отримання достовірної оцінки часу обслуговування запитів необхідно проводити кілька обчислень оцінки часу обробки запиту протягом дня та усереднювати отримане значення.

Проте запропонований підхід також має недолік: в результаті обчислюється лише час обробки запиту вебсервісом, але не враховується мережна затримка, що існує через віддаленість клієнта від вебсервісу. Ця проблема може бути вирішена за рахунок використання клієнтського програмного забезпечення, яке дозволяє дізнатися про цю затримку. Наприклад, можна використовувати програму ping або її аналог, але працює за протоколом HTTP.

Критерій пропускної спроможності у цій роботі не включений у розрахунки з кількох причин.

По-перше, згідно [21] пропускна спроможність вимірюється для вебсервісів, орієнтованих на виконання пакетних завдань (англ. Batch job). Найчастіше вебсервіси спрямовані виконання транзакційних завдань (англ. Online transaction processing, OLTP), т.к. використовуються у безпосередній роботі з користувачем.

По-друге, для вебсервісів з низькою чутливістю на практиці часто неможливо визначити пікову пропускну здатність через такі фактори:

1. Висока масштабованість інфраструктури таких вебсервісів, яка забезпечує від 1 мільйона активних користувачів. Внаслідок цього неможливо організувати досить продуктивний тестовий стенд, який дозволив би дізнатися пікову пропускну здатність.

2. Наявність систем виявлення мережевих атак. Все частіше вебсервіси навіть із середньою аудиторією використовують проміжне програмне та апаратне забезпечення, що фіксує та запобігає ряду мережевих атак.

Нині найпоширеніший тип атак – це розподілена атака типу «відмова у обслуговуванні» (англ. Distributed Denial of Service, DDoS), коли генерується велика кількість штучних запитів до системи, що призводить до відмови обслуговування запитів справжніх користувачів. У зв'язку з цим спроба визначення пропускної здатності може бути розцінена як DDoS – атака, що призведе до недостовірних результатів тесту.

Середній час між відмовами (напрацювання на відмову, англ. Mean Time Between Failures, MTBF) – прогнозований час, що минув між відмовами системи,

що відновлюється, протягом її роботи. Обчислюється як середнє арифметичне часу між відмовами.

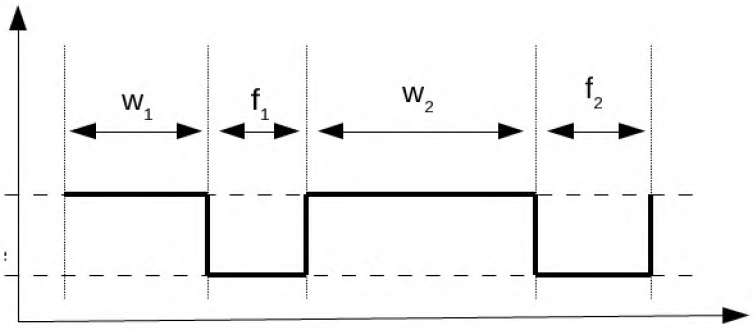

Рисунок 2.5 – Часова діаграма процесів роботи та відновлення

На рис. 2.5 показані позначення часів роботи та простою вебсервісу. Використовуючи наведені позначення, можна обчислити MTBF. Наводиться інкрементна формула обчислення (формула 2.7), яка може бути корисною при практичній реалізації ПЗ через те, що при кожному опитуванні вебсервісу на предмет працездатного стану необхідно щоразу перераховувати значення MTBF. У найпростішому випадку цієї операції знадобилося б зробити запит до БД на виїмку всіх значень w_i, порахувати їх кількість і лише тоді отримати середнє значення. Використання інкрементної формули дозволить не виконувати такі запити: достатньо буде зберігати значення загальної кількості проміжків безвідмовної роботи:

$$MTBF_n = \frac{(n-1) \cdot MTBF_{n-1} + w_n}{n} \tag{2.7}$$

де w_n – час безвідмовної роботи,

n – порядковий номер поточного проміжку безвідмовної роботи вебсервісу.

Середній час на відновлення (Mean Time To Repair (MTTR) – середній час, необхідний для відновлення системи. Також використовується інкрементна формула обчислення:

$$MTTR_m = \frac{(m-1) \cdot MTTR_{m-1} + f_m}{m} \tag{2.8}$$

де f_m – поточний час простою,

m – порядковий номер проміжку простою вебсервісу.

Доступність (готовність, англ. Availability) – ймовірність того, що об'єкт функціонуватиме в штатному режимі в даний момент часу. Існує кілька способів обчислення доступності:

1. Відношення середнього часу між відмовами до суми середнього часу на відновлення та середнього часу між відмовами.

$$Availability = \frac{MTBF}{MTBF + MTTR} \tag{2.9}$$

2. Відношення загального часу роботи в штатному режимі (англ. Uptime) до загального часу простою (англ. Downtime) та роботи в штатному режимі. Як правило, ця метрика використовується разом із визначенням періоду спостереження за системою.

$$Availability = \frac{TotalUptime}{TotalUptime + TotalDowntime} \tag{2.10}$$

У цій роботі використовується перший спосіб обчислення доступності через те, що величини MTBF і MTTR вже враховуються як критерії порівняння.

Безвідмовність (англ. Reliability) – ймовірність того, що запит буде коректно опрацьовано за умови, що вебсервіс функціонує у штатному режимі. Обчислюється як відношення числа успішно опрацьованих запитів до загальної кількості запитів під час роботи сервісу в штатному режимі:

$$Reliability = \frac{SuccessRequests}{TotalRequests} \tag{2.11}$$

Час роботи (англ. Uptime) – час роботи вебсервісу з останнього збою або з моменту збору статистики роботи вебсервісу. Далі час роботи вимірюється у годинах.

Вартість надання послуг зазвичай вимірюється у грошових одиницях за кількість запитів. Наприклад, на січень 2023 року сервіс Google Maps визначав вартість здійснення 1000 запитів 0.05$. Інформація про вартість зазвичай не

представлена у машиночитаному форматі, проте деякі постачальники послуг надають окремі інформаційні вебсервіси з планування витрат (наприклад, Amazon Web Services Cost Allocation). Далі вартість вимірюється у центах долара за виконання одного запиту.

Безкоштовний ліміт запитів. Часто цінова політика вебсервісів передбачає певну безкоштовну квоту кількості запитів за одиницю часу. Наприклад, на січень 2023 року безкоштовний ліміт використання сервісу Google Maps складав 25 000 запитів на день. Далі безкоштовний ліміт запитів вимірюється серед запитів на день.

2.6 Формування множини Парето

Нехай X_{par} – множина Парето (або парето-оптимальна множина) – множина недомінованих варіантів x таких, що не існує такого $x' \in X$, що $\forall j = 1, m\ x' R^j x$ та $\exists j^0 \in 1,m\ x' P^{j0} x$, де R^j – відношення несуворої переваги:

$$R^j = P^j \cup I^j,$$

P^j – відношення суворої переваги за j-им критерієм,

I^j – відношення рівноцінності (еквівалентності).

Для небагатьох варіантів і критеріїв досить швидко працює простий метод повного перебору, коли всі варіанти попарно порівнюються і множина Парето формують з недомінованих варіантів. Проте зі збільшенням розмірності завдання експоненційна складність алгоритму повного перебору призводить до неможливості завершення формування множини Парето за розумний час. З метою усунення цього недоліку у роботі використовується алгоритм, запропонований в [11].

Для опису алгоритму попередньо введемо нові позначення:

$S^j = x_{i1} Q_1^j x_{i2} Q_2^j \ldots Q_{n-1}^j x_{in}$ де $\forall \alpha = 1,n-1\ Q^j{}_\alpha \in \{P^j, I^j\}$ – несуворе ранжування варіантів за j-им критерієм;

$X_1^j(x_i)$ – підмножина ідентифікаційних номерів альтернатив, розміщених у ранжируванні S^j між найкращою (першою) та альтернативою x_i;

$X_2^j(x_i)$ – підмножина ідентифікаційних номерів альтернатив, розміщених у ранжируванні S^j між найкращою (першою) і альтернативою x_i, що виключає альтернативи, еквівалентні x_i по j -му критерію;

X_{max} – множина таких альтернатив, що для кожної з них є деяких критерій K^{j0}, $j^0 \in 1, m$ за яким ця альтернатива є найкращою та не має еквівалентних.

У ході роботи алгоритму необхідно буде обчислити множину X^*:

$$x_r \in X^* = \arg\min \sum^{m} |X_2^j(x_l)|, x_l \in Xj = 1 \qquad (2.12)$$

Алгоритм формування парето -оптимальних рішень має такий вигляд:

1. Формується X^* та X_{max} за формулою (2.12).
2. Вибирається перша альтернатива $x_r \in X^*$.
3. Будується множина:

$$S_{r2} = \bigcup_{j=1}^{m} X_2^j(x_r).$$

4. Формується множина:

$$\tilde{X} = Sr2 \setminus (X\,max \cup X^*).$$

5. Формується множина:

$$X' = \{x \in \tilde{X} : \bigcap_{j=1}^{m} \overline{X}_1^j(x) = \emptyset\},$$
$$\overline{X}_1^j(x) = X_1^j(x) \setminus \{x\}.$$

6. Визначається значення параметрів вебсервера:

$$Xpar = X' \cup X^* \cup X\,max.$$

Множина Парето зазвичай містить велику кількість варіантів, тому з метою звуження отриманої множини використовуються процедури виявлення переваг ОПР. Один із таких методів з невеликими змінами, що стосуються предметної області, описується в наступному розділі.

Висновки до розділу 2

У розділі розглянуто підхід до вирішення проблеми вибору найбільш оптимальної композиції вебсервісів. Використання теорії важливості критеріїв, що базується на кількісних шкалах вимірювання, надає значні переваги у порівнянні з методами на основі згортки, аналізу ієрархій, нелінійної оптимізації. Процедура вибору, що враховує людський фактор, має чітку форму представлення збитків та компенсацій для значень критеріїв у відповідних шкалах виміру. Використання ефективного алгоритму формування множини Парето дозволяє здійснювати вибір за прийнятний час, навіть за значної кількості альтернатив та критеріїв. У проведеному аналізі було оцінено критерії порівняння, значення яких можна отримати автоматизованим способом.

РОЗДІЛ 3
ВДОСКОНАЛЕННЯ МЕТОДУ ВИЗНАЧЕННЯ ОПТИМАЛЬНОЇ ЗА ПЕРЕВАГАМИ КОМПОЗИЦІЇ ВЕБСЕРВІСІВ

3.1 Вибір математичного апарату теорії важливості критеріїв

В аналізованій літературі у випадках урахування переваг у виборі вебсервісів вводиться єдина інтегральна функція Ф, що є зваженою сумою критеріїв, помножених на відповідні коефіцієнти важливості [12, 32]. При цьому значення кожного критерію описується різними значеннями коефіцієнтів (3.1). Чим важливіший критерій, тим більше значення набуває відповідного коефіцієнта.

$$\Phi = \alpha_1 K_1 + \alpha_2 K_2 + ... + \alpha_n K_n, \qquad (3.1)$$

де $\sum \alpha_i = 1$. Даний підхід має ряд недоліків [26, 34], усунення яких важливе для коректного вирішення описуваного завдання.

У роботах [12, 35] також на вирішення завдання вибору вебсервісів за перевагами використовується метод аналізу ієрархій (MAI). Як показано у ряді робіт (наприклад, [26]) MAI є теоретично некоректним методом, тому його не рекомендується використовувати для вирішення подібних завдань.

Існує кілька методів, прийнятних на вирішення цієї задачі. Турнірні методи [27] та метод малих сум [28] можуть бути корисні в ситуації, коли ОПР не визначилася у важливості тих чи інших критеріїв.

Метод триградаційних ранжувань [39] використовується у ситуації, коли пріоритетною є вимога відсутності низьких оцінок за критеріями. У роботі використовується підхід теорії важливості критеріїв, описуваний у роботах [16] та [20].

У вирішуваній задачі всі критерії неоднорідними, тобто мають різні шкали виміру. Використовуваний підхід дозволяє вирішити проблему вибору за наявності кількісної інформації про важливість критеріїв, не нормалізуючи і не наводячи

неоднорідні критерії єдиної порядкової шкали (ЄПШ). Такий підхід є кращим варіантом через те, що при переході до ЄПШ може бути втрачена частина інформації. Обмеженням підходу, що використовується, є необхідність наявності кількісних шкал у всіх критеріїв.

Слід зазначити, що є два основних підходи до постановки завдання вибору:

1. Пошук кращої композиції вебсервісів з урахуванням структури завдання. При цьому вводяться 4 базові структури об'єднання вебсервісів (послідовна, циклічна, умовна та паралельна), для кожного можливого поєднання реалізацій підзадач вебсервісами розраховується інтегральне значення критеріїв порівняння (цей процес, як правило, оптимізується) та вибирається варіант композиції з найкращими інтегральними оцінками значень критеріїв.

2. Пошук кращої композиції без урахування структури завдання. При цьому загальне завдання пошуку кращої композиції вебсервісів у рамках окремого завдання розглядається як пошук найкращих рішень для окремих завдань.

У цій роботі використовується другий підхід з огляду на те, що, по-перше, найчастіше виконується умова незалежності вебсервісів і, по-друге, цей підхід дозволяє наочно представляти результати вибору з урахуванням відносної важливості критеріїв, заданої ОПР. Насправді питання існування переваги першого підходу перед другим не розглянуто у відомій літературі та залишається відкритим.

3.2 Алгоритм звуження множини Парето на основі інформації про важливість критеріїв

Введемо основні визначення. Нехай x – функція, яку надає вебсервіс. Критерій $K_i(x)$ важливіший за критерій $K_j(x)$ із заданими позитивними параметрами v_i і v_j, якщо для будь-якої векторної оцінки $K(x) = (K_1(x),..., K_m(x))$ має місце співвідношення

$$K'(x) > K(x), \tag{3.2}$$

де $K'(x) = (K'_1(x),..., K'_m(x))$, причому

$$
\begin{aligned}
K_i'(x) &= K_i(x) + v_i\,, \\
K_j'(x) &= K_j(x) - v_j\,, \\
K_k'(x) &= K_k(x), k = \overline{1,m}, k \neq i,\ k \neq j
\end{aligned}
\tag{3.3}
$$

Іншими словами, критерій K_i важливіше критерію K_j, якщо з двох векторів K ОПР надає перевагу вектору K_i' ОПР готова пожертвувати певною кількістю v_j за менш важливим критерієм K_j задля отримання додаткової кількості (компенсації) v_i за більш важливим критерієм K_i за умови збереження значень решти критеріїв.

За допомогою чисел v_i та v_j можна кількісно оцінити зазначений рівень відносної важливості:

$$
\theta_{ij} = \frac{v_j}{v_i + v_j}, \theta_{ij} \in [0,1] \tag{3.4}
$$

Показник θ_{ij} показує частку втрати за менш важливим критерієм, на яку згоден піти ЛПР, порівняно із сумою зазначеної втрати та надбавки за більш важливим критерієм. Якщо коефіцієнт θ_{ij} близький до одиниці, то це означає, що ЛПР за відносно невелику надбавку за більш важливим i-му критерію готове платити досить великою втратою за менш важливим j-му критерієм. Таке положення відповідає ситуації, коли i-ий критерій має порівняно високий рівень важливості в порівнянні з j-им критерієм. У разі коли θ_{ij} рівний нулю, ОПР відповідно йде на втрати за менш важливим критерієм лише за умови отримання суттєвого збільшення за більш важливим критерієм. Це означає, що ступінь важливості i-го критерію порівняно невисока.

Якщо критерій K_i важливіший за критерій K_j із заданими позитивними параметрами v_i і $_j$, можна звузити множину Парето заміною векторного критерію $K(x)$ векторним критерієм $K^*(x) = (K^*_1(x),\ldots, K^*_m(x))$, компоненти якого обчислюються за формулами:

$$
\begin{aligned}
K_j^*(x) &= v_j K_i(x) + v_i K_j(x)\,, \\
K_k^*(x) &= K_k(x), k = \overline{1,m}, k \neq j\,,
\end{aligned}
\tag{3.5}
$$

Таким чином, векторний критерій K^* виходить з K' заміною менш важливого критерію K_j на лінійну комбінацію K_i та K_j із позитивними коефіцієнтами v_j та v_i. Перед обчисленням K^* всі значення критеріїв проходять процедуру стандартизації: критерії для яких більше значення є менш кращим (тобто вони мають негативний тренд), примножуються на −1, інші критерії зберігають свої значення.

Далі все нові векторні оцінки $K^*(x)$ проходять процедуру отримання множини Парето. Решта варіантів і є результатом вибору за перевагами.

Для уточнення переваг для ОПР метод можна доповнити людино-машинною процедурою. Спочатку можна дізнатися лише якісну інформацію про переваги. Припустимо, що критерій надійності важливіший за критерій середнього часу обробки запитів $\Omega = \{1 > 2\}$, тоді v_2 можна задати значенням, доречним у рамках шкали вимірювання середнього часу обробки запитів. Тоді, давши можливість ОПР змінювати v_1 в рамках шкали вимірювання K_1 можна наочно уявити які варіанти будуть оптимальні при заданому прирості критерію K_1 на величину v_1.

3.3 Визначення архітектури системи на прикладі додатку для пошуку найближчих лікарень

На рис. 3.1 представлена архітектура розробленого програмного комплексу. Основні функції системи:

1. Здійснення раціонального вибору композиції вебсервісів у межах певного завдання.

2. Ведення реєстру вебсервісів та їх функцій.

3. Моніторинг значень критеріїв порівняння вебсервісів.

Програмний комплекс реалізований як система з сервісноорієнтованою архітектурою, що складається з трьох основних вебсервісів.

Користувальницький вебінтерфейс компонента дозволяє користувачеві здійснити такі операції:

1. Додавання, редагування та видалення вебсервісів та їх функцій у реєстрі

вебсервісів.

2. Створення моделей завдань, які у свою чергу складаються з підзадач.

3. Зв'язування підзадач із функціями вебсервісів, які можуть реалізувати це підзавдання.

4. Здійснення людино-машинної процедури оптимального вибору композиції вебсервісів.

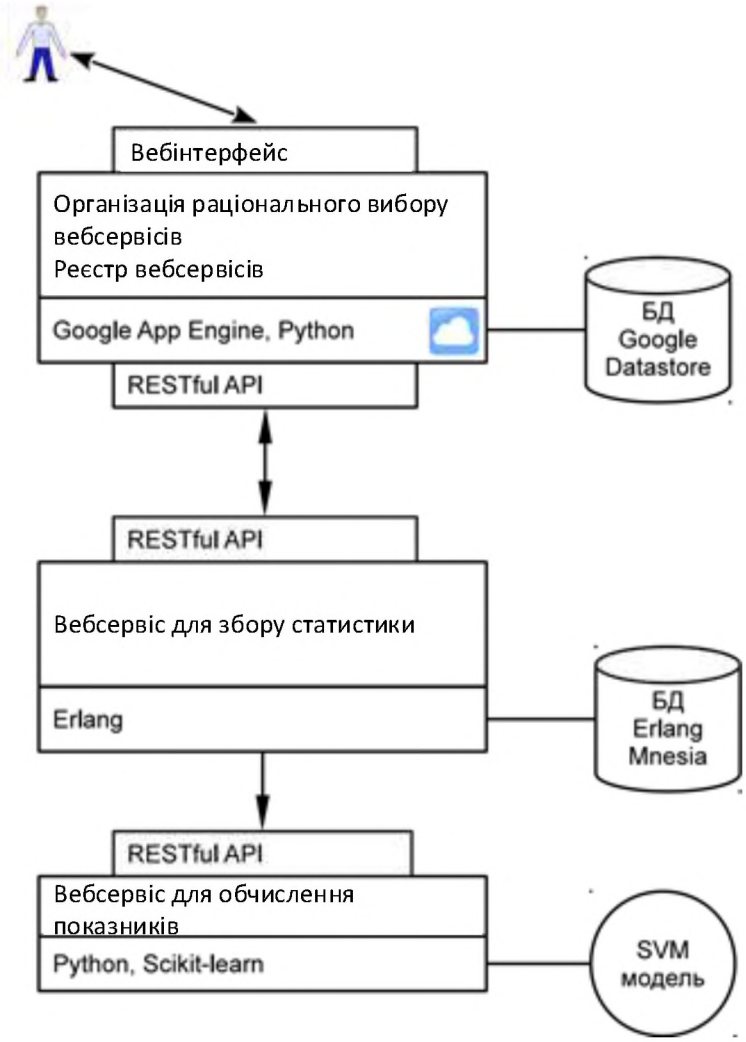

Рисунок 3.1 – Архітектура спроєктованої системи

Вебсервіс має триланкову архітектуру:

1. Клієнт реалізований за допомогою технологій HTML 5, CSS 3, Javascript. У роботі використовувалися такі бібліотеки з відкритим кодом:

1.1. Google AngularJS [11] – JavaScript фреймворк, що реалізує шаблон проєктування «Модель – Вид – Модель виду» [92], а також підтримує модульну розробку елементів вебдодатку.

1.2. JQuery [13] – JavaScript бібліотека для роботи з DOM структурою HTML-документа, AJAX запитами і т.д.

1.3. Twitter Bootstrap [14] – HTML, CSS, JavaScript фреймворк (платформа), що включає набір інтерфейсних елементів та обробників взаємодії з цими елементами.

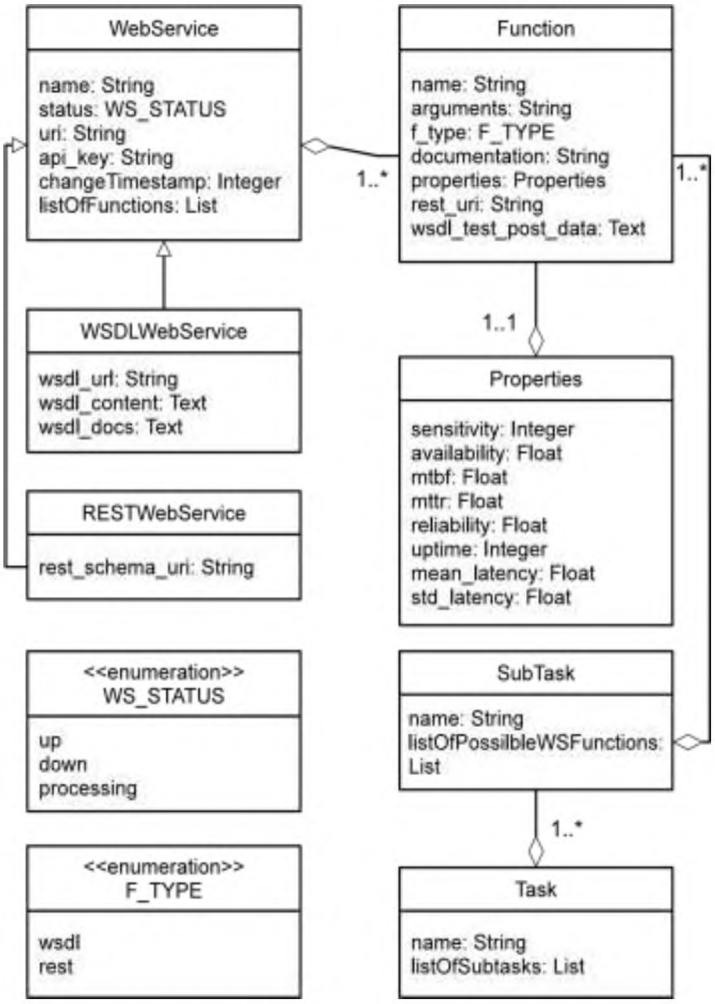

Рисунок 3.2 – UML діаграма основних класів вебсервісу раціонального вибору композицій вебсервісів та реєстру сервісів

2. Серверна частина реалізована з урахуванням мови програмування Python [15]. У роботі використовувалися такі бібліотеки з відкритим кодом:

2.1. WebApp2 [16] – Python фреймворк, що реалізує шаблон проєктування «Модель – Вид – Контролер».

2.2. Suds [17] – Python бібліотека для роботи з вебсервісами на базі технологій WSDL та SOAP.

3. Як сховище даних використовується розподілена нереляційна база даних типу «ключ-значення» Google Datastore. Цей вебсервіс розміщено в рамках хмарної інфраструктури Google App Engine [18]. Вибір цієї платформи обумовлений, по-перше, готовими можливостями до горизонтального масштабування вебдодатку та, по-друге, досить великим безкоштовним щоденним лімітом, що дозволяє вільно розробляти та використовувати вебдодаток за невеликої кількості користувачів.

UML діаграма основних класів програми представлена на рис. 3.2. Вебсервіс також містить багато службових класів, які стосуються обробки вхідних запитів. Ці класи є стандартними та поставляються з веб фреймворком WebApp2.

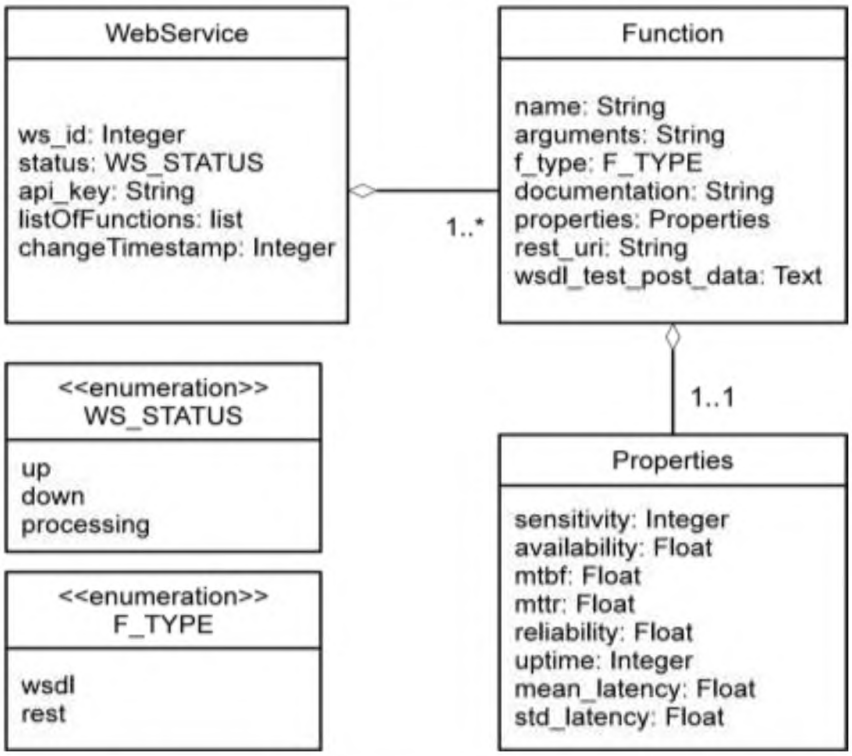

Рисунок 3.3 – UML діаграма основних класів вебсервісу збору статистичних даних для отримання оцінок сервісів за критеріями

Компонент збору статистичних даних періодично звертається до реєстру для синхронізації списку вебсервісів, що розглядаються. Зібрані статистичні дані використовуються для обчислення оцінок за такими критеріями (рис. 3.3):

– чутливість;

– середнє та стандартне відхилення часу обробки запитів;

– середній час між відмовами;

– середній час відновлення;

– доступність;

– безвідмовність.

Даний вебсервіс не має інтерфейсу користувача. Керування вебсервісом відбувається за допомогою RESTful [16] програмного інтерфейсу доступу. Компонент реалізований мовою програмування Erlang [13]. Вибір Erlang обумовлений двома причинами: по-перше, досить високою швидкістю розробки вебсервісів, які повинні працювати в постійному режимі (режим «daemon»), по-друге, архітектурні рішення мови забезпечують високу продуктивність та низьке споживання ресурсів на багатоядерних системах.

Під час розробки використовувалися такі бібліотеки з відкритим кодом:

1. Misultin [19] – бібліотека, що дозволяє реалізувати доступ до функцій вебсервісу за допомогою протоколу HTTP.

2. JSON [10] – бібліотека для роботи з форматом передачі JSON.

3. Mnesia – нереляційна база даних у стандартній поставці Erlang /OTP.

Структура основних класів даного вебсервісу представлена на рис. 3.3. У класі WebService ws_id – це ідентифікаційний номер зовнішнього об'єкта класу WebService компонента.

На рис. 3.4 зображено часову діаграму роботи даного вебсервісу.

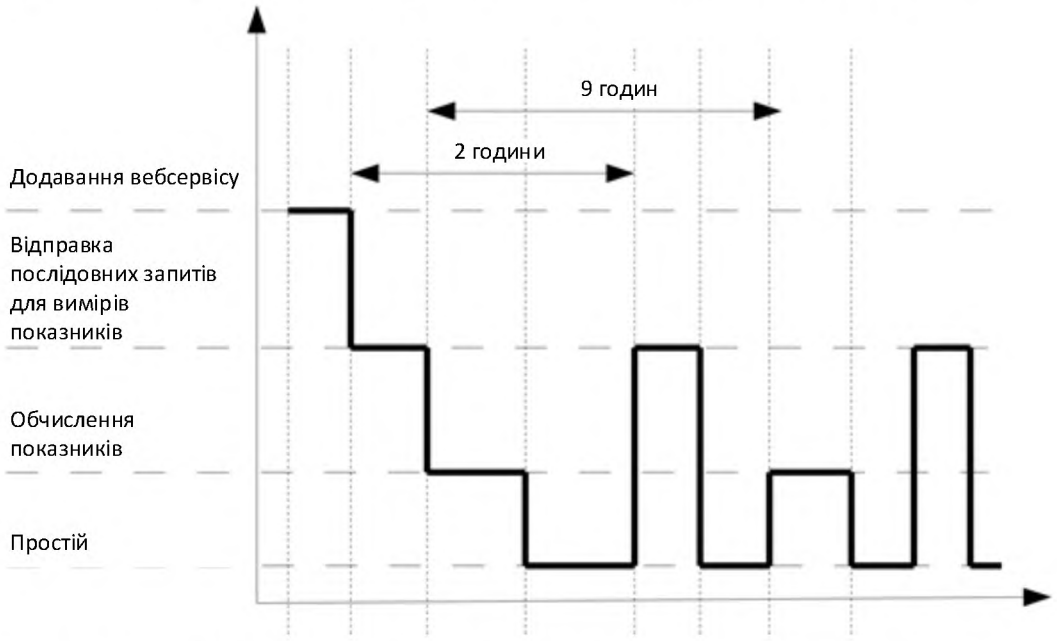

Рисунок 3.4 – Часова діаграма обчислення значень критеріїв порівняння

Варто зазначити, що збір статистичних даних для оцінки чутливості проводиться 3 рази з різницею в 9 годин. Це зроблено для того, щоб отримати усереднену оцінку чутливості за різними часами доби, оскільки багато вебсервісів мають великий перекіс завантаженості протягом проміжків часу протягом дня. Наприклад, відеосервіси мають найбільшу завантаженість у вечірній час, новинні – вранці.

Таблиця 3.1 – Опис та приклад вхідних даних вебсервісу оцінки чутливості

Час початку запиту, мс	Час обробки запиту, мс	Номер ітерації	Число запитів в ітерації
1348245358318	602	1	10
1348245358418	391	1	10
1348245358518	424	1	10
1348245358618	413	1	10
1348245358718	484	1	10
1348245358818	441	1	10
1348245358918	708	1	10
1348245359018	448	1	10
1348245359118	390	1	10
1348245359218	393	1	10
1348245360127	397	2	20
1348245360178	399	2	20
1348245360228	540	2	20
1348245360278	456	2	20

Як API, вебсервіс оцінки чутливості надає одну функцію обчислення оцінки чутливості при вхідних даних, представлених у певному форматі (табл. 3.1). Вхідні дані є результатами тесту. Вебсервіс реалізований мовою програмування Python. У роботі використовувалися такі бібліотеки з відкритим кодом:

1. Scikit-learn [11] – бібліотека із широким набором алгоритмів машинного навчання.

2. Twisted [21] – бібліотека, що дозволяє реалізувати доступ до функцій вебсервісу за допомогою протоколу HTTP.

У основі роботи сервісу лежить «навчена» модель з урахуванням методу опорних векторів. Програмно модель являє собою серіалізований та збережений у файл об'єкт класу sklearn.svm.SVC.

У спрощеному варіанті процес проєктування додатка, що використовує зовнішні вебсервіси, складається з наступних етапів:

1. Визначення мети програми.

2. Формування набору основних завдань, які необхідно вирішити з метою забезпечення функціонування програми.

3. Аналіз завдань та розбиття на відповідні підзавдання.

4. Пошук та аналіз вебсервісів, які можуть бути корисними у реалізації підзавдань.

5. Раціональний вибір композиції вебсервісів з урахуванням кількісних переваг особи, яка приймає рішення.

Основною метою додатка, є надання наочного списку найближчих лікарень і поліклінік. Наочність досягається за рахунок показу не тільки лікарень, а й розташування користувача на карті. Також як допоміжна функція надається коротка інформація про спеціалізацію установ, графік роботи та контактні дані.

Для вирішення поставленої мети необхідно реалізувати ряд завдань та відповідних підзавдань:

1. Отримання інформації про найближчі лікарні.

1.1. Знаходження розташування користувача на карті: визначення географічних координат за IP або MAC-адресою мережевого пристрою.

1.2. Визначення списку найближчих до користувача лікарень за географічними координатами.

1.3. Уточнення даних щодо знайдених лікарень.

2. Відображення розташування користувача та отриманої інформації по лікарнях на карті.

Розроблена система дозволяє спростити та автоматизувати етапи пошуку та аналізу вебсервісів, раціонального вибору композиції вебсервісів.

3.4 Обчислення показників ефективності системи з композицією вебсерверів

Проведемо процедури ковзного контролю та сіткового пошуку з метою вибору найкращої комбінації параметрів та оцінки узагальнюючої здатності алгоритму, запропонованого та описаного у розділі 2.

Розділимо випадковим чином вихідну вибірку на два підмножини: 80% – підмножина, що використовується в процедурах сіткового пошуку та ковзного контролю, 20% – контрольна підмножина. Задамо $k = 3$ для процедури k -кратного ковзного контролю. Задамо множину параметрів, серед яких необхідно знайти найкращу комбінацію за допомогою методу сіткового пошуку та крос-валідації:

1. Для лінійного ядра: $C = [1, 10, 100, 1000]$.
2. Для радіальної базисною функції: $\gamma = [0.001, 0.0001]$, $C = [1, 10, 100, 1000]$.

Таблиця 3.2 – Результати сіткового пошуку та крос-валідації

F -міра	Ядро	Параметри ядра
0.303 ± 0.010	РБФ	$C = 1$, $= 0.001$
0.303 ± 0.010	РБФ	$C = 1$, $= 0.0001$
0.470 ± 0.010	РБФ	$C = 10$, $= 0.001$
0.303 ± 0.010	РБФ	$C = 10$, $= 0.0001$
0.799 ± 0.019	РБФ	$C = 100$, $= 0.001$
0.470 ± 0.010	РБФ	$C = 100$, $= 0.0001$
0.862 ± 0.013	РБФ	$C = 1000$ $= 0.001$
0.799 ± 0.019	РБФ	$C = 1000$ $= 0.0001$
0.862 ± 0.022	Лінійне	$C = 1$
0.848 ± 0.013	Лінійне	$C = 10$
0.787 ± 0.013	Лінійне	$C = 100$
0.787 ± 0.013	Лінійне	$C = 1000$

Використовуючи бібліотеку з відкритим вихідним кодом, що містить алгоритми машинного навчання, Scikit Learn [31], здійснимо обчислювальний експеримент із заданими параметрами. Результати сіткового пошуку та крос-валідації (як оцінка використовується середня F -міра за всіма класами) показані в таблиці 3.2. Кращий набір параметрів спостерігається для ядра РБФ з параметрами

$C = 1000$, $\gamma = 0.001$. Проведемо оцінку класифікатора із заданими параметрами на контрольній підмножині.

Варто відзначити, що результати сіткового пошуку та ковзного контролю, а також оцінка F-заходів на контрольній підмножині залежать від початкового випадкового розбиття всієї вибірки на підмножини, тому кінцеве значення загальних F-заходів може змінюватись. Після кількох повторень всієї процедури навчання класифікатора оцінка контрольної підмножини F-заходів = 0.955±0.025.

Отримані результати можна подати візуально. Для цього зменшимо розмірність простору ознак з 9 до 2 за допомогою методу головних компонентів [32], налаштуємо параметри класифікатора, використовуючи отриманий простір ознак і побудуємо межі прийняття рішень за класами (рис. 3.5).

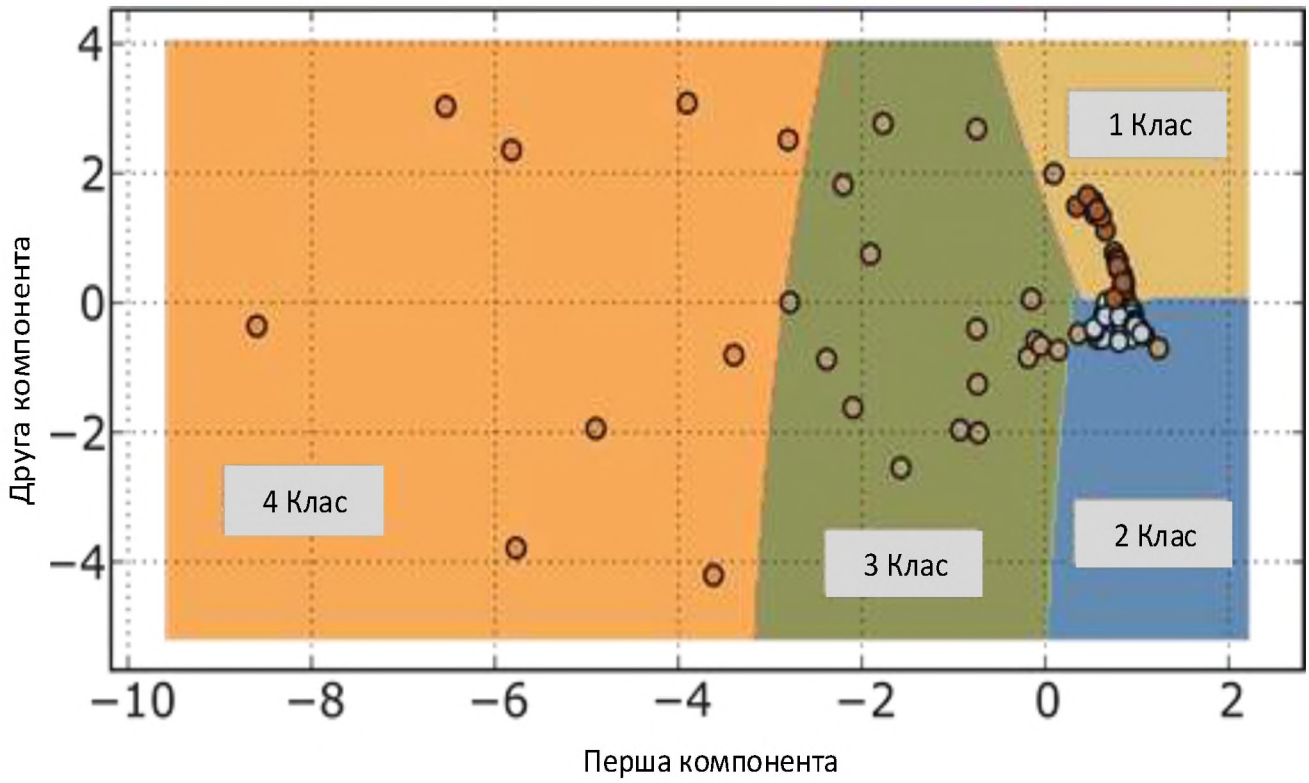

Рисунок 3.5 – Межі прийняття рішення класифікатора за класами

Методика визначення класу чутливості вебсервісу показує певну стійкість. В результаті обчислювального експерименту кожен вебсервіс проходив тестування 3 рази з різницею о 9 годині між тестами. Надалі для перевірки гіпотези стійкості методу кожен результат тесту використовувався як вхідні дані визначення

чутливості. 83% усіх вебсервісів показують однакове значення чутливості за три вимірювання, інші 17% вебсервісів лише один раз змінюють клас, тобто. були присутні послідовності класів виду 1 2 1, 3 2 2 та ін.

3.5 Модель функціонально справних станів вебсервісу

Систему (вебсервіс) можна описати як марківський процес із використанням дискретних станів та неперервного часу. На рис. 3.6 представлена модель системи у вигляді графа станів, де вузли відображають стани системи, а стрілки показують переходи між цими станами.

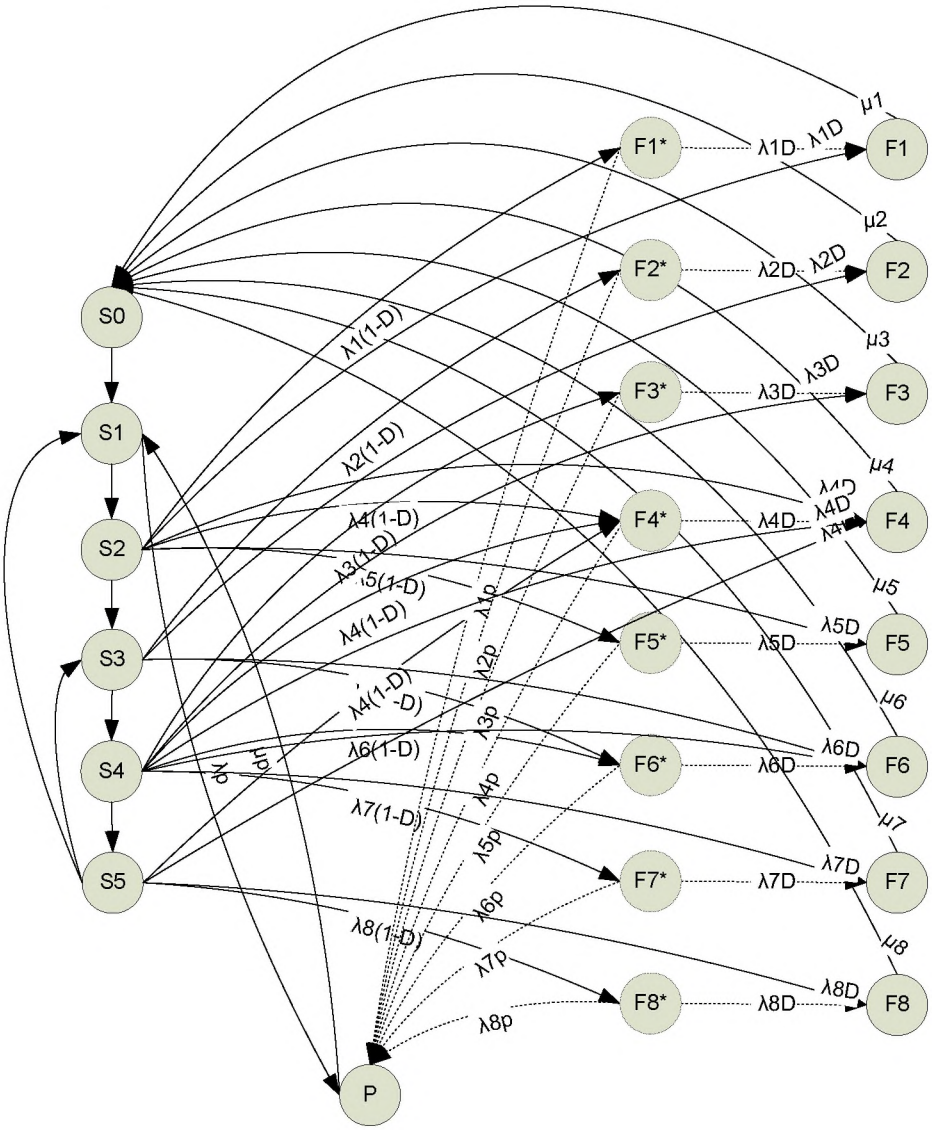

Рисунок 3.6 – Розмічений марковський граф моделі вебсервісу

На рис. 3.6 зображено модель з 24 станами, що охоплює такі стани:

S0: Підготовка черги вхідних запитів;

S1: Очікування надходження запиту (періодична перевірка черги);

S2: Отримання та розбиття клієнтського запиту;

S3: Відправлення запитів на обробку відповідним вебсервісам;

S4: Збір та перевірка результатів обробки на відповідність;

S5: Повернення результатів клієнту після підтвердження;

P: Регулярне обслуговування системи;

F*: Приховані відмови;

F1-F8: Виявлені відмови системи.

Для позначення цього марківського графа потрібно враховувати правила щодо інтенсивності переходів відповідно до логіки роботи моделі. Як відомо, простий потік має наступні ключові характеристики:

– стаціонарність;

– ординарність;

– відсутність післядії.

До цього моменту введемо два припущення, що визначають цільову модель:

– Заявки з різних джерел не взаємопов'язані.

– Заявки від одного джерела взаємопов'язані лише у випадку, якщо сервіс не надав послугу. Але ймовірність цього є дуже малою.

Встановимо параметри інтенсивності переходів між станами. Таким чином, переходи від працездатного стану ОК до стану профілактики P і назад позначені як λ_p і μ_p відповідно. Перехід з ОК до стану прихованої відмови F* позначено як $\lambda^*(1-D)$, до стану профілактики P – як λ_p, а до стану виявленої відмови F – як λ_D. Зрештою, переходи з ОК до F і назад позначені як λ_D і μ відповідно.

Цифрові характеристики навантаження вебсервісів оцінюються відповідно до значень інтенсивності запитів, які можуть коливатися від 1 до 100 запитів за секунду. Ці дані витікають з аналізу журналів доступу до вебсайтів, проведеного лабораторією Hewlett Packard Labs. Таким чином, середня інтенсивність запитів прийнята на рівні 10 запитів за секунду або 36,000 запитів за годину.

Аналіз даних системи BASIS WS, що вимірює показники гарантованої надійності систем орієнтованих на біомедицину, вказує на максимальну кількість запитів у розмірі 15,74 запитів на секунду (56,664 запитів за годину) і мінімальну – 2,16 запитів на секунду (7,776 запитів за годину). Отже, середня інтенсивність запитів від користувачів для реальних SOA-систем оцінюється на рівні 8,95 запитів за секунду (32,220 запитів за годину).

Оскільки активність користувачів вебпорталів змінюється в залежності від часу доби, сезонів відпусток та рекламних кампаній, різноманіття інтенсивності запитів може коливатися від 100 до 100000 запитів за годину. У визначенні інтенсивності переходу системи в стан збою та вторгнень, має значення кількість параметрів. Наприклад, для кожного сервера у кластерній системі, інтенсивність збою оцінюється на рівні 0,001 1/год (тобто 10^{-3}) [24], а інтенсивність відмов – 0,00001 1/год (тобто 10^{-5}). Інтенсивність програмних збоїв, викликаних програмним забезпеченням, вважається на порядок вище, ніж збої апаратного забезпечення системи, тобто 10^{-2} 1/год, а інтенсивність відмов, відповідно, 10^{-4} 1/год. Зокрема, інтенсивність збоїв операційних систем може бути вдвічі вищою, ніж збоїв системного програмного забезпечення (наприклад, СУБД, вебсервери і т.д.), а інтенсивність збоїв програмного забезпечення додатків – втричі вищою за збої системного ПЗ [22].

У моделюванні системи використовуються постійні дані, які представлені у таблиці 3.3, для врахування різноманітності системи при різних вхідних параметрах.

Таблиця 3.3 – Постійні значення параметрів марковської моделі вебсервісу

λ	Значення (1/годин)	μ	Значення (1/годин)
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

Інтенсивності обробки запитів вебсервісом, позначені як ρ1-ρ7, при відсутності затримок у обслуговуванні, мають однакові значення. Щодо інтенсивностей профілактики, λр та μр обчислюються за таким способом: μр = MIN(μ1 ... μ8) = μ1 = 0,5 (1/год); λр залежить від частоти проведення профілактичних робіт, наприклад: якщо профілактика виконується один раз на тиждень – Тр = 724 = 168 годин, то λр = 0,00595 (1/год); якщо профілактика виконується раз на два дні – Тр = 224 = 48 годин, то λр = 0,02083 (1/год); якщо профілактика виконується раз на добу – Тр = 24 години, то λр = 0,0417 (1/год).

Функція matrixA [27] використовується для формування матриці системи диференціальних рівнянь Колмогорова-Чепмена. Для розв'язання цієї системи диференціальних рівнянь Колмогорова було застосовано середовище Matlab та метод ode15s у межах часового інтервалу [0…10000] годин. Графічні результати розв'язання представлені на рисунку 3.7.

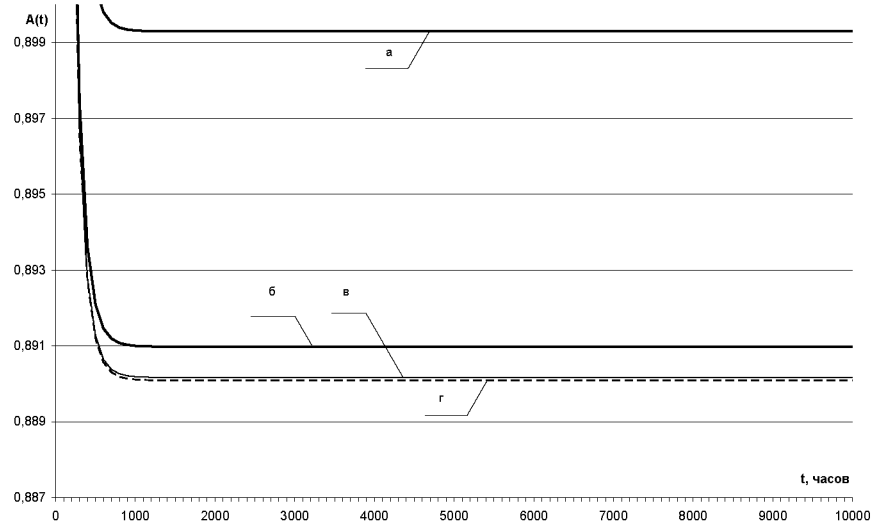

Рисунок 3.7 – Залежність коефіцієнта готовності системи А для різних значень параметрів ρ: а) ρ = 0,1; б) ρ = 1; в) ρ = 5; г) ρ = 8.

На рисунку 3.7 показані результати обчислень коефіцієнта готовності для занижених параметрів ρ. За графіками видно стабільну (постійну) динаміку зміни коефіцієнта готовності, а змінюються лише його стаціонарні значення. Наприклад, система досягає стаціонарних значень через 900 годин з похибкою 10-5 або через

1200 годин з похибкою 10-6 для будь-яких значень ρ. Це свідчить, що навіть при великих значеннях ρ динаміка коефіцієнта готовності залишається стабільною. Розрахунок коефіцієнта готовності у стаціонарному режимі можна виконати, вирішивши систему лінійних рівнянь. Усі подальші дослідження моделі вебсервісу проводилися у стаціонарному режимі, де вивчалися залежності коефіцієнта готовності від інших параметрів, а не від часу функціонування.

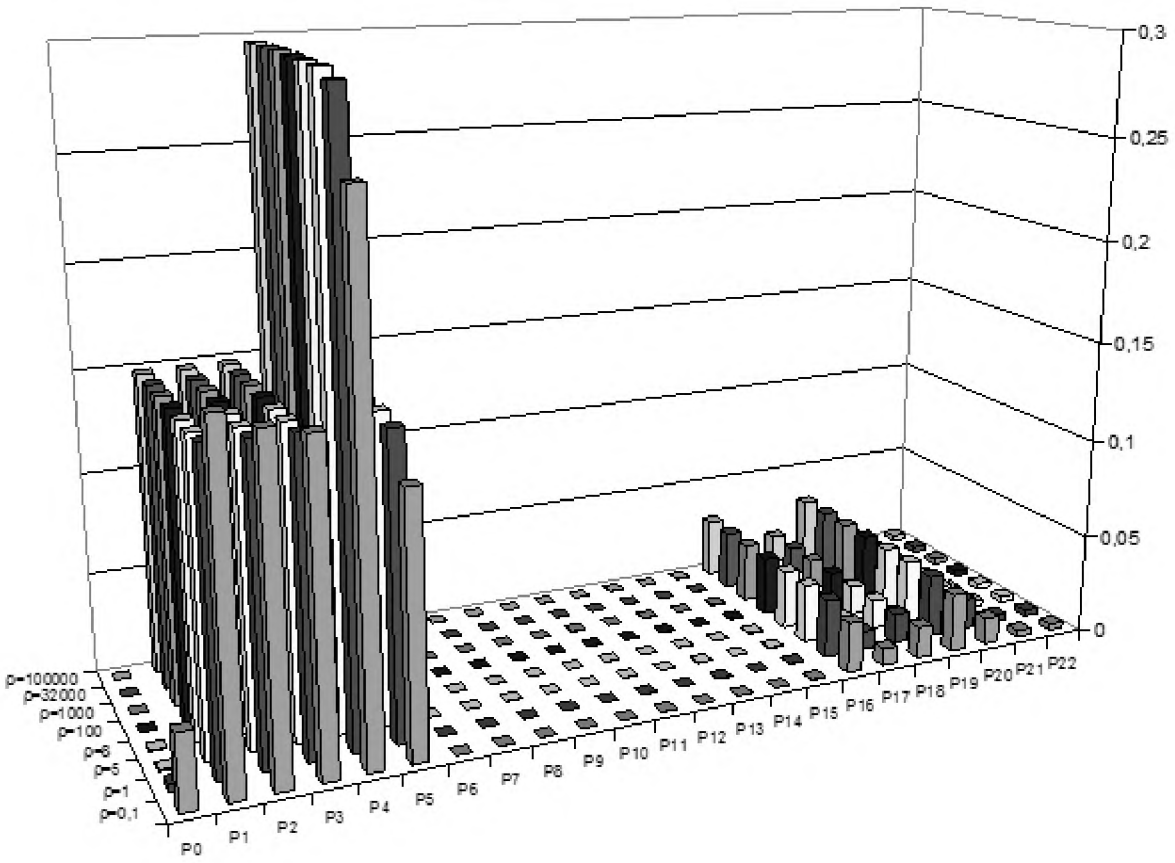

Рисунок 3.8 – Діаграма розподілу рівнів ймовірностей станів системи моделі вебсервісу при різних значеннях параметрів ρ1-ρ7

Аналіз даних, поданих на рис. 3.8, вказує на те, що при обраних параметрах системи головною загрозою для готовності вебсервісу є два основні аспекти. З одного боку, це можливість виникнення прихованих відмов у програмній компоненті (імовірності P16-P21). З іншого боку, є ризик прискореного проведення профілактичних заходів щодо цих відмов (P22), що також може вплинути на готовність системи.

3.6 Економічне обґрунтування проєкту з композиції вебсервісів

Для визначення терміну окупності використана економічна модель, запропонована в [40]. Ця модель призначена для власників бізнесу, щоб з'ясувати наступне:

– умови, за яких композиція вебсервісів буде прибутковою або неефективною для бізнесу;

– час і фінансові витрати, потрібні для досягнення прибутковості та повного окупання інвестицій;

– чи є доцільність вкладатися в композицію вебсервісів на даному етапі розвитку бізнесу.

Модель також пропонується використовувати фахівцям СОА з метою:

– оцінки ризику для клієнта (зменшення ймовірності, що послуги не окупляться або зменшення втрат клієнтів);

– підвищення конверсії замовлень (адаптація тарифів під можливості клієнта для забезпечення окупності);

– збільшення середнього чеку (пропозиція клієнту вищого тарифу на основі рентабельності бізнесу).

Деталі та вхідні параметри цієї моделі наведені у таблиці 3.4.

Таблиця 3.4 – Значення вхідних параметрів економічної моделі

№ з/п	Назва параметра	Позначення	Значення
1	Середня конверсія в оплачене замовлення	Ск	1%
2	Середній чек	Сч	3000 грн
3	Середня маржинальність	См	40%
4	Вартість робіт з композиції вебсервісів (на місяць)	Вс	30000 грн
5	Термін виконання робіт з композиції вебсервісів (місяців)	Т	6 місяців
6	Відвідувачі понад поточну відвідуваність	dКв	354

У табл. 3.5 представлені розрахункові показники моделі та відповідні їм розрахункові формули.

Таблиця 3.5 – Формули для розрахунку показників економічної моделі

№ з/п	Назва показника	Позначення	Розрахункова формула
1	Кількість відвідувачів вебресурсу	Кв	Кв0+dКв
2	Кількість замовлень	Кз	Кв·Ск
3	Виторг	Вит	Кз·Сч
4	Прибуток	Пр	Вит·См
5	Затрати на композицію вебсервісів	Зс	=IF(t<=T;Вс;0)
6	Прибуток з урахуванням послуг з композиції вебсервісів	Пр_с	Пр-Зс
7	Чистий прибуток	ЧПр	Пр_с(t-1)+ ЧПр(t)

Результати розрахунків наведені у табл. 3.6 та на рис. 3.7.

Таблиця 3.6 – Результати розрахунку моделі окупності

Мі-сяць	Кількість відвідувачів вебресурсу	Кількість замов-лень	Виторг	Прибуток	Затрати на композиції вебсервісів	Прибуток з урахуванням послуг з СОА	Чистий прибуток
1	300	3	9000,00	3600,00	30000,00	-26400,00	-26400,00
2	500	5	15000,00	6000,00	30000,00	-24000,00	-50400,00
3	800	8	24000,00	9600,00	30000,00	-20400,00	-70800,00
4	980	10	29400,00	11760,00	30000,00	-18240,00	-89040,00
5	1220	12	36600,00	14640,00	30000,00	-15360,00	-104400,00
6	1520	15	45600,00	18240,00	30000,00	-11760,00	-116160,00
7	1672	17	50160,00	20064,00	-	20064,00	-96096,00
8	1839	18	55176,00	22070,40	-	22070,40	-74025,60
9	2023	20	60693,60	24277,44	-	24277,44	-49748,16
10	2225	22	66762,96	26705,18	-	26705,18	-23042,98
11	2448	24	73439,26	29375,70	-	29375,70	6332,73
12	2693	27	80783,18	32313,27	-	32313,27	38646,00
13	2962	30	88861,50	35544,60	-	35544,60	74190,60
14	3258	33	97747,65	39099,06	-	39099,06	113289,66
15	3584	36	107522,41	43008,97	-	43008,97	156298,62
16	3942	39	118274,66	47309,86	-	47309,86	203608,49
17	4337	43	130102,12	52040,85	-	52040,85	255649,34
18	4770	48	143112,33	57244,93	-	57244,93	312894,27
19	5247	52	157423,57	62969,43	-	62969,43	375863,70
20	5772	58	173165,92	69266,37	-	69266,37	445130,07
21	6349	63	190482,52	76193,01	-	76193,01	521323,07
22	6984	70	209530,77	83812,31	-	83812,31	605135,38
23	7683	77	230483,84	92193,54	-	92193,54	697328,92
24	8451	85	253532,23	101412,89	-	101412,89	798741,81

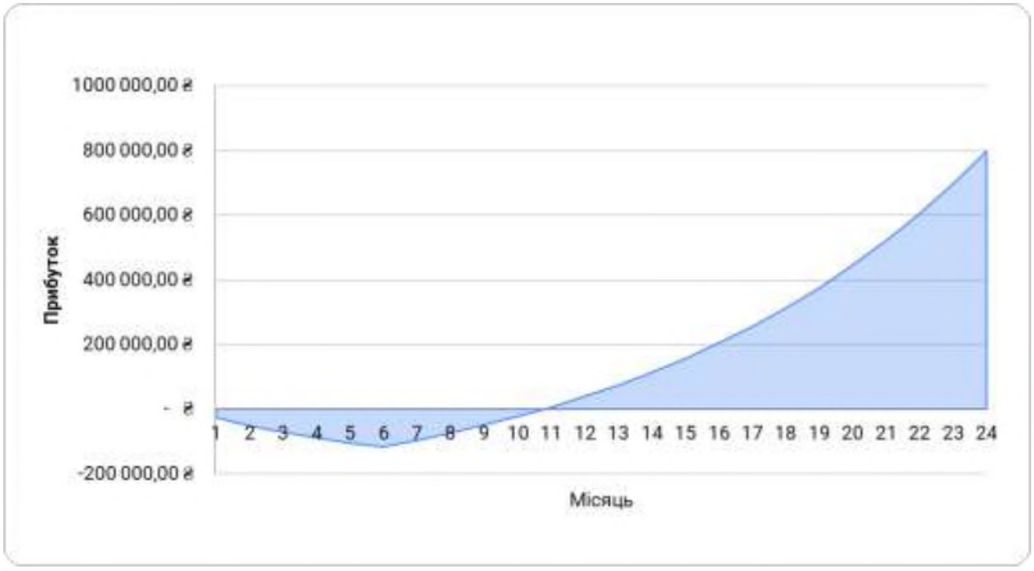

Рисунок 3.7 – Результати розрахунку терміну окупності композиції вебсервісів

Результати розрахунків показали, що за значень вхідних параметрів економічної моделі з табл. 3.4, термін окупності затрат на композицію вебсервісів складатиме 11 місяців.

Висновки до розділу 3

У третьому розділі послідовно вирішено задачу раціонального вибору композиції вебсервісів на прикладі додатку для пошуку найближчих лікарень. З використанням термінів теорій прийняття рішень та важливості критеріїв описується ефективний алгоритм формування множини Парето, а також вводиться людино-машинна процедура звуження множини Парето за рахунок виявлення рішення. Представлено технічний опис системи, що допомагає здійснити раціональний вибір композиції вебсервісів за рахунок реалізації алгоритму, представленому у попередньому розділі. Наводиться опис архітектури розробленого програмного комплексу, його основних компонентів та деталей їх реалізації. Демонструється детальний проєкт системи на прикладі розробки додатку для пошуку найближчих лікарень. Також у розділі проведено побудову марковського графа для моделі композитних сервісів. Розрахований коефіцієнт

готовності, а також проведений аналіз впливу вхідних параметрів на його сталий показник. Аналіз результатів моделювання, вказує на те, що при обраних параметрах системи головною загрозою для готовності вебсервісу є два основні аспекти. З одного боку, це можливість виникнення прихованих відмов у програмній компоненті. З іншого боку, є ризик прискореного проведення профілактичних заходів щодо цих відмов, що також може вплинути на готовність системи. Результати економічних розрахунків показали, що термін окупності затрат на композицію вебсервісів складатиме 11 місяців.

ВИСНОВКИ

У роботі була поставлена і вирішена актуальна наукова задача розробки та дослідження вдосконаленої моделі системи з сервісноорієнтованою архітектурою для вирішення завдання багатокритеріального вибору композицій вебсервісів.

У ході виконання кваліфікаційної роботи було виконано оцінку сучасного стану проблеми оптимальної композиції вебсервісів у рамках сервісноорієнтованої архітектури, підкреслено її актуальність та значущість. Вирішення цієї проблеми неможливе без використання моделювання вебсервісів, визначення ключових функціональних та нефункціональних характеристик, а також без створення моделі самої системи на основі сервісноорієнтованої архітектури. На основі роботи можна сформулювати наступні висновки.

1. Дотепер моделювання вебсервісів здійснювалось для оцінки їх статичних показників, але питанням моделювання поведінки вебсервісів приділялось недостатньо уваги. З іншого боку, правильне проєктування сервісів, застосування балансувальників навантаження, хмарних технологій може допомогти уникнути збільшення часу обробки запитів при зростанні навантаження. Таким чином, важливо мати такі характеристики вебсервісів, які відображають зміни їх продуктивності та надійності при збільшенні навантаження.

2. Використання теорії важливості критеріїв, що базується на кількісних шкалах вимірювання, надає значні переваги у порівнянні з методами на основі згортки, аналізу ієрархій, нелінійної оптимізації. Процедура вибору, що враховує людський фактор, має чітку форму представлення збитків та компенсацій для значень критеріїв у відповідних шкалах виміру.

3. Використання ефективного алгоритму формування множини Парето дозволяє здійснювати вибір за прийнятний час, навіть за значної кількості альтернатив та критеріїв. У проведеному аналізі було оцінено критерії порівняння, значення яких можна отримати автоматизованим способом.

4. Вирішено задачу раціонального вибору композиції вебсервісів на прикладі додатку для пошуку найближчих лікарень. З використанням термінів теорій

прийняття рішень та важливості критеріїв описано ефективний алгоритм формування множини Парето, а також запропонована людино-машинна процедура звуження множини Парето за рахунок виявлення рішення. Представлено технічний опис системи, що допомагає здійснити раціональний вибір композиції вебсервісів за рахунок реалізації алгоритму.

5. Побудовано марковський граф для моделі композитних сервісів. Розрахований коефіцієнт готовності, а також проведений аналіз впливу вхідних параметрів на його сталий показник. Аналіз результатів моделювання, вказує на те, що при обраних параметрах системи головною загрозою для готовності вебсервісу є два основні аспекти. З одного боку, це можливість виникнення прихованих відмов у програмній компоненті. З іншого боку, є ризик прискореного проведення профілактичних заходів щодо цих відмов, що також може вплинути на готовність системи..

7. Визначено порядок розрахунку економічної ефективності композиції вебсервісів для сервісноорієнтованої архітектури. Результати економічних розрахунків показали, що термін окупності затрат на композицію вебсервісів складатиме 11 місяців.

Таким чином, поставлені задачі розв‘язано у повному обсязі. Напрямком подальших досліджень є розробка та дослідження імітаційних моделей функціонування вебсервісів при непуасонівських потоках впливів на систему.