Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Моделі хмарного тестування програмного забезпечення

(тема)

Виконав:

студент II курсу, групи СПзм-21-1

Марченко В.В.
(прізвище, ініціали)

Спеціальність

123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: проф. Волк М.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Марченку Владиславу Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделі хмарного тестування програмного забезпечення

затверджена наказом по університету від " 24 " березня 2023 р. № 60 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 17 травня 2023 р.

3. Вхідні дані до роботи

Моделі хмарних обчислень.
Критерії оцінки якості хмарних сервісів
Рівні обслуговування хмарних обчислень
Методи хмарного тестування програмного забезпечення

4. Перелік питань, що потрібно опрацювати у роботі

Аналіз предметної області
Розроблення фреймворку хмарного тестування програмного забезпечення
Реалізація програмної системи хмарного тестування програмного забезпечення
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Презентація 14 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області	27.03.23 – 06.04.23	
2	Розробка моделей	06.04.23 – 12.04.23	
3	Реалізація алгоритмів	13.04.23 – 20.04.23	
4	Розробка структури програмних засобів	21.04.23 – 25.04.23	
5	Розробка програмних модулів	26.04.23 – 30.04.23	
6	Оформлення матеріалів кваліфікаційної роботи	01.05.23 – 10.05.23	
7	Подання кваліфікаційної роботи керівникові та	11.05.23 – 14.05.23	
	попередній захист		
8	Подання кваліфікаційної роботи на	14.05.23 – 17.05.23	
	рецензування		

Дата видачі завдання 27 березня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Волк М.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 56 с., 13 рис., 2 табл., 1 дод., 25 джерел.

МОДЕЛЬ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ТЕСТУВАННЯ, ХМАРНІ ТЕХНОЛОГІЇ.

Метою кваліфікаційної роботи є розроблення моделі середовища хмарних обчислень для тестування програмного забезпечення з різними типами хмар.

Модель є основою розробки хмарних фреймворків, створених для вирішення проблем у галузі комп'ютерної інженерії, які пов'язані з інтеграцією сервісів тестування у розподілене мережеве середовище та перевіркою хмарних програмних систем, оцінкою програмних середовищ виконання завдань та управлінням обчислювальним середовищем. З точки зору розробки програмного забезпечення, робота демонструє, як нова парадигма веб-сервісу дозволяє відносно просто включати як старе так і нове програмне забезпечення та тестувати їх взаємодію.

ABSTRACT

Master’s thesis: 56 pages, 13 figures, 2 tables, 1 appendice, 25 sources.

MODEL, SOFTWARE, TESTING, CLOUD TECHNOLOGIES.

The purpose of this qualification work is to develop a cloud computing environment model for testing software with different types of clouds.

The model is the basis for the development of cloud frameworks created to solve problems in the field of computer engineering, which are related to the integration of testing services in a distributed network environment and the verification of cloud software systems, the evaluation of software environments for the execution of tasks and the management of the computing environment. From a software development perspective, the paper demonstrates how the new web service paradigm makes it relatively easy to incorporate both old and new software and test their interactions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення

API – Application Programming Interface

IaaS – Infrastructure as a Service

IDE – Integrated Development Environment

IoT – Internet of Things

IT – Information Technology

P2P – Peer-to-Peer

PaaS – Platform as a service

SaaS – Software as a service

SLA – Service Level Agreement

TaaS – Testing as a service

QoS – Quality of service

UDDI – Universal Description Discovery & Integration

WSDL – Web Services Description Language

ВСТУП

Хмарні обчислення — це нова платформа сервісних обчислень, призначена для швидкої та динамічної доставки гарантованих обчислювальних ресурсів. Хмарні обчислення передбачають угоди про рівень обслуговування (SLA) для гарантованої доступності безперебійної роботи для забезпечення зручного мережевого доступу за вимогою до розподілених і спільних обчислювальних ресурсів. Хоча парадигма хмарних обчислень зберігає свій потенційний статус у сфері розподілених обчислень, хмарні платформи ще не в центрі уваги більшості дослідників і практиків. Якщо говорити точніше, спільнота дослідників і практиків все ще має фрагментовані та недосконалі знання про принципи та методи хмарних обчислень. В цьому контексті, однією з головних мотивацій роботи є виявлення різноманітних переваг парадигми хмарних обчислень, і, отже, метою цієї роботи є розроблення моделі хмарного тестування програмного забезпечення.

У сучасному світі веб-сервіси є відомим сприйняттям для всіх користувачів, які користуються Інтернетом. Процес дослідження веб-сервісу передбачає виявлення, вибір і ранжування сервісів. Виявлення – це процес зіставлення запиту користувача до хмарної програмної системи. Одним з елементів загальної роботи є розробка моделі для виявлення веб-сервісів із комбінованим підходом вибору та рейтингу сервісів. У роботі запропоновано техніку процесу виявлення веб-сервісів, поєднуючи пошук за ключовими словами та семантичний пошук і ранжування сервісів. Результати впровадження показують, що запропонована модель працює краще для процесу виявлення веб-сервісу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Парадигма хмарних обчислень

Хмарні обчислення – це нещодавня еволюція парадигми розподілених обчислень, яка може підтримувати спільний доступ до послуг за вимогою з вищим рівнем гнучкості та динамічної масштабованості. Гнучкість і масштабованість у середовищі хмарних обчислень може бути досягнута шляхом балансування навантаження екземплярів додатків, які працюють окремо в різних операційних системах і підключаються через веб-сервіси [1].

Хмарні обчислення забезпечують угоди про рівень обслуговування (SLA) для гарантованої доступності безвідмовної роботи. Крім того, хмарні обчислення використовують модель для забезпечення зручного мережевого доступу на вимогу до розподілених і спільних обчислювальних ресурсів, які можна конфігурувати [2,3]. Найбільша перевага хмарних обчислень полягає в тому, що користувачеві не потрібно знати технологію та контролювати інфраструктуру в хмарному середовищі.

Хоча хмарна парадигма зберігає свій потенційний статус у сфері розподілених обчислень, хмарні платформи ще не в центрі уваги більшості людей. Точніше кажучи, спільнота дослідників і практиків все ще має фрагментарні та недосконалі знання про принципи та методи парадигма хмарних обчислень. Завдяки цьому інші парадигми здаються кращими, але цей сценарій незабаром буде змінено на користь реальних привабливостей хмарних обчислень, включаючи масштабованість і економічно ефективні послуги [2,4]. Отже, дослідники зобов’язані змусити практиків усвідомити особливості середовища хмарних обчислень. У цьому контексті однією з головних мотивацій роботи, представленої роботі, є виявлення різноманітних переваг парадигми хмарних обчислень, і, отже, метою цієї роботи є виявлення надзвичайного значення парадигми хмарних обчислень через

прикладне середовище. Ця розробка спрямована в широкому обсязі, і її значення реалізуються в обмеженому обсязі, який може бути розширений у відповідній мірі.

Після прийняття рішення про впровадження хмарного середовища розробник повинен прийняти стандартну процедуру для його впровадження. Але наразі не існує такої стандартної процедури чи вказівок для побудови середовища хмарних обчислень, тому тут послуги реалізуються за допомогою концепцій веб-сервісів. Технологія систем, заснованих на веб-сервісах, за останні роки викликала великий інтерес серед користувачів і розробників через її здатність стати новою парадигмою концептуалізації, проектування та реалізації програмних систем. Ця характеристика особливо приваблива для створення програмного забезпечення, яке працює в розподілених і відкритих середовищах, таких як Інтернет, і ця властивість збільшує потребу в системах веб-сервісів, які складаються з сервісів, які спілкуються одноранговим способом. Центральним для розробки та ефективної роботи таких веб-сервісів є основний набір питань і дослідницьких питань, які протягом багатьох років вивчалися розподіленою спільнотою [5-7]. Це великомасштабні системи, і дослідницьке співтовариство прагне до співпраці веб-сервісів, щоб досягти своїх функцій у дуже гнучкий спосіб. Але, на жаль, в існуючих системах занадто мало описано з достатньою кількістю деталей, щоб прийняти їх для реальних розробок [8-9]. Рішення полягає в тому, щоб побудувати та використовувати об’єктивні специфічні стандарти, які найкраще підходять для даного проблемного простору.

З точки зору керування хмарами, критична проблема створення ефективних хмар — зробити їх стійкими до потенційних збоїв. У деяких випадках системні збої можуть спричинити низьку продуктивність програми, хоча вона добре розроблена та розгорнута. Таким чином, на додаток до функціональних особливостей, програма повинна мати можливість координуватися з системою, яка обробляє конкретну програму. З іншого

боку, хмара також повинна мати такі можливості для координації з додатками. Таким чином, відповідна обробка збоїв або виняткових ситуацій у хмарній структурі призначена не лише для покращення якості системи з точки зору доступності, а також для покращення продуктивності домену програми. У цьому вигляді можливий набір винятків і механізм обробки винятків також описані, щоб допомогти в управлінні хмарами, представленими в цьому документі. Ці механізми також можна розглядати як загальні та застосовувати їх там, де це необхідно.

1.2 Моделі хмарних обчислень

Хмарні обчислення використовують Інтернет і центральні віддалені сервери для підтримки даних і програм і, таким чином, дозволяють користувачам отримувати доступ до технологічних послуг з Інтернету без знання, досвіду або контролю над технологічною інфраструктурою. що їх підтримує. Динамічність і спільне використання ресурсів є ключовими архітектурними показниками хмарних обчислень, і ця технологія дозволяє набагато ефективніше обчислювати за рахунок централізації зберігання, пам’яті, обробки та пропускної здатності. Загалом, хмарні сервіси можна згрупувати в три великі категорії (моделі) [1] [3-7].

Програмне забезпечення як послуга (SaaS) – це свого роду програма, яка працює виключно в хмарі. Сервіси, як правило, розміщуються та керуються у власному центрі обробки даних і роблять їх доступними через Інтернет.

Платформа як послуга (Paas) – ці категорії служб є локальними програмами, які мають створюватися на основі спільного використання платформи у хмарі. Ця система полегшує розробку та розгортання додатків без витрат і складнощів, пов’язаних із купівлею базової інфраструктури та керування нею, а також розширює функціональні можливості локальних додатків за допомогою доступу до послуг, що надаються в хмарі. Їх можна

розглядати як набір прикріплених служб, доступних лише певній програмі.

Інфраструктура як послуга (PaaS) – це надання апаратного забезпечення та відповідного програмного забезпечення як послуги. Хмарна платформа надає хмарні послуги для створення додатків, а не створення власної спеціальної основи, їх можна створити на хмарній платформі.

Загальна взаємодія між локальними додатками та хмарними службами [1] представлена на рисунку 1.1. Наша робота базується на категорії PaaS, яка включає служби додатків локальних і хмарних середовищ.

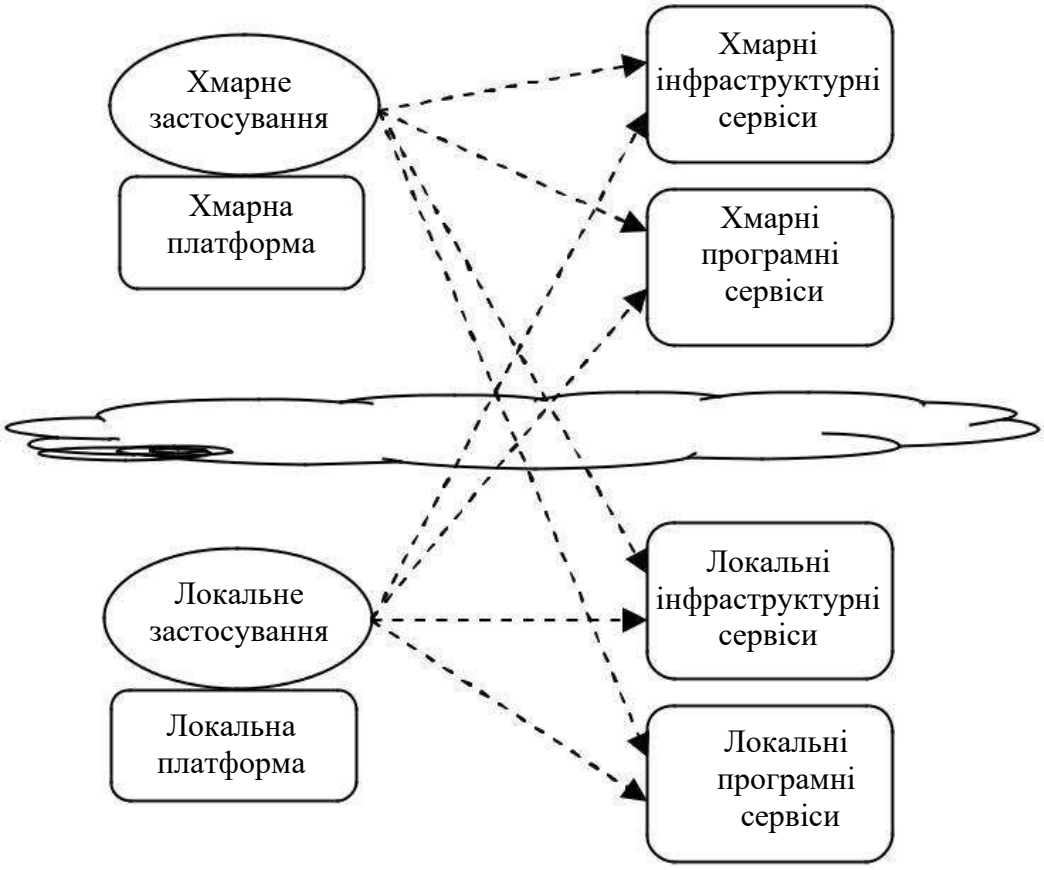

Рисунок 1.1 – Взаємозв’язок між локальними програмами та хмарою

1.3 Оцінка якості хмарних сервісів

Багато дослідників зосереджуються на технології веб-сервісів, оскільки в наш час все більше веб-сервісів надають постачальники послуг. Тому вибір

веб-сервісів лише на основі функціональних вимог не задовольняє вимоги користувачів. Якість обслуговування (QoS) є нефункціональним параметром, який слід враховувати для вибору найкращого сервісу. Таким чином, на основі QoS користувач може отримати точні та найкращі послуги для свого бізнесу. Для наших експериментальних цілей ми використали набори даних реального світу. Існує багато веб-сайтів, таких як XMethods.net тощо, які надають сховище для веб-служб. У більшій частині нашого дослідження ми аналізували, що процес виявлення базується на вимогах користувача.

Багато дослідників запропонували кілька алгоритмів для виявлення, вибору та ранжування послуг. Традиційний процес виявлення використовує лише пошук за ключовими словами, пошук не здатний знайти всі пов’язані служби для користувачів [10,11]. У процесі пошуку за ключовими словами є можливість пропускати послуги відповідно до вимог користувача. Тому ми пропонуємо модель із поєднанням пошуку за ключовими словами та семантичного пошуку. І ми оцінюємо послуги на основі нормалізації значень параметра QoS. Рисунок 1.2 показує стандартну модель веб-сервісу. Вона в основному складається з трьох компонентів: постачальника послуг, споживача послуг і UDDI (універсальний опис, виявлення та інтеграція).

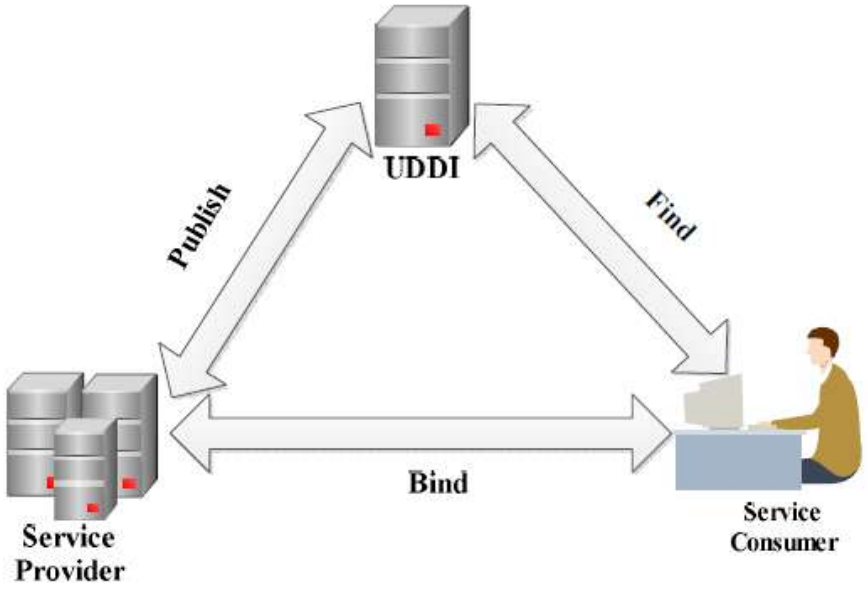

Рисунок 1.2 – Модель веб-сервісу

Постачальник послуг реєструє послугу в UDDI з усіма деталями, такими як місцезнаходження, назва постачальника тощо. Деталі зберігатимуться в сховищі UDDI [10,11]. Споживач, якому потрібні послуги, запитує репозиторій UDDI відповідно до своїх потреб. Тоді UDDI приймає запит користувача та шукає відповідні служби в реєстрі. Потім відповідні результати будуть надані споживачу послуги.

Роботи у цьому напрямку в основному зосереджуються на нефункціональних параметрах і оцінюють послуги відповідно до довіри та якості обслуговування постачальників послуг [12,13]. Інша річ, вони стверджують, що децентралізований підхід не дуже ефективний [14]. Вони запропонували модель виявлення веб-сервісу з QoS на основі переваг користувача та запровадили трирівневий механізм відповідності [15].

Пропонується архітектура для виявлення веб-сервісів на основі відповідності агента та QoS, вибору та ранжирування. Модель також отримує потрібні параметри якості обслуговування для точності та ефективності [16,17]. Вони в основному зосереджені на динамічному виявленні веб-сервісів із QoS на основі моделі SOA, техніка, яку вони дотримуються, це автомати кінцевого стану, щоб уникнути неправильних переходів [18].

Токаж запропоновано семантичну модель і розроблено P2P підхід для підхід до розподіленого виявлення веб-служб [11,19], оскільки старий традиційний UDDI є неефективним, а wsdl не включає семантичну специфікацію [20].

В іншої роботі запропоновано модель виявлення семантичної веб-служби на основі кількох агентів [11,19], коли специфікація брокера не змінюється під час виявлення служби. Модель аналізує запит користувача та відповідає запиту в реєстрі [21,22]. Вони обговорюють огляд і обмеження контекстно-орієнтованих методів у виявленні послуг, а також вказали на контекстну інформацію. Було відмічено, що визначення контексту все ще неможливо ідентифікувати [16].

В роботі [23] було змодельовано спеціалізовану пошукову систему для

виявлення служб, яка витягує інформацію з мов опису веб-служб (WSDL) із тегів і анотацій. Вони аналізують недоліки UDDI на основі брокера та пропонують деякі плани просування існуючих стандартів [24,25]. В роботі пропонують багатокритеріальний підхід до виявлення послуг на основі уподобань користувача та включають семантику та використовують меншу кількість параметрів QoS [10]. Грунтуючись на обговоренні споріднених робіт, було доказано актуальность ношої дослідницької роботи.

1.4 Хмарне тестування програмного забезпечення

У зв’язку з раптовим зростанням використання програмних додатків у всьому світі підприємствам стає надзвичайно важко встигнути задовольнити вимоги ринку. Групи забезпечення якості підприємства, здатні виявляти помилки якнайшвидше, матимуть більше часу для роботи на різних інших етапах розробки, а також для підвищення якості додатків. З появою технології хмарних обчислень підприємства скористалися кількома інноваційними можливостями в тестуванні та розгортанні програмного забезпечення. Це відкрило нові можливості, зокрема у сфері тестування та обслуговування програмного забезпечення.

Хмарне тестування програмного забезпечення – це набір процедур, інструментів і процесів, які тестувальники використовують для неефективного та точного тестування програмного забезпечення. Завдяки використанню моделей хмарних послуг підприємства можуть впроваджувати тестування як послугу без необхідності повністю інвестувати в тестові лабораторії, інструменти чи інфраструктуру. Хмарні сервіси стосуються не лише тестування, а й усього, починаючи від хмарної безпеки, розробки програмного забезпечення, використання ресурсів тощо.

Без належного тестування програмне забезпечення завжди буде вразливим до загроз, помилок, порушень безпеки та багатьох інших факторів, які суттєво впливатимуть на взаємодію з клієнтами. Тому, обираючи

інструмент тестування, інфраструктуру чи службу автоматизації з хмари, тестувальники повинні повністю розуміти сумісність платформи, підтримку ресурсів, гнучкість і вартість послуг. Повністю впроваджена система забезпечення якості на багатьох етапах впровадження та адміністрування хмари має бути основним завданням. Це означає, що підприємства повинні здійснювати надійне планування та умови контролю якості, щоб гарантувати, що хмарні служби успішно охоплюють увесь процес автоматизації тестування. Крім того, підприємства повинні чітко усвідомлювати зростаючий рівень регулятивного контролю хмарних технологій. Клієнти очікують доказів того, що підприємства повністю розробили та впровадили стратегії тестування програмного забезпечення на основі хмари для даних, гнучкості та виходу з програм.

Технологія хмарних обчислень зробила революцію в тому, як підприємства використовують IT-ресурси протягом життєвого циклу розробки програмного забезпечення. Від розробки та тестування до остаточного розгортання, це покращило життєвий цикл програмного забезпечення та значно скоротило час і вартість. Хмарні обчислення визначаються як платформа для сприяння повсюдному, релевантному, бюджетному, мережевому доступу на вимогу до кількох обчислювальних ресурсів одночасно. Підприємства можуть використовувати все: від мереж і серверів до сховищ і інструментів без необхідності купувати все рішення. Ці ресурси можна швидко надати з хмари та впровадити з мінімальними зусиллями адміністратора або взаємодії постачальника послуг.

Це робить ще більш важливим для підприємств використовувати хмарні сервіси в різних бізнес-операціях.

Ось деякі переваги для підприємств, які використовують технології хмарних обчислень.

Самообслуговування на вимогу: обчислювальні ресурси, такі як час продуктивності сервера та хмарне сховище, можуть безперешкодно використовуватися підприємством. Після того як підприємство

зареєструється для отримання необхідних послуг, не буде необхідності взаємодіяти з постачальником хмарних послуг.

Широкий доступ до мережі: взаємодія з хмарою або зв'язок між пристроями базується на службах хмарної мережі. Таким чином, великі підприємства, які використовують кілька пристроїв IoT, інтелектуальних пристроїв, машин або навіть ноутбуків і телефонів, можуть безперешкодно з'єднувати пристрої та отримувати доступ до них через хмарні мережі.

Об'єднання ресурсів: загальнодоступні обчислювальні ресурси розподіляються та доступні декільком споживачам за межами одного підприємства. Цей тип спільного використання ресурсів допомагає компаніям використовувати обробку, зберігання, пам'ять, центри обробки даних і багато іншого з різних географічних місць.

Швидка еластичність: обчислювальні ресурси можуть автоматично надаватися підприємствам на певний час залежно від потреби. Ресурси здебільшого знаходяться під політикою оплати за використання, що полегшує підприємствам їх впровадження або скасування, коли вони не потрібні.

Більшість IT-компаній і розробників програмного забезпечення зараз переносять свої застарілі системи в хмарну екосистему для кращих послуг автоматизації тестування. Завдяки хмарному тестуванню їхні програми є масштабованими, гнучкими та легко адаптованими. Ось кілька причин, чому підприємства використовують хмарне тестування програмного забезпечення замість традиційного або ручного тестування програм.

Це значно скорочує витрати та цикли процесу за рахунок спільного використання ресурсів під час виконання стратегії тестування. Це пов'язано з тим, що хмарне тестування як послуга (TaaS) дозволяє IT-розробникам і розробникам програмного забезпечення ініціалізувати практичні експериментальні тести на хмарних платформах без необхідності мати ліцензії чи купувати ресурс. Це зменшує витрати на тестування та покращує спільне використання ресурсів і використання послуг.

Краще тестове середовище тестування та віртуальної інфраструктури. Гнучкість хмарних технологій дозволяє підприємствам використовувати TaaS з будь-якої точки земної кулі, якщо місце має хороше підключення до Інтернету. Крім того, хмара забезпечує краще віртуальне середовище для тестування та рішень SaaS, які підтримують весь життєвий цикл тестування, включаючи розробку. Завдяки цій віртуальній інфраструктурі підприємствам не доведеться витрачати багато на справжні лабораторії чи генератори трафіку, а просто орендувати ресурси з хмари.

Політика оплати за використання хмарних сервісів є найбільш помітним фактором для підприємств. На відміну від традиційного тестування програмного забезпечення, у хмарному тестуванні програмного забезпечення підприємства можуть вибрати ресурси, інструменти або технології саме на потрібний їм час. Їм доведеться платити лише за послуги на основі часу використання та можуть припинити використання хмарних служб після завершення тестування.

Хмарні обчислення та хмарне тестування програмного забезпечення відкривають нові можливості для команд із забезпечення якості та компаній, які займаються розробкою програмного забезпечення. Особливо під час поточної кризи COVID-19 та війни, коли існують величезні обмеження на спілкування з людьми, ручне тестування зіткнеться з серйозними проблемами. Завдяки тестуванню програмного забезпечення на основі хмари підприємства можуть використовувати хмарні рішення з будь-якої точки земної кулі в будь-який час. Навіть постковідна ера побачить зростання технологічних інновацій, де більше додатків матимуть сучасні досягнення, такі як штучний інтелект і машинне навчання, інтегровані в їхню архітектуру. Для ефективного тестування таких програм і забезпечення нульової помилки підприємства повинні інвестувати в хмарні обчислення та тестувати послуги автоматизації.

Парадигма зсуву вліво життєвого циклу розробки програмного забезпечення показує, що тестування може бути реалізоване на ранніх

стадіях життєвого циклу розробки програмного забезпечення, оскільки беззаперечно, що будь-яка помилка у виробництві коштує дорожче, ніж найбільший збій у підготовчому виробництві.

Тестування є обов’язковим для доставки надійного програмного забезпечення вашим кінцевим користувачам. Оскільки сучасні програми мають розподілену архітектуру, розміщені в хмарі або локально, їх важко протестувати, а ще важче усунути неполадки, коли тестування не вдається.

Завдяки можливості збирати дані з програми в режимі реального часу розробники та інженери з контролю якості отримують 100% спостережуваність на рівні коду та швидше вирішують помилки тестування. Засоби хмарного тестування замінюють ітеративний негнучкий процес, який сьогодні потрібен для налагодження невдалих збірок у конвеєрах CI, бо вони не залежать від платформи, працюють локально, у хмарі, на контейнерах і безсерверному коді.

2 РОЗРОБЛЕННЯ ФРЕЙВОРКУ ХМАРНОГО ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Розроблення моделі хмарного тестування програмного забезпечення

При моделюванні запропонованої системи буде легше зрозуміти модель, виразивши її у математичної формі визначення продукту. У цій формі модель використовується для визначення різних наборів послуг тестування щодо продукту, який потрібно перевірити. У всіх випадках друга категорія є підмножиною попередньої. З точки зору продукту, пропоновану систему можна визначити з точки зору конкретних послуг продукту.

Визначення: нехай A_i – набор хмар, необхідних для тестування продукту P_i, тоді A можна визначити як у рівнянні (2.1).

$$A_{P_i} = \left\{ \begin{array}{l} \left(sm_i, tc_{i1}, tc_{i3}, tc_{i3}, \ \ldots \ , tc_{iy}\right), \\ tc_{i1} = \left(ts_{i(11)}, ts_{i(12)}, \ldots, ts_{i(1K_{i1})},\right), \\ tc_{i2} = \left(ts_{i(21)}, ts_{i(22)}, \ldots, ts_{i(2K_{i2})},\right), \\ \vdots \\ tc_{iy} = \left(ts_{i(y1)}, ts_{i(y2)}, \ldots, ts_{i(yK_{iy})}\right) \end{array} \right\}, \qquad (2.1)$$

де sm_i – сервіс-менеджер продукту P_i і належить до служб локальних програм, а H – кількість продуктів, які потрібно перевірити одночасно, $0 < i \le H$;

tc_{ij} – одна з тестових хмар з A, а $tc_{i(ju)}$ – один з сервісів тестування з tc_{ij}, $0 < j \le y, 0 < u \le K_{ij} - 1$;

y – це кількість різних тестових хмар, необхідних для продукту P_i;

K_{ij} – максимальна кількість послуг тестування a_{ij} і K_{ij} відноситься до

загальної кількості послуг тестування, доступних у певних хмарах тестування j продукту P_i.

Оскільки ця хмарна структура забезпечує середовище тестування скалярного типу, у будь-який момент, $A_{i_1} \cap A_{i_2} = \varnothing$, де, $0 < i_1, i_2 \le H \; i \; i_1 \ne i_2$, тобто на якомусь конкретному сервісі, ані одна хмара тестування, ані служби тестування не можуть використовуватися одночасно більш ніж одним продуктом. Логічний макет запропонованої хмарної структури для тестування програмного забезпечення показано на рисунку 2.1.

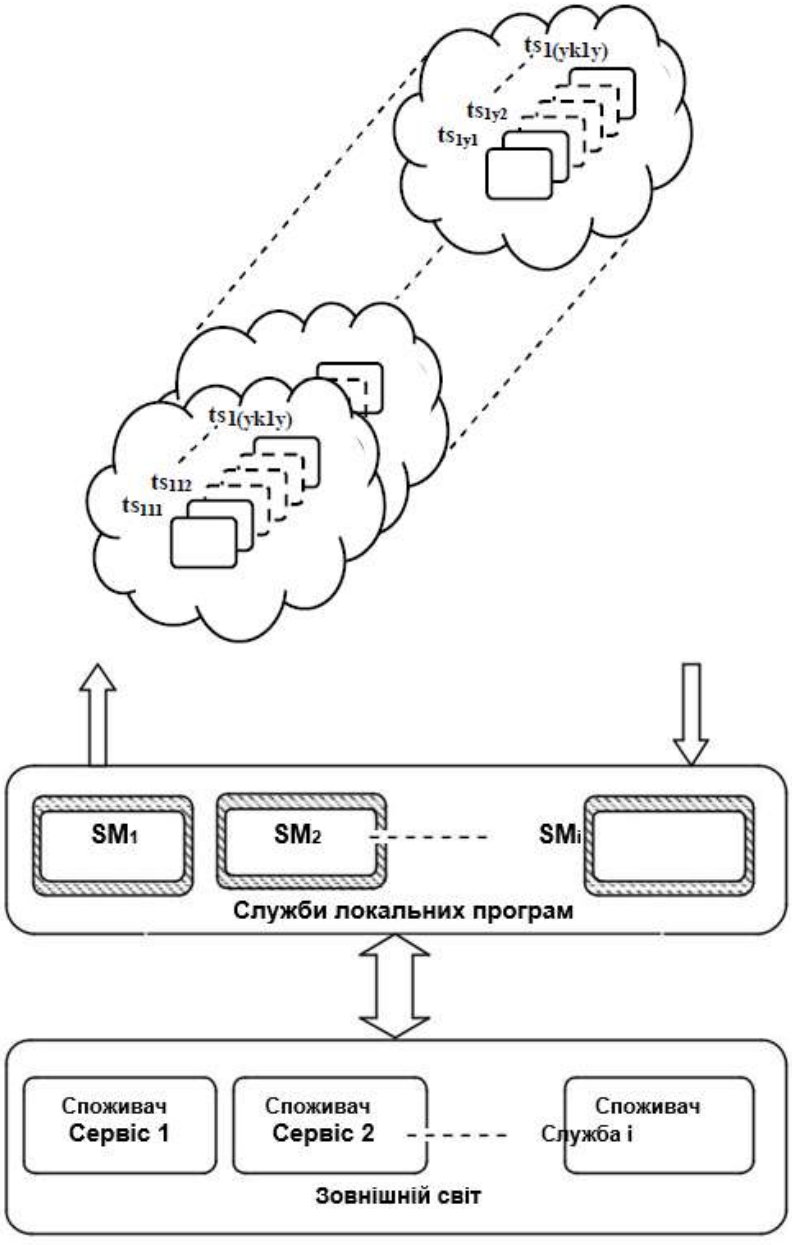

Рисунок 2.1 – Запропоноване хмарне середовище тестування

Загалом є три набори компонентів: сервіс менеджер $(\{sm_1, sm_2, \ldots, sm_z\})$, хмара тестування $(\{tc_1, tc_2, \ldots, tc_x\})$ і клони тестових сервісів $(\{ts_{11}, ts_{12}, \ldots, ts_{1K_1}\}, \ldots, \{ts_{x1}, ts_{x2}, \ldots, ts_{xK_{x1}}\})$, кожен з яких працює локально на різних машинах у мережі. Логічні зв'язки між службами стосуються залежності рівня обслуговування між службами в структурі. Така залежність може означати той факт, що одна служба залежить від іншої для досягнення мети, завдання, яке потрібно виконати, або ресурсу, який буде доступним.

Під час уточнення запропонованої системи було виявлено, що дуже бажано пояснити модель у багаторівневому підході; існує два типи (рівні) послуг: локальні служби додатків і служби хмарних додатків. Рівень хмарних сервісів додатків є динамічним, який створюється під час процесів тестування. Загальна підтримка складається з підрозділів, які спеціалізуються на підтримці різних послуг тестування.

Менеджер служби відповідає за нагляд і координацію основної діяльності відповідних рівнів загального середовища. Отримавши завдання від користувача, менеджер служби (sm_i) сформує хмару тестування і може визначити множину $(\{tc_1, tc_2, \ldots, tc_x\})$. Процес визначення набору необхідних хмар залежить від методів тестування, необхідних для будь-якого продукту P_i. Потім, залежно від навантаження, кожна хмара тестування вибере одну або кілька додаткових служб тестування з відповідними методами та визначить набори служб тестування. Ці додаткові служби мають називатися вторинними службами з множині Service Manager.

Кожна хмара відповідає за надання певного стандарту виводу, якому має відповідати вивід завдання. Стандарт виводу залежить від типу завдання та необхідних властивостей виводу, таких як кількість тестів, знайдених дефектів, час, витрачений на тестування тощо. Вхідні дані cs_i подаються від тестувальника і включають набір продукту для тестування, специфікацію часу для тестування, оцінки детектора дефектів і специфікацію щодо необхідних методів тестування. Це передається до диспетчера послуг sm_i з

cs_i. Приймальна служба отримує вхідні дані від cs_i та повинна проаналізувати специфікацію щодо необхідних методів тестування. На основі специфікацій служби тестування відповідний набір хмар може бути визначено та ідентифіковано за допомогою процесу узгодження в службах хмарних додатків. За погодженням утворюються хмари $К_у$.

Потім призначення буде розподілено в ідентифікованому наборі Testing Cloud, а їхні результати можна отримати для інтеграції. Звіти про випробування навколишнього середовища з ідентифікованих хмар тестування будуть інтегровані в блок керування тестуванням, а потім передані у зовнішній світ (cs_i) через службу cm_i. Менеджер служби відповідає за всі види координаційної діяльності в цій системі. Початковий вхід до Testing Cloud tc_{ij} надходить для керування програмами тестування. Він включає в себе набір методик тестування, специфікацію часу для тестування, оцінки дефектоскопа та оцінку складності виробу. Це передається до Testing Clouds tc_{ij} з набором оціночних значень середнього розміру модулів продукту та прогнозованих значень загальної кількості тестових випадків, які має створити tc_{ij}, середній розмір тесту та середній час, необхідний для створення та виконання модульного тесту.

Режим розподілу навантаження визначає менеджер сервісу на основі вхідних даних служби споживачів. Він також відповідає за визначення кількості служб тестування, які необхідні для створення хмари. Інформація про хмару зберігається у базі даних та використовується паралельними процесами, які поступово вибірають тести з бази. Крім того, необхідно виконати процеси реєстрації, розподілу навантаження та збору результатів від служб тестування. У той же час він розподілить навантаження на хмари тестування того самого tc_{ij}. Результати хмарного тестування, тобто звіти про часткові випробування навколишнього середовища від служб тестування, будуть оброблені менеджером програм тестування. Потім створений звіт про випробування tc_{ij} буде передано до бази даних.

2.2 Розроблення структури системи хмарного тестування програмного забезпечення

Виявлення сервісу є процесом майнінгу через обслуговування великого сховища сервісів. Існує потреба у вдосконаленні процесу виявлення сервісів. Структура пропонується на базі дослідження літератури і запропонованої моделі системи хмарного тестування програмного забезпечення, як показано на рисунку 2.2.

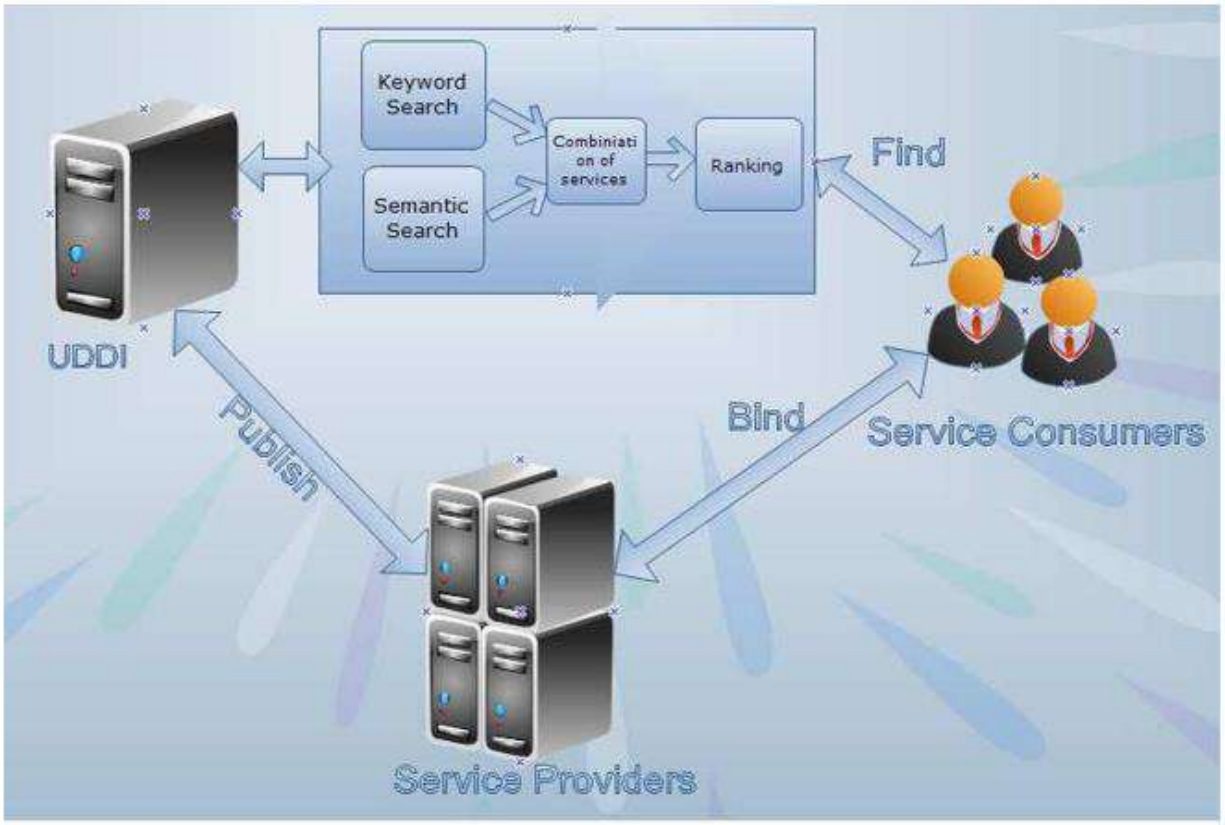

Рисунок 2.2 – Модель виявлення та ранжирування веб-служб

Пропонована система складається з постачальника послуг, споживача послуг і UDDI (універсальний опис, виявлення та інтеграція). Модуль WSDSR складається з пошуку за ключовими словами, семантичного пошуку та ранжирування. Спочатку виконується пошук на основі ключових слів, а потім пошук на основі семантики, після чого обидва результати об’єднуються разом, і буде отримано набір сервісів. Після цього для

ранжування значення параметрів QoS будуть нормалізовані до тієї ж шкали. Після процесу нормалізації послуги будуть ранжовані, отримані послуги будуть надані кінцевому користувачеві. Розрахунок нормалізації QoS наведено нижче.

Нормалізація для позитивної оцінки якості:

$$QoS^{P}(i) = \frac{QoS_i - QoS_{i\min}}{QoS_{i\max} - QoS_{i\min}} \tag{2.2}$$

Нормалізація для негативної оцінки якості:

$$QoS^{N}(i) = \frac{QoS_{i\max} - QoS_i}{QoS_{i\max} - QoS_{i\min}} \tag{2.3}$$

Під час тестування виникає ряд проблем і труднощів хмари та хмарне програмне забезпечення. Тут ми їх обговорюємо з наступних чотирьох областей.

Побудова тестового середовища на вимогу. Хоча нинішня хмара технології підтримують автоматичне надання необхідні обчислювальні ресурси для кожного SaaS (або додаток) у хмарі, немає підтримки рішення, щоб допомогти інженерам налаштувати необхідну тестове середовище в хмарі за допомогою економічно ефективного спосіб. Необхідно надати тест на вимогу середовище для клієнтів TaaS. Для цього TaaS постачальники повинні надати системне рішення створити необхідне тестове середовище на основі на вибір користувача. Крім того, інженери також виявлено, що існує відсутність рентабельності рішення для них, щоб легко використовувати свої хмарні програми (або SaaS) у хмарі за допомогою існуючі інструменти тестування, оскільки більшість із них такими не є з підтримкою хмари.

Тестування масштабованості та продуктивності. Є багато опублікованих робіт обговорюють систему тестування продуктивності та

оцінка масштабованості за останні два десятиліття більшість із них звертаються питань і рішень у звичайних розподілених програмне забезпечення або веб-системи програмного забезпечення.

Відповідно до нашого нещодавнього огляду літератури про на цю тему зосереджено більшість існуючих документів метрики та рамки оцінки масштабованості для паралельних і розподілених систем. Оскільки ці системи налаштовані з попередньо налаштованими системними ресурсами та інфраструктурою, тестування продуктивності та оцінка масштабованості зазвичай проводиться в статичній і попередньо фіксованій системі середовища (наприклад, тестова лабораторія), тому існуючі метрики оцінки, рамки та рішення не враховував особливості хмари тестування, наприклад, динамічна масштабованість, масштабована середовища тестування, вимоги на основі SLA, і моделі витрат.

Тестування безпеки та вимірювання в хмарах стало гарячим дослідженням тема з багатьма відкритими питаннями в поточному спільнота тестувальників програмного забезпечення. З безпеки стає головною проблемою в хмарах і служби безпеки стають необхідною частиною сучасні SaaS та хмарні технології, інженери повинні вирішувати проблеми та виклики безпеки перевірка та гарантія якості для SaaS і хмари. Ось деякі пов’язані з цим проблеми та виклики:

забезпечити безпеку хмарних процесів додатків і бізнесу дані в сторонній хмарній інфраструктурі;

стандарти QoS для орієнтованого на безпеку забезпечення якості для наскрізного процес подання заявки та відповідні бізнес-дані в/над/над хмарами;

моделі тестів, адекватність тесту, тест техніки та інструменти для тестування безпеки для кінцеві програми в/на/над хмарами;

забезпечити та оцінити конфіденційність користувачів у хмарній інфраструктурі;

Тестування інтеграції в хмарах обговорюється в численних

опублікованих дослідницьких роботах. Але вирішення питань тестування інтеграції програмного забезпечення і стратегії, не так багато результатів досліджень застосовані в реальній інженерній практиці.

Однією з головних причин є наявність програмне забезпечення та компоненти розроблені без передові технології та рішення для підтримки та полегшити систематичну інтеграцію програмного забезпечення. В хмарній інфраструктурі, з якою повинні мати справу інженери інтеграція різних SaaS і програм в/над хмарами у вигляді чорної скриньки на основі їх надані API та протоколи підключення. Це може спричинити багато додаткових витрат на інтеграцію та труднощі через такі проблеми.

Відсутність чітко визначеної перевірки методів та стандартів забезпечення якості для вирішення протоколів підключення, інтерфейси взаємодії та API служби надається SaaS і хмарними API.

Існує брак економічно ефективної інтеграції рішення та структура для полегшення інтеграція програмного забезпечення та збірка всередині хмар і над хмарами.

Проблеми та проблеми з тестуванням на вимогу TaaS послуги тестування програмного забезпечення повинні бути контролюється та керується на вимогу запити на тестування. Це свого роду новий сервіс тестування модель поставила кілька проблем і проблем.

Підтримка перевірки програмного забезпечення на вимогу в хмарі повинні вирішити проблеми регресійного тестування і проблеми, спричинені змінами програмного забезпечення та виправлення помилок. Однак більшість існуючих досліджень в програмному регресійному тестуванню приділяється найбільша увага для повторного тестування певної версії програмного забезпечення в попередньо налаштованому тестовому середовищі. Багаваріативна особливість хмар може викликати труднощі з застосувати поточну дослідницьку роботу в хмарному тестуванні, спеціально для програмної регресії на вимогу послуга тестування при зміні програмного забезпечення.

Крім того, також бракує динамічного програмного забезпечення методи перевірки та рішення для вирішення динамічні функції SaaS і хмари, наприклад функції автоматичного надання/деактивації.

Існує ряд основних потреб у хмарному тестуванні. Вони обговорюються нижче.

Потреба №1: Адекватні тестові моделі та критерії Для ефективної підтримки хмарного тестування інженерам потрібні нові адекватні тестові моделі та критерії в наступних областях.

Моделі масштабованості для SaaS/Cloud програми – інженерам потрібні чітко визначені адаптивні тестові моделі та метрики оцінки для тестування масштабованості та продуктивності в хмари для перевірки та вимірювання динамічної системи масштабованість (наприклад, збільшення та зменшення масштабу) та продуктивність системи (наприклад, покращення та деградація). Ці моделі та показники повинні адреса масштабованих обчислювальних ресурсів, динамічна навантаження системи та попередньо визначені економічні продажі (моделі вартість/ціна).

Адекватні моделі та критерії інтеграції: до вирішення проблем інтеграції програмного забезпечення в хмару тестування, інженерам потрібні відповідні тестові моделі та критерії, що стосуються трьох типів інтеграції в хмарне тестування:

- хмарні протоколи підключення та API перетинають хмари;
- SaaS (або додатки) API та взаємодії до застарілих систем поза хмарами;
- наскрізна інтеграція додатків хмари.

Потреба №2: процеси тестування TaaS і стандарти QoS Сучасні комерційні SaaS і хмари пропонують інше типи угод про рівень обслуговування (SLA) для користувачів.

Однак як постачальники, так і користувачі потребують чіткого визначення стандарти тестування та забезпечення якості як теорія базу для встановлення чесних і надійних угод про рівень обслуговування між ними.

Ось кілька конкретних прикладів.

Безпека системи і даних програми має бути зазначено як частина вимог до SaaS і хмари, сучасні хмарні технології та основні гравці перераховують лише вимоги безпеки для хмарна інфраструктура. Однією з основних причин є відсутність стандартів тестування та QoS, які стосуються хмари і SaaS безпеки в різних аспектах, включаючи, безпека даних програми, наскрізна транзакція безпека, безпека бізнес-процесів і користувач конфіденційність.

Оскільки TaaS – це нова концепція та модель обслуговування для тестування програмного забезпечення бракує чітко визначених процесів TaaS і стандартів QoS послуги тестування на вимогу, наприклад, ціна моделі.

Потреба №3: Інноваційні методи тестування та рішення. Нові функції та нові вимоги до хмарних технологій програми та SaaS пред'являють деякі вимоги до нового тесту методи та рішення.

Безперервна перевірка та регресійне тестування рішення – Оскільки висока доступність системи дуже важливі для SaaS і хмарних програм, інженерам потрібні автоматичні методи повторного тестування для вирішення проблеми мультиорендування хмар кожного разу, коли програмне забезпечення змінюється для виправлення помилок або покращення функції.

Нові рішення для автоматичного тестування для хмари сумісність – оскільки і хмари, і SaaS надає свої протоколи підключення та API, це вимагає від інженерів забезпечення якості сумісність хмарних програм перетинаючи різні хмари в підключенні протоколи, брандмауери, взаємодія між SaaS і застарілі системи.

Інноваційні тестові технології для хмари тестування на сумісність – хмарні програми і SaaS повинні підтримувати глобальних онлайн-користувачів отримати доступ до наданих послуг за допомогою різноманітних клієнтські платформи, браузери та технології, це говорить про те, що постачальникам потрібні нові ефективні тестові технології для підтримки валідації сумісність додатків на різних платформах, клієнтські технології та браузери.

Хмарне тестування вимагає великомасштабних тестових навантажень у реальному часі в а масштабоване розподілене середовище на основі Інтернету. Це наводить на думку потреба в кількох тестових інструментах і рішеннях. Один із них потужний симулятор тестування, який забезпечує широкомасштабне веб-моделювання тестування та генерацію даних у хмарі за допомогою віртуальні та фізичні обчислювальні ресурси. Щоб ефективно використовувати існуючі рішення продуктивності, зрозуміло, що потрібні певні гнучкі інтерфейси та шлюзи симулятор для підключення до існуючого тесту продуктивності інструменти. Наступне — це інноваційне рішення для тестування інтеграції який підтримує легку та бездоганну інтеграцію між SaaS і програми, що перетинають хмари. Це означає, що на основі певного стандарту API та з підтримкою підключення для підтримки інтеграції необхідні тестові рамки над хмарами. По-третє, це хмарна продуктивність і рішення для масштабованості, які підтримують перевірку інженерів нефункціональні вимоги для SaaS і хмарних технологій програми на різних рівнях, щоб забезпечити їх продуктивність і масштабованість на основі заданих SLA та певних економічні масштаби (або моделі витрат). Крім того, ан потрібне інноваційне наскрізне рішення для відстеження програми для підтримки тестування програмного забезпечення, виправлення помилок і обслуговування хмарних програм на різних рівнях. Це рішення дозволяє інженерам розуміти, тестувати та контролювати наскрізні процеси додатків, транзакції, сервісні функції, і взаємодії між SaaS і хмарами.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ ХМАРНОГО ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка алгоритму хмарного тестування програмного забезпечення

Алгоритм виявлення, вибору та ранжирування веб-сервісів (WSDSR) спочатку отримує вхідні дані від користувача, а на основі введених даних отримує служби зі сховища. Після цього набір сервісів витягується зі сховища. Алгоритм спочатку шукає збіги на основі тестів, якому надає перевагу користувач. На другому кроці алгоритм виконує семантичне зіставлення з використанням WordNet. І тестувальна хмара, і семантичні результати об’єднуються разом, усуваючи однакові служби. Алгоритм тестування на базі веб служб представлено в лістингу 3.1.

Лістинг 3.1 – Алгоритм хмарного тестування програмного забезпечення

```
Step 1: Input: Name of the web service (S)
       User Requirements (UR)
       Output: Retrieved Collection of Service with Quality of
       Services (QoS) with ranking (WS_QoS)
Step 2: Select Web Service (WS) from the repository (R) with
       keyword Search || Semantic Search
       Keyword_ Match= (Select Services w.r.t. to Keyword match)
          For (i=0; i< Keyword_ Match.length;i++)
               Keyword Search []={K_WSi, K_WS…..K_WSn }w.r.t
               keyword search
       End
       //Retrieving the matched services using Semantic Search
       using WordNet
       //Semantic Match SM:=(Select S w.r.t to semantic meaning
       using wordnet)
          For (i=0; i< Semantic Match;i++)
               Keyword Search []={K_WSi, K_WS…..K_WSn }w.r.t
               keyword search
          End
       Keyword_Search []={K_WSi, K_WS…..K_WSn }w.r.t keyword
       search
       Semantic_Search[] ={ S_WSi, S_WS…..S_WSn }w.r.t semantic
```

```
       search
Step 3: Combine both the search results
       Final_Collections [WSi to WSn][WQoSi to WQoSn]= {
       Keyword_Search }
       U { Semantic_Search }
     End
Step 4: Normalizing the QosParmeters Values from the
     Final_Collections
     QoS Values Ranges between 0 to 1
     Services [services][QoS]=Normalized values w.r.t to
     services
Step 5: Summing up the QoS Normalized Values for each services
     For (i=0;i<Services.length;i++)
          QoS[i] =.Services i[Qos1 + Qos2 +…. Qosn]
     End
Step 6: Ranking the Services
     For (i=0;i<QoS.length;i++)
          Sort the values in ascending order from
          WS_QoS= QoS [i] toQoS[n]
          Display the sorted list.
     End Return (WS_QoS)
```

За допомогою алгоритму отримуються остаточні тестові сервіси. Нормалізація - це процес масштабування значень від 0 до 1. Отримані значення параметрів якості обслуговування спочатку не нормалізуються, ми повинні нормалізувати значення, щоб виконати обчислення щодо якості послуг. Тоді результати будуть піддані процесу нормалізації. Рівняння (2.2) і (2.3) використовуються для розрахунку нормалізації. Атрибут QoS нормалізується відповідно до позитивних і негативних атрибутів. Параметри QoS, такі як доступність, пропускна здатність, вважаються позитивними параметрами, а такі атрибути, як вартість і час відгуку, вважаються негативними атрибутами. Споживач вимагає більше значень позитивних атрибутів і менше значень негативних атрибутів. Після процесу нормалізації значення QoS будуть нормалізовані в тому ж масштабі. Ми повинні додати всі значення QoS для кожної служби, відповідно до значень tit сортується в порядку зростання, і буде отримано рейтинг послуг. Список ранжованих послуг буде надано користувачеві.

3.2 Опис фреймворку для досліджень хмарного тестування

На рисунку 3.1 представлено структуру, яку ми розробили, щоб охарактеризувати дослідження хмарного тестування. Прагнучі до повноти, структура розгорталася в хмарі поступово. Спочатку було отримано проект схеми на основі нашої множені тестів та сервісів; ця схема включала шість напрямів і кілька тестів для кожного напряму. Потім ми використали цей проект схеми для класифікації сервісів під час тестування, але також продовжували додавати в кожну область нові підтеми за потреби.

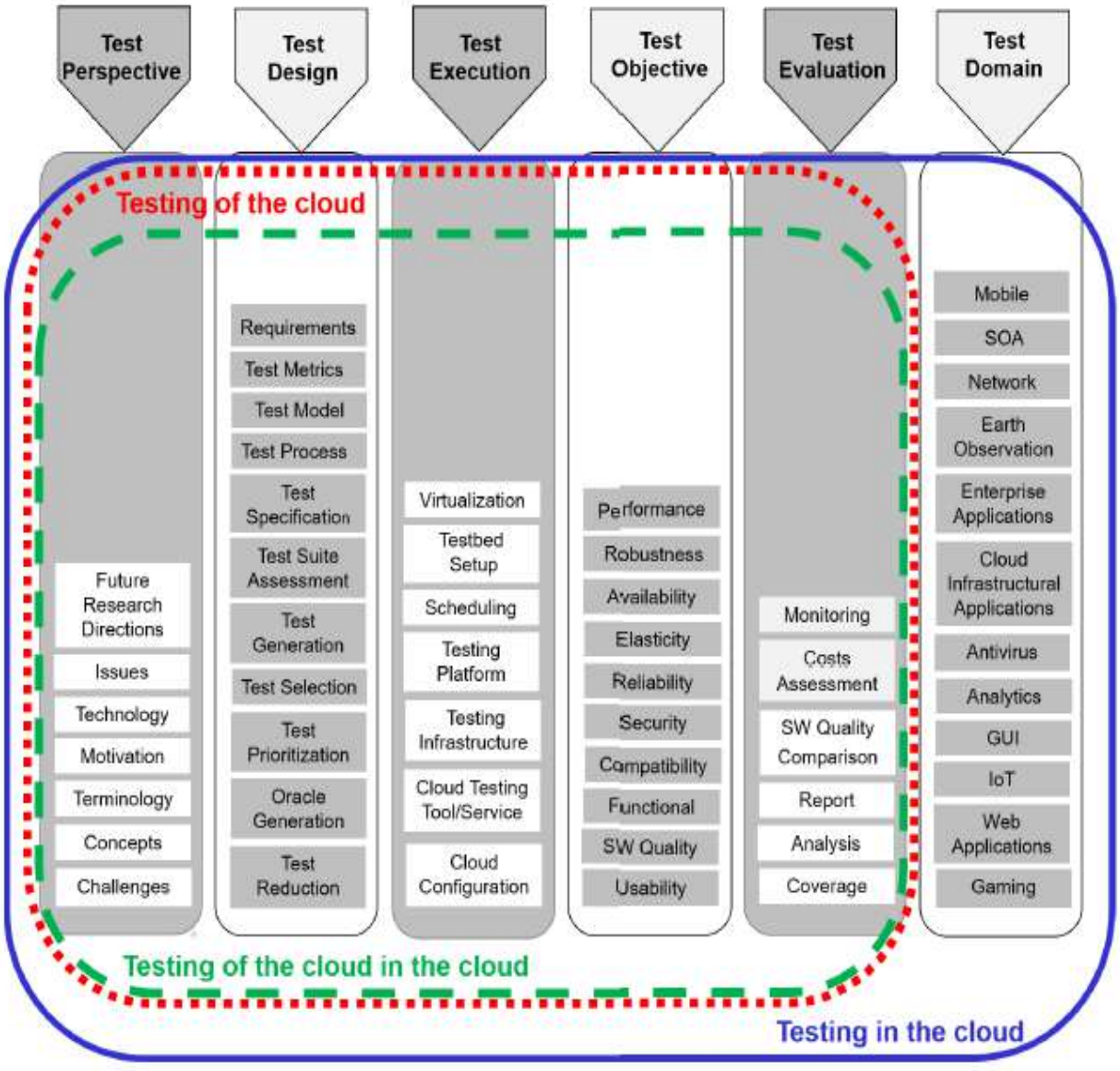

Рисунок 3.1 – Фрейворк хмарного тестування

Під час аналізу даних ми стандартизували/уніфікували отримані параметри на базі нормуючих виразів (2.2) та (2.3). Таким чином, отримана структура сама по собі є корисним внеском для отримання моментального знімка тенденцій у дослідженні хмарного тестування.

Основними особливостями, які треба враховувати при проведенні тестування наступні.

Перевірка перспективи. Тести, що належать до цієї галузі, представляють нові погляди на дослідження хмарного тестування. Вони розглядають такі теми, як основні концепції, термінологія, виклики та майбутні напрямки дослідження хмарного тестування.

Дизайн тесту. Сервіси, що належать до цієї області, містять рішення, спрямовані на етап проектування тестової діяльності. Зокрема, вони представляють аналіз вимог до тестування, визначення тестової моделі або тестової метрики, а також різні стратегії тестування для створення тестових випадків або вибір, скорочення тестових випадків або оцінка набору тестів. Крім того, ця область також містить аналіз вхідних даних для сервісів, що визначають процес хмарного тестування.

Виконання тесту. Тести представляють артефакти, задіяні на етапі виконання тестової діяльності. Зокрема, вони представляють платформи, інфраструктури, інструменти чи служби для хмарного тестування, а також візуалізацію чи конфігурацію хмарних підходів або рішень для налаштування тестового стенда.

Мета тесту. Тести стосуються різних цілей хмарного тестування, таких як перевірка відповідності систем функціональним специфікаціям або показу визначених нефункціональних властивостей, таких як продуктивність, надійність, надійність, зручність використання тощо.

Оцінка тесту. Тести та сервіси стосуються оцінювання діяльності та результатів тестування, надання підтримки для звітів про тестування, аналізу витрат на тестування або оцінки якості різних рішень для тестування.

• Тестовий домен. Фреймворк тестування представляє хмарні рішення

для тестування для потреб, що виникають у певних областях застосування, таких як мобільні або веб-додатки.

Загалом, як і очікувалося, ми організували тести за трьома різними категоріями, які охоплюють зазначені вище області, як показано кольоровими рамками на рисунку 3.1:

- тестування в хмарі (синя безперервна рамка на рисунку 3.1) відноситься до тестування програмного забезпечення, яке виконується шляхом використання масштабованих хмарних технологій, рішень і обчислювальних ресурсів для перевірки програмного забезпечення. Ця категорія включає тестові рішення для різних доменів застосувань, таких як мобільне або веб-середовище, які перевіряються за допомогою широкомасштабного використання симуляції та еластичні ресурси, які пропонує хмара. Основні переваги цих рішень полягають у зниженні витрат за рахунок використання, очевидно, необмежених обчислювальних ресурсів у хмарі; уникнення розробки та підтримки тестової інфраструктури (шаффолдингу); тестові послуги на вимогу, що надаються третьою стороною для проведення онлайн-підтвердження для великомасштабних програмних систем;

- тестування хмари (червона пунктирна рамка на рисунку 3.11) відноситься до перевірки якості (функціональних і нефункціональних властивостей) програм та інфраструктур, які розгортаються в хмарі. Основна увага приділяється конкретним проблемам тестування, які виникають у системах, що знаходяться в хмарі, тому тести, що належать до цієї категорії, спрямовані на перевірку наданих автоматичних хмарних функціональних служб, а також на перевірку їх продуктивності, масштабованості, еластичності та безпеки. Крім того, програмні додатки можна розгортати на різних хмарах (напрприватні, загальнодоступні чи гібридні), отже, тестування також може зосереджуватися на сумісності та взаємодії між різнорідними хмарними ресурсами;

- тестування хмари в хмарі (зелена пунктирна рамка на рисунку 3.1)

стосується додатків та інфраструктур, розгорнутих у хмарі та перевірених за допомогою хмарних платформ. Тести, що належать до цієї категорії, заповнюють область перетину між тестуванням хмари і тестуванням в хмарі.

На відміну від тестування звичайного веб-програмного забезпечення, тестування хмари та хмарне програмне забезпечення має кілька унікальних тестів цілі забезпечення якості, вимоги та відмінні особливості. На рисунку 3.2 показані основні функції хмарного тестування.

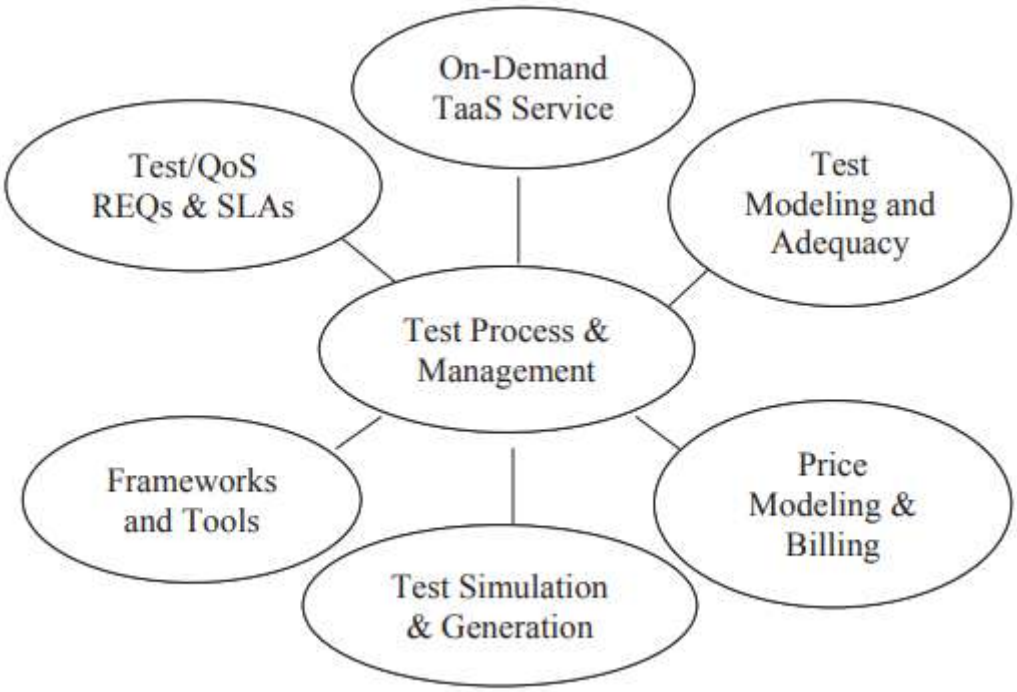

Рисунок 3.2 – Функції хмарного тестування програмного забезпечення

Хмарне середовище тестування використовує обрану хмарну інфраструктуру (або платформу) як основу для формування випробувального стенду, оснащеного різноманітними і масштабованими обчислювальними ресурсами, системною інфраструктурою та ліцензованими інструментами, які виділяються за допомогою автоматичного надання на основі статичних/динамічних запитів. І віртуальні, і фізичні обчислювальні ресурси можуть бути включені і розгорнуті всередині хмари.

Угоди про рівень обслуговування (SLA) у хмарі обчислення. Усі хмари

SaaS та програми зазвичай надають їм різноманітні послуги кінцевих користувачів і клієнтів із чітко визначеними угодами про рівень обслуговування. Природно, ці угоди стають частиною тестування та вимог до забезпечення якості, такі як надійність, доступність, безпека та продуктивність угоди.

Цінові моделі та виставлення рахунків за послуги – починаючи з послуг з проведення обчисленнь, є одним із основних понять і функцій у хмарних обчисленнях, тому моделі цін і корисності виставлення рахунків стає основними частинами та послугами для тестування як послуга. Іншими словами, необхідні обчислювальні ресурси та інфраструктури (включаючи інструменти) і послуги з тестування стягуватимуться на основі заздалегідь визначені моделі витрат.

Великомасштабні хмарні дані та трафік симуляція. Застосування та імітація широкомасштабного доступу користувачів до Інтернету та даних трафіку (або повідомлення) в інтерфейсах з'єднання є необхідним у хмарному тестуванні, зокрема на системному рівні перевірки функцій і тестування продуктивності.

У хмарному тестуванні є кілька відмінних особливостей. Одним із них є тестування як послуга (TaaS). Це ан інноваційна концепція, і вона стосується надання послуг статичного/динамічного тестування на вимогу в/на/над хмарами для третіх осіб у будь-який час і в будь-який час (365/7/24).

Однією з першочергових цілей є скорочення IT-бюджету підприємств зосередити свою основну діяльність на аутсорсингу завдання тестування програмного забезпечення третій стороні за допомогою служби TaaS моделі. Відповідно до Вікіпедії, TaaS передбачає виконання тестів на вимогу чітко визначених наборів тестів матеріалів, як правило, на сторонній основі. Виконання можна виконувати на сайті клієнта або віддалено з випробувальні лабораторії/засоби зовнішніх постачальників.

Відповідно до URL://http://www.tieto.com/, тестування як а сервіс (TaaS) спочатку був представлений як концепція Tieto в Данії в 2009 році, і його

рішення TaaS було номінований IBM на премію Software Innovation Award 2009. Зараз TaaS отримав широку увагу завдяки його перевага в масштабованому середовищі тестування, вартість скорочення, моделі обслуговування на основі корисності та на вимогу послуги тестування.

Як показано на рисунку 3.3, робочий процес TaaS включає наступні основні можливості служби TaaS.

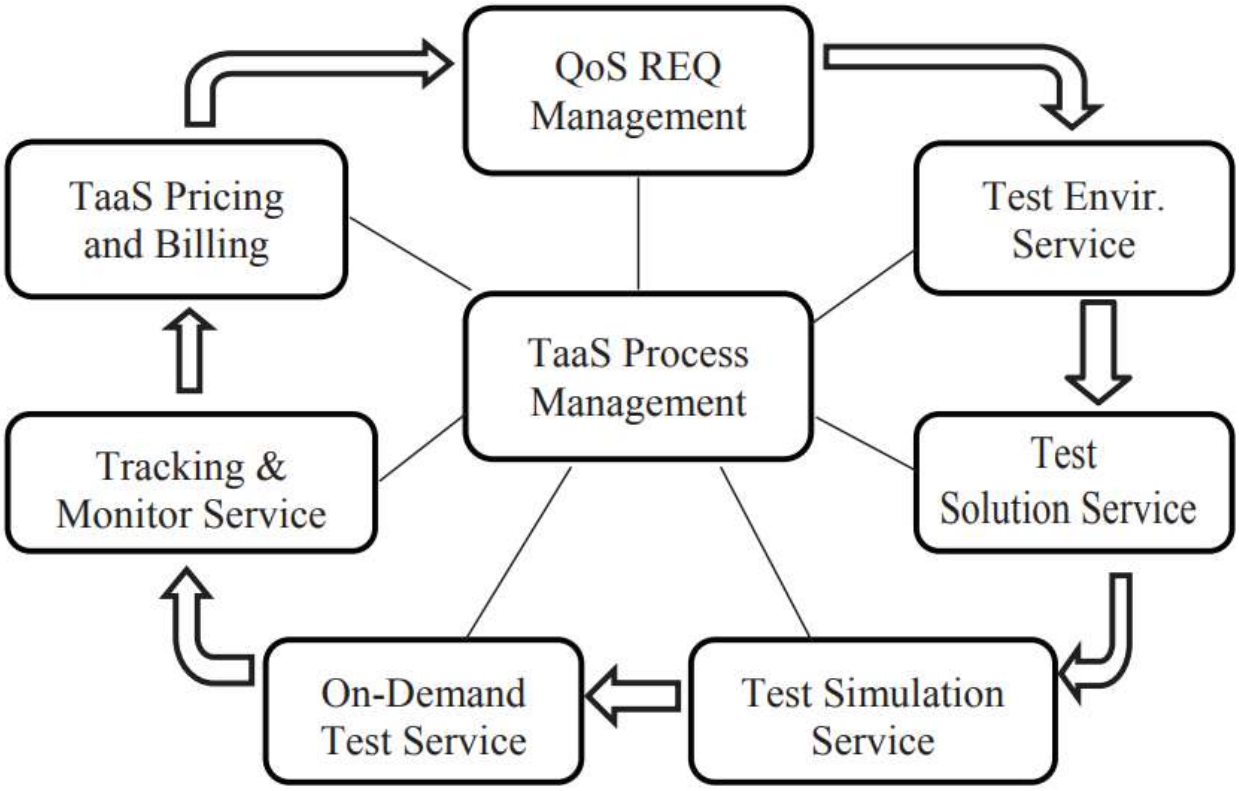

Рисунок 3.3 – Послідовність виконання етапів хмарного тестування

Управління процесом TaaS –пропонує тест управління проектами та контроль процесів.

Керування вимогами QoS, що підтримує ведення бухгалтерського обліку та моделювання тестування ПЗ і вимоги до якості обслуговування, включаючи якість моделювання впевненості.

Служба тестового середовища, яка надає послуги тестового середовища на вимогу для встановлення необхідна віртуальна (або фізична) хмара обчислювальні ресурси та інфраструктури, а також як необхідні

інструменти.

Служба тестування рішень, яка пропонує різноманітні рішення для систематичного тестування (наприклад, тест моделювання та методи тестування), а також тестове обладнання послуги з генерації та управління.

Служба тестового моделювання, яка встановлює середовища тестового моделювання на вимогу вибрані засоби (наприклад, інструменти) і підтримка необхідні тестові дані/генерація повідомлень.

Сервіс тестування на вимогу, який надає послуги виконання тесту на вимогу на основі вибраного графіки та тестові вироби.

Служба відстеження та моніторингу, що дозволяє тестувати інженерів для відстеження та моніторингу різноманітних програм поведінки на різних рівнях у/на/над хмарами для мета тестування.

Ціноутворення та виставлення рахунків TaaS, що дозволяє TaaS постачальники, щоб запропонувати клієнтам вибіркове тестування контракти на обслуговування на основі заздалегідь визначених цін моделі та послуга виставлення рахунків.

Хмарне тестування стає гарячою темою досліджень в програмній інженерної. Як розвиток хмарних технологій і тестування як послуг тощо необхідно провести дослідницьку роботу для вирішення відкритих питань і проблеми в хмарному тестуванні та TaaS. Більш інноваційний методи тестування та рішення, а також стандарти QoS необхідні для підтримки служб тестування на вимогу в масштабованій хмарній інфраструктурі.

3.3 Результати експериментів

Ми реалізували нашу запропоновану роботу в Net Beans IDE 17 з використанням мови Java як інтерфейсом і Oracle 10 як сервером. Наша система містить 1000 веб-сервісів, 3 основних наборів тестів і використовуючи їх, ми реалізували наш запропонований підхід і отримали результати, як показано нижче в таблицях 3.1 та 3.2.

Таблиця 3.1 – Результати хмарного тестування сервісів

Назва сервісу	Місцезнаходження сервісу (url)	Доступність (%)	Час тестування (ms)
Sms	http://www.2sms.com/soap/2smsmessaging.wsdl	89	3030
AWSECommerceService	http://www.smartsms.se/IPX/services/FileShop/fshandler.asmx?wsdl	80	4822
ExchangeRateService	http://www.smsdome.com/portalvbvs/services/smsgateway.asmx?wsdl	84	1265
ExchangeRateService	http://www.jaredmonaco.com/ATTSMS.asmx?WSDL	91	4562
FpML Validation	http://www.directsdi.com.au/info/api/SmsGateway-HTTP.wsdl	42	1354
IXML WS service	http://www.info-mesms.it/ws.php?wsdl	83	2150
Location	http://www.info-mesms.it/sms.wsdl	98	3562
MatchService	http://ws.acrosscommunications.com/SMS.asmx?WSDL	90	1953
prophecyService	http://sms.cellcom.co.il/SmsGate/SmsGate2.asmx?WSDL	89	1945
wordcountService	http://www.scottnichol.com/samples/hellowsdl2.php?wsdl	84	5120

Таблиця 3.2 – Результати оцінки якості QoS хмарного тестування сервісів

Доступність сервісу			Час тестування (ms)		
Test 1	Test 2	Test 3	Test 1	Test 2	Test 3
1	2	3	4	5	6
62	45	65	4565	6526	8741
23	39	52	3212	1565	2369

Продовження таблиці 3.2

1	2	3	4	5	6
86	63	45	8456	4532	8541
36	45	68	9535	9575	2569
95	25	15	6545	5698	8745
76	16	68	8575	7551	6547
94	89	63	2545	6224	1458
56	63	21	4512	7486	6345
86	84	25	3575	3214	2541
28	33	58	9654	4789	1268

3.4 Аналіз результатів експериментів

Оцінки результатів чітко показують, що виявлення послуг тестування на основі QoS покращено, а також підтримується надійність послуг у запропонованій системі. Відповідні результати наведені на рисунку 3.4.

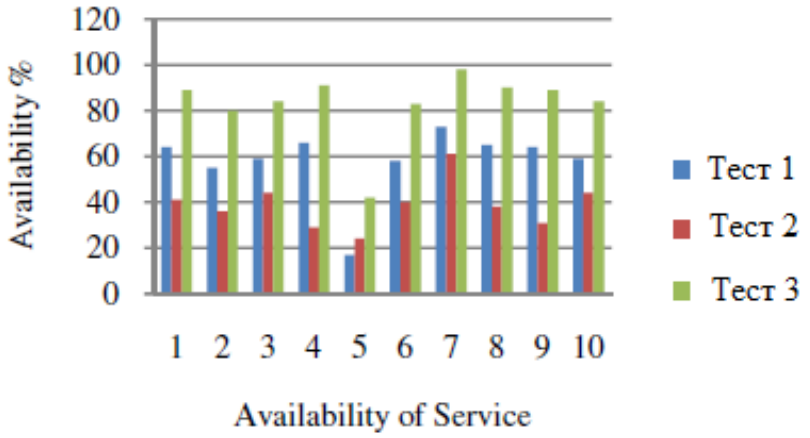

Рисунок 3.4 – Результати хмарного тестування сервісів

Рисунок 3.4 побудовано на даних з таблиці 3.1 і визначає доступність послуги та її варіабельність при проведенні трьох пакетів хмарних тестів.

Рисунок 3.5 описує час відгуку та ілюструє відповідні пом’якшувальні заходи служби.

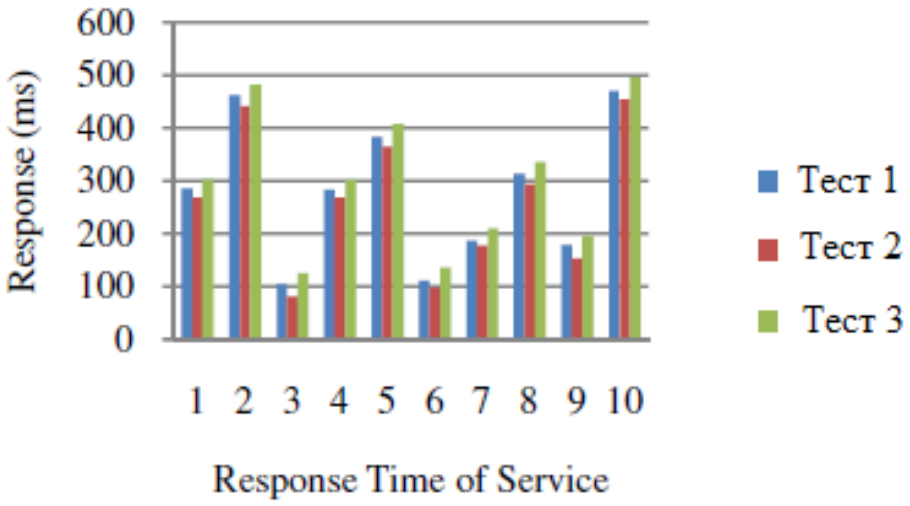

Рисунок 3.5 – Оцінка часу відповіді на основі QoS

Крім того, на рисунке 3.6 пропускна здатність різних постачальників послуг і частота відмов також різко зменшуються в запропонованій системі.

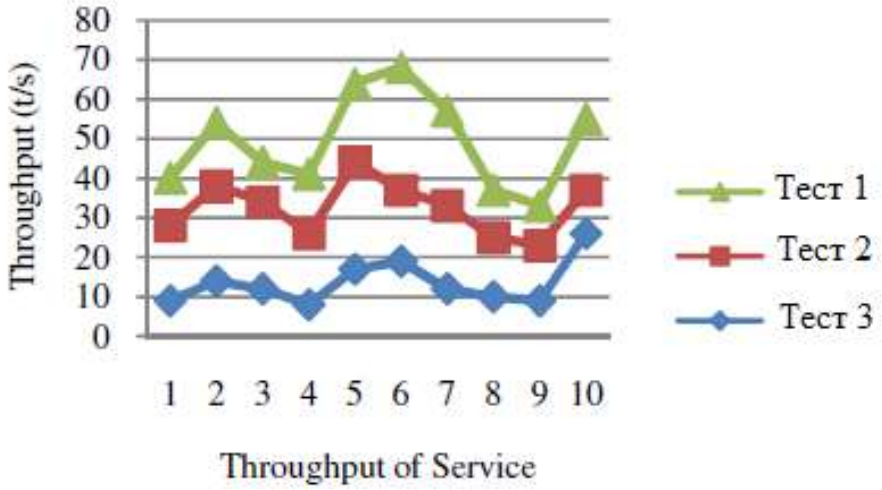

Рисунок 3.6 – Перевірка пропускної здатності на основі QoS різних послуг

Подібним чином на рисунку 3.7 його відповідна оцінка вартості пошуку та обслуговування такого виду тестування.

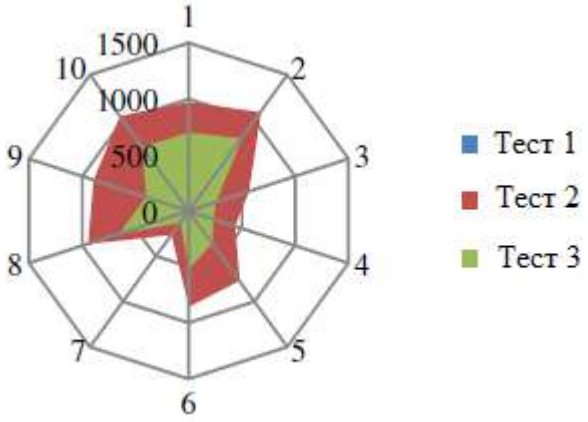

Рисунок 3.7 – Оціночна вартість послуги хмарного тестування

Рисунок 3.8 демонструє результати якості хмарного тестування QoS по сервісам, які було протестовано.

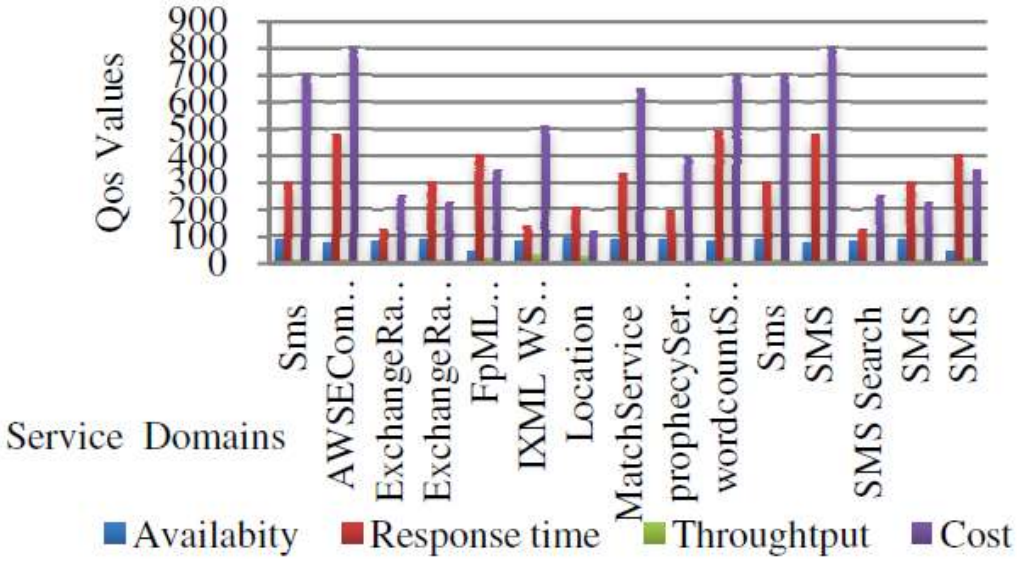

Рисунок 3.8 – Оцінка якості тестування хмарних сервісів

На основі результатів тестування є можливість обчислити загальний рейтинг сервісів, враховуя ріність нормованих (за виразами (2.2) та (2.3)) параметрів. Результати обчислення рейтингу наведено на рисунку 3.9.

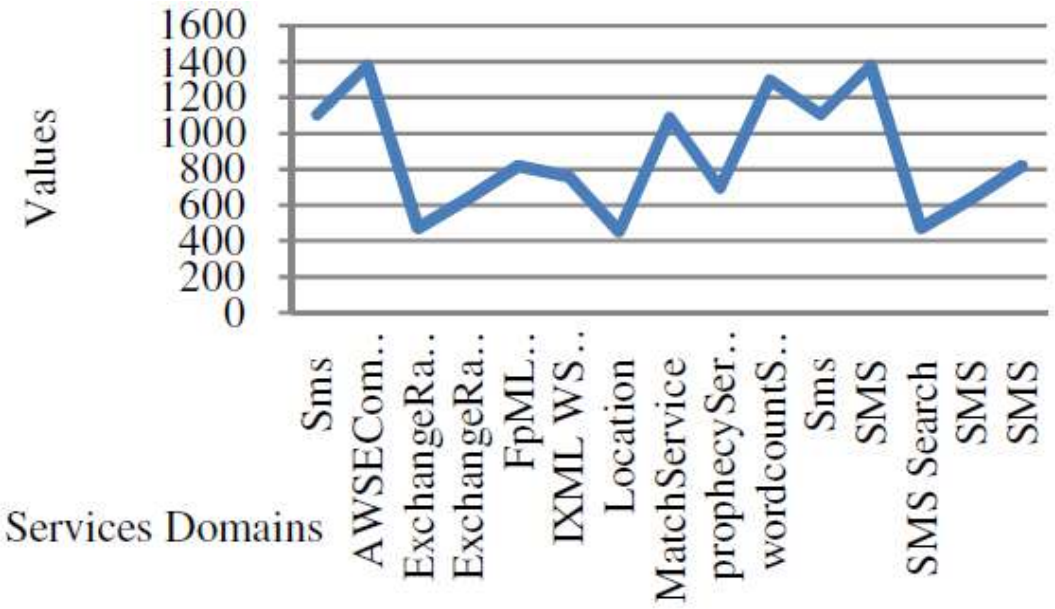

Рисунок 3.9 – Узагальнений рейтинг хмарних сервісів на основі тестування

ВИСНОВКИ

У роботі була розроблена модель середовища хмарних обчислень для тестування програмного забезпечення з різними типами хмар. Ця робота є одним із небагатьох прикладів хмарних фреймворків, створених для вирішення проблем у галузі системного програмування, які явно інтегруються як сервіс у розподіленого мережного середовища та перевіряються кількома складками. У роботі також проведена оцінка діапазону використання щодо середовища застосування та оцінка управління обчислювальним середовищем. З точки зору розробки програмного забезпечення, це продемонструвало ефективність концепції хмарного тестування.

В ході виконання кваліфікаційної роботи проаналізовані проблеми виявлення та ранжирування тестів на основі веб-сервісів, запропоновано та реалізовано новий алгоритм хмарного тестування. Запропонований підхід із наборами хмарних тестів є ефективним. Результати впровадження показують, що цей підхід є кращою реалізацією виявлення та ранжирування веб-служб на основі хмарного тестування. Результати показують, що запропонований підхід дає кращі результати. У майбутньому можливе розширення алгоритму за допомогою більшої кількості параметрів QoS.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. David Cheppal et. al, “A Short Introduction Cloud Platforms – An Enterprise Oriented View”, Technical Report, August 2018.

2. Khurshid Ahmed, Al-Nayeem Abdullah and Gupta Indranil, “Performance Evaluation of the Illinois Cloud Computing Testbed”, Technical Report, June 2019. http://hdl.handle.net/2142/12983.

3. Judith M. Myerson, “Cloud computing versus grid computing”, White Paper IBM, March 2019.

4. Rajkumar Buyya, Chee S. Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic, “Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility”, Future Generation Computer Systems, Vol. 25, No. 6, pp. 599-616, June 2019.

5. Bethea, W.; Cole, R.; Harshavardhana, P., “Automated discovery of information services in heterogeneous distributed networks”, Military Communications Conference, MILCOM 2018, Page(s): 1 – 6, 2018.

6. Chou, Wu; Li, Li; Liu, Feng, “Web Services for Service-Oriented Communication”, International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2016, Page(s): 1-8

7. Feng Liu; Gesan Wang; Li Li; Wu Chou, “Web Service for Distributed Communication Systems”, IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI '16, Page(s): 1030 – 1035, 2016.

8. Dhavachelvan P and Uma. G.V, “Multi-agent Framework Construction for Intra Class Testing of Object-Oriented Software”, Lecture Notes in Computer Science (LNCS), Vol.2869, pp. 992-999, Springer Verlag, September-2013.

9. Dhavachelvan P and Uma. G.V, “Reliability Enhancement in Software Testing: An Agent-Based Approach for Complex Systems ”, Lecture Notes in Computer Science (LNCS), Vol. 3356, pp. 282-291, Springer Verlag, December 2014.

10. Azadeh Ghari Neiat, Mehran Mohsenzadeh, Rana Forsati, Amir Masoud Rahmani. An Agent-based Semantic Web Service Discovery Framework. International Conference on Computer Modeling and Simulation, p.194-198, DOI: 978-0-7695-3562-3/09.

11. P. Victer Paul, D. Rajaguru, N. Saravanan, R. Baskaran and P. Dhavachelvan. Efficient service cache management in mobile P2P networks. Future Generation Computer Systems, Elsevier, 2021; 29 (6), p.1505–1521.

12. Ourania Hatzi, Georgios Batistatos, Mara Nikolaidou, Dimosthenis Anagnostopoulos. A Specialized Search Engine for Web Service Discovery. IEEE 19th International Conference on Web Services (ICWS). DOI: 10.1109/ICWS.2007.

13. GaoShu, Omer F. Rana, Nick J. Avis, Chen Dingfang. Ontology-based semantic matchmaking approach. Advances in Engineering Software. 2017; vol. 38, p. 59-67.

14. FatihEmekci, Ozgur D. Sahin, DivyakantAgrawal ,Amr El Abbadi.A Peer-to-Peer Framework for Web Service Discovery with Ranking. Proceedings of the IEEE International Conference on Web Services (ICWS'04.) DOI: 10.1109/ICWS.2014.1314739, 2014. pp.192-199.

15. KaiZheng, Hailing Xiong.Semantic Web Service Discovery method Based on User Preference and QoS. 12nd International Conference on Consumer Electronics, Communications and Networks (CECNet), 2022. pp.3502-3506.

16. Hong Qing Yu; Dietze, S.; Benn, N., Autonomous Matchmaking Web Services. International Conference on Computer Information Systems and Industrial Management Applications (CISIM), 2018; 8(18), pp.420-425. doi: 10.1109/CISIM.2018.5643504.(4-7).

17. Colin Atkinson, Philipp Bostan, Oliver Hummel and Dietmar Stoll. A Practical Approach to Web Service Discovery and Retrieval. IEEE International Conference on Web Services (ICWS 2017), p.24-248.

18. Agarwal, S Studer, R. Automatic Matchmaking of Web Services, International Conference on Web Services-ICWS '19, vol. no. pp.45-54.

19. Kritikos, K.; Plexousakis, D. Evaluation of QoS-Based Web Service Matchmaking Algorithms. IEEE Congress on Services; 6(18), p.567,574. doi: 10.1109/SERVICES-1.2018.

20. Murugaiyan.S.R, Sambasivam.G, Vengattaraman.T, Dhavachelvan.P, Venkatachalapathy.V.S.K. An improved matchmaking technique for Web service discovery. International Journal of Applied Engineering Research, 2014; 9(13), p.2359-2368.

21. Chiranjeevi Manike, P.P.S Naik, Kavadi Durga Prasad. An Efficient Framework for High-Quality Web Service Discovery.CSI Sixth International Conference on Software Engineering (CONSEG) 2022; p.1-7. DOI: 10.1109/CONSEG.2022.6349480.

22. Mohan.R, Murugaiyan.S.R, Sambasivam.G, Vengattaraman.T, Dhavachelvan.P. User preferential model for semantic web service discovery based on concept lattices. International Journal of Engineering and Technology, 2014. 5(4), p.3470-3483.

23. ZhenqiWang,Yuanyuan. An Approach for Semantic Web Service Discovery Based on P2P Network. 4th International Conference on Wireless Communications, Networking and Mobile Computing- WiCOM '08. p. 1-4, DOI: 10.1109/WiCom.2008.1230.

24. Balaji.N, Sambasivam.G, Basha.M.S.S, Vengattaraman.T, Dhavachelvan.P. SLA based architecture for web service selection and ranking with QoS. International Conference on Human Computer Interactions, ICHCI 2013, DOI: 10.1109/ICHCIIEEE.2022.6887783.

25. Agarwal, S Studer, R. Automatic Matchmaking of Web Services, International Conference on Web Services-ICWS '06, vol. no. pp.45- 54, doi: 10.1109/ICWS.2006.35.