ДИПЛОМНА РОБОТА
БАКАЛАВРА

ДРБ. ІП - 47.00.00.000 ПЗ

Група ІП-18-1

Чичул Наталія

2022

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

УДК 004.4

ДИПЛОМНА РОБОТА БАКАЛАВРА

Тема: «Композиційні моделі налаштування програмного забезпечення та сервісів»

(назва згідно з наказом ректора)

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДРБ.ІП - 47.00.00.000 ПЗ

(позначення)

Студентка

ІП-18-1 /Н.В.Чичул/
(шифр групи) (підпис) (дата) (розшифровка підпису)

Керівник роботи

проф.,д.т.н. /Д.Ф.Тимків/
(посада) (підпис) (дата) (розшифровка підпису)

Консультанти:

з питань нормоконтролю

доц., к.т.н /В.Я. Піх/
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту
Завідувач кафедри

проф., д.т.н. /В.І.Шекета/
(посада) (підпис) (дата) (розшифровка підпису)

2022
(рік)

Івано-Франківський національний технічний університет нафти і газу

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедрою **ІПЗ**
проф. **В.І. Шекета**
“ ____ ” _____2022 р.

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ БАКАЛАВРА СТУДЕНТОВІ

Чичул Наталії Вікторівні

(прізвище, ім’я, по-батькові)

1. Тема проекту (роботи) “**Композиційні моделі налаштування програмного забезпечення та сервісів”**

керівник проекту (роботи) Тимків Дмитро Федорович,професор, д.т.н.

затверджені наказом закладу вищої освіти від “ 25 ” травня 2022 р. № ____

2. Строк подання студентом проекту (роботи) 12 червня 2022р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1.Огляд існуючих концепцій, рішень та сервісів в даній області

2.Побудова моделі або алгоритму власного рішення

3.Документування реалізації власного оригінального рішення вибраними засобами

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1.Чотиришарова модель стеку веб-служб (рис.1.2,ст.10)

2.Алгоритм IDDFS (рис.2.1,ст.25)

3.Багатоцільовий еволюційний алгоритм (рис.2.4.,ст.28)

4.Обмеження різних підходів (рис.2.5,ст.31)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання **17 травня 2022**

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обґрунтування теми роботи	01.02.2022	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	01.03.2022	виконано
3	Побудова моделі або алгоритму власного рішення	01.04.2022	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	01.06.2022	виконано
5	Тестування програмного продукту	08.06.2022	виконано
6	Оформлення пояснювальної записки кваліфікаційної роботи	10.06.2022	виконано

Студент _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

В дипломній роботі запропоновано підхід який не тільки використовує вимоги клієнтів для визначення послуг-кандидатів, але й враховує чутливість при виборі найбільш підходящих для конкретної програмної системи, виготовленої на замовлення. Це покращення вибору та складу послуг призводить до розробки розширеної структури життєвого циклу інтеграції послуг, робота якої представлена в цій роботі в новому алгоритмі. На додаток до деталей структури, також включено тематичне дослідження для вивчення процесу ідентифікації кандидатів, аналізу чутливості, вибору послуг та складу системи для налаштованої системи покупок в Інтернеті.

КЛЮЧОВІ СЛОВА: ПРОГРАМНА СИСТЕМА, КОМПОЗИЦІЙНА МОДЕЛЬ, АЛГОРИТМ, ІДЕНТИФІКАЦІЯ, ЖИТТЄВИЙ ЦИКЛ ПРОДУКТУ, ВЕБ-СЕРВІС

ANNOTATION

The thesis proposes an approach that not only uses customer requirements to determine candidate services, but also takes into account sensitivity when choosing the most suitable for a specific custom-made software system. This improvement in service selection and composition leads to the development of an extended service integration lifecycle framework, the work of which is presented in this work in a new algorithm. In addition to the details of the framework, a case study is also included to examine the candidate identification process, sensitivity analysis, service selection, and system composition for a customized online shopping system.

KEY WORDS: SOFTWARE SYSTEM, COMPOSITION MODEL, ALGORITHM, IDENTIFICATION, PRODUCT LIFE CYCLE, WEB SERVICE

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ……7

ВСТУП……..8

1 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ТА СЕРВІСІВ

1.1 Довідкова інформація……..9

1.2 Огляд веб-сервісів……10

1.3 Вибір послуги……...11

1.4 Сервіс-орієнтований метод проектування……11

1.5 Склад послуг……12

1.6 Переваги……...……15

1.7 Лінія продуктів та рішень……...……17

1.8 Оптимізація сервісу……..17

1.9 Виявлення веб-сервісу……18

1.10 Семантичний відповідник……18

1.11 Підходи до аналізу чутливості……..19

1.12Попередні дослідження……..20

2 АЛГОРИТМІЧНІ ТА МАТЕМАТИЧНІ РІШЕННЯ ДЛЯ ЗАДАЧ ДАНОГО КЛАСУ

2.1 Різні підходи до складу служби……..22

2.2 Обмеження підходів……..……..30

2.3 Мета дослідження SOA імплементацій……..31

2.4 Мова визначення веб-служби (WSDL)……...32

2.5 Якість послуг та їх виконання……34

2.6 Структура проектного рішення……...34

2.7 Промислове призначення……35

3 ВЛАСНЕ ОРИГІНАЛЬНЕ АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ РІШЕННЯ НАЛАШТУВАННЯ СЕРВІСІВ

3.1 Основа процедури проектування……...36

3.2 Вимоги до експерименту……...41

3.3 Зразки послуг……...42

3.4 Приклад проектованого сервісу……43

3.5 Аналіз отриманих результатів……44

3.6 Аналіз отриманих сервісів……..48

ВИСНОВКИ……..49

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА……...54

БІБЛІОГРАФІЧНА ДОВІДКА

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

SOA - Сервісно-орієнтована архітектура

БД – База даних

ПЗ - Програмне забезпечення

ПП – програмний продукт

ВСТУП

Сервісно-орієнтована архітектура (SOA) відноситься до процесу, який використовує Інтернет для зв'язування компонентів програмного забезпечення. Це включає вибір послуг, композицію, узгодження, проектування та відображення. Існує багато доступних технологій. Залежно від використовуваної технології, опис інтерфейсу, підтримка протоколів і реєстр послуг забезпечуються відповідним чином.

Система використовує власну техніку для побудови сервісів, досліджуються різні підходи та може бути використаний найкращий можливий метод. Також можна провести порівняльне дослідження різних підходів. Після виявлення послуги, реєстрації та складу необхідно перевірити якість послуги.

Перевірка послуг зробить систему кращою. Розробка програмного забезпечення на основі компонентів допомагає планувати етапи та підтримувати модулі. Фреймворк корисний для зв'язку між різними модулями, які допомагають у повторному використанні, обслуговуванні та підтримці. Взаємодія людини з комп'ютером заснована на взаємодії між користувачем і комп'ютером. Користувач буде вказувати вимоги до системи. Система вибере найкращий веб-сервіс для розробки.

Композиція веб-сервісів передбачає відкритий, заснований на стандартах підхід до з'єднання веб-сервісів разом для створення нового програмного забезпечення для бізнесу. Такий склад допомагає зменшити трудомісткість, час і витрати, підвищуючи загальну ефективність. Для складу сервісу створюється динамічна, гнучка та адаптована структура. Структура повинна забезпечувати чітке розуміння між процесом логіки та задіяними веб-сервісами. Динамічна композиція веб-сервісу включає багато методів. В цій роботі будуть розглянуті деталі SOA, які останні технології використовуються в індустрії програмного забезпечення. Аналіз буде включати вибір послуги, відповідність і склад. Обговорюються різні алгоритми з їх перевагами та недоліками. Далі впроваджується новий запропонований метод. Експеримент призначений для

оцінки ефективності знову запропонованого алгоритму та його порівняння з існуючими алгоритмами. Експеримент демонструє ефективне використання композиції сервісу, що зменшить час, витрачений на створення нового програмного забезпечення.

1 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ТА СЕРВІСІВ В ПРЕДМЕТНІЙ ОБЛАСТІ

1.1 Довідкова інформація

Архітектура, орієнтована на обслуговування (SOA) — це модель програмного забезпечення, в якій логіка автоматизації розділена на менші окремі логічні одиниці. Разом ці підрозділи складають більшу частину логіки автоматизації бізнесу. Ці одиниці можуть бути розподілені окремо. Кожна одиниця логіки слабо пов’язана і може повторно використовуватися в кількох програмах, а також для досягнення кількох різних цілей проекту.

У моделі програмного забезпечення SOA є три важливі компоненти, а саме: постачальник послуг, реєстр послуг і запитувач послуг, як показано на рис. 1.1 .

• Постачальник послуг – здатний створювати та публікувати службу в реєстрі та робить її доступною через Інтернет.

• Запитувач послуг – виконує операції виявлення служби в реєстрі служб, щоб знайти потрібну послугу, а потім отримує доступ до цієї служби.

• Реєстр послуг – допомагає постачальникам послуг і запитувачам знайти один одного, виконуючи роль реєстру послуг.

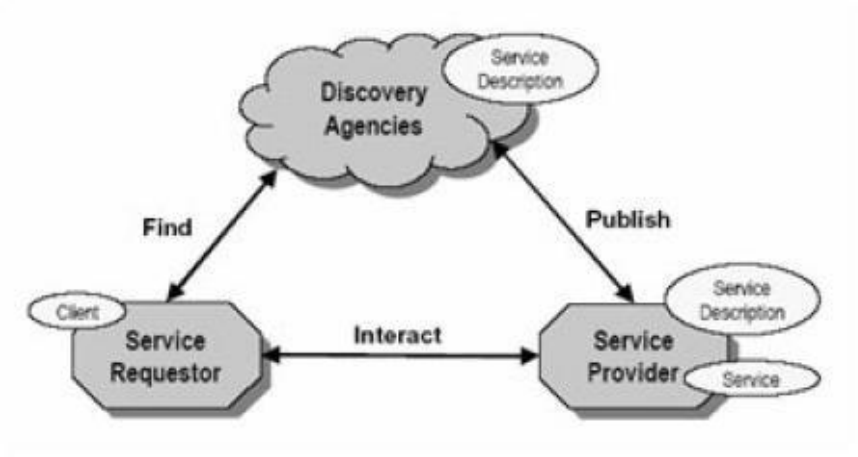


Рисунок 1.1 - Основні компоненти SOA

Кожен блок повинен відповідати базовому набору принципів і стандартів, які дозволять йому розвиватися незалежно, зберігаючи здатність працювати разом з іншими підрозділами в логіці. Сервіс розроблено на основі їх опису, встановлення, доопрацювання та адаптації до бізнес-процесів. Хоча відповідного методу для розробки сервісно-орієнтованих бізнес-додатків не було, увага приділяється складу, дизайну та детальності послуг.

Служби визначаються як один або кілька логічних блоків, які інкапсульовані. Це дозволяє розробникам використовувати одиницю логіки з власних програм. Сервіси, по суті, є структурою зв'язку між додатками та одиницею логіки. Основна перевага SOA полягає в тому, що вона зменшує залежності між клієнтом і постачальником.

1.2 Огляд веб-сервісів

Веб-сервіси є одним із типів постачальників послуг для реалізації моделі SOA. Веб-служби — це автономні веб-додатки, що самоописують, які публікуються, розташовуються та динамічно запускаються в мережі. Вони підтримують сумісний міжмашинний зв'язок через мережу. Веб-сервіс являє собою чотирирівневу модель. На наступному малюнку представлена структура моделі.

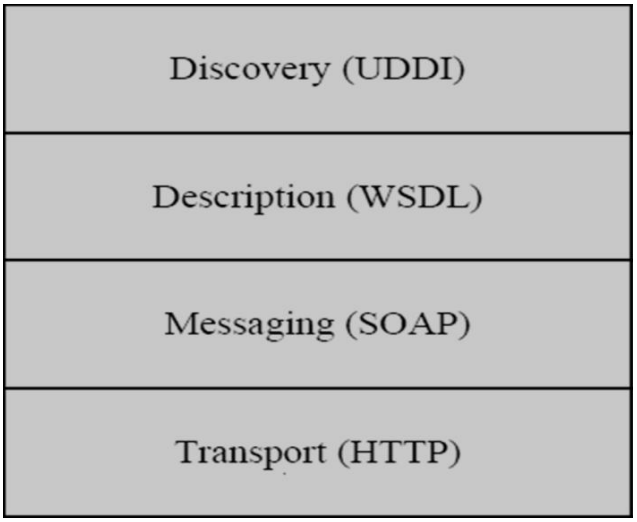


Рисунок 1.2 - Чотиришарова модель стеку веб-служб

Транспортний рівень: транспортний рівень використовує HTTP для передачі повідомлення між мережею додатків. Це низькорівневий протокол, який використовується для транспортування між Інтернетом і моделлю.

Рівень обміну повідомленнями: рівень повідомлень використовує формат XML, який може зрозуміти одержувач. SOAP (Simple Oriented Access Protocol) — поширений формат для обміну даними веб-сервісів. HTTP. Рівень опису: використовується для опису конкретної веб-служби. Зазвичай для опису цієї інформації використовується файл WSDL (мови визначення веб-служб).

1.3 Вибір послуги

Вибір послуги здійснюється на основі опису послуг. Опис послуги визначає метадані, які повністю описують характеристики служб, розроблених у мережі. Ці метадані є важливими і мають основне значення для досягнення слабкого зв’язку, пов’язаного з SOA. Він надає абстрактне визначення інформації, необхідної для розгортання та взаємодії з сервісом. Вибраний сервіс має 3 різні рівні. Це служба суб'єктів, служба завдань та комунальна служба.

Рівень виявлення: це звичайний реєстр. UDDI (Universal Description Discovery and Integration) використовується для пошуку існуючих служб.

Сумісність досягається за допомогою набору відкритих стандартів на основі XML, таких як WSDL, SOAP і UDDI. Оскільки є лише дослідницькі роботи, які показують, що UDDI можна використовувати, немає промислового використання, WSDL розглядається для дослідження складу служби.

1.4 Сервіс-орієнтований метод проектування

Сервісно-орієнтований дизайн включає деякі кроки та дії для класифікації. Це п’ять кроків, як і загальна модель процесу. Нижче перелічено етапи процесу з коротким уявленням про виконану дію.

Оглядаючи системний рівень, можна помітити, що послуги пошарово залежать від їхньої відповідальності, повторного використання та багатовимірної стратегії. Ці шари також аналізуються.

Варіаційний аналіз: як наступний крок у процесі проектування створюється композиція, інтерфейс і логіка компонента сервісу. На основі цього розробляється модель і також проводиться групування. Ідентифікація послуги на етапі проектування: Групування різних послуг є передумовою для ефективного дизайну. Ці служби контролюються та позиціонуються.

Огляд детальності: деталізацію можна використовувати двома способами: по-перше, вона застосовується до області домену, який реалізує вся служба. По-друге, він застосовується до області дії, яку реалізує кожен метод в інтерфейсі.

Моделювання процесу: на цьому етапі моделюються відносини з бізнес-процесами. Це робиться автоматично з використанням плати, оскільки ручне відображення спричиняє неефективність та невідповідність.

1.5 Склад послуг

Моделі послуг допомагають встановити фундаментальні контексти та функціональні граничні служби, надаючи загальні та попередньо визначені типи. Композиція послуг передбачає розробку індивідуальних послуг, які часто складаються з виявлення, інтеграції та виконання існуючих послуг. Справа не тільки в споживанні послуги, однак, а також про надання послуг. Це можна зробити таким чином, що вже існуючі служби об’єднаються в одну або кілька нових служб, які краще підходять до вашої складеної програми.

Існує 2 види композиції. Статична композиція проектує послуги до розробки програмного забезпечення. Динамічна композиція допомагає розробляти послуги на основі введених користувачів. Для нашої вимоги ми будемо використовувати статичну композицію. Це використовується в

семантичній композиції служби. Сервіси призначені для повторного використання. Перед впровадженням необхідно створити структуру дизайну. Діаграми бізнес-процесів перевіряються для створення нових послуг. Перш ніж розробити належну структуру дизайну, необхідно пам’ятати про особливості послуг.[20]

• Сервіси мають добре підібрану деталізацію

• Послуги є цілісними та повними

• Деталі впровадження служби інкапсулюють

• Послуги вміщують декілька шаблонів виклику

• Сервіси використовують інтерфейси

• Моделювання послуг здійснюється за допомогою державних транзакцій

Враховуючи схему процесу, послуги розраховані на запропоновану ідею. Розробник повинен надати інформацію для наступного. [20]

• Відповідальність служби

• Правила обслуговування

• Метод, який використовується для реалізації

• Відсутність дублювання послуг

Умови застосування послуги, тобто якщо послуга має специфічний бізнес-процес, вона повинна бути описана та включена в дизайн послуги

• Учасники, які беруть участь у використанні сервісу

• Порядок виклику послуги

• Підтвердження викликаної послуги

Крім цього, користувач також повинен пам’ятати про технічні аспекти розробки сервісу. Деякі з них:

• Формат, у якому використовується послуга – XML, SOAP або будь-який інший механізм

• Протокол, який використовується для зв'язку із сервісом – протоколи HTTP або RPC

• Доступність послуги

• Очікуване підтвердження надісланої послуги

SOA є одночасно масштабованим і надійним. SOA спрощує розробку і може використовувати різні сервіси та процеси для впровадження. Розробники повинні мати знання предметної області та базові навички програмування. Існує безліч шляхів досягнення нашої мети. Для нашої вимоги ми б дотримувались наведених шляхів.

• Декларативні прийоми

• Абстракції

• Генерація коду

• Оснастка

• Розробка на основі моделі

Принципи та прийоми

SOA надала набір принципів, методів і теорій для побудови бізнес-процесу. Служби складаються з дій (1) з'єднання в поєднанні з потоками даних і керування та (2) описані з умовами входу/виходу. Кожен вид діяльності служби має свою мету. Це потрібно для розробки додатків.

Спочатку служби використовували компоненти як веб-сервіс. Це було зроблено на основі вставки рівня SOAP/WSDL/UDDI поверх наявної програмної програми або компонентів, які створюють вебпослуга можлива для роботи.

Для розробки сервіс-орієнтованих додатків використовуються новітні технології, такі як J2EE, .Net, SOAP, UDDI та багато інших. Вони допомагають об'єднати існуючі служби у великі функціональні одиниці. Для складу служби розглядаються 2 аспекти.

1) Автоматично розробити композицію послуг і точно їх інтегрувати. Здебільшого послуги описуються на основі їх семантичних знань, тобто з використанням мови опису та реєструються на рівні UDDI/SOAP. Хоча на ринку є багато мов опису, ми зазвичай підтримуємо лише одну мову опису. [3]

2) Це допомагає забезпечити успішне виконання композиції сервісу в динамічному середовищі. Цей склад визначається на етапі проектування. [3]

Склад веб-сервісу

Веб-сервіси — це автономні модульні блоки логіки додатків, які можна використовувати в бізнес-функціональності для забезпечення з'єднань для інших програм за допомогою сервера та клієнта або підключення до Інтернету. Веб-послуги описуються за допомогою мови визначення веб-сервісів WSDL, яка є мовою на основі XML. Композиція веб-сервісу повинна забезпечувати динамічну, гнучку та адаптивну структуру. Фреймворк повинен мати можливість розділяти логіку процесу та використовувані веб-сервіси. Крім того, структура повинна мати можливість складати служби вищого рівня для нових або модифікації вимог.

1.6 Переваги SOA технології

SOA – це технологія, що вибухає, і є багато переваг використання цієї технології в будь-якій реалізації. Вони корисні в галузі інформаційних технологій. Деякі з переваг перераховані нижче.

• Оскільки SOA реалізується в мережі, його додаток можна отримати з будь-якого місця у всесвітній мережі, оскільки він відомий своєю незалежністю.

• Можливість повторного використання є високою в SOA, оскільки існують вільні служби, об'єднані разом для виконання однієї функції або програми.

• Додаток SOA повністю не залежить від платформи та мови.

• SOA та веб-сервіси є взаємосумісними, оскільки вони функціонують у масштабах Інтернету та Інтернету.

• SOA мости між несумісними технологіями.

• Оскільки SOA надає можливість повторного використання, ефективність створення нових додатків стає легшою та простішою

• Простіші системи: оскільки SOA базується на галузевих стандартах, складність зменшується і стає легше отримати рішення.

• Зменшення витрат на технічне обслуговування: простота та легке обслуговування зменшує витрати.

• Підвищення архітектурної гнучкості: SOA підтримує композитні рішення. Це допомагає взаємодіяти з кількома системами, використовуючи простіший інтерфейс користувача.

Оскільки в останнє десятиліття SOA став популярним у промислових спільнотах, головне завдання — знайти послуги, які задовольняють продуктивність і вимоги. Чутливість сервісу дає розробникам можливість знайти відповідний програмний сервіс, а його узгодженість та оптимальність перевіряються. Підхід зосереджений на техніці повторного використання однієї і тієї ж служби для різних цілей. Доступно кілька програмних служб, які допомагають досягти того самого бажаного результату. Запитувачі послуг мають вибір щодо того, яку послугу вони можуть включити в свою програму. Хоча ці послуги спрямовані на досягнення однакового результату, вони можуть давати різні результати для тих самих вхідних параметрів. Тому, пам’ятаючи про це, служба програмного забезпечення має бути впевнена, що вони відповідають вимогам. Сервісна чутливість є корисним інструментом у моделі побудови та оцінки.

1.7 Лінія продуктів та рішень

Було виявлено, що практика лінійки програмних продуктів знижує як вартість програмного забезпечення, так і кількість дефектів. Це розглядається для розробки систем сервіс-орієнтованої архітектури (SOA). Ідея використання мови опису семантичної веб-служби для кодування конфігурацій продукту та

використання її як вхідних даних для пропонованого фреймворку допомагає досягти реального застосування. Налаштування веб-сервісів на основі функціональних і нефункціональних вимог для досягнення підходу до розробки лінійки продуктів обговорюється в статті, написаній Хонгю, Робін Р. Лут і Самік Басу.

Для управління механізмом веб-сервісів слід пам’ятати, що лінійка програмних продуктів повинна бути задоволена. Умови викладені у вигляді ділових лошат, встановлених системами для виконання поставленого завдання.

1.8 Оптимізація сервісу

Вибір відповідного веб-сервісу для конкретного завдання є складним завданням, оскільки сховище служб буде мати багато схожих функціональних служб. Це включатиме як функціональні, так і нефункціональні властивості служби. Функціональна властивість посилається на те, що може робити служба, а нефункціональна властивість — на те, як служба може це робити.

Для розробників, які створюють програму на основі SOA, яка залежить від сервісів та його складу, оптимізація сервісу дуже важлива. Основна ідея полягає в тому, щоб гарантувати, що обраний ними сервіс допоможе виконати точно відповідне масштабу проекту.

Розробники можуть запускати служби, щоб перевірити їхню продуктивність, і варіанти однієї і тієї ж функціональної служби порівнюються за допомогою оператора порівняння, щоб визначити оптимальний сервіс. Запровадження вибору послуг сприяє кращому розвитку. Коли постачальник послуг створює нове програмне забезпечення, основні вимоги полягають у тому, чи використовує вимога лише один сервіс або 2-3 послуги.

Крім того, ще одна головна проблема полягає в тому, що можуть бути служби з такою ж функціональністю.

1.9 Виявлення веб-сервісу

Виявлення веб-сервісу означає діяльність з пошуку веб-сервісу, що відповідає певним функціональним критеріям. Відбір передбачає діяльність з оцінки та ранжування виявлених веб-сервісів. Вибір сервісу здійснюється на основі описів веб-сервісів, які містяться в онтологічних концепціях. Алгоритм заснований на аналізі чутливості сервісу. Основна увага при цьому методі полягає в тому, що якщо є одна або кілька веб-сервісів, які відповідають вимогам, вибирається найкращий сервіс. Цей модуль є попередньою роботою Мохіта Суда та доктора Юаня. Основна ідея цієї роботи – знайти оптимальний сервіс для заданих вимог.

1.10 Семантичний відповідник

База знань необхідна для встановлення відповідності та вибору послуг. Механізм онтологічного висновку має опис веб-сервісу та опис запиту. Щоб отримати точну послугу, потрібно підібрати різні параметри. Зі списку потенційно доступних послуг вибирається найбільш підходяща послуга.

Послуга відбирається за методикою онотології. Rest не є стандартом, мовою чи протоколом, але використовується як архітектурна база для роботи з Інтернетом та технологіями протоколів (наприклад: HTTP і XML). REST простіше, ніж SOAP. REST допомагає краще розробити програмне забезпечення. Це допомагає створити сервер і клієнт. Це найбільша перевага технологій веб-сервісів, які роблять все однаковим.

Необхідні серверний і клієнтський компонент. За взаємодію з клієнтами відповідає серверний компонент. Сервер буде тут сховищем з усіма потенційними веб-сервісами. Кількість клієнтів може збільшуватися в залежності від набору популярності програми. Постачальник послуг може створити нову послугу та надати запитувачам доступ до них для своїх програм.

Неможливо, щоб усі послуги були унікальними. Результат при виборі послуги залежить від мети програми. Також потрібно, щоб запитувач послуг вибрав найбільш підходящий сервіс для даної умови, інакше програма буде зруйнована.

Основна мета цієї дипломної роботи – розглянути композицію веб-сервісу. Оскільки композиція веб-сервісу також включає вибір веб-сервісу, йому надається деяке значення.

1.11 Підходи до аналізу чутливості

Існують численні підходи, які за своєю природою подібні до аналізу чутливості, оскільки він застосовується до різних моделей, і не обмежується областю програмного забезпечення. Ці підходи не були застосовані до моделі SOA як засобу ефективної оцінки ефективності послуги для виконання гарантій якості та ефективного вибору послуг.

Brute Force — працює лише на невеликих моделях, розв’язання яких займає короткий проміжок часу, змінення вихідних даних і повторне вирішення моделі, щоб побачити, які результати ви отримаєте. Класичний аналіз чутливості – застосовується до дуже великих моделей, вирішення яких займає багато часу. Класичний метод чутливості покладається на зв’язок між початковою таблицею та будь-якою наступною таблицею для швидкого оновлення оптимального рішення, коли вносяться зміни до коефіцієнтів вихідної таблиці.

Комп’ютерне визначення діапазону – проста інформація про те, як можуть змінитися певні коефіцієнти, перш ніж поточне оптимальне рішення буде фундаментально змінено. Джон В. Чиннек, професор Університету Карлтона, провів значні дослідження щодо використання аналізу чутливості, щоб визначити, які дані мають найбільший вплив на результати в програмній системі. Він використав комбінацію лінійного програмування та комп’ютерного визначення діапазону, щоб встановити цю інформацію.

Дисперсійний аналіз (ANOVA) – підхід ANOVA дуже поширений у статистиці. Це включає виконання обчислень, щоб визначити, чи є середнє значення двох наведених методів лікування еквівалентними чи ні. Хоча цей підхід успішно зарекомендував себе при оцінці різних статистичних моделей, його методи найкраще підходять для аналізу чутливості на основі двох факторів.

Проектування експерименту (DOE) – експерименти DOE не обмежуються програмними системами і можуть виконуватися на різних речах, включаючи; програмні системи, люди, рослини, тварини тощо. DOE дозволяє спостерігати та судити про важливість для виходу вхідних змінних, які можуть діяти окремо або в поєднанні одна з одною. DOE можна вважати найдавнішим походженням аналізу чутливості. Його підхід не був адаптований до програмного забезпечення.

1.12 Попередні дослідження

Аналіз чутливості – це практична механіка оптимізації, яка використовувалася для встановлення впевненості та оптимізації продуктивності в багатьох речах, включаючи статистичні дослідження та програмне забезпечення. Наразі не було достатньої кількості досліджень, спрямованих на застосування аналізу чутливості до успіху та вдосконалення SOA продемонстрували методику, яка дозволяє проводити двофакторний аналіз чутливості послуг на основі SOA. Там методика застосовує аналіз чутливості до SOA з метою оцінки та прогнозування якості програмного забезпечення, а також для визначення оптимальних конфігурацій програмних послуг лише з двома факторами окреслили методологію багатофакторного аналізу чутливості систем на основі SOA. Їх підхід використовується для того, щоб визначити, які окремі фактори чи спільні фактори мають значний вплив на продуктивність програмної системи .

2 АЛГОРИТМІЧНІ ТА МАТЕМАТИЧНІ РІШЕННЯ ДЛЯ ЗАДАЧ ДАНОГО КЛАСУ

2.1 Різні підходи до складу служби

Щоб виконати нескладне завдання, у кожного буде певний план для виконання. Під час реєстрації на курс кафедра повідомить студентів про наявність курсу. Далі професор вирішує, скільки студентів він хоче мати на курсі. Таким чином, все відбувається відповідно до процесу чи плану. Це відомо як процес синтезу. Це більше відомо як статична композиція. Для менших проблем це може добре спрацювати. Але для автоматичної композиції веб-сервісів цьому бракує гнучкості та масштабованості. Вимоги користувачів враховуються до автоматичної композиції веб-сервісу, інкапсуляції змінних і динамічних дій у домені. Для опису правил використовується семантика та онтологія.

Згідно з посиланням , для динамічного виявлення сервісів вибір і композиція будуть задіяти онтологію процесу. Це включатиме набір правил, відношення вибору, виключне відношення, попереднє відношення, неупорядковане відношення та паралельну композицію. Це засновано на 3-х кортежах N (набір вузлів), E (набір ребер) і R (набір правил). Вони витягуються як відносини та правила, які складають дерево онтологій.

Наступним кроком, за запитом служби, система шукає цільовий вузол в дереві онтології процесу і перевіряє їх взаємозв’язок перед створенням. Для виконання цієї дії автори ввели ще два визначення глибини вузла і батьківського вузла, на основі яких написані два алгоритми. Цей процес не можна порівняти з жодним іншим підходом. В основному це пошук, що виконується за допомогою специфікації користувача, що включає можливості та поведінку послуг. Оскільки це техніка, схожа на робочий процес, вона може складати бізнес-процеси та динамічно інтегруватися.

Алгоритм пошуку на основі графіка

Композиція сервісу, в основному, має на меті зменшити людське втручання під час виявлення сервісу, зіставлення, ранжування, фільтрації та аргументації використовував алгоритм пошуку на основі графіка для виявлення та підбору партнерів на основі семантичної анотації служб. Це допомагає знайти всі можливі комбінації для обґрунтування та перевірки послуг.

Семантика служб та їх властивості фіксуються для складу послуг. Мови опису, які здатні захоплювати семантику, - це OWL-S, WSDL і RDF. Вони надають інформацію про властивості послуг. Наприклад, OWL-S матиме 2 параметри про семантику сервісу. Вони бувають функціональними (наприклад, введення, вихід, умови) і нефункціональними (QoS). Це допомагає у виявленні служби. Узгодження послуг є важливим питанням для складу послуг. [5] Алгоритм пошуку на основі графіка заснований на модифікованому алгоритмі пошуку першого дихання. З посиланням на описаний алгоритм. Для складу послуг має бути сформована деревоподібна структура. Це матиме вузол, який є унікальним індексом, ребро, що з'єднує два вузли з виходом з одного вузла як вхід до іншого вузла, а гілка і композиція є підграфами загального графа, що містить відмінний набір вузлів і ребер.

Початковим етапом буде виявлення сервісів і відповідність сервісів на основі семантичних властивостей. Перевірка проводиться, щоб переконатися, що система побудована правильно. Пізніше для реалізації цього алгоритму здійснюється прототипування. Прототип реалізації буде виглядати як на малюнку нижче.

Еволюційний алгоритм

На основі запропонованої ідеї, обговорюються мешапи та еволюційні алгоритми. Mashups надає мову програмування модель, за допомогою якої можна створювати різні веб-додатки. Відповідний фреймворк mashup дозволить користувачеві комбінувати веб-сервіси для розробки. На основі цієї ідеї

інформація стає доступною через модель користувача, для якої застосовується еволюційний алгоритм для того, щоб існуючий додаток збирав інформацію про веб-сервіси.

Еволюційні алгоритми - термін, що часто використовується для загального опису алгоритмів пошуку, оптимізації або навчання, що засновані на формалізованих принципах природного еволюційного процесу. Еволюційні методи призначені для пошуку бажаних рішень і засновані на статистичному підході до дослідження ситуацій та ітераційному наближенні системи до шуканого стану. На відміну від точних методів математичного програмування еволюційні методи дозволяють знаходити рішення, близькі до оптимальних, за прийнятний час, а на відміну від відомих евристичних методів оптимізації характеризуються істотно меншою залежністю від особливостей додатку (більш універсальні) і в багатьох випадках забезпечують кращу ступінь наближення до оптимального рішення.

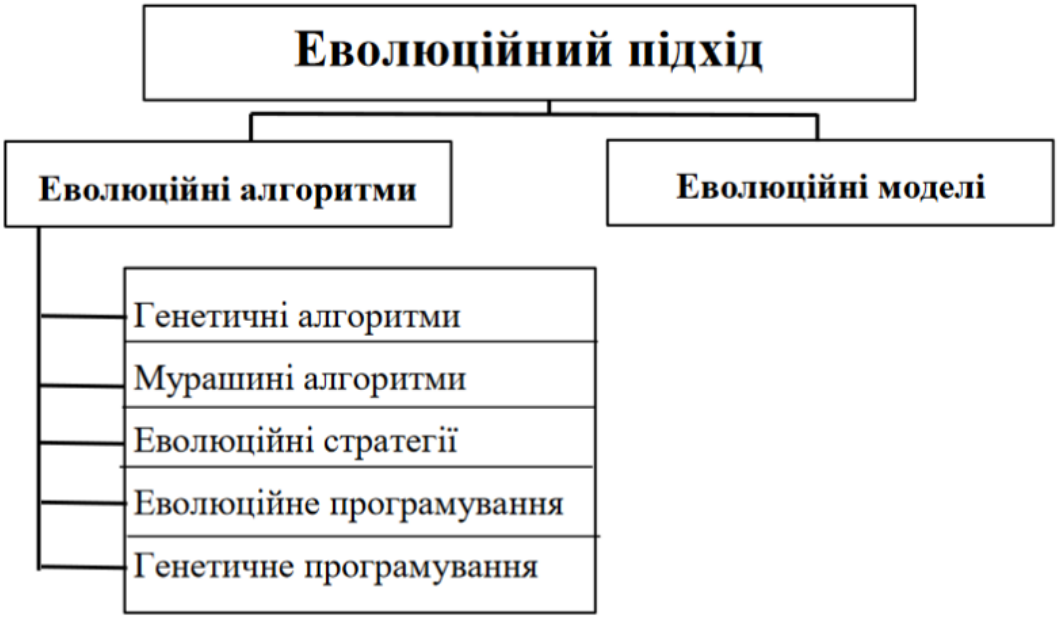


Рисунок 2.1 – Класифікація еволюційного підходу

Еволюційна стратегія - евристичний метод оптимізації в розділі еволюційних алгоритмів, який засновано на адаптації та еволюції. Еволюційна стратегія схожа з генетичним алгоритмом, але існує декілька суттєвих відмінностей.

Сконцентруємо увагу на найважливіших подібностях і розходженнях між еволюційними стратегіями і генетичними алгоритмами. Очевидно, що головна подібність полягає в тому, що обидва методи використовують популяції потенційних рішень і реалізують принцип селекції і перетворення найбільш пристосованих особин. Однак обговорювані підходи сильно відрізняються один від одного.

Перше розходження полягає в способі представлення особин. Еволюційні стратегії оперують векторами дійсних чисел, тоді як генетичні алгоритми - двійковими векторами.

Друге розходження між еволюційними стратегіями і генетичними алгоритмами криється в організації процесу селекції. При реалізації еволюційної стратегії формується проміжна популяція, що складається з усіх батьків і деякої кількості нащадків, створених у результаті застосування генетичних операторів.

За допомогою селекції розмір цієї проміжної популяції зменшується до величини батьківської популяції за рахунок виключення найменш пристосованих особин.

Сформована в такий спосіб популяція утворює чергове покоління. Напроти, у генетичних алгоритмах передбачається, що в результаті селекції з популяції батьків відбирається кількість особин, рівна розмірності вихідної популяції, при цьому деякі (найбільш пристосовані) особи можуть відбиратися багаторазово.

У той же час, менш пристосовані особи також мають можливість з’явитися в новій популяції. Однак шанси їхнього відбору пропорційні величині пристос. Незалежно від застосовуваного в генетичному алгоритмі методу селекції (наприклад, рулетки або рангового) більш пристосовані особи можуть

відбиратися багаторазово. При реалізації еволюційних стратегій особи відбираються без повторень. В еволюційних стратегіях застосовується детермінована процедура селекції, тоді як у генетичних алгоритмах вона має випадковий характер.

Третє розходження між еволюційними стратегіями і генетичними алгоритмами стосується послідовності виконання процедур селекції і рекомбінації (тобто зміни генів у результаті застосування генетичних операторів). При реалізації еволюційних стратегій спочатку виконується рекомбінація, а потім селекція. У випадку виконання генетичних алгоритмів ця послідовність інвертується. При здійсненні застосування еволюційних стратегій нащадок утворюється в результаті схрещування двох батьків і мутації. Формована в такий спосіб проміжна популяція, що складається з усіх батьків і отриманих від них нащадків, надалі піддається селекції, що зменшує розмір цієї популяції до розміру вихідної популяції. При виконанні генетичних алгоритмів спочатку виконується селекція, що приводить до утворення перехідної популяції, після чого генетичні оператори схрещування і мутації застосовуються до особин (обираних з ймовірністю схрещування) і до генів (обираних з ймовірністю мутації).

Наступне розходження між еволюційними стратегіями і генетичними алгоритмами полягає в тому, що параметри генетичних алгоритмів (такі, як ймовірності схрещування і мутації) залишаються сталими протягом усього процесу еволюції, тоді як при реалізації еволюційних стратегій ці параметри піддаються безперервним змінам (так звана самоадаптація параметрів). Ще одне розходження між еволюційними стратегіями і генетичними алгоритмами стосується трактування обмежень, що накладаються на розв'язувані оптимізаційні задачі. Якщо при реалізації еволюційних стратегій на деякій ітерації нащадок не задовольняє всім обмеженням, то він відкидається і включається в нову популяцію. Якщо таких нащадків виявляється багато, то еволюційна стратегія запускає процес адаптації параметрів, наприклад, шляхом

збільшення ймовірності схрещування. У генетичних алгоритмах такі параметри не змінюються. У них може застосовуватися штрафна функція для тих особин, що не задовольняють накладеним обмеженням, однак ця технологія має багато недоліків.

Створення простого мешапу вимагає від користувача технічних знань. Аналогічно, у mashup є певні недоліки, але все ж він забезпечує хорошу агрегацію з джерелами даних. Важливо досягти автоматичного та адаптивного змішування. WSDL використовується для захоплення повідомлень, кінцевих точок та інтерфейсів для служб. Семантичне знання фіксується OWL-S. Таким чином, це допомагає паралельному виконанню композиції веб-сервісу.Архітектура системи - це каркас, який повідомляє про потік інформаційного вмісту між різними модулями. Представлення знань та онтології є важливими для мешапів. Це використовується для поєднання даних з різних джерел. Інформація в Інтернеті представлена за допомогою RDF (Resource Description Framework). У модулі композиції використовується еволюційний алгоритм.[2] Еволюційний процес протікає так, як показано на схемі. Процес починається з отримання можливого набору вхідних даних. Визначається найкраща комбінація. Таким чином композиція веб-сервісу планується, репрезентується та оптимізується. При цьому обробляються основні цілі, такі як правильність, повнота та довжина, зважені конкретною лінійною комбінацією. Ці функції обчислюються за допомогою різних функцій. [2]Тому цей підхід допомагає перевірити масштабованість та коректність.

Композиція на основі ознак

Композиція на основі ознак допомагає визначити вибрані ознаки для отримання результату. Він заснований на шаблонах/моделях. Це підхід зверху вниз, тобто спочатку отримують кінцевий результат або програмний продукт, а звідти класифікують набір функцій. Він розглядає 3 основні фази.

1) моделювання ознак і аспектів щодо використовуваної мови

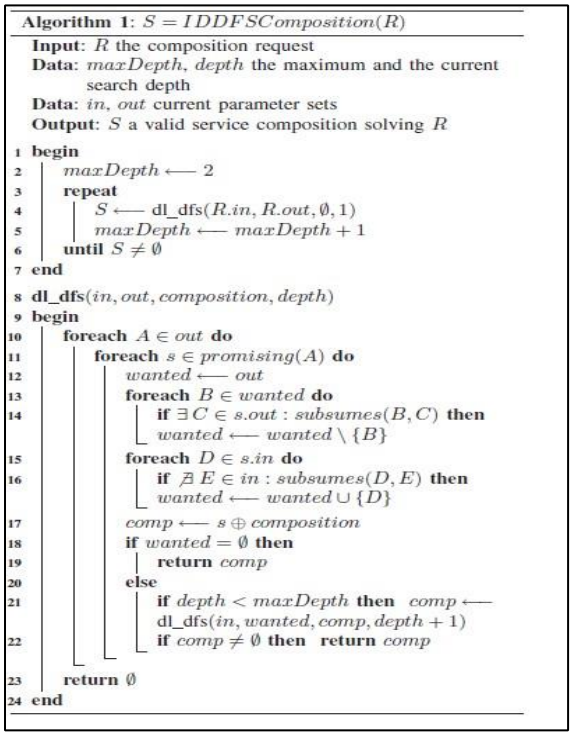
2) аналіз обмежень, що стосується рівня ознак і аспектів

3) композиція моделі, яка розповідає про процес отримання окремого продукту.

Цей метод не досліджує можливість повторного використання, а також динамічний вибір функцій для композиції нашого веб-сервісу.

Алгоритм IDDFS (ітеративний поглиблення пошуку в глибину)

Автори Weise, Steffen, Comes і Kurt обговорювали наступні три алгоритми у своїй статті «Різні підходи до складу семантичних веб-сервісів», яка була представлена на третій міжнародній конференції з Інтернету та веб-додатків, що відбулася в 2008 році. Цей алгоритм є простим підходом. для створення веб-сервісів за допомогою уніфікованого пошуку. Алгоритм пошуку не використовує ніяких спеціальних математичних формул для досягнення результату. Він використовує ітераційний пошук у глибину. Це допомагає швидко знаходити рішення, коли сховище послуг є торговим центром, і намагається надати оптимальне рішення, якщо проблема вимагає вичерпного пошуку.



Algorithm 1: $S = IDDFSComposition(R)$

Input: R the composition request
Data: $maxDepth$, $depth$ the maximum and the current search depth
Data: in, out current parameter sets
Output: S a valid service composition solving R

```
1  begin
2      maxDepth ⟵ 2
3      repeat
4          S ⟵ dl_dfs(R.in, R.out, ∅, 1)
5          maxDepth ⟵ maxDepth + 1
6      until S ≠ ∅
7  end
8  dl_dfs(in, out, composition, depth)
9  begin
10     foreach A ∈ out do
11         foreach s ∈ promising(A) do
12             wanted ⟵ out
13             foreach B ∈ wanted do
14                 if ∃ C ∈ s.out : subsumes(B, C) then
                       wanted ⟵ wanted \ {B}
15             foreach D ∈ s.in do
16                 if ∄ E ∈ in : subsumes(D, E) then
                       wanted ⟵ wanted ∪ {D}
17             comp ⟵ s ⊕ composition
18             if wanted = ∅ then
19                 return comp
20             else
21                 if depth < maxDepth then comp ⟵
                       dl_dfs(in, wanted, comp, depth + 1)
22                 if comp ≠ ∅ then return comp
23     return ∅
24 end
```

Рисунок 2.2 - Алгоритм IDDFS

Відповідно до алгоритму композиція веб-сервісу будується на основі дійсних веб-сервісів. Цикл викликається ітераційно, збільшуючи межу глибини на кожному кроці, щоб забезпечити дійсне рішення. Недоліком цього підходу є те, що він повільний для великого сховища пам'яті. Крім цього, він не використовує жодної додаткової інформації для своїх критеріїв пошуку. Ефективність можна значно підвищити, якщо усунути ці недоліки.

Жадібний алгоритм

Алгоритм Greedy виконує сортування всередині за допомогою функції порівняння. Виходячи з вимоги, дві найкращі послуги входять до короткого списку. Сервіси порівнюються за допомогою функції. Найкращі послуги перевіряються на основі відповідних параметрів. В основному береться послуга з коротшими параметрами, оскільки композиція стане маленькою і короткою.

Algorithm 2: $S = greedyComposition(R, c)$

Input: R the composition request
Data: X the sorted list of compositions to explore
Output: S the solution composition found, or $\emptyset$

```
1  begin
2      X ⟵ ⋃_{∀A∈R.out}(promising(A))
3      while X ≠ ∅ do
4          X ⟵ sort(descending, X, c)
5          S ⟵ popLastElement(X)
6          if isGoal(S) then return S
7          foreach A ∈ wanted(S) do
8              foreach s ∈ promising(A) do
9                  X ⟵ appendList(X, s ⊕ S)
10     return ∅
11 end
```

Рисунок 2.2 - Жадібний алгоритм 1

Функція порівняння включає наступні кроки.

Algorithm 3: $r = c_{cmp}(S_1, S_2)$

Input: S_1, S_2 two composition candidates
Output: $r \in \mathbb{Z}$ indicating whether S_1 $(r < 0)$ or S_2 $(r > 0)$ should be expanded next

```
1  begin
2      i1 ⟵ |wanted(S1)|
3      i2 ⟵ |wanted(S2)|
4      if i1 = 0 then
5          if i2 = 0 then return |S1| − |S2|
6          else return −1
7      if i2 = 0 then return 1
8      e1 ⟵ |eliminated(S1)|
9      e2 ⟵ |eliminated(S2)|
10     if e1 > e2 then return −1
11     else if e1 < e2 then return 1
12     if i1 > i2 then return −1
13     else if i1 < i2 then return 1
14     if |S1| ≠ |S2| then return |S1| − |S2|
15     return |known(S1)| − |known(S2)|
16 end
```

Рисунок 2.3 - Жадібний алгоритм 2

Багатоцільовий генетичний алгоритм

Це генетичний підхід до вирішення алгоритмів метаевристичної оптимізації. Він містить такі біологічні терміни, як мутація, кросовер, природний відбір та виживання найсильніших. У порівнянні з іншими підходами, які обговорювалися раніше, це має перевагу оптимізації рішення на основі кількох зроблених припущень.

Стохастична оптимізація є одним з основних підходів в обчислювальній статистиці. Стохастична оптимізація означає мінімізацію (або максимізацію) цільових функцій за наявності випадковості в процесі оптимізації. Випадковість може бути пов'язана як із застосуванням методу Монте-Карло в процедурі моделювання, так із шумом у вимірюваннях або ж з обома випадками одночасно.

Стохастичні методи оптимізації дуже зручні для оцінки цільових функцій за наявності шуму. В останнє десятиліття одними з найбільш

перспективних та широковживаних методів стохастичної оптимізації є генетичний алгоритми з недомінантним сортуванням - NSGA-II [2] та NSGA-III [3].

В цих алгоритмах після ініціалізації популяції, особини в популяції групуються в набір фронтів за принципом недомінантності. Перший фронт розглядають як абсолютно недомінантний набір у поточній популяції, на другому фронті переважають особини лише першого фронту і так далі. Для кожного окремого учасника на кожному фронті призначається оцінний ранг, що базується на номері фронту, до якого вони належать. Особинам у першому фронті надають рангове значення 1, особинам у другому -- значення 2 і так далі.

Окрім значення рангу, вводять новий параметр, який називається щільністю скупчення, і його розраховують для кожної особини. Щільність скупчення означає, наскільки близько до своїх сусідів розташована особина. Більша середня щільність призводить до більшого розмаїття популяції. Батьків добирають з популяції за допомогою бінарних турнірів з урахуванням рангу та щільності скупчення. Між двома особинами добирається та, яка має менший ранг, або більшу щільність скупчення. За допомогою операторів кросовера та мутацій можна дібрати популяцію для генерування потомків. Популяцію повторно сортують за принципом недомінантності та до неї добирають лише найкращі N особин для створення нової популяції, де N - чисельність популяції.

При застосуваннях багато-цільових генетичних алгоритмів до задач багато-критеріальної оптимізації їх ефективність перевіряється рішенням двох задач:

а) спроможністю еволюційного алгоритму давати збіжність до Парето-оптимального фронту ("задача збіжності")

б) давати гарний розподіл оптимальних розв’язків по всьому Парето фронту ("задача поширення")

Еволюційний алгоритм передбачає кілька етапів у процесі складання сервісу. На першому етапі створюються всі сервіси. Усі ці послуги тестуються, і для складу вибирається найкращий сервіс.

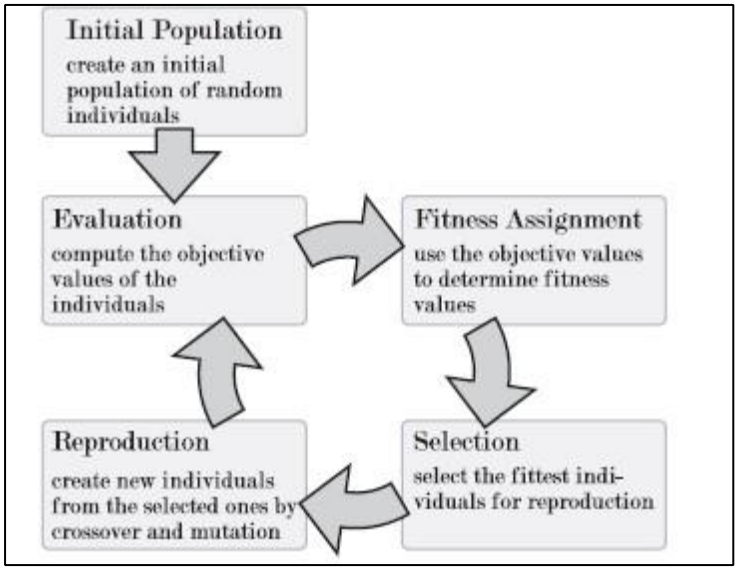


Рисунок 2.4 - Багатоцільовий еволюційний алгоритм

Підхід «знизу вгору» інтегрує веб-технології для композиції веб-сервісу. Це враховує BPEL (мова виконання бізнес-процесів) для його інтеграції. Основна точка зору цього підходу полягає в тому, що він використовує зрозумілі для машини описи необхідних послуг у процесі для реалізації семантичних описів. Це допомагає динамічно знаходити потрібні послуги. Якщо одна служба не відповідає вимозі, використовується алгоритм рекурсивного зворотного ланцюга, щоб знайти службу, яку можна викликати. Основним недоліком цього підходу є те, що він не враховує попередні та постумови виявлення та композиції служби.

Підхід до METEOP-C

Розробляється динамічний інструмент для веб-композиції. Це дозволяє дизайнерам розробляти процес на основі процесу та бізнес-обмежень. Концепція, в якій це працює, полягає в тому, що надається специфікація служби,

на основі якої здійснюється виявлення служб. Це абстрактний процес. Більше функціональних специфікацій використовується для вибору послуги. Дизайнери використовують BPEL для моделювання процесів. Основним недоліком цього підходу є те, що вручну вибір послуги виконується для складу служби.

Композиція на основі шаблону: підхід AI

Штучний інтелект можна використовувати в будь-якій сфері. Відповідно до цього підходу абстрактні дії можуть використовуватися для опису послуги, яка використовується. Послуги виявляються на основі їх діяльності. Основна ідея цього підходу полягає в тому, що він намагається зосередитися на цінності переваг. Завдяки цьому послуги з високою відповідною специфікацією займають одне з перших місць у наборі доступних послуг. Це буде використовувати технологію Semantic Web (OWL) для написання шаблону специфікації послуг. Планування AI використовується для виявлення послуг.

Підхід до складу WSMO

WSMO communications також розробила інструмент для динамічної композиції для веб-сервісів. Інструмент складання дозволяє розробникам згадати про свої цілі, посередників і контрольний потік. Процес включає перелік цілей, з яких враховуються конкретні голи. Невідповідність типів між входом і виходом підтримується посередниками. Тут також використовується OWL-S (технологія семантичної веб-сторінки)

Автоматична композиція за допомогою SHOP2

Цей підхід описує, як систему планування ШІ можна використовувати для опису веб-сервісу, щоб автоматично складати веб-сервіси. Хоча цей підхід стосується планування, він лише частково підтримує створення веб-сервісів. У той же час, це також не підтримує складений процес з усіма підтримуваними OWL-S конструкціями. [4]

Вважається, що цей метод надає набір інструментів для складання веб-сервісів. Концепція експертної системи, заснованої на правилах, допомагає визначити автоматичне створення складених сервісів з використанням існуючих сервісів. Негативний аспект цього методу полягає в тому, що його ідея обмежується вибором веб-сервісів для компонування на основі входу та виходу, а також певних передумов, які впливатимуть.

Це програмна система, яка забезпечує планування складу послуг. Тут використовується метамодельний підхід. Мета-модель тут означає, що це буде використовувати шаблони для виявлення служби. Тут беруть участь два основних компоненти: композитор і актор. Композитор несе відповідальність за формування складу алгоритму планування пошуку для виконання дії. Розпорядник відповідає за планування та виконання послуг у складі

2.2. Обмеження підходів

Існують серйозні проблеми з існуючими підходами, які узагальнено та наведено в таблиці нижче. Деякі з основних проблем динамічної композиції:

Виявлення та вибір сервісу на основі відповідності функціональної та нефункціональної семантики: це стосується основної функціональної семантики, тобто умов введення, виведення, попередніх та постулаційних умов. Нефункціональна семантика стосується додаткової інформації (курс валюти, поштовий індекс тощо). Це більше стосується вибору однієї служби з набору семантичних послуг.

Прив’язка послуги та посилання: тут обговорюється те, як послуга вибирається для композиції. В основному тут описано робочий процес та планування AI.

Стратегія композиції: це включає композицію семантичних веб-сервісів. З планування визначається остаточна композиція, чи можна це зробити автоматично чи напівавтоматично.

Виконання: тут фокусується кінцевий результат виконання.

Семантична веб-технологія: обговорюється технологія веб-сервісу, яка стосується цього підходу.

Composition Approach	Service Discovery		Service Selection		Service Binding & Referencing	Composition Strategy	Execution	Semantic Web Technology
	Functional Semantics	Non-Functional Semantics	Functional Semantics	Non-Functional Semantics				
Bottom-up Approach	*Partial*	*No*	*Partial*	*Yes*	*Run-time*	*Dynamic*	*Yes*	*OWL-S*
METEOR-S	*Yes*	*Partial*	*Yes*	*Partial*	*Deployment/Design time*	*Dynamic*	*Yes*	*WSDL-S*
Template Based Composition	*Partial*	*Partial*	*Partial*	*Partial*	*Dynamic*	*Automatic*	*Yes*	*OWL-S*
WSMO Approach	*Yes*	*Partial*	*Yes*	*Partial*	*Dynamic*	*Semi-Automatic*	*Yes*	*WSMO*
HTN Planning using SHOP2	*Partial*	*Partial*	*Partial*	*Partial*	*Dynamic*	*Automatic*	*Yes*	*OWL-S*
SWORD	*Partial*	*No*	*Partial*	*No*	*Off-line/ Composition time*	*Semi-automatic*	*Yes*	*Independent of Standards*
Plængine	*Yes*	*No*	*Yes*	*No*	*Dynamic*	*Automatic*	*Yes*	*Integrated Meta-Model*

Рисунок 2.5 - Обмеження різних підходів

Успішна модель вибору та складання веб-сервісів передбачає вибір/відкриття та композицію веб-сервісу. Це досягається лише використанням концепцій семантичної мережі.

Сервісно-орієнтована архітектура (SOA) — це нещодавня розробка в області інформатики. Завдяки цьому програмне забезпечення розробляється ефективно та легко. Все пов’язано з онлайн-активністю в Інтернеті. Усі організації мають свої веб-сайти та сервіси. Це допомагає легко автоматизувати процеси, а також забезпечує глобальну видимість. Ця магістерська робота буде зосереджена на динамічному складі цих веб-сервісів для будь-якої даної вимоги.

Зокрема, розроблено алгоритм оцінки динамічного складу. Ця техніка додатково перевіряється на якість вибраних послуг та їх ефективність.

2.3 Мета дослідження SOA імплементацій

Сервісно-орієнтована архітектура допомагає інтегрувати веб-додатки. Це також можна згадати, оскільки веб-сервіси можуть реалізовувати сервісно-орієнтовану архітектуру. Перевага веб-сервісів полягає в тому, що вони надають основні протоколи, до яких може отримати доступ будь-яка система в мережі. В Інтернеті є багато веб-сервісів, які виконують різноманітні функції.

Веб-сервіси — це автономні модульні програми, що самоописують, які можна опублікувати, розташувати та викликати в будь-якому місці мережі. В основному, це допомагає швидко розвивати систему за допомогою вихідних сервісів. Оскільки існує більше веб-сервісів з тією ж функціональністю, важливо вибрати найбільш підходящий. Крім того, вимога може включати більше однієї служби, композиція повинна бути виконана належним чином.

Склад веб-сервісу

Веб-сервіс є останнім трендом індустрії інформаційних технологій. Композиція веб-сервісу полягає у створенні різних веб-сервісів для досягнення заданої вимоги. Композиція веб-сервісу допомагає швидко складати складні веб-сервіси. Таким чином, він створює потужний продукт веб-сервісу. Служби можуть складатися як статично, так і динамічно. Послуга, обрана під час розробки, є статичним підходом. Більшість сучасних технологій працює статичним методом. Існує лише кілька кейсів і жодного практичного експерименту для динамічної реалізації композиції веб-сервісу. У динамічній композиції послуги вибираються під час виконання. Послуги шукаються в репозиторії, а потім вибирається найкращий сервіс.

2.4 Мова визначення веб-служби (WSDL)

Мова визначення веб-сервісу (WSDL) допомагає забезпечити спосіб зв'язку з постачальниками послуг і веб-сервером/клієнтом. Існують певні специфікації, які необхідні для WSDL.

Файл WSDL — це файл мови опису веб-служб, написаний мовою XML для опису веб-служб. Він містить інформацію про кінцеві точки або інтерфейси для підключення до зовнішнього світу в загальноприйнятому форматі. Специфікації формату файлу WSDL (підтримується W3C.org) визначають умови публікації каналів даних для обміну даними, щоб мати віддалений доступ програм через порти та кінцеві точки.

Багаторазове використання опису та незалежних проектів концерну регулюється веб-службою. Це досягається за допомогою кількох різних типів елементів, включаючи типи (визначення типів даних), повідомлення (передані дані), операції (дії) і протоколи, які використовує служба. Усі вони керуються на абстрактному рівні. Прив’язка деталей транспортного та дротового форматів визначається зв’язкою, яка групує кінцеві точки разом для реалізації спільного інтерфейсу.

Limthanmaphon і Zhang описали про це у своїй статті «Склад веб-сервісу з Case-Основане міркування». Автори зазначають, що для встановлення сервісу в мережі необхідно сім елементів. Однією з найпоширеніших мов є WSDL.

• Тип надає інформацію про визначення типів даних для опису повідомлення, яким обмінюються.

• Повідомлення містить деталі абстрактного визначення даних, які мають бути передані або передані.

• Операція створює абстрактний опис дії, яку може підтримувати служба. Це стосується вхідного та вихідного повідомлення.

• Тип порту – це операція, встановлена однією або кількома кінцевими точками.

• Прив'язка згадує конкретний протокол і специфікацію формату даних для операцій і повідомлень.

• Порт відноситься до URL-адреси прослуховуваної веб-служби.

• Сервіс – це сукупність пов'язаних портів

2.5 Якість послуг та їх виконання

Веб-послуги включають мережевий протокол. Існує багато можливостей, які вплинуть на якість та продуктивність складу послуги. Веб-сервіси передбачають стандарти протоколів. Тому якість послуг (QoS) є пріоритетом для постачальника послуг. Оскільки веб-сервіси складаються динамічно, це може вплинути на якість послуг. Необхідно переконатися, що вимоги щодо QoS веб-сервісів акцентують увагу як на функціональних, так і на нефункціональних. Це покращує продуктивність, надійність, цілісність, доступність, сумісність та безпеку.

У галузі QoS надається велике значення. Існують проблеми з низькою продуктивністю та масштабованістю через неправильний вибір послуг. Існує багато методів підвищення якості та швидкості реагування на послуги. Ця дипломна робота буде зосереджена лише на якості послуг фреймворку веб-композиції. Унікальна особливість динамічної композиції веб-сервісу вимагає ефективного тестування та вибору. Ці веб-сервіси описуються мовою визначення інтерфейсу веб-сервісу, відомому як WSDL (мова виявлення веб-служб). Якість веб-сервісів слід перевірити обидва якісно і якісно.

2.6 Структура проектованого рішення

З вищесказаного неочевидно видно, що склад служби не має певної моделі. Навіть якщо підходів небагато, усі вони статичні, а не динамічні.

Ключовою проблемою тепер став вибір послуг та їх склад для будь-якої даної вимоги.

Цей дипломний звіт буде зосереджено на новому методі, який допоможе розробникам створювати служби під час виконання. Оцінка створеного нового програмного забезпечення стане простою. Крім того, вимірюється продуктивність і якість послуги.

Буде проведено експеримент, щоб показати склад веб-сервісу та його ефективність. В експерименті враховано три веб-сервіси, які разом допоможуть покращити роботу модуля. Це дозволить розробнику писати менше рядків коду для розробки. Додаток створюється простіше та швидше.

2.7 Промислове призначення

Впровадження цієї нової ідеї в сферу сервісно-орієнтованої розробки дозволить розробникам швидко та ефективніше створювати кращі програмні додатки. Що стосується розробки, кодери повинні написати кілька тисяч рядків коду для розробки. Тоді як якщо це використовується, розробники можуть використовувати існуючі веб-сервіси з функціональністю, яка відповідала б вимогам для розробки.

Незважаючи на те, що це дає дизайн програмного додатка, він майже завершив роботу, за винятком підключення до бази даних. Дизайн програмного додатка може бути показаний клієнту. Коли клієнт схвалює, кодер повинен просто працювати над репозиторієм бази даних. Це не тільки скорочує час розробки, але й якість продукту буде кращою.

3 ВЛАСНЕ ОРИГІНАЛЬНЕ АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ РІШЕННЯ ЗАДАЧІ

3.1 Основа процедури проектування

Автори у своїй статті «Життєвий цикл інтеграції для композиції семантичних веб-сервісів» про різні підходи, а також про модель життєвого циклу. При цьому використовується основа цієї каркасної моделі.

Нижче наведено життєвий цикл інтеграції та композиції семантичних веб-сервісів. Життєвий цикл базується на підході зверху вниз, що означає, що аналіз бізнес-потреб починається на початку, потім йде композиція веб-сервісу і закінчується на виконанні. Існуючі численні процеси — це розробка бізнес-процесу, додавання технічних і бізнес-обмежень до процесів, створення моделі композиції та, нарешті, розгортання проекту та виконання — це завершальний етап.

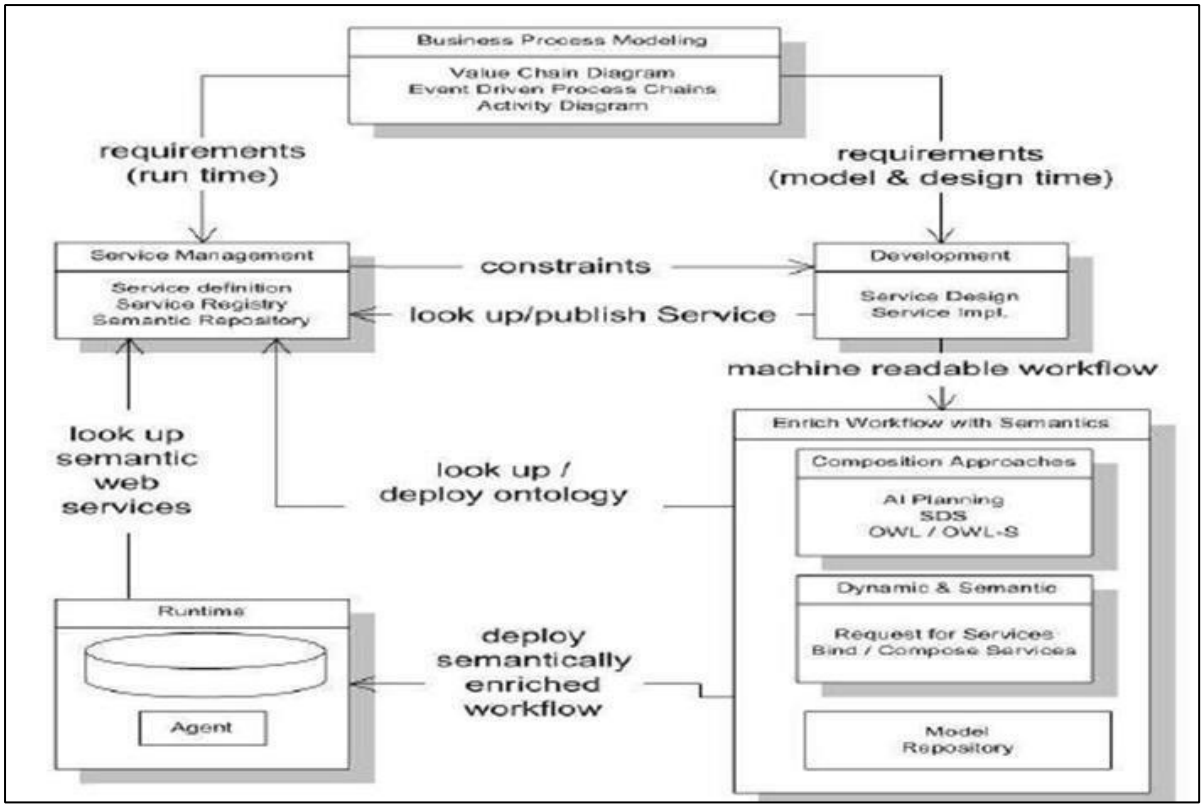


Рисунок 3.1 - Фреймова модель інтеграції веб-сервісів

3.2 Методологія проектованого рішення

Будь-яка структура або архітектура повинні мати етап проектування. Хоча SOA не має визначеного методу проектування чи архітектури для розробки, на основі ідеї, отриманої з моделі програмного процесу та аналогічний метод пропонується для проектування сервісів. Цей процес є повторюваним, що допомагає визначити завдання користувача та як діяти далі. Запропонований метод матиме наступні етапи та функції.

Модифікується каркасна модель із вибором послуг та складом послуг.

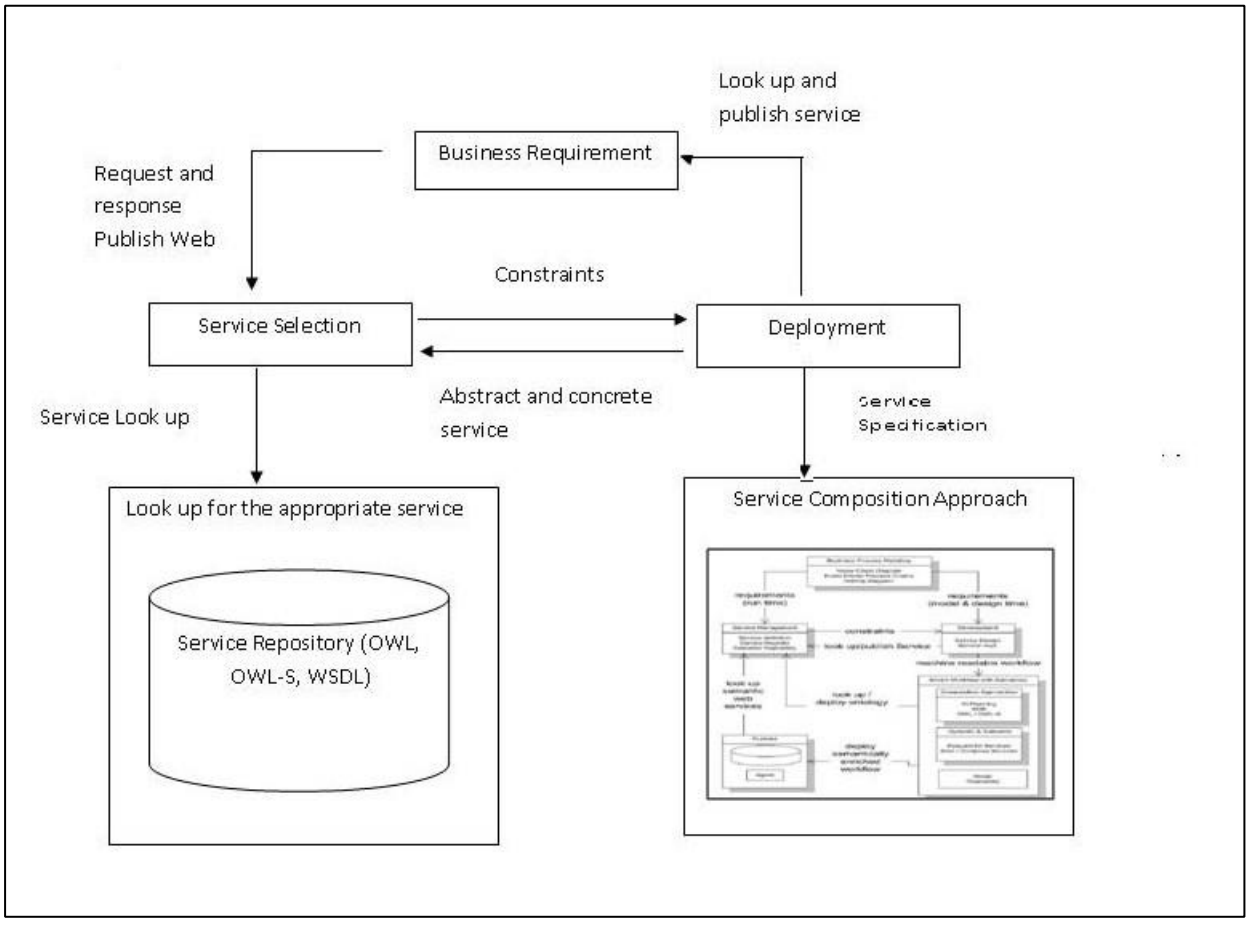


Рисунок 3.2: Структура запропонованого методу

Блок-схема пояснює структуру, в якій виконуються дані для запропонованого методу.

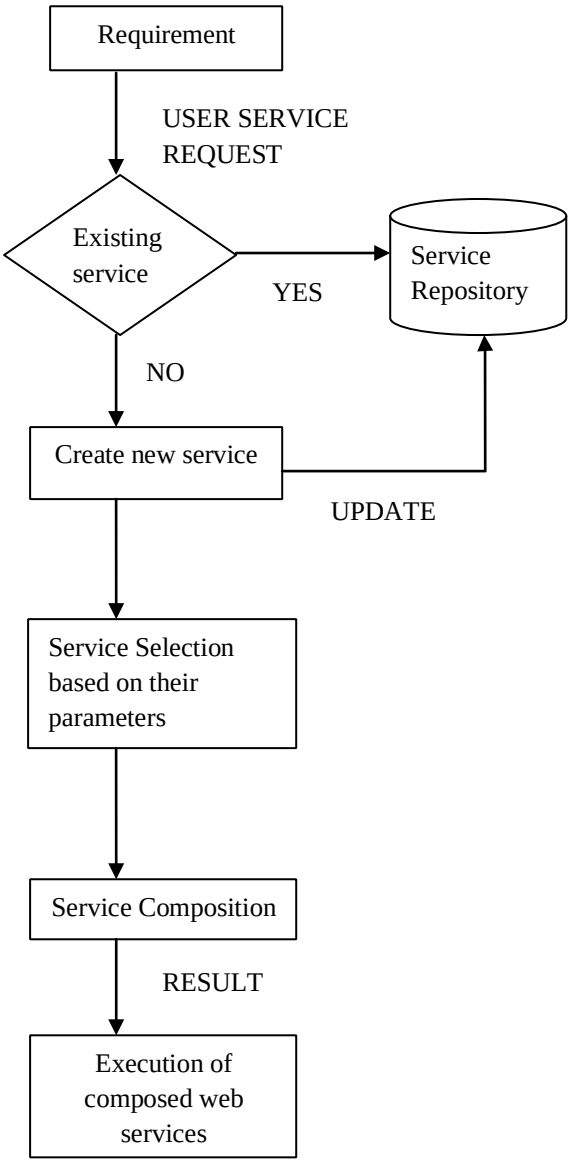


Рисунок 3.3 - Блок-схема запропонованого методу

Нижче наведено схему роботи алгоритму.

1. Аналіз вимоги

2. Виявляє всі потенційні послуги для вимоги, інакше відомої як вибір служби, який базується на описі служб, які зберігаються в репозиторії онтологій

3. Для оцінки створюється найкращий сервіс

4. Вимірюється продуктивність композиції.

Перевага цього нового підходу полягає в тому, що він включає як роботу, щоб зробити його повною програмною системою. Крім того, новою концепцією є композиція сервісу, заснована на взаємодії людини з комп'ютером.

Проводиться подальша оцінка послуг. Нижче описано процес роботи алгоритму в деталях.

Перевага цього нового підходу полягає в тому, що він включає як роботу, щоб зробити його повною програмною системою. Крім того, новою концепцією є композиція сервісу, заснована на взаємодії людини з комп'ютером. Проводиться подальша оцінка послуг.

```
Composite Algorithm
input R- the user request  S- Service  repository S1, S2,S3...
 result C- composition
 data X- set of services

 1   begin
 2    for each R do
 3     X belongs to S;
 4     append (X,C);
 5    end
 6    while X not equal to null do
 7     add X;
 8     if ( X, S)
 9      if(S1 = S2)
10        Check the parameters;
11        if (S1 > S2)
12          return S1;
13        else return S2;
14        end;
15        if valid (C,R) then
16          return C;
17        end
18      end
19     end
21    while R is not in S
22     create new service
23     update the S
24    end
25   end
```

Рисунок 3.4 - Алгоритм запропонованого способу

Крок 1 - Аналіз послуг

Усі наявні або наявні послуги аналізуються. Користувач не матиме можливості вибрати послугу. Користувач надасть розробнику вимогу щодо

розробки нового додатка. Або додаток, за допомогою якого користувач може вказати вимоги та систему, вибираючи найкращий сервіс для програми.

Крок 2 – Вибір послуг

Оскільки всі програми базуються на використанні Інтернету, склад служби буде використовувати веб-сервіси для розробки програми або нового продукту. Відповідно до W3C, веб-сервіси називаються системою програмного забезпечення, яка взаємодіє між машинами в мережі. Вони розділили веб-сервіси на два типи.

1) Простий веб-сервіс: він надає функціональні можливості типу запиту та не підтримує транзакції.

2) Комплексний веб-сервіс: забезпечує основу для співпраці між бізнесом та управління бізнес-процесами.

У більшості випадків буде використовуватися простий веб-сервіс, оскільки це обмін повідомленнями «точка-точка» і забезпечує веб-безпеку.

Крок 3 – Склад послуг

Склад послуг базується на підборі послуг. Склад служби може бути автоматичним або напівавтоматичним або ручним. Існують різні підходи. Деякі з методів обговорюються нижче. Усі ці підходи будуть використовувати SOAP або якийсь інший протокол, схему RDF, WSDL, XML.

Структуру сервісу неможливо зрозуміти без технічних знань. Для композиції необхідні семантичні знання про служби. Це семантичне знання описується за допомогою WSDL (мова опису веб-сервісу). Потім це відображається в XML для передачі служби. Передача здійснюється за допомогою протоколу SOAP. Здебільшого SOAP використовує HTTP для передачі. Universal Description, Discovery and Integration — це реєстр, у якому тепер зберігатимуться документи WSDL та повідомлення SOAP, які пізніше можна використовувати повторно.

Крок 4 – Якість послуг

Після вибору сервісу необхідно перевірити його гнучкість, масштабованість, надійність та продуктивність. Це взагалі відомо як якість послуг (QoS). Це перевіряє мережу, в яку передаються веб-служби. Існують різні алгоритми перевірки якості послуг.

Крок 5 - Відображення з моделлю

Після реєстрації сервісу, дизайну та складання сервісу як наступного кроку, веб-сервіс має бути зіставлений для виконання дії зі створення нової системи. Усі ідентифіковані нові веб-сервіси зберігаються в сховищі служб для майбутнього використання. Це стає дуже корисним для створення нової системи простим способом. Нижче наведено огляд архітектурної моделі веб-сервісу.

Склад веб-сервісу визначається шляхом проведення експерименту, щоб продемонструвати переваги динамічної композиції веб-сервісу. Завдяки цьому експерименту композиція вирішує її проблеми. Модель програмного забезпечення SOA є базовою архітектурою як для постачальників послуг, так і для запитувачів послуг. Для композиції веб-сервісу використовуються три веб-сервіси. Усі три веб-сервіси мають різні функції. Ці унікальні функціональні служби об’єднуються разом для виконання операції разом.

Перша частина експерименту стосується визначення відповідних сервісів для даної вимоги. Друга частина експерименту демонструє склад послуг. Це дозволить переконатися, що веб-служби складаються динамічно, тобто під час виконання. Якість виробництва програмного забезпечення краща.

3.2 Вимоги до експерименту

Для того, щоб успішно розгорнути цей експеримент із залученням веб-сервісів та їх складу, необхідно, щоб там були дотримані наступні вимоги до апаратного та програмного забезпечення. У наведених нижче таблицях представлені вимоги до апаратного та програмного забезпечення відповідно.

Таблиця 3.1 - Вимоги до обладнання

Продукт	Опис	Опис	Технічні характеристики
ПК	Комп'ютер з конфігураціями.	Комп'ютер з конфігураціями.	Точні технічні характеристики вони повинні мати 1 гігагерц або вище процесор. • 1 гігабайт оперативної пам’яті або більше. • Доступно 16 ГБ жорсткого дискового простору • Графіка Direct X 9 пристрій або вище (щоб переглянути результат)

Таблиця 3.2 - Вимоги до програмного забезпечення

Назва програмного езпечення	Опис	Доступність
Microsoft Windows Vista	Операційна система в	Доступно від Microsoft
Netbeans	комп'ютер	http://www.netbeans.org
Java SDK	Ця IDE допомагає в	http://www.sun.com
Сервер Glassfish	розвиток	http://www.glassfish.java.net

3.3 Зразки послуг

Для цього експерименту потрібні 3 зразки веб-сервісів із пов’язаною функціональністю для виконання тесту веб-композиції. Експеримент завершується використанням цих створених служб. Усі ці сервіси мають свої відмінні функції. Основна мета базується на результатах пошуку товару, оплаті вартості та доставці.

Використовується інший сервіс з таким же функціоналом пошуку товару. Це використовується для того, щоб динамічна композиція сервісу могла визначити найбільш відповідну послугу для композиції. Метод реалізації включає ці 3 служби, які засновані на REST (репрезентаційна передача стану). Це намагається описати HTTP або протоколи для обмеження інтерфейсу.

Архітектура на основі REST також може використовувати WSDL для опису базової архітектури. Перевага REST в тому, що він допомагає розробникам використовувати операції GET, POST, PUT, DELETE для взаємодії з іншими веб-сервісами.

3.4 Приклад проектованого сервісу

Дослідження стосується електронного бізнесу/покупок в Інтернеті, які сьогодні популярні. Вимога від кінцевого користувача висувається за допомогою онотології.

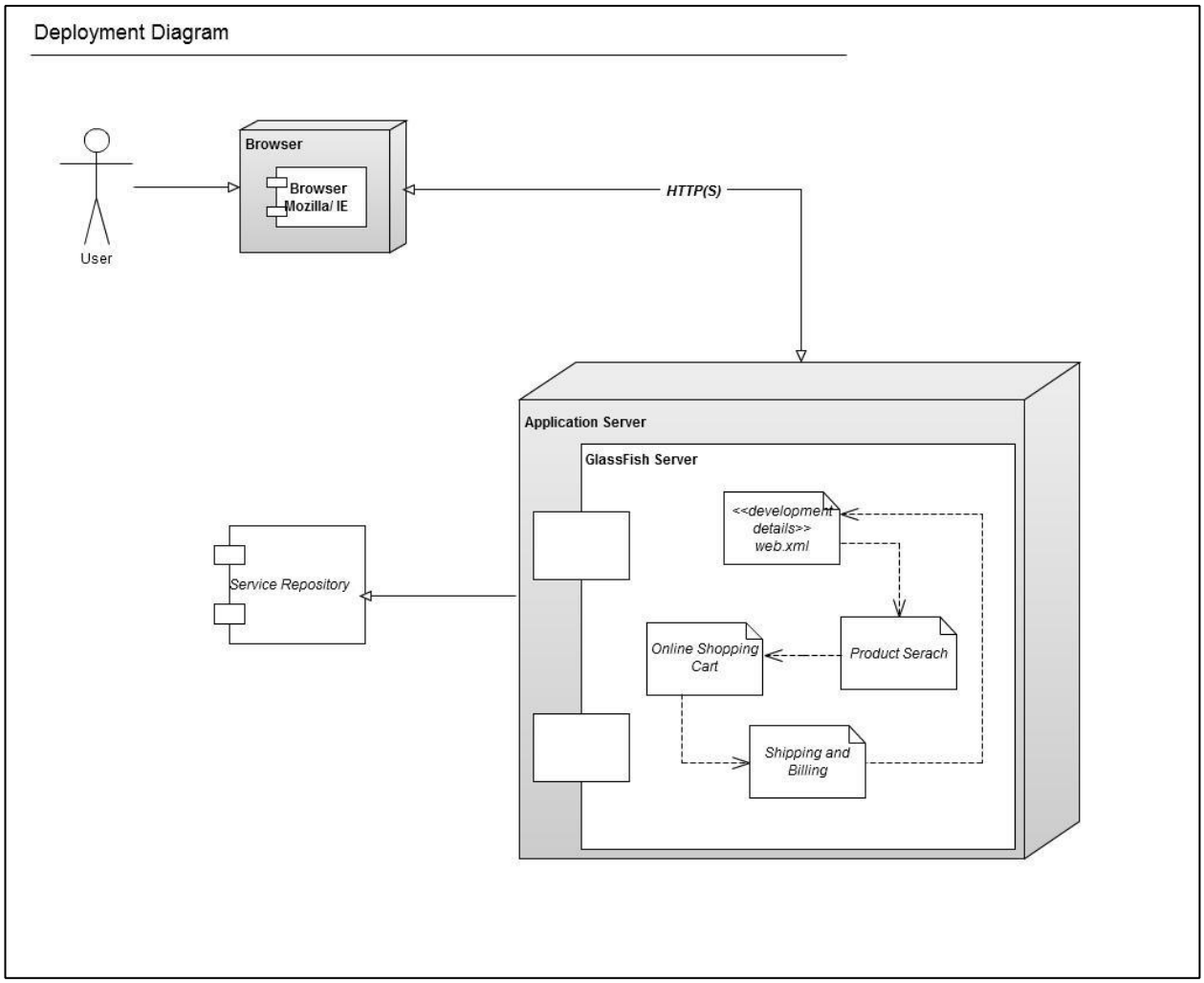


Рисунок 3.1 - Схема розгортання для прикладу

Розробник використовує пакет програми, який міститиме клієнт служби та його адаптери. У сховищі ведеться пошук відповідної служби. Послуга, яка відповідає вимогам, зберігається у списку вибраних послуг. Нарешті, коли всі послуги з вимогами збираються, вони об'єднуються для пакету заявок. Наступна діаграма класів заснована на прикладі електронної комерції. Це пояснює перебіг процесу в додатку.

Інтернет-магазини в наші дні збільшуються. У цій електронній комерції проводиться експеримент зі створення веб-сервісу. Для базового веб-сайту покупок знадобляться три основні служби (1) пошук товарів (2) відображення продуктів пошуку з їх вартістю (3) виставлення рахунку та доставка товару. Головна сторінка в розробці. Тут запускається веб-служба пошукового продукту. У критеріях пошуку продукту вводиться назва продукту. Відповідно до назви продукту відображатимуться результати пошуку. Протягом цього часу здійснюється виклик відповідної веб-служби. Останнім кроком є вибір товару для виставлення рахунку та доставки. Ця веб-служба викликається. Кінцевим результатом цього експерименту є те, що веб-сторінка легко розробляється користувачем у інтерактивному режимі, просто використовуючи доступні веб-служби.

Припустимо, у сховищі є два види пошуку продуктів, вибір послуги вибирає найбільш підходящий сервіс. Наприклад, якщо клієнт хоче послуги з електронних продуктів, а сховище послуг має два види сервісу, який шукає в одному електроніку, а інший — для відеоігор. Вибирається служба результатів пошуку продукту відповідної послуги. Наступна структура архітектури дасть краще уявлення про цю концепцію.

3.5 Аналіз отриманих результатів

Запропонована методологія апробується в системі інтернет-магазинів. В якості початкового кроку розглядаються 3 послуги для динамічної композиції

послуг. Наступним кроком був створений ще один сервіс із тими ж функціями кредитної картки. Вибирається найкращий або найбільш підходящий веб-сервіс.

Наступні малюнки показують результат запуску кожної веб-служби. Перша сторінка – це сторінка .jsp, яка дає основний контур сторінки. У цьому випадку здійснюється пошук продукту, тому відображається сторінка пошуку.

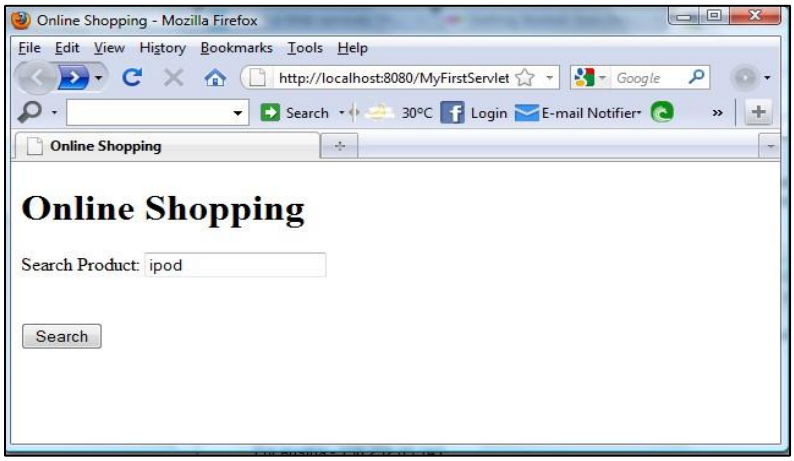


Рисунок 3.2 - Домашня сторінка інтерфейсу користувача

Шукається товар «ipod». Результати пошуку (з їх вартістю) представлені на скріншоті нижче. Відбувається доступ до веб-служби результатів пошуку продуктів і надаються результати пошуку за ключовим словом. Якщо пошукового продукту немає, з'явиться екран із повідомленням «Пошуковий продукт не знайдено».

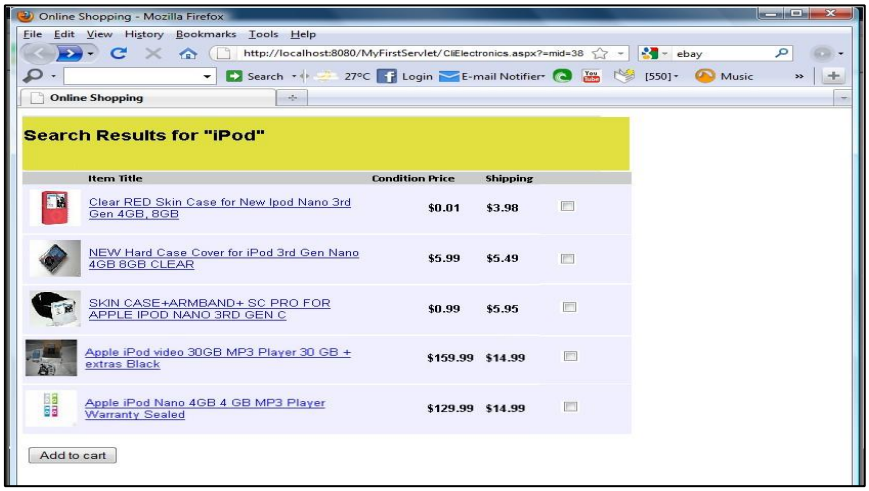


Рисунок 3.3 - Сторінка пошуку в інтерфейсі користувача

Наступним кроком вибирається продукт, який потрібно виставити на рахунок і відправити. На скріншоті нижче показано перевірений продукт.

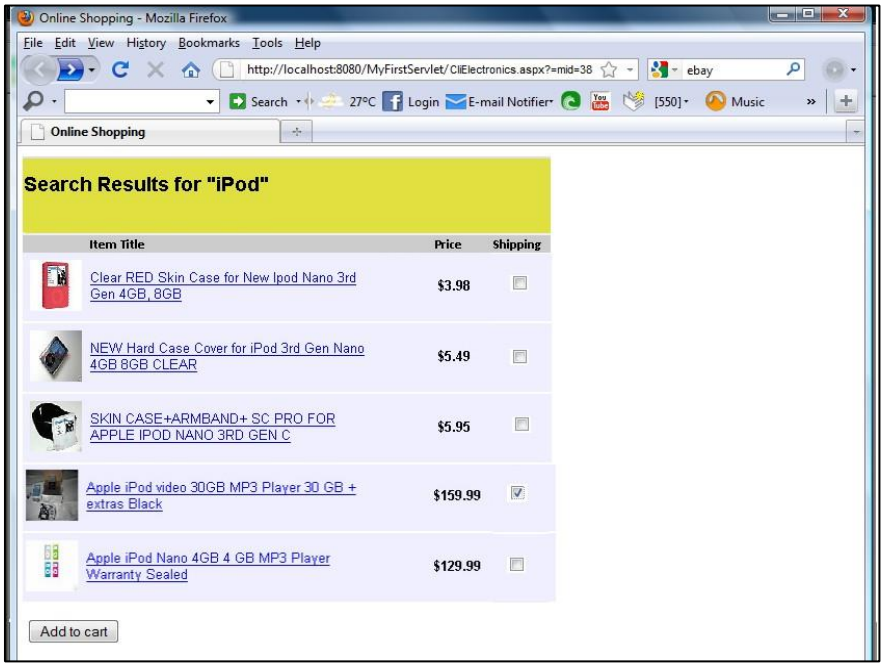


Рисунок 3.4 - Сторінка пошуку в інтерфейсі користувача 2

Нарешті, крок включає адресу виставлення рахунку та дані кредитної картки. Коли натискається кнопка «додати в кошик», вона викликає дані третьої веб-служби кредитної картки.

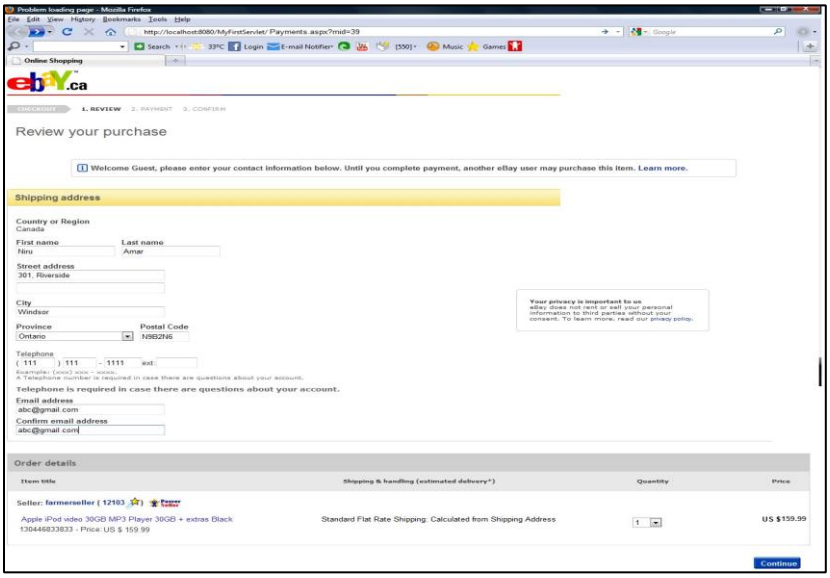


Рисунок 3.5 - Варіанти оплати через інтерфейс користувача

Проводиться оплата товару, що купується.

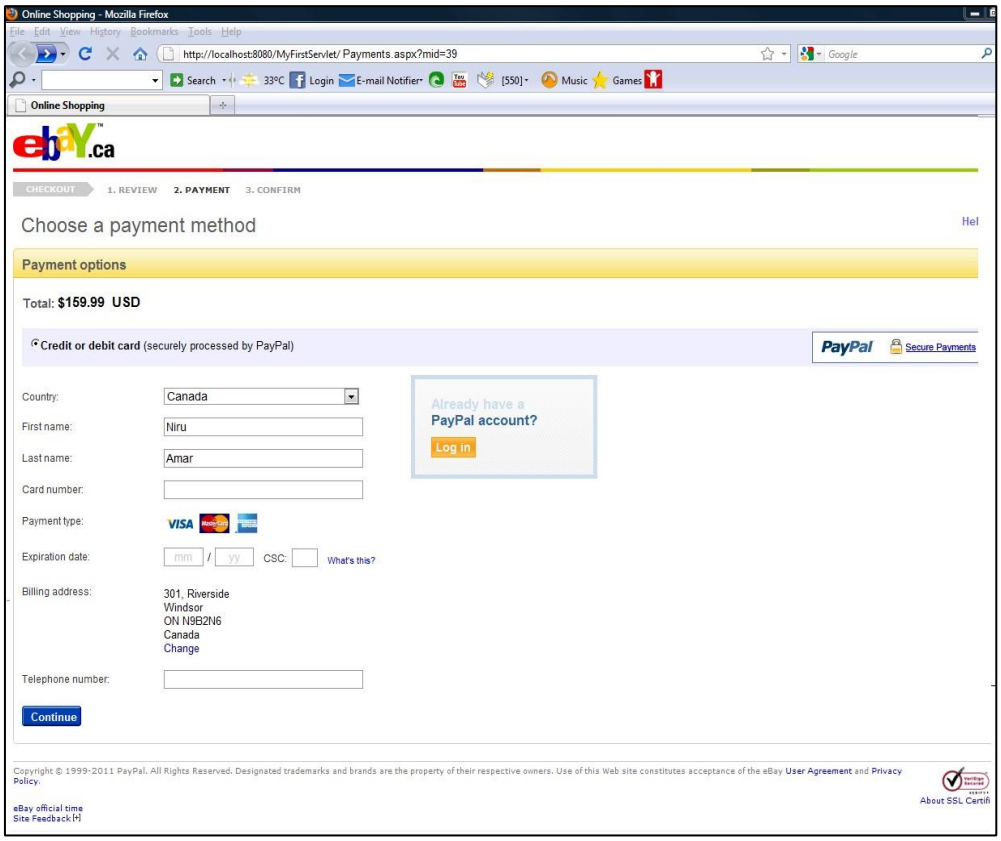


Рисунок 3.6 - Варіанти оплати в інтерфейсі користувача2

Нарешті, підтвердження говорить, що оплата здійснена успішно, і товар буде доставлено.

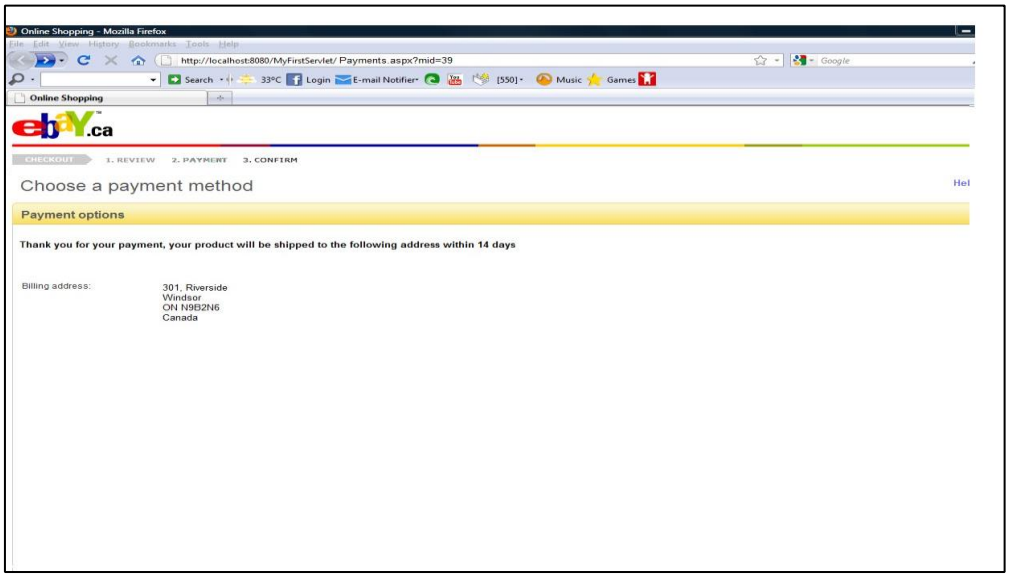


Рисунок 3.7 - Сторінка підтвердження інтерфейсу користувача

Це проект фактичної програмної системи. Якщо ці служби підключаються до програмної системи, це робить систему завершеною.

3.6 Аналіз отриманих сервісів

Вибір послуги відбувається на останньому етапі впровадження прийнятого прикладу. Вибирається найкращий сервіс. Для даного прикладу користувач надає дані кредитної картки для виставлення рахунку. У цей момент має бути запущена відповідна послуга кредитної картки.

Цей процес відбувається в платіжному шлюзі. Платіжний шлюз – це постачальник послуг електронної комерції, який авторизує платежі. Це також гарантує, що дані кредитної картки захищені. Користувачі мають право вибору послуг.

Таблиця 3.3 - Таблиця продуктивності

	Послуга А Послуга Б	Послуга А Послуга Б
Вартість	7	3.50
Середній час відповіді	350	700
Середнє значення переданих даних	10	20
Цінність ефективності з кожною категорією зважена на 33%	82.5%	66%

Тут виконується виконання чутливості служби. У наступній таблиці наведено вартість і продуктивність.

Деякі користувачі можуть бути готові платити більше за кращий сервіс, але інші можуть віддати перевагу пожертвувати продуктивністю за меншу вартість. Клієнта цікавляться цими факторами для ідеального вибору послуги. Згідно з наведеним прикладом, усі ці деталі надаються раніше, а два веб-сервіси для оплати кредитною карткою мають свою власну різницю в продуктивності. Якщо

клієнт обирає веб-сервіс «Master Card», тобто послугу А, продуктивність буде кращою, ніж послуга В, тобто «VISA».

ВИСНОВКИ

В дипломній роботі використана технологія, орієнтована на сервіси (SOA) – це нова технологія, яка була адаптована та прийнята підприємствами та розробниками у всьому світі. Кажуть, що це майбутнє Всесвітньої мережі. SOA покращує можливість повторного використання завдяки використанню нестрогих зв'язків і сервісів. Майбутнє буде здебільшого за веб-основою, і SOA відіграватиме важливу роль. Процес розробки нових систем скоротиться, а використання існуючих послуг/технік для побудови системи прийде, оскільки це буде легше через менші витрати часу та з меншою кількістю помилок. Хоча у SOA є деякі недоліки, вони несуттєві у порівнянні з перевагами.

Компонування та дизайн веб-сервісу можна виконувати автоматично за допомогою різних алгоритмів. Їх продуктивність і якість обслуговування також можна виміряти. Таким чином можна розробити надійну та ефективну систему. Подальше впровадження чутливості сервісу та динамічного складу сервісу допоможе розробити кращу програмну систему, яка є тенденцією сьогодення. Це є також причиною візуалізації програмного забезпечення.

СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА

1 Kelly S, Gibson N, Holland CP. and Light B. A Business Perspective of Legacy Systems, Communications of the Association for Information Systems 1999, 2(Article 7 – July). http://cais.isworld.org

2 Ward J. Principles of Information Management. Routledge: London, 1995; 252- 253.

3 Bennett K. Legacy Systems: Coping with Success. IEEE Software 1994; 12(1): 19-23.

4 Kim Y. Improving Legacy Systems Maintainability. Information Systems Management 1997; 14(1): 7-11.

5 Adolph W.S. Cash Cow in the Tar Pit: Reengineering a Legacy System. IEEE Software 1996; 10(5): 41

6 Lehman MM. Programs, Life-Cycles, and the Laws of Software Evolution. Proceedings of the IEEE 1980; 68(9): 1060-1076.

7 Hammer M. Reengineering work: Don’t automate, obliterate. Harvard Business Review 1990; 68 (4): 104-112.

8 Davenport TH and Short JE. The New Industrial Engineering: Information Technology and Business Process Redesign. Sloan Management Review 1990; 32(4): 11-27.

9 Price Waterhouse. Price Waterhouse Information Technology Review 1995/1996. Price Waterhouse: London, 1996; 27. Pre-publication draft – accepted in Special Issue on the Maintenance of Large Package Software Maintenance in the Journal of Software Maintenance: Research and Practice. Not for any form of distribution without the prior written consent of the author. Copyright © 2001 John Wiley & Sons, Ltd. J. Softw. Maint: Res. Pract, 2001;

10 KPMG. Exploiting Packaged Software. KPMG: London, 1998; 5.

11 Scheer AW and Habermann F. Making ERP a Success, Communications of the Association for Computing Machinery 2000. 43(4): 57-61.

12 Butler J. Risk Management Skills Needed in a Packaged Software Environment. Information Systems Management 1999. 16(3): 15-20.

13 Klaus H, Rosemann M, and Gable GG. What is ERP? Information Systems Frontiers 2000. 2(2): 141-162.

14 Sumner M. Risk Factors in Enterprise-wide/ERP Projects. Journal of Information Technology 2000. 15(4): 317-327.

15 Willcocks L and Sykes R. The Role of the CIO and IT Function in ERP. Communications of the Association for Computing Machinery 2000. 43(4): 33- 38

16 Bingi P, Sharma MK, and Godla JK. Critical Issues Affecting An ERP Implementation. Information Systems Management 2000. 16(3): 7-14.

17 Light B and Holland C. Enterprise Resource Planning Systems: Impacts and Future Directions. In Henderson P. (ed.) Systems Engineering for Business Process Change – 2000. Springer: London; 117-126.

18 Holland C and Light B. Global Enterprise Resource Planning Implementation. In Proceedings Of The 32nd Annual Hawaii International Conference On System Sciences – 1999. IEEE Computer Society Press: Los Alamitos, CA. (CD-ROM)

19 Holland C and Light B. A Critical Success Factors Model for ERP Implementation. IEEE Software 1999. 16(3): 30-36.

20 Davenport TH. Putting The Enterprise Into The Enterprise System. Harvard Business Review 1998. 76(4): 121-131.

21 Stedman C. Reebok Steps Up Use of SAP Apparel Apps, 1999. http://www.computerworld.com [29 August 2001].

22 Stedman C. Apparel Makers Anxiously Await SAP Fixes, 1999. http://www.computerworld.com [29 August 2001].

23 Markus ML, Axline S, Petrie D, and Tanis C. Learning from Adopters' Experiences with ERP: Problems Encountered and Success Achieved. Journal of Information Technology 2000. 15(4): 245-265.

24 Lucas HC Jr, Walton EJ, and Ginzberg MJ. Implementing Packaged Software. Management Information Systems Quarterly 1988. 12(4): 536-549.

25 Soh C, Kien SS, and Tay-Yap J. Cultural Misfits: is ERP a Universal Solution? Communications of the Association for Computing Machinery 2000. 43(4): 47- 51.

26 Scott JE and Kaindl L. Enhancing Functionality in an Enterprise Software Package. Information and Management 2000. 37(3): 111-122.

27 Pettigrew AM. Contextualist Research: A Natural Way to Link Theory and Practice. In Lawler EE, Mohrman AM, Mohrman SA, Ledford GE, Cummings TG and Associates. (eds) Doing Research that is Useful in Theory and Practice - 1985. Jossely-Bass: San Fransisco CA; 22-274.

28 Miles MB and Huberman AM. Qualitative Data Analysis: An Expanded Sourcebook, 2nd Edition, 1994. Sage Publications Inc: Thousand Oaks CA; 10. Pre-publication draft – accepted in Special Issue on the Maintenance of Large Package Software Maintenance in the Journal of Software Maintenance: Research and Practice. Not for any form of distribution without the prior written consent of the author. Copyright © 2001 John Wiley & Sons, Ltd. J. Softw. Maint: Res. Pract, 2001;

29 Yin RK. Case Study Research: Design and Methods, 2nd Edition, 1994. Sage Publications: London; 4- 30 Gummesson E. Qualitative Methods in Management Research 1991. Sage Publications, Inc: Newbury Park, CA; 75.

31 Chapin N, Hale JE, Khan KM, Ramil JF, and Tan W. Types of Software Evolution and Software Maintenance. Journal of Software Maintenance: Research and Practice 2000. 13(1): 3-30.

32 Stake RE. The Art Of Case Study Research 1995. Sage Publications Inc.: Thousand Oaks CA; 113.

33 Gill J and Johnson P. Research Methods For Managers 1991. Paul Chapman Publishing: London; 159

34 Silverman D. Qualitative Research: Meanings or Practices? Information Systems Journal 1998. 8(1): 3-

35 Light B, Holland CP, and Wills K. ERP and Best of Breed: A Comparative Analysis. Business Process Management Journal 2001. 7(3): 195-204

36 G. Amadio and A. Ananya and J.Apostolakis and A.Arora and M.Bandieramonte and A.Bhattacharyya and C.Bianchini and R.Brun and P.Canal and F.Carminati and L.Duhem and D.Elvira and A.Gheata and M.Gheata and I.Goulas and R.Iope and S.Jun and G.Lima and A.Mohanty and T.Nikitina and M.Novak and W.Pokorski and A.Ribon and R.Sehgal and O.Shadura and S.Vallecorsa and S.Wenzel and Y.Zhang, GeantV: from CPU to accelerators, Journal of Physics: Conference Series, 762, 1, p.012019, 2016

37. K. Deb and A. Pratap and S. Agarwal and T. Meyarivan,A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, 2002, Vol.6, p.182-197,Doi:10.1109/4235.996017

38. Seada, Haitham, and Kalyanmoy Deb. "U-NSGA-III: A A Unified Evolutionary Optimization Procedure for Single, Multiple, and Many Objectives", IEEE Trans. Evolutionary Computation 20(3): 358-369 (2016)

39. K. Deb and L. Thiele and M. Laumanns and E. Zitzler, Scalable Test Problems for Evolutionary Multi-Objective Optimization, Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, Springer, 2005

БІБЛІОГРАФІЧНА ДОВІДКА

Тема дипломної роботи бакалавра: "Композиційні моделі налаштування програмного забезпечення та сервісів"

Обсяг пояснювальної записки: 58 аркушів

Дата закінчення дипломної роботи 13 червня 2021р.

Підпис студента _____